# SBRC 2014

Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

# Anais da Trilha Principal e do Salão de Ferramentas

Florianópolis - SC
5 a 9 de Maio de 2014


# XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

5 a 9 de Maio de 2014
Florianópolis - SC

# Anais

## Trilha Principal e Salão de Ferramentas

**Editora**

Sociedade Brasileira de Computação (SBC)

**Organizadores**

Luciano Paschoal Gaspary (UFRGS)

Markus Endler (PUC-RIO)

Alfredo Goldman (USP)

Joni da Silva Fraga (UFSC)

Frank Siqueira (UFSC)

**Realização**

Universidade Federal de Santa Catarina (UFSC)

**Promoção**

Sociedade Brasileira de Computação (SBC)

Laboratório Nacional de Redes de Computadores (LARC)

**Capa**: Vanessa Umbelino (PostMix)
**Produção Editorial**: Roberto Willrich (UFSC)

**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 (32: 2014: Florianópolis, SC)

Anais / 32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos; organizado por Joni da Silva Fraga... [et al.] - Porto Alegre: SBC, c2014

1081 p.

SBRC 2014
Realização: Universidade Federal de Santa Catarina
ISSN: 2177-496X

1. Redes de Computadores - Congressos. 2. Sistemas Distribuídos- Congressos. I. Fraga, Joni da Silva. II. Sociedade Brasileira de Computação. III. Título.

# Promoção

## Sociedade Brasileira de Computação (SBC)

### Diretoria

**Presidente**
Paulo Roberto Freire Cunha (UFPE)

**Vice-Presidente**
Lisandro Zambenedetti Granville (UFRGS)

**Diretora Administrativa**
Renata de Matos Galante (UFRGS)

**Diretor de Finanças**
Carlos André Guimarães Ferraz (UFPE)

**Diretor de Eventos e Comissões Especiais**
Altigran Soares da Silva (UFAM)

**Diretora de Educação**
Mirella Moura Moro (UFMG)

**Diretor de Publicações**
José Viterbo Filho (UFF)

**Diretora de Planejamento e Programas Especiais**
Claudia Lage Rebello da Motta (UFRJ)

**Diretor de Secretarias Regionais**
Marcelo Duduchi Feitosa (CEETEPS)

**Diretor de Divulgação e Marketing**
Edson Norberto Caceres (UFMS)

**Diretor de Relações Profissionais**
Roberto da Silva Bigonha (UFMG)

**Diretor de Competições Científicas**
Ricardo de Oliveira Anido (UNICAMP)

**Diretor de Cooperação com Sociedades Científicas**
Raimundo José de Araujo Macêdo (UFBA)

**Diretor de Articulação de Empresas**
Avelino Francisco Zorzo (PUC-RS)

# Promoção

## Sociedade Brasileira de Computação (SBC)

### Conselho

#### Mandato 2013-2017

Alfredo Goldman (IME/USP)

José Palazzo Moreira de Oliveira (UFRGS)

Maria Cristina Ferreira de Oliveira (ICMC/USP)

Thais Vasconcelos Batista (UFRN)

Wagner Meira Junior (UFMG)

#### Mandato 2011-2015

Ariadne Carvalho (UNICAMP)

Carlos Eduardo Ferreira (IME - USP)

Jose Carlos Maldonado (ICMC - USP)

Luiz Fernando Gomes Soares (PUC-Rio)

Marcelo Walter (UFRGS)

#### Suplentes - 2013-2015

Alessandro Fabrício Garcia (PUC-Rio)

Aline Maria Santos Andrade (UFBA)

Daltro José Nunes (UFRGS)

Karin Koogan Breitman (PUC-Rio)

Rodolfo Jardim de Azevedo (UNICAMP-IC)

# Promoção

## Laboratório Nacional de Redes de Computadores (LARC)

### Diretoria 2012-2014

**Diretor do Conselho Técnico-Científico**

Elias P. Duarte Jr. (UFPR)

**Diretor Executivo**

Luciano Paschoal Gaspary (UFRGS)

**Vice-Diretora do Conselho Técnico-Científico**

Rossana Maria de C. Andrade (UFC)

**Vice-Diretor Executivo**

Paulo André da Silva Gonçalves (UFPE)

**Membros Institucionais**

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

# Realização

## Comitê de Organização

### Coordenação Geral

Joni da Silva Fraga (UFSC)

Frank Siqueira (UFSC)

### Coordenação do Comitê de Programa

Luciano Paschoal Gaspary (UFRGS)

Markus Endler (PUC-RIO)

### Coordenação de Palestras e Tutoriais

Antonio Alfredo Ferreira Loureiro (UFMG)

### Coordenação de Painéis e Debates

Carlos Kamienski (UFABC)

# Realização

## Coordenação de Minicursos

Carlos Alberto Maziero (UTFPR)

## Coordenação de Workshops

Carlos André Guimarães Ferraz (UFPE)

## Coordenação do Salão de Ferramentas

Alfredo Goldman (USP)

## Comitê Consultivo

Dorgival Guedes (UFMG)
Jussara Almeida (UFMG)
Elias Procópio Duarte Jr (UFPR)
José Ferreira de Rezende (UFRJ)
Jacir Luiz Bordim (UnB)
Rafael Timóteo de Sousa Júnior (UnB)
William Ferreira Giozza (UnB)
Carlos André Guimarães Ferraz (UFPE)
José Augusto Suruagy Monteiro (UFPE)

# Realização

## Organização Local

Carlos Barros Montez (UFSC)

Edison Tadeu Lopes Melo (UFSC)

Guilherme Eliseu Rhoden (PoP-SC)

Leandro Becker (UFSC)

Mário A. R. Dantas (UFSC)

Michelle Wangham (Univali)

Ricardo Felipe Custódio (UFSC)

Roberto Willrich (UFSC)

Rodrigo Pescador (PoP-SC)

Rômulo Silva de Oliveira (UFSC)

## Secretaria do SBRC 2014

Juliana Clasen (UFSC)

Jade Zart (UFSC)

# Mensagem dos Coordenadores Gerais

Boas-vindas a todos os participantes do 32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2014). 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 novamente em Florianópolis, Santa Catarina.

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 2014 engloba um conjunto de atividades bastante abrangente e de alta qualidade técnica. São 21 sessões técnicas nas quais serão apresentados 66 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 cinco palestras proferidas por pesquisadores internacionalmente renomados e três painéis abordando temas extremamente atuais: SDN, 5G e BDaaS. São oferecidos seis minicursos, voltados à formação e atualização dos participantes sobre tópicos selecionados através de chamada pública. Adicionalmente, nove 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 Markus Endler, Luciano Gaspary, Antonio Loureiro, Carlos Kamienski, Carlos Ferraz, Carlos Maziero e Alfredo Goldman pelo tempo e empenho na efetivação das várias trilhas do SBRC 2014.

Por fim, exaltamos o trabalho intenso e extremamente competente realizado pelo integrantes do Comitê de Organização Local. Agradecemos ainda às diretorias do SBC e do LARC, promotores do SBRC, pela confiança depositada em nós, e pelo competente apoio organizacional prestado pela equipe administrativa da SBC. Somos também gratos aos integrantes do Comitê Consultivo do SBRC e à coordenação da Comissão Especial de Redes de Computadores e Sistemas Distribuídos da SBC, pelos aconselhamentos e pelo suporte financeiro inicial prestado à organização do SBRC 2014. Gostaríamos de agradecer ainda aos patrocinadores do simpósio: o Comitê Gestor da Internet no Brasil, às agências governamentais de fomento – CNPq, CAPES e FAPESC – 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.

## Mensagem dos Coordenadores Gerais

Por fim, agradecemos aos Departamentos de Automação e Sistemas e de Informática e Estatística da UFSC, por prestarem o indispensável suporte para a realização do SBRC.

Desejamos a todos os participantes que tenham uma excelente estadia em Florianópolis e que tirem proveito de todo o conhecimento que as atividades o SBRC 2014 tem a lhes oferecer.

**Joni da Silva Fraga (UFSC)**

**Frank Siqueira (UFSC)**

Coordenadores Gerais do SBRC 2014

# Mensagem dos Coordenadores do Comitê de Programa

Ao longo de mais de três décadas de realização, o SBRC tornou-se 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 32ª edição, realizada em Florianópolis, a comunidade continuou a prestigiar o evento com um excelente número de submissões. De um total de 221 artigos completos submetidos, foram aceitos 66 para apresentação e publicação, perfazendo uma taxa de aceitação em torno de 30%.

Os 221 artigos foram submetidos a uma avaliação criteriosa executada pelos 112 membros do Comitê de Programa e por 133 revisores associados a eles. Cada artigo foi avaliado por, pelo menos, 3 especialistas na área, sendo que a maior parte 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 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.

Nos anais encontram-se os textos dos artigos completos selecionados, cobrindo tópicos como redes de rádios cognitivos, redes veiculares, redes definidas por software, Internet do futuro, computação em nuvem e segurança de redes e serviços. 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, pela primeira vez na história do SBRC, todos os artigos aceitos para a Trilha Principal - e apresentados no SBRC - serão publicados no IEEE Xplore, alavancando esses artigos para um novo patamar de visibilidade.

# Mensagem dos Coordenadores do Comitê de Programa

Gostaríamos de expressar os nossos sinceros agradecimentos aos membros do Comitê de Programa e revisores pelo árduo trabalho realizado. A grande maioria dos pareceres foi de uma qualidade excepcional, demonstrando grande objetividade, imparcialidade e com muitas recomendações para os autores. Ficamos muito satisfeitos, e 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. Um agradecimento especial também vai aos coordenadores gerais deste 32° SBRC – professores Joni da Silva Fraga e Frank Siqueira – pela confiança depositada em nosso trabalho e pelo apoio prestado ao longo de todo o processo. Além disso, estendemos o nosso muito obrigado para a equipe de suporte do JEMS, que atendeu prontamente os nossos pedidos. Por fim, queremos agradecer aos autores de artigos e participantes do Simpósio pelo seu interesse no evento, bem como pelo trabalho investido no aprimoramento dos artigos e na preparação das apresentações. Esperamos e desejamos que todos vocês apreciem a programação técnica e que os trabalhos e as conversas no café sirvam de inspiração para novos projetos e cooperações. Tenham ótimos dias aqui em Floripa, a Ilha da Magia!

Florianópolis, maio de 2014.

**Luciano Paschoal Gaspary (UFRGS)**

**Markus Endler (PUC-RIO)**

Coordenadores do Comitê de Programa

## Mensagem do Coordenador do Salão de Ferramentas

Bem-vindos ao Salão de Ferramentas do SBRC!

Ao ser chamado para organizar o Salão deste ano, fiquei particularmente contente, por diversas razões. Primeiro, foi a minha estreia no quadro de organizadores do SBRC, eu já havia participado antes como autor, apresentador, membro do comitê de programa e organizador de workshop. Mas, confesso que participar de forma mais efetiva na organização foi uma experiência muito enriquecedora. Saber um pouco de todo o trabalho que há nos bastidores foi bastante motivador, ao menos espero cumprir de forma satisfatória a parte que me coube.

Segundo, com a experiência que venho adquirindo após o meu doutoramento, acredito cada vez mais na importância de aproximar a pesquisa em geral de algo mais concreto e com potencial de aplicação na prática. Neste sentido, a implementação de ferramentas vai bem além das simulações, do pseudocódigo e de garantias de otimalidade.

Finalmente, o papel de coordenador deste evento foi extremamente facilitado, por duas razões, o apoio do ótimo comitê de programa e o alto nível das ferramentas submetidas. Neste ano, tivemos 18 submissões, sendo que destas 9 foram aceitas para apresentação. Cada ferramenta foi revisada por ao menos três revisores. Sendo que para as ferramentas aceitas, não houve nenhuma recomendação negativa, após uma fase de discussão entre os revisores. O comitê foi formado por 27 pesquisadores, sendo que foram indicados por eles mais 4 revisores adicionais.

Para terminar, gostaria de agradecer todo o apoio que tive dos organizadores, Joni e Frank, que pacientemente responderam às minhas dúvidas e me apoiaram em tudo que foi necessário.

Um ótimo Salão a todos!!!!

**Alfredo Goldman (USP)**

Coordenador do Salão de Ferramentas

## Comitê de Programa do SBRC 2014

Aldri dos Santos (UFPR)
Alex Borges Vieira (UFJF)
Alfredo Goldman (IME-USP)
Aline Carneiro Viana (INRIA, França)
Ana Paula Couto da Silva (UFMG)
André Costa Drummond (UnB)
André Castelo Branco Soares (UFPI)
Anelise Munaretto (UTFPR)
Antônio Jorge Gomes Abelém (UFPA)
Antônio Tadeu Azevedo Gomes (LNCC)
Antonio Alfredo F. Loureiro (UFMG)
Antonio Augusto de A. Rocha (UFF)
Artur Ziviani (LNCC)
Bruno Schulze (LNCC)
Carlos André Guimarães Ferraz (UFPE)
Carlos Kamienski (UFABC)
Carlos Alberto Vieira Campos (UNIRIO)
Carmelo José A. Bastos Filho (UFPE)
Célio Vinicius N. de Albuquerque (UFF)
Christian E. Rothenberg (UNICAMP)
Daniel Fernandes Macedo (UFMG)
Daniel Ratton Figueiredo (UFRJ)
Daniel Sadoc Menasche (UFRJ)
Dênio Mariz Timóteo de Sousa (IFPB)
Djamel Fawzi Hadj Sadok (UFPE)
Dorgival Olavo Guedes Neto (UFMG)
Edmundo de Souza e Silva (UFRJ)
Edmundo Madeira (UNICAMP)
Eduardo Cerqueira (UFPA)
Elias Duarte Jr. (UFPR)
Fabíola Gonçalves Pereira Greve (UFBA)
Fabio Costa (UFG)
Fabio Kon (IME-USP)
Fatima Duarte-Figueiredo (PUC Minas)
Fábio Luciano Verdi (UFSCAR)
Fernando Luis Dotti (PUCRS)
Fernando Pedone (University of Lugano, Suíça)
Flavia Coimbra Delicato (UFRJ)
Francisco Vilar Brasileiro (UFCG)
Geraldo Robson Mateus (UFMG)
Gustavo Bittencourt Figueiredo (UFBA)
Gustavo Souza Pavani (UFABC)
Humberto T Marques-Neto (PUC Minas)
Igor Monteiro Moraes (UFF)
Italo Cunha (UFMG)
Jacir Luiz Bordim (UnB)
Javier Baliosian (University of the Republic, Uruguai)
Jó Ueyama (ICMC-USP)
Jean-Marie Farines (UFSC)
Joaquim Celestino Júnior (UECE)
Joni da Silva Fraga (UFSC)
José Neuman de Souza (UFC)
José Ferreira de Rezende (UFRJ)
José Augusto Suruagy Monteiro (UFPE)
José Marcos Nogueira (UFMG)
Jussara Marques de Almeida (UFMG)
Keiko Fonseca (UTFPR)
Kelvin Lopes Dias (UFPE)
Kleber Vieira Cardoso (UFG)
Lau Cheuk Lung (UFSC)
Leandro Aparecido Villas (UNICAMP)
Liane M; Rockenbach Tarouco (UFRGS)
Linnyer Beatrys Ruiz (UEM)
Lisandro Z. Granville (UFRGS)
Luci Pirmez (UFRJ)
Lucia Drummond (UFF)
Luciano Paschoal Gaspary (UFRGS)
Luis Felipe de Moraes (UFRJ)
Luis Rodrigues (INESC-ID/IST - Universidade de Lisboa, Portugal)

## Comitê de Programa do SBRC 2014

Luis Henrique M. K. Costa (UFRJ)
Luiz Fernando Bittencourt (UNICAMP)
Magnos Martinello (UFES)
Marcel William Rocha da Silva (UFRRJ)
Marcelo Dias de Amorim (UPMC, França)
Marcelo Pias (Globosense e University of Cambridge, UK)
Marcelo Gonçalves Rubinstein (UERJ)
Marco Aurélio Spohn (UFFS)
Marcos Salvador (CPqD)
Marcos José Santana (ICMC-USP)
Marinho P. Barcellos (UFRGS)
Markus Endler (PUC-Rio)
Mauro Sergio Pereira Fonseca (PUCPR)
Michael Stanton (RNP e UFF)
Michele Nogueira (UFPR)
Michelle Silva Wangham (Univali)
Miguel Elias Mitre Campista (UFRJ)
Moises Renato Nunes Ribeiro (UFES)
Nabor Mendonca (UNIFOR)
Natalia Castro Fernandes (UFF)
Nazareno Andrade (UFCG)
Nelson Fonseca (UNICAMP)
Noemi Rodriguez (PUC-Rio)
Orlando Loques (UFF)
Paulo Aguiar (UFRJ)
Paulo Cunha (UFPE)
Paulo Ferreira (INESC-ID/IST - Universidade de Lisboa)
Paulo de Figueiredo Pires (UFRJ)
Paulo André da Silva Gonçalves (UFPE)
Pedro Braconnot Velloso (UFF)
Rafael Timóteo de Sousa Jr (UnB)
Raimundo José de A. Macêdo (UFBA)
Raquel A. de Freitas Mini (PUC Minas)
Ronaldo Alves Ferreira (UFMS)
Ronaldo Moreira Salles (IME)
Rosa Maria Meri Leão (UFRJ)
Rossana Maria de Castro Andrade (UFC)
Silvana Rossetto (UFRJ)
Stênio Fernandes (UFPE)
Thais Vasconcelos Batista (UFRN)
Vinod Rebello (UFF)
Wagner Meira Jr. (UFMG)
William Ferreira Giozza (UnB)

## Revisores do SBRC 2014

Adriana Centurion (USP)
Alberto Sampaio (UFC)
Alberto Schaeffer-Filho (UFRGS)
Aldelir Fernando Luiz (UFSC)
Aldri dos Santos (UFPR)
Alex Vieira (UFJF)
Alexandre Gonçalves (UFF)
Alextian Liberato (IFES)
Alfredo Goldman (USP)
Aline Viana (INRIA, França)
Allan Vidal (UFSCAR)
Ana Paula Couto da Silva (UFMG)
Anderson Nascimento (UNB)
André Cardoso (UFCE)
André Drummond (UNB)
André Oshiro (UFES)
André Soares (UFPI)
Andrey Brito (UFCG)
Anelise Munaretto (UTFPR)
Antônio Abelém (UFPA)
Antonio A. Ferreira Loureiro (UFMG)
Antonio Mury (LNCC)
Antonio Rocha (UFF)
Antonio Tadeu Azevedo Gomes (LNCC)
Ariel da Silva Dias (USP)
Arlindo Conceição (UNIFESP)
Arthur Callado (UFC)
Artur Ziviani (LNCC)
Atslands Rocha (UFC)
Avelino Zorzo (PUCRS)
Bruno Schulze (LNCC)
Bruno T. Kuehne (USP)
Carlos Alberto Vieira Campos (UNIRIO)
Carlos Corrêa (UFF)
Carlos Ferraz (UFPE)
Carlos Henrique Nicodemus (UFF)
Carlos Kamienski (UFABC)
Carlos Raniery P. dos Santos (UFRGS)
Carlos Senna (UNICAMP)
Célio V. Neves de Albuquerque (UFF)
Christian E. Rothenberg (UNICAMP)
Clarissa Marquezan (Duisburg-Essen University, Alemanha)
Claudio de Farias (UFRJ)
Claudio Monteiro (IFTO)
Cristiano Both (UFRGS)
Dalbert Mascarenhas (CEFET-RJ)
Daniel Batista (USP)
Daniel Chaves (UFPE)
Daniel Cordeiro (USP)
Daniel Fernandes Macedo (UFMG)
Daniel Figueiredo (UFRJ)
Daniel Menasche (UFRJ)
Dênio Mariz Sousa (IFPB)
Diego Passos (UFF)
Djamel Fawzi Hadj Sadok (UFPE)
Dorgival Guedes (UFMG)
Edmundo de Souza e Silva (UFRJ)
Eduardo Cerqueira (UFPA)
Eduardo Julio (UFJF)
Elias Duarte Jr. (UFPR)
Emanuel Rodrigues (UFC)
Emerson Ribeiro de Mello (IFSC)
Eros Spalla (IFES)
Everson Borges (IFES)
Ewerton Salvador (UFMG)
Fabio Costa (UFG)
Fabio Kon (USP)
Fábio Luciano Verdi (UFSCAR)
Fabíola Greve (UFBA)
Fatima Duarte-Figueiredo (PUC Minas)
Felipe Henriques (CEFET-RJ)
Fernanda G. Oliveira (UFF)
Fernando Augusto Teixeira (UFSJ)
Fernando Dettoni (UFSC)
Fernando Dotti (PUCRS)
Fernando Garcia (IFCE)
Fernando Pedone (University of Lugano, Suiça)
Flavia Delicato (UFRJ)

## Revisores do SBRC 2014

Flávio Assis Silva (UFBA)
Flavio Deus (UNB)
Flávio Rabello de Souza (UFES)
Francisco Brasileiro (UFCG)
Frederico Lopes (UFRN)
Gabriela Dias (IME)
Geraldo Robson Mateus (UFMG)
Giacomo Mc Evoy (LNCC)
Gilmar Vassoler (UFES)
Gustavo Figueiredo (UFBA)
Gustavo Pavani (UFABC)
Helio Waldman (UFABC)
Humberto Marques (PUC Minas)
Hylson Netto (UFSC)
Igor Dos Santos (UFRJ)
Igor Moraes (UFF)
Islene Garcia (UNICAMP)
Italo Cunha (UFMG)
Jacir Bordim (UNB)
Jair Silva (UFES)
Jauberth Abijaude (UESC)
Javier Baliosian (University of the Republic, Uruguai)
Jean-Marie Farines (UFSC)
Jesus Talavera Portocarrero (UFRJ)
Jó Ueyama (USP)
Joao Paulo da Costa (UNB)
Joaquim Celestino Júnior (UECE)
Joaquim Martins-Filho (UFPE)
Joilson Alves Jr. (UFPR)
José Augusto Suruagy Monteiro (UFPE)
José De Souza (UFC)
José Ferreira de Rezende (UFRJ)
José Viterbo (UFF)
Jose-Marcos Nogueira (UFMG)
Juliano Naves (UFF)
Julio Duarte (IME)
Júlio Estrella (USP)
Jussara Almeida (UFMG)
Karcius Assis (UFBA)
Keiko Fonseca (UTFPR)
Kelvin Dias (UFPE)
Kleber Cardoso (UFG)
Leandro Magnabosco (UFSC)
Leandro Resendo (IFES)
Leandro Sales Pinto (Politecnico di Milano, Itália)
Leandro Villas (UNICAMP)
Leobino Sampaio (UFBA)
Leonardo Oliveira (UFMG)
Leopoldo Mauricio (UERJ)
Liane M. Rockenbach Tarouco (UFRGS)
Linnyer Ruiz (UEM)
Lisandro Z. Granville (UFRGS)
Luci Pirmez (UFRJ)
Lucia Drummond (UFF)
Luciano Chaves (UNICAMP)
Luciano Paschoal Gaspary (UFRGS)
Luciene Demenicis (IME)
Lucio Agostinho (UFSCAR)
Luis Felipe de Moraes (UFRJ)
Luis Henrique Costa (UFRJ)
Luis Nakamura (USP)
Luis Rodrigues (INESC-ID, Portugal)
Luiz Fernando Bittencourt (UNICAMP)
Luiz Filipe Vieira (UFMG)
Luiz Henrique Correia (UFLA)
Marcel William Rocha da Silva (UFRRJ)
Marcelo D. de Amorim (UPMC, França)
Marcelo Pias (Globosense / Univ Cambridge, Reino Unido
Marcelo Rubinstein (UERJ)
Marcelo Santos (IFSudesteMG)
Marco Antonio Casanova (PUC-RIO)
Marco Spohn (UFFS)
Marcos Salvador (Lenovo)
Marcos Vieira (UFMG)
Marcus Sandri (UFSCAR)

## Revisores do SBRC 2014

Marinho Barcellos (UFRGS)
Markus Endler (PUC-RIO)
Matheus Bandini (LNCC)
Matheus Lehmann (UFRGS)
Mauro Fonseca (PUCPR)
Michael Stanton (RNP & UFF)
Michele Nogueira (UFPR)
Michelle Wangham (UNIVALI)
Miguel Elias Mitre Campista (UFRJ)
Miguel Franklin de Castro (UFC)
Nabor Mendonca (UNIFOR)
Natalia Castro Fernandes (UFF)
Nazareno Andrade (UFCG)
Nelson Fonseca (UNICAMP)
Nelson Rosa (UFPE)
Neumar Malheiros (UNICAMP)
Noemi Rodriguez (PUC-RIO)
Odorico Mendizabal (FURG)
Orlando Loques (UFF)
Osvaldo A. Carvalho Junior (USP & UNIFAL)
Paulo Aguiar (UFRJ)
Paulo André da Silva Gonçalves (UFPE)
Paulo Cunha (UFPE)
Paulo Eustáquio (USP)
Paulo Ferreira (INESC-ID, Portugal)
Paulo Mafra (UFSC)
Paulo Pires (UFRJ)
Paulo Sausen (UNIJUÍ)
Pedro Velloso (UFF)
Rafael Antonello (UFPE)
Rafael Costa (UFRJ)
Rafael Emerick Zape de Oliveira (UFES)
Rafael Lopes Gomes (UNICAMP)
Rafael Silva Guimaraes (IFES)
Rafael Sousa (UNB)
Rafael Vencioneck (UFES)
Rafaelli Coutinho (UFF)
Raimundo J. de Araujo Macedo (UFBA)
Raphael Gomes (IFG)
Raphael Guerra (UFF)
Raphael Machado (INMETRO)
Raquel Mini (PUC Minas)
Rasha Hasan (PUC-RS)
Raul Almeida Jr. (UFMG)
Reginaldo Nunes (IFES)
Reinaldo Gomes (UFCG)
Renato de Moraes (UNB)
Ricardo Carrano (UFF)
Ricardo Nogueira (USP)
Ricardo Rabêlo (UESPI)
Rick Lopes de Souza (UFSC)
Roberto Ierusalimschy (PUC-RIO)
Robson Albuquerque (UNB)
Robson Gomes de Melo (UFPR)
Rodolfo Ribeiro Gomes (UFES)
Rodolfo Villaça (UFES)
Rodrigo de Freitas (UEA)
Rodrigo de Souza Couto (UFRJ)
Ronaldo Ferreira (UFMS)
Ronaldo Salles (IME)
Rone Silva (UFSJ)
Rosa Leão (UFRJ)
Rossana Andrade (UFC)
Sand Correa (UFG)
Sergio Campello Oliveira (UFPE)
Sérgio Cardoso (IME)
Sérgio Carvalho (UFG)
Sergio Charpinel (UFES)
Silvana Rossetto (UFRJ)
Stenio Fernandes (UFPE)
Thais Braga (UFV)
Thais Vasconcelos Batista (UFRN)
Thamires Campos Luz (UFSCAR)
Thiago Genez (UNICAMP)
Tulio Ribeiro (UFSC)
Ubiratam De Paula (UFF)
Valdir Stumm Junior (IFC)
Valério Aymoré Martins (UNB)
Vinicius Mota (UFMG)

## Revisores do SBRC 2014

Vinod Rebello (UFF)
Wagner Meira Jr. (UFMG)
Weverton Cordeiro (IFPA)
William Giozza (UNB)

## Comitê de Avaliação do Salão de Ferramentas

Antonio Rocha (IC/UFF)
Bruno Schulze (LNCC)
Carlos Kamienski (UFABC)
Daniel Batista (IME - USP)
Daniel Cordeiro (IME - USP)
Divanilson Campelo (UFPE)
Eduardo Cerqueira (UFPA)
Fabio Costa (UFG)
Flavia Delicato (UFRJ)
Gerson Geraldo H. Cavalheiro (UFPel)
Gustavo Figueiredo (UFBA)
Humberto Marques (PUC Minas)
Igor Moraes (UFF)
Julian Monteiro (MAPS)
Jussara Almeida (DCC-UFMG)
Lisandro Zambenedetti Granville (UFRGS)
Luiz Fernando Bittencourt (UNICAMP)
Mario Dantas (UFSC)
Márcio Castro (UFRGS)
Nazareno Andrade (UFCG)
Nelson Rosa (UFPE)
Olga Goussevskaia (UFMG)
Paulo Pires (UFRJ)
Paulo André da Silva Gonçalves (UFPE)
Raphael Camargo (UFABC)
Ricardo R. Oliveira (UFOP)
Ronaldo Ferreira (UFMS)

## Revisores do Salão de Ferramentas

Jesus Talavera Portocarrero (UFRJ)
Matheus Bandini (LNCC)
Taniro Rodrigues (UFRN)

# Sumário

## Trilha Principal do SBRC 2014

# Sumário

# Sumário

# Sumário

# Sumário

## Sumário

# Sumário

# Sumário

# Sumário

# Sumário

# Sumário


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 1*

## *Virtualização e Computação em Nuvem*

# Alocação Ótima de Recursos para Infraestruturas Virtuais Confiáveis

**Gustavo A. S. Cavalcanti, Rafael R. Obelheiro e Guilherme P. Koslovski**

Programa de Pós-Graduação em Computação Aplicada (PPGCA)
Universidade do Estado de Santa Catarina (UDESC) – Campus Joinville
Email: {gasc@joinville.udesc.br, rro@joinville.udesc.br, guilhermek@joinville.udesc.br}

***Abstract.*** *Virtual infrastructures (VIs) consolidated the dynamic provisioning of computing and communication resources. Their infrastructure providers (InPs) face a challenge in choose the better approach to allocate and reserve these resources. Resource allocation is a complex problem that needs to satisfy different goals: users expect to run their applications on survivable VIs, while InPs aim to maximize profits, minimize costs and reduce substrate fragmentation. There is a dichotomy between minimizing substrate fragmentation, by co-locating VIs, and maximizing VI survivability, by sparsely allocating resources to decrease the impact of substrate failures. In this context, we propose a MILP model to allocate resources considering the joint coordination of fragmentation and survivability. Experimental results suggest that it is possible to enhance VI's survivability without significantly impacting substrate fragmentation.*

***Resumo.*** *Infraestruturas Virtuais (IVs) consolidaram o provisionamento dinâmico e provedores de infraestruturas (InPs) enfrentam um desafio na escolha da melhor abordagem para alocação e reserva de recursos computacionais e de rede. A alocação de recursos é um problema complexo que precisa satisfazer diferentes objetivos: usuários esperam executar suas aplicações em IVs confiáveis, enquanto InPs objetivam maximizar lucros, minimizar custos e reduzir a fragmentação do substrato físico. Há uma dicotomia entre minimizar a fragmentação do substrato, co-alocando IVs, e maximizar a confiabilidade de IVs, alocando recursos esparsos para diminuir o impacto de falhas no substrato. Nesse contexto, é proposto um modelo de programação inteira para alocar recursos considerando a coordenação conjunta de fragmentação e confiabilidade. Resultados experimentais sugerem que é possível aprimorar a confiabilidade de IVs sem impactar significativamente na fragmentação do substrato.*

## 1. Introdução

A computação em nuvem introduziu o provisionamento dinâmico de recursos e serviços virtualizados guiados pelos requisitos dos usuários. Esse conceito revolucionou o gerenciamento de TI (Tecnologia da Informação) por entregar computação, rede e armazenamento como serviços. Neste cenário, usuários não necessitam investimentos iniciais significativos para compor o parque de TI, podendo apenas alugar um conjunto de serviços ofertado por provedores de infraestruturas (InPs, *infrastructure providers*) seguindo um modelo *pay-as-you-go* [Mell and Grance 2009]. Atualmente, InPs podem oferecer infraestruturas completamente virtualizadas, possibilitando que usuários executem suas aplicações em conjuntos confinados e privados de recursos. Essas *infraestruturas*

*virtuais* (IVs) podem ser definidas como um conjunto de máquinas virtuais (MVs) interconectadas por recursos virtuais de rede [Anhalt et al. 2010].

Um desafio para os InPs é alocar recursos físicos para hospedar IVs. Usualmente, *frameworks* de gerenciamento consideram os objetivos dos InPs, orientados pelas suas perspectivas financeiras. InPs objetivam maximizar suas receitas alocando o maior número possível de IVs, usando para isso o mínimo de infraestrutura computacional que for viável. De fato, trabalhos anteriores tentaram minimizar a fragmentação do substrato físico (i.e. a quantidade de recursos físicos ativos) para diminuir os custos de provisionamento [Greenberg et al. 2008]. Este objetivo pode também beneficiar os usuários, uma vez que, normalmente, também é diminuída a latência da comunicação entre os recursos internos da IV [Koslovski et al. 2011]. No entanto, a fragmentação é minimizada pelo agrupamento dos recursos provisionados, o que pode prejudicar a disponibilidade das IVs e, portanto, colocar usuários e provedores em conflito: uma única falha física pode tornar indisponíveis ou inacessíveis várias IVs e as aplicações que elas hospedam.

Interrupções não planejadas no funcionamento de *data centers* (DCs) são comuns. Um estudo recente [Ponemon 2013] indica que empresas americanas sofreram, em uma média dos últimos 24 meses, 2,04 interrupções completas e 10,16 parciais de 107 minutos cada uma. Outro estudo [Ponemon 2011] estimou que cada minuto de interrupção no funcionamento de um DC custa em média US$ 5.600. É sabido que DCs de nuvens também são afetados por interrupções, as quais em alguns casos podem durar várias horas [McCarthy 2012, Raphael 2013]. Provedores populares de nuvens divulgam dados sobre a disponibilidade e/ou confiabilidade do substrato (e.g., 99,95% e 99,9%), mas, na prática, quando há ocorrência de indisponibilidade, os usuários somente recebem créditos para reexecutarem suas aplicações. Da perspectiva do usuário, uma única falha pode comprometer toda a execução de uma aplicação. Para evitar essa situação, algumas aplicações são tolerantes a faltas, mas isso normalmente afeta seu tempo de execução e dificulta seu desenvolvimento e manutenção. Sobretudo, uma falha pode resultar em uma violação de SLA (*Service Level Agreement*). InPs estão cientes deste fato e podem mitigar o impacto em algumas aplicações levando em consideração falhas de recursos (ou requisitos de disponibilidade) durante a alocação no substrato [Bodík et al. 2012]. Geralmente, isso requer a alocação em recursos físicos esparsos [Rahman et al. 2010, Barla et al. 2012, Koslovski et al. 2010, Yu et al. 2010, Xu et al. 2012], o que aumenta o custo de provisionamento. De fato, conforme o InP adiciona redundâncias (servidores, roteadores, *switches* e enlaces de *backup*), aumentam tanto o consumo de energia quanto a complexidade de gerenciamento. Apesar dos recursos de *backup* poderem ser compartilhados entre diversos usuários (até um certo ponto) [Yeow et al. 2010], os custos associados com confiabilidade são, em última instância, transferidos para os usuários finais.

Nesse contexto, este artigo propõe um novo modelo de alocação ótima para fornecer IVs confiáveis. O principal objetivo desse modelo é selecionar os recursos físicos que devem hospedar recursos virtuais de forma a minimizar tanto a fragmentação do substrato quanto o número de recursos virtuais afetados por uma falha no substrato. Para alcançar esse objetivo é apresentada uma formulação de programação linear inteira mista que incorpora as perspectivas dos InPs (redução da fragmentação) e as dos usuários (aumento da confiabilidade). Resultados experimentais discutem o *trade-off* entre fragmentação e confiabilidade. São três as principais contribuições deste trabalho: (i) um modelo de alocação

ótima que considera tanto a fragmentação do substrato físico quanto a confiabilidade de IVs; (ii) a introdução de métricas de falhas locais e globais, que fornecem medidas de confiabilidade da perspectiva dos usuários e dos InPs, respectivamente; e (iii) resultados experimentais que fornecem uma linha de base ótima do *trade-off* entre fragmentação e confiabilidade e servem como referência para análises futuras.

Este artigo está organizado da seguinte forma: a Seção 2 introduz conceitos de base, enquanto a Seção 3 apresenta a formulação do modelo de programação linear inteira mista. A Seção 4 discute os resultados experimentais. A Seção 5 revisa os trabalhos relacionados e a Seção 6 conclui o artigo e indica as perspectivas futuras.

## 2. Conceitos e definições

O modelo busca a alocação ótima de recursos físicos para hospedar IVs confiáveis. Este problema envolve conjuntos de variáveis, parâmetros e funções para representar os objetivos dos usuários e dos InPs. A Fig. 1 ilustra os principais conceitos. Inicialmente, a Fig. 1(a) define $s$ como um *substrato físico* composto por máquinas físicas (MFs) $m^p$ e *switches*[1] $s^p$, interconectados por enlaces unidirecionais $e^p$. Todos os recursos físicos são agrupados em *racks* ($d^r$) e conectados em circuitos elétricos ($d^c$). Em seguida, a Fig. 1(b) ilustra duas *infraestruturas virtuais*, $v_a$ e $v_b$, ambas compostas por recursos virtuais: máquinas ($m^a$ e $m^b$), *switches* ($s^a$), e enlaces ($e^a$ e $e^b$). Tomando as Figs. 1(a) e 1(b) como grafos, os pesos das arestas denotam as capacidades de enlace no substrato e as demandas por largura de banda em uma IV. Similarmente, as capacidades/demandas de CPU e memória são associadas aos vértices dos grafos. O objetivo é que as demandas por recursos de uma IV sejam reservadas pelo InP no substrato físico, como exemplificado pelas Figs. 1(c) e 1(d). Particularmente, a Fig. 1(c) mostra que um enlace virtual (e.g., $e_1^b$) pode ser alocado[2] em um caminho físico com múltiplos saltos ($e_3^p, e_2^p$).

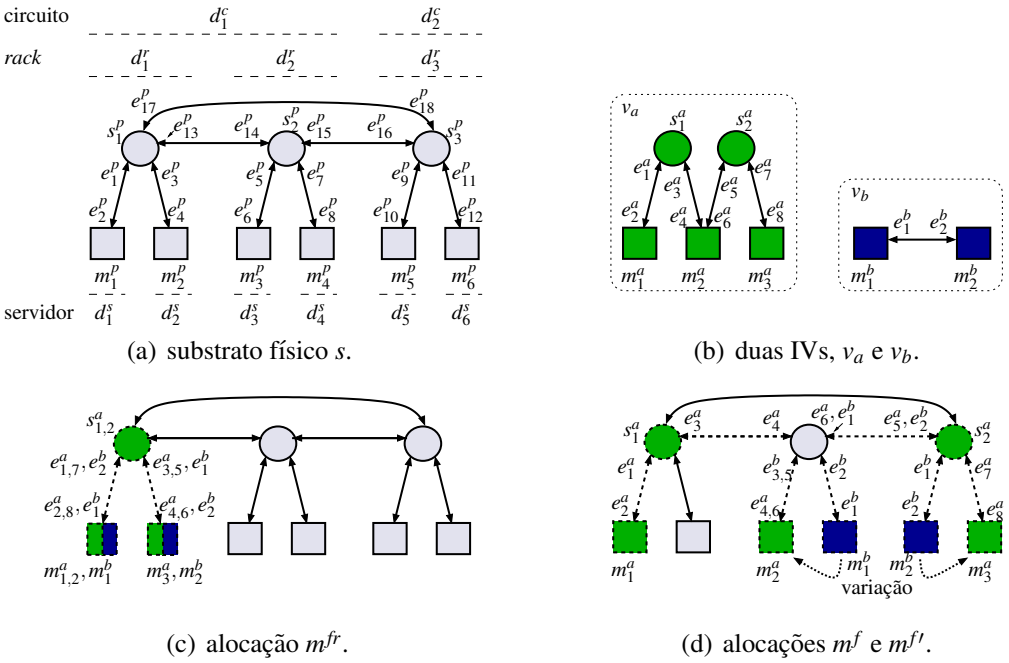
**Tabela 1. Métricas de fragmentação e falha para a Fig. 1.**

| | Frag. (%) | | Falha (%) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | circuito ($c$) | | *rack* ($r$) | | servidor ($s$) | | enlace ($e$) | |
| $m$ | FrN | FrE | FLN($v_a/v_b$) | FGN | FLN($v_a/v_b$) | FGN | FLN($v_a/v_b$) | FGN | FLE($v_a/v_b$) | FGE |
| $m^{fr}$ | 33 | 22 | 100/100 | 100 | 100/100 | 100 | 67/50 | 60 | 50/50 | 50 |
| $m^f$ | 78 | 78 | 60/50 | 57 | 40/50 | 43 | 33/50 | 20 | 25/50 | 33 |
| $m^{f\prime}$ | 56 | 56 | 60/50 | 57 | 40/50 | 43 | 33/50 | 40 | 25/50 | 50 |

As Figs. 1(c) e 1(d) ilustram como as IVs $v_a$ e $v_b$ podem ser alocadas no substrato $s$ de acordo com diferentes *trade-offs* entre fragmentação e confiabilidade. A alocação $m^{fr}$ reduz *a fragmentação do substrato físico*, definida como a razão entre o número de recursos físicos ativos (i.e., aqueles que hospedam recursos virtuais, de acordo com as alocações efetuadas) e o total de recursos físicos. São consideradas duas métricas de fragmentação: fragmentação de nós (FrN) e fragmentação de enlaces (FrE). A fragmentação de nós é dada pela Eq. 1, onde $|\mathcal{N}_a^p|$ é o número de nós ativos (tanto máquinas físicas $m^p$ quanto *switches* $s^p$) e $|\mathcal{N}_t^p|$ é o total de nós. Similarmente, a fragmentação de enlaces é dada pela Eq. 2, onde $|\mathcal{E}_a^p|$ é o número de enlaces ativos e $|\mathcal{E}_t^p|$ é o total de enlaces. Diminuindo a fragmentação (minimizando a soma de (1) e (2)), os InPs podem reduzir os custos desativando os recursos ociosos. Por exemplo, na alocação $m^{fr}$ (Fig. 1(c)),

[1] Os termos *switch* e *roteador* são usados indistintamente no restante do artigo.

[2] Seguindo a literatura, os termos *alocação* e *mapeamento* possuem o mesmo significado neste artigo.




(a) substrato físico $s$. (b) duas IVs, $v_a$ e $v_b$.

(c) alocação $m^{fr}$. (d) alocações $m^f$ e $m^{f\prime}$.

**Figura 1. Um exemplo de mapeamento. O substrato $s$ (Fig. 1(a)) pode hospedar as IVs $v_a$ e $v_b$ (Fig. 1(b)), como mostram as alocações nas Figs. 1(c) e 1(d).**

somente um *rack* ($d_1^r$) está hospedando ambas as IVs $v_a$ e $v_b$, e portanto os *racks* $d_2^r$ e $d_3^r$ estão ociosos. A Tabela 1 mostra os valores de fragmentação para as alocações nas Figs. 1(c) e 1(d). Para a alocação $m^{fr}$ observa-se 33% de fragmentação de nós e 22% de fragmentação de enlaces, enquanto a alocação $m^f$ – que favorece a confiabilidade – tem 78% de fragmentação de nós/enlaces (2,3 e 3,5 vezes os valores para $m^{fr}$). No entanto, uma pequena diferença na alocação $m^{f\prime}$ (a variação na Fig. 1(d)) resulta em um menor aumento nas fragmentações de nós e de enlaces (1,7 vezes maior que em $m^{fr}$).

$$\mathrm{FrN} = \frac{|\mathcal{N}_a^p|}{|\mathcal{N}_t^p|} \qquad (1) \qquad\qquad \mathrm{FrE} = \frac{|\mathcal{E}_a^p|}{|\mathcal{E}_t^p|} \qquad (2)$$

Os recursos físicos estão sujeitos a falhas de parada (*crash*); uma falha pode afetar um único recurso (e.g., um servidor) ou um grupo de recursos (e.g., um *rack* ou um DC inteiro). Para quantificar o impacto destas falhas, [Bodík et al. 2012] definiu um *domínio de faltas* (DF) como um conjunto de recursos físicos que compartilham de um único ponto de falha. O mesmo nó pode pertencer a múltiplos DFs. Em um ambiente de nuvem, uma IV pode ser distribuída sobre diversos recursos físicos pertencentes a diferentes DFs. Para quantificar o impacto de falhas no substrato, definimos uma *métrica de falha* como a fração de recursos virtuais (nós e enlaces) afetada pelo pior caso de falha em um recurso físico (i.e., a falha física que mais afeta recursos virtuais). Essa métrica é calculada para todos os *tipos de domínios de faltas* (ou simplesmente *tipos*) definidos pelo InP. Por exemplo, o substrato físico $s$ (Fig. 1(a)) tem quatro tipos: *servidor*, *rack*, *circuito elétrico* e *enlace*. Cada DF é denotado por $d$, e cada tipo é denotado por $t$.

Esta métrica é exemplificada analisando o impacto de uma falha em um *rack* na Fig. 1. Neste cenário, cada *rack* $d_n^r$ contém duas MFs e um *switch*. Se um *rack* falha, todos os recursos virtuais hospedados nas máquinas ou *switch* deste *rack* são perdidos. Considerando três DFs para o tipo *rack* ($d_1^r$, $d_2^r$ e $d_3^r$), o pior caso na alocação $m^{fr}$ é a falha

do *rack* $d_1^r$, que induz a perda de todos os recursos virtuais. Considerando as alocações $m^f$ e $m^{f\prime}$, o pior caso é dado pela falha de $d_3^r$, a qual afeta 40% de $v_a$ e 50% de $v_b$, como resumido na Tabela 1. *Nesse contexto, é observado que usuários e provedores devem minimizar as métricas de falhas para consequentemente alcançar uma confiabilidade ótima, independentemente dos objetivos financeiros.*

Embora o *objetivo geral de confiabilidade* seja minimizar as métricas de falha de uma alocação, usuários e InPs têm diferentes perspectivas. Um usuário tipicamente deseja maximizar a confiabilidade de suas próprias IVs sem se preocupar com a confiabilidade das outras IVs, ao passo que um InP tentará proporcionar uma boa confiabilidade global (a qual pode ser sub-ótima para certos usuários). Essas perspectivas distintas são traduzidas em métricas de *falha local* e de *falha global*, respectivamente. A métrica de falha local (FL) considera as métricas de falha de nós e de enlaces ($\text{FLN}_{v,t}$ e $\text{FLE}_{v,t}$, respectivamente). Para uma dada IV $v$ e um tipo de DF $t$, a métrica de falha local de nós ($\text{FLN}_{v,t}$) é dada pela razão de nós falhos ($|\mathcal{N}_{v,t}^f|$) e o total de nós ($|\mathcal{N}_{v,t}|$) de $v$ alocados em $t$, como mostrada na Eq. 3; uma definição análoga para $\text{FLE}_{v,t}$ é dada pela Eq. 4. Ao diminuir $\text{FLN}_{v,t}$ e $\text{FLE}_{v,t}$, a confiabilidade de $v$ aumenta. A coluna *Falha* na Tabela 1 ilustra essa métrica para as alocações $m^{fr}$, $m^f$ e $m^{f\prime}$. Em $m^{fr}$, $v_a$ tem uma alta FLN, visto que todos os nós são perdidos quando falha um circuito ou *rack*. Mesmo se um servidor físico falhar, somente um único nó virtual sobrevive. Nas alocações $m^f$ e $m^{f\prime}$, as $\text{FLN}_{v,t}$ para $v_a$ e $v_b$ diminuem para todos os tipos exceto para falhas de servidor afetando $v_b$.

$$\text{FLN}_{v,t} = \frac{|\mathcal{N}_{v,t}^f|}{|\mathcal{N}_{v,t}|} \quad (3) \qquad \text{FLE}_{v,t} = \frac{|\mathcal{E}_{v,t}^f|}{|\mathcal{E}_{v,t}|} \quad (4) \qquad \text{FGN}_t = \frac{|\mathcal{N}_t^f|}{|\mathcal{N}_t|} \quad (5) \qquad \text{FGE}_t = \frac{|\mathcal{E}_t^f|}{|\mathcal{E}_t|} \quad (6)$$

A métrica de falha global (FG) segue o mesmo princípio da FL, com a diferença que todas as IVs hospedadas são indistintas. A Eq. 5 define a métrica de falha global de nós como $\text{FGN}_t$, a razão entre os recursos falhos ($|\mathcal{N}_t^f|$) e o total de recursos ($|\mathcal{N}_t|$) alocados no tipo de DF $t$; a Eq. 6 providencia uma definição análoga para a métrica de falha global de enlaces ($\text{FGE}_t$). Assim como as métricas de falha local, os InPs objetivam minimizar $\text{FGN}_t$ e $\text{FGE}_t$ para fornecer um serviço com confiabilidade. A alocação $m^{fr}$ ilustra um cenário no qual $\text{FGN} = 100\%$ quando ocorre uma falha de circuito ou *rack*. Por outro lado, ela oferece um valor aceitável para falhas de servidor (ao menos 60% dos nós são capazes de sobreviver no pior caso). As métricas de falha global para $m^f$ e $m^{f\prime}$ indicam um maior nível de confiabilidade para falhas de circuito e de *rack*. Considerando o tipo servidor, para $m^f$, a $\text{FGN}_s$ é $1/3$ da $\text{FGN}_s$ para $m^{fr}$, e a migração de $v_b$ em $m^{f\prime}$ traz a $\text{FGN}_s$ para $2/3$ da de $m^{fr}$.

Encontrar um *trade-off* ótimo entre fragmentação e confiabilidade não é uma tarefa trivial. Comparada com $m^f$, a alocação $m^{f\prime}$ na Fig. 1(d) diminui a fragmentação de nós (FrN) em quase 29%, sem impactar nas métricas global e local para os tipos circuito e *rack*. De fato, os valores dessas métricas para $m^f$ e $m^{f\prime}$ são idênticos. Em contrapartida, para o tipo servidor, $m^f$ é significativamente mais confiável do que $m^{f\prime}$, uma vez que metade do número de servidores falham no pior caso. Além do mais, em $m^f$, qualquer falha isolada afetará no máximo uma IV, enquanto em $m^{f\prime}$ ambas as IVs podem ser afetadas.

## 3. Modelo Proposto

Esta seção introduz, em quatro partes, um modelo de programação linear inteira mista que proporciona confiabilidade de IVs sem aumentar a fragmentação no substrato físico.

### 3.1. Alocação de grafos

Antes dos modelos de fragmentação e confiabilidade serem apresentados, será introduzida a formulação tradicional para a alocação de recursos de IV. Esse problema tenta resolver a alocação de recursos em um substrato físico (ou grafo de uma infraestrutura física) para hospedar recursos virtuais (um grafo de IV), o qual é sabidamente um problema NP-difícil [Chowdhury et al. 2009]. A Tabela 2 sumariza a notação usada na formulação. O substrato físico é modelado como um conjunto de máquinas ($\mathcal{M}^p$) que se comunicam através de *switches* ($\mathcal{S}^p$). Um conjunto de nós $\mathcal{N}^p$ é usado para agrupar as máquinas e *switches* físicos. Um conjunto de enlaces $\mathcal{E}^p$ define todas as interconexões físicas; um enlace $e^p$ entre dois nós $n_1^p$ e $n_2^p$ é denotado pela tupla $(n_1^p, n_2^p)$. Similarmente, uma IV $v$ é composta por conjuntos de máquinas ($\mathcal{M}^v$), *switches* ($\mathcal{S}^v$) e enlaces ($\mathcal{E}^v$) virtuais. O conjunto $\mathcal{N}^v$ denota a união de $\mathcal{M}^v$ e $\mathcal{S}^v$. Uma máquina $m$ (de $\mathcal{M}^v$ ou $\mathcal{M}^p$) tem capacidades $\text{CM}_{m,r}$, onde $r \in \mathcal{R}^m$ indexa atributos individuais (e.g., CPU, memória, armazenamento) dessa máquina. Para nós físicos, CM representa a capacidade total disponível, e para nós virtuais indica a demanda de recurso. Do mesmo modo, um *switch* $s$ (de $\mathcal{S}^v$ ou $\mathcal{S}^p$) tem capacidades $\text{CS}_{s,r}$, onde $r$ é um recurso pertencente a $\mathcal{R}^s$. Finalmente, um enlace $e$ entre dois nós tem capacidades $\text{CE}_{e,r}$, onde $r \in \mathcal{R}^e$ é um de seus atributos (e.g, latência, largura de banda). Escolheu-se modelar capacidade de CPU para as máquinas, capacidade de memória para os *switches* e largura de banda para os enlaces. Como discutido, o modelo indica a presença de conjuntos para modelar esses atributos, podendo ser facilmente estendido para cobrir mais atributos funcionais e não funcionais.

**Tabela 2. Notação para o modelo de alocação de grafo.**

| Notação | | Descrição |
|---|---|---|
| Conjuntos | $m^v \in \mathcal{M}^v$ | uma máquina virtual (MV) |
| | $m^p \in \mathcal{M}^p$ | uma máquina física (MF) |
| | $r \in \mathcal{R}^m$ | uma capacidade de máquina (e.g., CPU ou memória) |
| | $s^v \in \mathcal{S}^v$ | um *switch* virtual |
| | $s^p \in \mathcal{S}^p$ | um *switch* físico |
| | $r \in \mathcal{R}^s$ | uma capacidade de *switch* (e.g., memória) |
| | $n^v \in \mathcal{N}^v = \mathcal{M}^v \cup \mathcal{S}^v$ | um nó virtual (máquina ou *switch*) |
| | $n^p \in \mathcal{N}^p = \mathcal{M}^p \cup \mathcal{S}^p$ | um nó físico (máquina ou *switch*) |
| | $(n^v, n^p) \in \mathcal{A}^n \subseteq \mathcal{N}^v \times \mathcal{N}^p$ | um nó virtual pré-alocado em um nó físico |
| | $e^v := (n_1^v, n_2^v) \in \mathcal{E}^v \subseteq \mathcal{N}^v \times \mathcal{N}^v$ | um enlace virtual unidirecional |
| | $e^p := (n_1^p, n_2^p) \in \mathcal{E}^p \subseteq \mathcal{N}^p \times \mathcal{N}^p$ | um enlace físico unidirecional |
| | $r \in \mathcal{R}^e$ | uma capacidade de enlace (e.g., largura de banda) |
| | $(e^v, e^p) \in \mathcal{A}^e \subseteq \mathcal{E}^v \times \mathcal{E}^p$ | um enlace virtual pré-alocado em um físico |
| Parâmetros | $\text{CM}\{m \in \mathcal{M}^v \cup \mathcal{M}^p, r \in \mathcal{R}^m\}, \in \mathbb{R}^+$ | as capacidades de nós |
| | $\text{CS}\{s \in \mathcal{S}^v \cup \mathcal{S}^p, r \in \mathcal{R}^s\}, \in \mathbb{R}^+$ | as capacidades de *switch* |
| | $\text{CE}\{e \in \mathcal{E}^v \cup \mathcal{E}^p, r \in \mathcal{R}^e\}, \in \mathbb{R}^+$ | as capacidades de enlace |
| Variáveis | $mn\{n^v \in \mathcal{N}^v, n^p \in \mathcal{N}^p\}, \in \{0,1\}$ | o mapeamento entre os nós virtuais e físicos |
| | $me\{e^v \in \mathcal{E}^v, e^p \in \mathcal{E}^p\}, \in \{0,1\}$ | o mapeamento entre os enlaces virtuais e físicos |

O objetivo da alocação de IV é encontrar um mapeamento combinado para alocar nós virtuais em nós físicos e enlaces virtuais em caminhos físicos. Um nó virtual $n^v$ é

mapeado em um único nó físico $n^p$, o qual pode hospedar múltiplos nós virtuais ($mn\colon \mathcal{N}^v \uparrow \mathcal{N}^p$); se $n^p$ hospeda $n^v$, então $mn_{n^v,n^p} = 1$, senão 0. Um enlace virtual $e^v$ é mapeado em um caminho físico composto por um ou mais enlaces. Um enlace físico $e^p$ pode hospedar múltiplos enlaces virtuais ($me\colon \mathcal{E}^v \uparrow \mathcal{E}^p$); se $e^p$ hospeda $e^v$, então $me_{e^v,e^p} = 1$, senão 0.

Na alocação *online* de IVs, usuários podem submeter pedidos de IVs em momentos diferentes. Nesse cenário, quando uma IV $v_a$ já está alocada e outra IV $v_b$ é solicitada, o InP deve escolher uma de duas abordagens: (*i*) realocar todas as IVs, o que pode melhorar as métricas de fragmentação do substrato e confiabilidade mas aumenta o tempo de alocação e introduz problemas de migração de IVs; ou (*ii*) manter os nós e enlaces pré-alocados em seus respectivos recursos físicos, o que evita descumprir SLAs existentes. Optou-se pela estratégia (*ii*), atualmente adotada por provedores populares de nuvens, o que envolve o gerenciamento de dois novos conjuntos: $\mathcal{A}^n$ para um nó virtual $n^v$ pré-alocado em um nó físico $n^p$, denotado pela tupla $(n^v, n^p)$; e $\mathcal{A}^e$ para um enlace $e^v$ pré-alocado em um enlace físico $e^p$, denotado pela tupla $(e^v, e^p)$. As restrições para o modelo de alocação são especificadas pelas Eqs. 7–15.

$$\sum_{m^v \in \mathcal{M}^v} \mathrm{CM}_{m^v,r} \cdot mn_{m^v,m^p} \leq \mathrm{CM}_{m^p,r}, \quad \forall r \in \mathcal{R}^m, \forall m^p \in \mathcal{M}^p \tag{7}$$

$$\sum_{s^v \in \mathcal{S}^v} \mathrm{CS}_{s^v,r} \cdot mn_{s^v,s^p} \leq \mathrm{CS}_{s^p,r}, \quad \forall r \in \mathcal{R}^s, \forall s^p \in \mathcal{S}^p \tag{8}$$

$$\sum_{e^v \in \mathcal{E}^v} \mathrm{CE}_{e^v,r} \cdot me_{e^v,e^p} \leq \mathrm{CE}_{e^p,r}, \quad \forall r \in \mathcal{R}^e, \forall e^p \in \mathcal{E}^p \tag{9}$$

$$\sum_{n^{p\prime} \in \mathcal{N}^{p\prime}} mn_{n^v,n^{p\prime}} = 1, \quad \forall n^v \in \mathcal{N}^v \tag{10}$$

$$\sum_{n^{p\prime\prime} \in \mathcal{N}^{p\prime\prime}} mn_{n^v,n^{p\prime\prime}} = 0, \quad \forall n^v \in \mathcal{N}^v \tag{11}$$

$$\sum_{(n_2^{p\prime\prime\prime}) \in \mathcal{N}^{p\prime\prime\prime}} me_{n_1^v,n_2^v,n_1^p,n_2^{p\prime\prime\prime}} + \sum_{(n_2^{p\prime\prime\prime\prime}) \in \mathcal{N}^{p\prime\prime\prime\prime}} me_{n_1^v,n_2^v,n_2^{p\prime\prime\prime\prime},n_1^p} = mn_{n_1^v,n_1^p} + mn_{n_2^v,n_1^p}, \quad \forall (n_1^v, n_2^v) \in \mathcal{E}^v, \forall n_1^p \in \mathcal{N}^p \tag{12}$$

$$mn_{n_1^v,n^p} + mn_{n_2^v,n^p} \leq 1, \quad \forall n^p \in \mathcal{N}^p, \forall (n_1^v, n_2^v) \in \mathcal{E}^v \tag{13}$$

$$mn_{n^v,n^p} = 1, \quad \forall n^v \in \mathcal{N}^v, \forall n^p \in \mathcal{N}^p\colon \quad (n^v, n^p) \in \mathcal{A}^n \tag{14}$$

$$me_{e^v,e^p} = 1, \quad \forall e^v \in \mathcal{E}^v, \forall e^p \in \mathcal{E}^p\colon \quad (e^v, e^p) \in \mathcal{A}^e \tag{15}$$

Inicialmente, a Eq. 7 garante que uma MF não hospedará mais MVs do que as suas capacidades permitem; restrições similares são aplicadas aos *switches* (Eq. 8) e enlaces (Eq. 9). As Eqs. 10 e 11 garantem que um nó virtual será alocado em somente um nó físico e esses nós serão do mesmo tipo (i.e., MVs devem ser hospedadas em MFs, não em *switches*, e vice-versa). Para satisfazer essas condições, são definidos subconjuntos de $\mathcal{N}^p$, na Eq. 10 como $\mathcal{N}^{p\prime} := \forall n^p \in \mathcal{N}^p\colon \ (n^v \in \mathcal{M}^v \wedge n^p \in \mathcal{M}^p) \vee (n^v \in \mathcal{S}^v \wedge n^p \in \mathcal{S}^p)$, e na Eq. 11 como $\mathcal{N}^{p\prime\prime} := \forall n^p \in \mathcal{N}^p\colon \ (n^v \in \mathcal{M}^v \wedge n^p \in \mathcal{S}^p) \vee (n^v \in \mathcal{S}^v \wedge n^p \in \mathcal{M}^p)$. Portanto, se $n^v$ é uma MV, então $\mathcal{N}^{p\prime}$ conterá MFs e $\mathcal{N}^{p\prime\prime}$ conterá *switches* (e vice-versa). A Eq. 12, baseada na formulação proposta por [Chowdhury et al. 2009], garante que um enlace virtual $e^v$ entre quaisquer dois nós virtuais, $n_1^v$ e $n_2^v$, será alocado em um ou mais enlaces físicos que formam um caminho conexo entre os nós físicos $n_1^p$ e $n_2^p$ que hospedam $n_1^v$ e $n_2^v$. Para

satisfazer essas condições, são definidos subconjuntos de $\mathcal{N}^p$ como $\mathcal{N}^{p'''} := \forall n_2^p \in \mathcal{N}^p$: $(n_1^p, n_2^p) \in \mathcal{E}^p$, na primeira soma, e como $\mathcal{N}^{p''''} := \forall n_2^p \in \mathcal{N}^p$: $(n_2^p, n_1^p) \in \mathcal{E}^p$, na segunda soma. Em ambos os casos, $\mathcal{N}^{p'''}$ e $\mathcal{N}^{p''''}$ terão somente nós físicos $n_2^p$ conectados com $n_1^p$. A Eq. 13 determina que dois nós virtuais diretamente conectados não podem estar alocados em um mesmo nó físico. Para garantir que IVs pré-alocadas permanecerão hospedadas em seus recursos físicos durante sua existência, as Eqs. 14 e 15 garantem que nós e enlaces virtuais permanecerão nos mesmos nós e caminhos físicos, respectivamente. Por fim, se todas essas restrições forem satisfeitas, a IV requisitada é alocada, caso contrário, se pelo menos uma dessas restrições não for satisfeita, a IV requisitada não é alocada.

### 3.2. Fragmentação

A Tabela 3 sumariza as notações usadas no modelo de fragmentação. Neste modelo, a fragmentação é considerada no nível de nós e enlaces físicos, como adotado em outros trabalhos [Greenberg et al. 2008, Koslovski et al. 2011]. Como discutido na Seção 2, se uma alocação $a$ requer menos nós/enlaces físicos para hospedar uma dada IV do que uma alocação $b$, então $a$ induz uma menor fragmentação no substrato do que $b$. A fragmentação de nós é denotada por $frn$, e a fragmentação de enlaces por $fre$.

**Tabela 3. Notação para o modelo de fragmentação.**

| Notação | | Descrição |
|---|---|---|
| Variáv. | $frn\{n^p \in \mathcal{N}^p\}, \in \{0,1\}$ | a fragmentação dos *nós* físicos |
| | $fre\{e^p \in \mathcal{E}^p\}, \in \{0,1\}$ | a fragmentação dos *enlaces* físicos |

As Eqs. 16 e 17 especificam as restrições de fragmentação. A Eq. 16 verifica se um nó físico hospeda ao menos um nó virtual. O primeiro membro define a razão entre o número de nós virtuais alocados e o total de nós virtuais. O segundo controla o estado de fragmentação do nó físico: em uso, se $frn = 1$, ou ocioso, se 0. Deve-se ressaltar que, aqui, o número de nós virtuais hospedados é irrelevante, o que importa é se um recurso físico está *ocupado* ou *em uso*. Analogamente, a Eq. 17 verifica o estado dos enlaces físicos (i.e., se hospedam ao menos um enlace virtual), seguindo a mesma abordagem.

$$\frac{\sum_{n^v \in \mathcal{N}^v} mn_{n^v,n^p}}{|\mathcal{N}^v|} \leq \lceil frn_{n^p} \rceil, \quad \forall n^p \in \mathcal{N}^p \quad (16) \qquad \frac{\sum_{e^v \in \mathcal{E}^v} me_{e^v,e^p}}{|\mathcal{E}^v|} \leq \lceil fre_{e^p} \rceil, \quad \forall e^p \in \mathcal{E}^p \quad (17)$$

As Eqs. 18 e 19 definem as métricas de fragmentação usadas na função objetivo (Eq. 28, discutida na Seção 3.4). Ambas as métricas devem ser minimizadas para diminuir o número de recursos físicos em uso. Elas denotam a razão entre os recursos físicos em uso e o número total de recursos (Eq. 18 para nós e Eq. 19 para enlaces).

$$\text{FrN} := \frac{\sum_{n^p \in \mathcal{N}^p} frn_{n^p}}{|\mathcal{N}^p|} \quad (18) \qquad \text{FrE} := \frac{\sum_{e^p \in \mathcal{E}^p} fre_{e^p}}{|\mathcal{E}^p|} \quad (19)$$

### 3.3. Confiabilidade

A Tabela 4 sumariza as notações usadas no modelo de confiabilidade. Todos os nós virtuais de uma dada IV têm um identificador único $v$, para serem distinguíveis das outras IVs (i.e., $\text{REQ}: \mathcal{N}^v \uparrow \mathcal{V}$; se $n^v \in$ IV $v$, então $\text{REQ}_{n^v,v} = 1$, senão 0). Como discutido na

Seção 2, as *métricas de falha local* medem o impacto de uma falha física em uma única IV, e as *métricas de falha global* medem o impacto em todos os nós e enlaces, indiferente a qual IV eles pertencem. Para alcançar um valor ótimo para essas métricas, os recursos virtuais devem ser distribuídos pelos domínios de faltas, levando em conta todos os tipos de DF (i.e., circuitos, *racks*, servidores, enlaces). No entanto, distribuir por DFs os nós e enlaces de *uma* IV (métrica de falha local) não é o mesmo que distribuir por DFs *todos* os nós e enlaces (métrica de falha global). Por exemplo, a primeira métrica pode distribuir uma IV $v_a$ em dois *racks* e uma IV $v_b$ nos *mesmos* dois *racks*, mesmo se há um terceiro desocupado; o pior caso de falha de *rack* afeta uma fração de cada IV, mas *sempre* afeta ambas. A segunda pode distribuir todos os nós nos três *racks*, ocupando todos eles, mas deixar os nós de $v_a$ concentrados em um e os de $v_b$ em outro; o pior caso de falha de *rack* afetaria uma porção menor de nós do que antes, mas seria *fatal* para uma IV.

**Tabela 4. Notação para o modelo de confiabilidade.**

| Notação | | Descrição |
|---|---|---|
| Conjuntos | $v \in \mathcal{V}$ | um identificador de IV |
| | $d \in \mathcal{D}^n$ | um DF para *nó* FD |
| | $t \in \mathcal{T}^n$ | um tipo de DF para *nós* |
| | $d \in \mathcal{D}^e$ | um DF para *enlace* |
| | $t \in \mathcal{T}^e$ | um tipo de DF para *enlace* |
| Parâmetros | $\text{REQ}\{n^v \in \mathcal{N}^v, v \in \mathcal{V}\}, \in \{0,1\}$ | o mapeamento de nós virtuais e IVs |
| | $\text{MND}\{n^p \in \mathcal{N}^p, d \in \mathcal{D}^n\}, \in \{0,1\}$ | o mapeamento de *nós* físicos e DFs |
| | $\text{MDTN}\{d \in \mathcal{D}^n, t \in \mathcal{T}^n\}, \in \{0,1\}$ | o mapeamento de DFs e tipos para *nós* |
| | $\text{MED}\{e^p \in \mathcal{E}^p, d \in \mathcal{D}^e\}, \in \{0,1\}$ | o mapeamento de *enlaces* físicos e DFs |
| | $\text{MDTE}\{d \in \mathcal{D}^e, t \in \mathcal{T}^e\}, \in \{0,1\}$ | o mapeamento de DFs e tipos para *enlaces* |
| | $\text{EFLN}\{v \in \mathcal{V}, t \in \mathcal{T}^n\}, \in \mathbb{Z}^+$ | os *fln* máximos estimados |
| | $\text{EFGN}\{t \in \mathcal{T}^n\}, \in \mathbb{Z}^+$ | os *fgn* máximos estimados |
| | $\text{EFLE}\{v \in \mathcal{V}, t \in \mathcal{T}^e\}, \in \mathbb{Z}^+$ | os *fle* máximos estimados |
| | $\text{EFGE}\{t \in \mathcal{T}^e\}, \in \mathbb{Z}^+$ | os *fge* máximos estimados |
| Variáveis | $\mathit{fln}\{v \in \mathcal{V}, t \in \mathcal{T}^n\}, \in \mathbb{Z}^+$ | o núm. máx. de *nós de uma IV* em um DF de um tipo |
| | $\mathit{fgn}\{t \in \mathcal{T}^n\}, \in \mathbb{Z}^+$ | o núm. máx. de *todos os nós* em um DF de um tipo |
| | $\mathit{fle}\{v \in \mathcal{V}, t \in \mathcal{T}^e\}, \in \mathbb{Z}^+$ | o núm. máx. de *enlaces de uma IV* em um DF de um tipo |
| | $\mathit{fge}\{t \in \mathcal{T}^e\}, \in \mathbb{Z}^+$ | o núm. máx. de *todos os enlaces* em um DF de um tipo |

Um nó ou enlace físico pertence a um ou mais tipos de domínios de falta. Por exemplo, uma máquina $m^p$ é um *servidor* dentro de um *rack*, conectado a um *circuito elétrico*, então $m^p$ pertence a três tipos. Adicionalmente, um *switch* $s^p$ está dentro de um *rack* e também conectado a um *circuito*, mas não é hospedado em um *servidor* (consequentemente, $s^p$ pertence a dois tipos). O *mapeamento entre DFs e tipos* é denotado como MDTN (Mapeamento entre DFs e Tipos para Nós), para nós (i.e., $\text{MDTN}: \mathcal{D}^n \uparrow \mathcal{T}^n$), e como MDTE (Mapeamento entre DFs e Tipos para Enlaces), para enlaces (i.e., $\text{MDTE}: \mathcal{D}^e \uparrow \mathcal{T}^e$). Se um DF $d$ pertence a um tipo $t$, então $\text{MDTN}_{d,t} = 1$, senão 0. O *mapeamento entre recursos físicos e DFs* é denotado como MND (Mapeamento entre Nós e DFs), para nós (i.e., $\text{MND}: \mathcal{N}^p \uparrow \mathcal{D}^n$), e como MED (Mapeamento entre Enlaces e DFs), para enlaces (i.e., $\text{MED}: \mathcal{E}^p \uparrow \mathcal{D}^e$); se $n^p$ está no tipo $d$, então $\text{MND}_{n^p,d} = 1$, senão 0 (o mesmo se aplica aos enlaces).

A função objetivo de confiabilidade minimiza as métricas de falha local e global. As Eqs. 20–23 especificam as restrições para alcançar a função objetivo ótima. As Eqs. 20

e 21 garantem que, para cada tipo de DF $t$, o número de nós (ou enlaces) virtuais de uma dada IV afetados por uma falha deste tipo (no pior caso) é no máximo *fln* (ou *fle*). As Eqs. 22 e 23 fornecem restrições análogas para as métricas de falha global. Para simplificar a formulação, a Eq. 20 define um subconjunto de $\mathcal{N}^v$ como $\mathcal{N}^{v\prime} := \forall n^v \in \mathcal{N}^v$: $\text{REQ}_{n^v,v}$ para considerar somente nós de uma dada IV $v$, e a Eq. 22 define um subconjunto de $\mathcal{N}^p$ como $\mathcal{N}^{p\prime} := \forall n^p \in \mathcal{N}^p$: $\text{MND}_{n^p,d}$ para considerar somente nós físicos no DF $d$. As mesmas abordagens são válidas para enlaces: na Eq. 21, um subconjunto de $\mathcal{E}^v$ é definido como $\mathcal{E}^{v\prime} := \forall (n_1^v, n_2^v) \in \mathcal{E}^v$: $\text{REQ}_{n_1^v,v}$ para considerar somente enlaces de uma IV $v$, e, na Eq. 23, um subconjunto de $\mathcal{E}^p$ é definido como $\mathcal{E}^{p\prime} := \forall e^p \in \mathcal{E}^p$: $\text{MED}_{e^p,d}$ para incluir somente enlaces no DF $d$.

$$\sum_{n^{p\prime}\in\mathcal{N}^{p\prime}} \sum_{n^{v\prime}\in\mathcal{N}^{v\prime}} mn_{n^{v\prime},n^{p\prime}} \leq fln_{v,t}, \quad \forall v \in \mathcal{V}, \forall d \in \mathcal{D}^n, \forall t \in \mathcal{T}^n\text{: } \text{MDTN}_{d,t} \tag{20}$$

$$\sum_{e^{p\prime}\in\mathcal{E}^{p\prime}} \sum_{e^{v\prime}\in\mathcal{E}^{v\prime}} me_{e^{v\prime},e^{p\prime}} \leq fle_{v,t}, \quad \forall v \in \mathcal{V}, \forall d \in \mathcal{D}^e, \forall t \in \mathcal{T}^e\text{: } \text{MDTE}_{d,t} \tag{21}$$

$$\sum_{n^{p\prime}\in\mathcal{N}^{p\prime}} \sum_{n^v\in\mathcal{N}^v} mn_{n^v,n^p} \leq fgn_t, \quad \forall d \in \mathcal{D}^n, \forall t \in \mathcal{T}^n\text{: } \text{MDTN}_{d,t} \tag{22}$$

$$\sum_{e^{p\prime}\in\mathcal{E}^{p\prime}} \sum_{e^v\in\mathcal{E}^v} me_{e^v,e^p} \leq fge_t, \quad \forall d \in \mathcal{D}^e, \forall t \in \mathcal{T}^e\text{: } \text{MDTE}_{d,t} \tag{23}$$

As Eqs. 24–27 são as componentes de falha da função objetivo (Eq. 28). Elas minimizam o número de falhas de nós e enlaces virtuais, tendendo a distribuir os recursos de IVs pelo substrato físico. Conforme as definições da Seção 2, a Eq. 24 (e Eq. 26 para enlaces) denotam a razão entre a soma das falhas locais (e.g., soma das falhas de circuito, *rack* e servidor para cada IV) e o produto entre o número de tipos e o número de IVs solicitadas. Similarmente, na Eq. 25 (Eq. 27), esses valores denotam a razão entre as falhas globais e o número de tipos. Essas componentes são normalizadas pelos valores de pior caso estimados para as falhas globais de nós e enlaces: EFLN (Estimativa de Falha [EF] Local de Nó), EFGN (EF Global de Nó); e para as falhas locais de nós e enlace: EFLE (EF Local de Enlace), EFGE (EF Global de Enlace). Essas estimativas representam o número de recursos em cada tipo de DF. Por exemplo, na Fig. 1, $v_a$ tem três MVs e dois *switches* (cinco nós) e $v_b$ somente duas MVs; quando alocadas no mesmo *rack* (e.g. alocação $m^{fr}$), $\text{EFLN}_{v_a,r} = 5$, $\text{EFLN}_{v_b,r} = 2$ e $\text{EFGN}_r = 7$. Cada IV tem um número de nós e enlaces comprometidos por um pior caso de falha, denotados como *fln* e *fle*, respectivamente. Esses valores representam as métricas de falha local (nós e enlaces). Similarmente, considerando todos os recursos virtuais indistintamente de suas IVs, as métricas de falha global são denotadas como *fgn* e *fge* para nós e enlaces, respectivamente. As métricas locais e globais têm diferentes valores para cada tipo de DF.

$$\text{FLN} := \frac{\sum_{t\in\mathcal{T}^n} \sum_{v\in\mathcal{V}} \frac{fln_{v,t}}{\text{EFLN}_{v,t}}}{|\mathcal{T}^n| \cdot |\mathcal{V}|} \tag{24}$$

$$\text{FGN} := \frac{\sum_{t\in\mathcal{T}^n} \frac{fgn_t}{\text{EFGN}_t}}{|\mathcal{T}^n|} \tag{25}$$

$$\text{FLE} := \frac{\sum_{t\in\mathcal{T}^e} \sum_{v\in\mathcal{V}} \frac{fle_{v,t}}{\text{EFLE}_{v,t}}}{|\mathcal{T}^e| \cdot |\mathcal{V}|} \tag{26}$$

$$\text{FGE} := \frac{\sum_{t\in\mathcal{T}^e} \frac{fge_t}{\text{EFGE}_t}}{|\mathcal{T}^e|} \tag{27}$$

### 3.4. Modelo Combinado

A função objetivo (Eq. 28) para o modelo combinado minimiza as métricas de fragmentação (Eqs. 18 e 19) e falha (Eqs. 24–27). Os pesos $\alpha$ e $\beta$ podem ser ajustados pelo InP para obter o equilíbrio desejado entre fragmentação e confiabilidade. Por exemplo, um provedor pode aumentar $\beta$ para oferecer melhor confiabilidade por uma tarifa *premium*. A relação entre $\alpha$ e $\beta$ é explorada mais adiante na Seção 4.

$$\begin{aligned} \text{minimizar:} \quad & \alpha \cdot \left( \frac{\text{FrN} + \text{FrE}}{2} \right) + \beta \cdot \left( \frac{\text{FLN} + \text{FGN} + \text{FLE} + \text{FGE}}{4} \right) \\ \text{s.a.:} \quad & \text{Eqs. 7–17, 20–23} \end{aligned} \tag{28}$$

## 4. Avaliação

Nesta seção o modelo proposto é avaliado, focando no *trade-off* entre fragmentação e confiabilidade (Eq. 28). O modelo foi implementado em CPLEX[3] e executado em um computador AMD Phenom II X4 com 4 GB de RAM e Linux Ubuntu 12.04.02. As topologias das infraestruturas físicas e virtuais foram geradas usando um simulador escrito em Python. A topologia da infraestrutura física é baseada em um projeto de DC de três camadas conforme a arquitetura de referência [Cisco 2007], a qual é normalmente implementada em grandes DCs de alta resiliência. Ao todo são 120 nós em dois DCs interconectados. São considerados quatro tipos de DF: circuito, *rack*, servidor e enlace.

O número de recursos virtuais em uma IV e suas capacidades foram aleatoriamente gerados de acordo com uma distribuição uniforme. IVs têm entre dois e cinco nós (uma MV se liga a um nó, e um *switch* se liga a dois outros nós; essa regra gera um conjunto de grafos direcionados que representam cenários reais), e as demandas dos seus recursos são dimensionadas como uma fração das capacidades físicas; assim, uma MV demanda 25–50% da capacidade de CPU de uma MF, um *switch* virtual demanda 15–25% da memória de um *switch* físico, e um enlace virtual demanda 1–3% da largura de banda de um enlace de acesso. Esses parâmetros são similares àqueles usados em trabalhos relacionados [Bays et al. 2012, Oliveira et al. 2013].

O modelo é avaliado em dois cenários: um substrato com baixa (BC) e outro com alta (AC) carga de uso. Em cada cenário são geradas 150 IVs a serem alocadas no substrato. A cada iteração chega uma IV nova, o modelo é executado e a IV é alocada se o substrato não estiver saturado, permanecendo ativa por um tempo de vida definido. Em média, o modelo leva 8,9 segundos para alocar uma IV. O tempo de vida de uma IV é de cinco (cenário BC) ou 85 (cenário AC) iterações. A Fig. 2 ilustra os resultados para cada cenário utilizando diferentes pesos de fragmentação ($\alpha$) e falha ($\beta$) na função objetivo (Eq. 28). Os pesos satisfazem a condição $\alpha + \beta = 1$. Os valores do peso de falha menores que 0,5 foram omitidos por motivos de clareza, uma vez que resultavam em uma confiabilidade pior sem diminuir significantemente a fragmentação. Para fins comparativos, também são mostradas as curvas de fragmentação "pura" ($\alpha = 1$, $\beta = 0$) e confiabilidade "pura" ($\alpha = 0$, $\beta = 1$), as quais fornecem linhas de base (LBs) de fragmentação e falha; o principal objetivo é avaliar os cenários com $\beta \in [0{,}5; 0{,}9]$.

As Figs. 2(a)–2(h) ilustram as falhas local e global de nó e enlace (FLN, FGN, FLE e FGE na Eq. 28) para cada cenário. Nesses cenários, a LB de fragmentação representa o *limite superior de falha* (i.e., os piores valores) e a LB de falha representa o *limite*

[3] Versão 12.5.1, `http://goo.gl/GfzpYx`

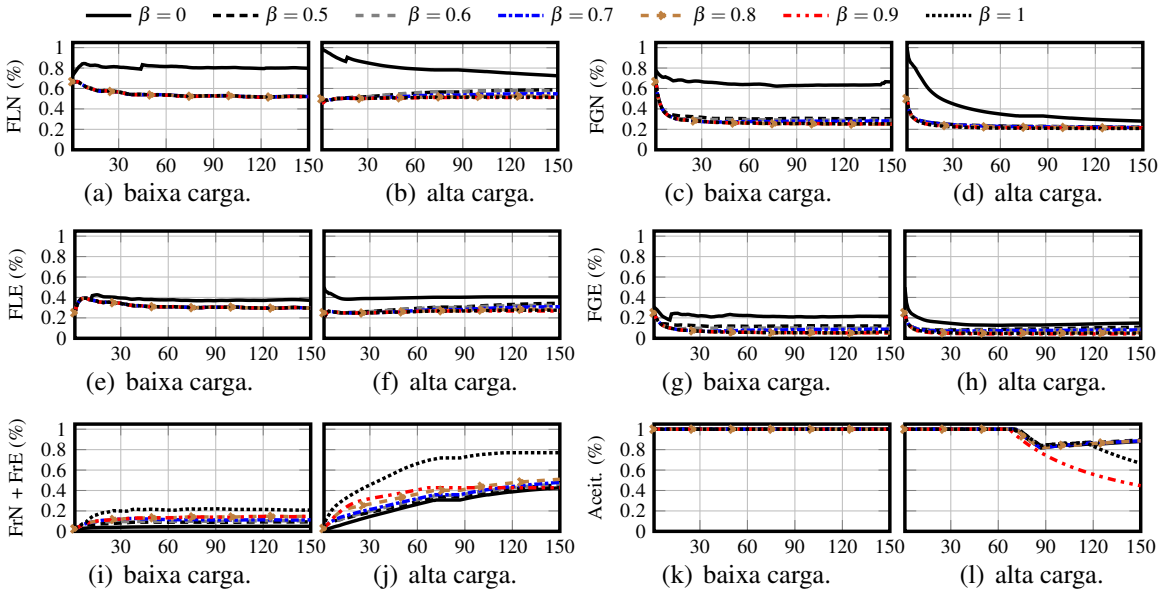



**Figura 2. Resultados experimentais para os cenários BC e AC.**

*inferior de falha* (i.e., os valores ótimos). Pode ser observado que, nos dois cenários, $\beta \in [0{,}5; 0{,}9]$ resulta em efeitos similares nas métricas de falha, com as curvas próximas às da LB, estando somente 1,18 pontos percentuais (pp) acima, em média. Só a LB de fragmentação é visivelmente pior em termos de falhas, estando 16,62 pp acima, em média, principalmente nas métricas de falha local e global de nó no cenário de baixa carga de uso (BC, Figs 2(a) e 2(c)), as quais têm uma piora de 33,32 pp.

As Figs. 2(i) e 2(j) ilustram a fragmentação combinada de nó e enlace (FrN e FrE na Eq. 28) para cada cenário. Nesses cenários, a LB de falha representa o *limite superior de fragmentação* (i.e., os piores valores) e a LB de fragmentação representa o *limite inferior de fragmentação* (i.e., os valores ótimos). No cenário BC (Fig. 2(i)), as métricas resultam em valores muito próximos, com $\beta \in [0{,}5; 0{,}9]$ estando somente 7,15 pp acima da LB de fragmentação e 8,75 pp abaixo da de falha, em média. No cenário AC (Fig. 2(j)), $\beta \in [0{,}5; 0{,}9]$ está somente 0,83 pp acima da LB de fragmentação e 30,26 pp menos fragmentada que a de falha, em média.

É possível perceber que todos os valores de $\beta \in [0{,}5; 0{,}9]$ resultam em boas métricas. Contudo, levando em consideração a taxa de aceitação para o cenário AC (Fig. 2(l)), de alta carga, os valores para $\beta \in [0{,}9; 1]$ tiveram alta taxa de rejeição de IVs, 44,67 % e 66,67 %, respectivamente. É importante notar, ainda, que para $\beta \in [0{,}5; 0{,}8]$ a taxa de aceitação foi 0,83 pp maior que a da fragmentação "pura" ($\beta = 0$).

Assim, levando em consideração ambas as perspectivas (fragmentação e confiabilidade) e a taxa de aceitação, é possível dizer que os pares de $(\beta = 0{,}5, \alpha = 0{,}5)$ até $(\beta = 0{,}8, \alpha = 0{,}2)$ fornecem os melhores *trade-offs*. Um InP realizando alocações com esse modelo e usando esses pesos poderá aprimorar a confiabilidade das IVs balanceando estes objetivos conflitantes. Enquanto outros cenários possam requisitar diferentes valores para $\alpha$ e $\beta$, os resultados evidenciam que o modelo proposto pode ser usado para encontrar um *trade-off* entre fragmentação e confiabilidade na alocação de IVs. Eles destacam que InPs podem oferecer IVs confiáveis sem comprometer as métricas de fragmentação.

## 5. Trabalhos Relacionados

A alocação de IVs considerando requisitos de confiabilidade é discutida em [Koslovski et al. 2010], onde isso é alcançado reservando recursos de *backup*, e em [Yu et al. 2010], onde é proposta uma solução para mapeamento e realocação de IVs confiáveis em um substrato federado. Essa abordagem parte de uma solução ótima de mapeamento, refinada por uma heurística que adiciona recursos de *backup* considerando uma sequência de possíveis cenários de falha, o que aumenta os custos de provisionamento.

A literatura que trata requisitos de confiabilidade na alocação de redes virtuais é mais extensa. Em [Bodík et al. 2012] é proposto um modelo para mapeamento entre MVs e recursos de *data centers* levando em conta requisitos de disponibilidade. Os nós no substrato são agrupados em DFs distintos e a função objetivo busca maximizar a confiabilidade no pior caso de falha e minimizar a utilização da largura de banda (favorecendo a hospedagem de uma rede virtual em um único *rack* ou sob um mesmo *switch* de agregação). Contudo, como cada rede virtual é somente considerada isoladamente (abordagem é análoga à métrica de falha local apresentada neste artigo), isso pode levar a uma situação indesejável da perspectiva das falhas globais. Em [Rahman et al. 2010] são introduzidos modelos para a alocação de redes virtuais confiáveis que minimizam as penalidades por violações de SLA e utilização de largura de banda reservando recursos esparsos no substrato. Em [Barla et al. 2012] é proposto um modelo para a minimização da latência em redes virtuais, que são espalhadas por diferentes *data centers* para aumentar a confiabilidade. Nessa abordagem, o mapeamento entre nós virtuais e *data centers* pode ser controlado tanto pelo usuário (que define um DC primário e um de *backup* para a sua rede virtual) quanto pelo provedor (que determina quais DCs devem ser usados). No entanto, o modelo não considera a capacidade de recursos virtuais, e o esquema de confiabilidade não apenas usa uma maior granularidade de DFs (e.g., regiões geográficas), mas também delega ao usuário tarefas com objetivos conflitantes (distribuir nós para melhorar a confiabilidade enquanto tenta minimizar a latência). Recentemente, esse trabalho foi estendido para fornecer resiliência fim a fim nas camadas física e virtual [Barla et al. 2013]. De forma similar a este artigo, os autores consideram que um InP pode oferecer para o usuário um serviço de resiliência (confiabilidade).

Modelos ótimos para alocação de redes virtuais com suporte a requisitos tanto de resiliência de enlace [Oliveira et al. 2013] quanto de confiabilidade [Bays et al. 2012] também foram propostos. A fragmentação de recursos não é considerada, o que pode levar a uma utilização ineficaz do substrato.

Todas as propostas citadas resolvem problemas de confiabilidade reservando recursos de contingência ou distribuindo os recursos virtuais pelo substrato, o que aumenta a fragmentação e, consequentemente, os custos para o InP.

## 6. Conclusão

Este artigo apresentou um modelo de alocação ótima de recursos para IVs confiáveis. Resultados experimentais demonstraram que o modelo é eficiente em otimizar simultaneamente a fragmentação, minimizando a utilização do substrato e diminuindo os custos do provedor, e a confiabilidade, beneficiando os usuários por executar suas aplicações em IVs confiáveis e, ao mesmo tempo, garantindo ao InP o cumprimento de um maior número de SLAs. Este é o primeiro trabalho a considerar conjuntamente a fragmentação do substrato

e a confiabilidade de IVs. Como perspectivas de trabalhos futuros, podem ser investigadas heurísticas para soluções em tempo polinomial, bem como aumentar a expressividade do modelo, permitindo aos usuários especificarem quais componentes de suas IVs devem ser confiáveis (e, então, alocadas em recursos que falham de forma independente).

## Agradecimentos

Os autores gostariam de agradecer ao PROMOP, programa de apoio financeiro da UDESC.

## Referências

Anhalt, F., Koslovski, G., and Primet, P. V.-B. (2010). Specifying and Provisioning Virtual Infrastructures with HIPerNET. *Int. J. Network Management*, 20(3):129–148.

Barla, I., Schupke, D., Hoffmann, M., and Carle, G. (2013). Optimal Design of Virtual Networks for Resilient Cloud Services. In *Proc. DRCN*, pages 218–225.

Barla, I. B., Schupke, D. A., and Carle, G. (2012). Resilient Virtual Network Design for End-to-End Cloud Services. In *Proc. IFIP Networking*, pages 161–174.

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*, pages 378–384.

Bodík, P., Menache, I., Chowdhury, M., Mani, P., Maltz, D. A., and Stoica, I. (2012). Surviving Failures in Bandwidth-Constrained Datacenters. In *Proc. ACM SIGCOMM*, pages 431–442.

Chowdhury, N. M. K., Rahman, M. R., and Boutaba, R. (2009). Virtual Network Embedding With Coordinated Node and Link Mapping. In *Proc. IEEE INFOCOM*, pages 783–791.

Cisco (2007). Cisco Data Center Infrastructure: 2.5 Design Guide.

Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. (2008). The Cost of a Cloud: Research Problems in Datacenter Networks. *ACM SIGCOMM Comp. Comm. Review*, 39(1):68–73.

Koslovski, G., Soudan, S., Gonçalves, P., and Vicat-Blanc, P. (2011). Locating Virtual Infrastructures: Users and InP Perspectives. In *Proc. IFIP/IEEE IM*, pages 153–160.

Koslovski, G., Yeow, W.-L., Westphal, C., Huu, T. T., Montagnat, J., and Vicat-Blanc, P. (2010). Reliability Support in Virtual Infrastructures. In *Proc. IEEE CloudCom*, pages 49–58.

McCarthy, J. (2012). The 10 Biggest Cloud Outages of 2012. *CRN*. `http://goo.gl/0civLK`.

Mell, P. and Grance, T. (2009). The NIST Definition of Cloud Computing. `http://goo.gl/PsJP5R`.

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.

Ponemon (2011). Understanding the Cost of Data Center Downtime: An Analysis of the Financial Impact on Infrastructure Vulnerability. `http://goo.gl/2jM3c`.

Ponemon (2013). 2013 Study on Data Center Outages. `http://goo.gl/v5DBJO`.

Rahman, M. R., Aib, I., and Boutaba, R. (2010). Survivable Virtual Network Embedding. In *Proc. IFIP Networking*, pages 40–52.

Raphael, J. R. (2013). The Worst Cloud Outages of 2013 (So Far). *Infoworld*. `http://goo.gl/6Q7oK7`.

Xu, J., Tang, J., Kwiat, K., Zhang, W., and Xue, G. (2012). Survivable Virtual Infrastructure Mapping in Virtualized Data Centers. In *Proc. IEEE GLOBECOM*, pages 196–203.

Yeow, W.-L., Westphal, C., and Kozat, U. (2010). Designing and Embedding Reliable Virtual Infrastructures. In *Proc. ACM SIGCOMM VISA*, pages 33–40.

Yu, H., Qiao, C., Anand, V., Liu, X., Di, H., and Sun, G. (2010). Survivable Virtual Infrastructure Mapping in a Federated Computing and Networking System under Single Regional Failures. In *Proc. IEEE GLOBECOM*, pages 1–6.

# Energy-Efficient Virtual Machines Placement

**Albert P. M. De La Fuente Vigliotti**[1], **Daniel Macêdo Batista**[1]

[1]Department of Computer Science – University of São Paulo (USP)
Rua do Matão, 1010 – Cidade Universitária – 05508-090 – São Paulo – SP – Brazil

{albert, batista}@ime.usp.br

***Abstract.*** *Energy efficiency on computer systems is a topic that is gaining a lot of interest. Even more in the cloud computing era, where data centers consumption corresponds to near 1.5% of total world wide power consumption. In this paper we present two novel approaches for virtual machines (VMs) placement consolidation. The two approaches aim to maximize the placed VMs on a host and therefore minimize the number of hosts used on a cloud computing environment. The first proposed approach is based on the Knapsack problem and the second one is based on an Evolutionary Computation heuristic. Both strategies have shown consumed energy reduction starting from 40.33% and up to 92.21% compared to a strategy that does not consider energy efficiency.*

## 1. Introduction

In the age of climate change and global warming, energy efficiency has become one of the most important design requirements for computing systems such as data centers and cloud systems. Data centers consume enormous amounts of electrical power, which leads to significant emissions of carbon dioxide into the environment. For example, the current IT infrastructure is responsible for about 2% of total world wide power consumption and $CO_2$ footprints. This corresponds to the typical electricity consumption of 120 million households per year. This consumption problem is rapidly growing [Beloglazov et al. 2010, Fettweis and Zimmermann 2008, Rich Brown 2007].

The performance per watt ratio of computers has been constantly increasing every year. If this trend continues, the cost of the energy consumed by a server during its lifetime will exceed the hardware acquisition cost. The problem is even worse for clusters, data centers and large-scale compute infrastructures. An energy consumption growth of 16-20% per year can be observed in the last years, corresponding to a doubling every 4-5 years [Beloglazov et al. 2010].

As reported by members of the Open Compute Project [The Open Compute Project Foundation 2013], Facebook's Oregon data center is one of the most energy efficient data center in the world. The data center had a power usage effectiveness (PUE) of 1.09, which means that more than 91% of energy is consumed by the computing resources. Now it is important to focus on resources management efficiency.

In this paper we propose two energy efficient strategies for virtual machines placement on homogeneous cloud environments. The proposal considers several resources like CPU, memory, network and I/O utilization. The first approach is based on a Knapsack algorithm and the second one is based on an Evolutionary Computation (EC) algorithm. Both approaches are compared with a strategy that doesn't consider energy efficiency.

Experiments showed significant energy savings starting from 40.33% and up to 92.21% on simulations performed with real workloads. Besides, a framework in Python has been developed along with the algorithms to facilitate the simulations.

The reminder of this paper is structured as follows: a more detailed overview on the concepts used can be found on the rest of this section. Afterward we review the related work on Section 2. We introduce the problem on Section 3 and the algorithms on Section 4. The framework developed during this work is discussed on Section 5. The performance evaluation is presented on Section 6. Conclusions and future directions can be found on Section 7.

### 1.1. Virtualization

Virtualization is a technology that allows to create logical computing units called Virtual Machines (VMs) that can accommodate an individual OS, allowing a physical server behavior. These VMs are able to run applications transparently isolated from the physical hosts where the VMs run.

When talking about virtualization there is an important choice to make: isolation of the guest and host vs. resources sharing. *Hypervisor-based virtualization* (VMware [VMware, Inc. 2013], Xen [Linux Foundation Collaborative Project 2013] and KVM [Redhat Emerging Technologies 2013]) prioritizes isolation of resources, libraries and OS from each VM from the host, over sharing, by running a full OS both on the guest and on the host with the Consequent overhead. As technology evolved and many hardware virtualization extensions like Intel-VT and AMD-V got better over time, the performance got better as well. Nevertheless there is still a performance gap, specially regarding I/O operations.

With recent developments of *container-based virtualization* CGroups (Control Groups) [Jackson and Lameter 2013], Linux Containers (LXC - *"chroot on steroids"*) [LXC Development Group 2013], OpenVZ [Community project, supported by Parallels, Inc. 2013] and Linux-VServer [Linux-VServer Development Group 2013] there was a step forward achieving almost native performance by using the same host OS within the guest VM by sharing the host libraries. Other typical scenarios include separating applications to improve security, hardware independence and migration for load balancing of containers in a cluster.

Recent studies have shown how container based virtualization could be used even in High Performance Computing (HPC) environments [Xavier et al. 2013], that is still not even an option with hypervisor-based virtualization technology. The two approaches share advantages, but some features are unique to each virtualization platform (e.g., full network virtualization, migration, etc.) therefore some users still prefer hypervisors instead of containers. In terms of energy savings, the use of containers is better because it does not have the overhead of having two operating systems. In this work we consider that virtualization is managed by containers and that it is possible to retrieve each resource usage percentage at any time.

### 1.2. Cloud Computing

There are many definitions for *Cloud computing*, often related to cloud and web services for marketing reasons, but cloud computing is more than that. A complete definition has

been made by the *National Institute of Standards and Technology (NIST)*:

*"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction..."* [Mell and Grance 2011]

*Cloud computing* has many benefits, among them we can highlight: (a) *costs savings* – organizations no longer have to invest money in their own IT infrastructure, as services are available on demand paid as they are used, so there are decreasing investments and running costs; (b) *efficiency* – cloud providers can operate better, faster and in a more cost-efficiently way than own IT infrastructure; (c) *improve availability* – it is common to find load balancing, redundant connections and state of the art infrastructure setup to meet business needs. It also improves (d) *scalability* – services can be scaled for business needs as resources can be allocated dynamically [Brian et al. 2008].

## 2. Related Work

There are some studies regarding optimization of energy consumption on hypervisor-based virtualized cloud computing environments, The work [Beloglazov and Buyya 2013] proposes an approach for solving the problem of host overload detection. They maximize the mean inter-migration time under the specified QoS goal based on a Markov chain model. The algorithm handles unknown non-stationary workloads using a multisize sliding window workload estimation technique.

OpenStack Neat [Beloglazov and Buyya 2012] is an open source software framework for distributed dynamic VM consolidation in cloud data centers based on the OpenStack platform. The framework acts independently of the OpenStack platform and applies VM consolidation processes by invoking public Application Programming Interfaces (APIs) of OpenStack.

CloudSim [Rajkumar Buyya 2013] is an extensible cloud simulation framework developed in Java that enables seamless modeling, simulation, and experimentation of cloud computing infrastructures. It supports modeling and simulation of energy-aware computational resources.

The VMs consolidation approaches cited until now are limited to a single resource utilization analysis (e.g., CPU), nevertheless we believe that other resources indicators like: network, I/O and memory could be used to improve the decision making process. These resource indicators can be retrieved by using container based virtualization which motivated this work.

Regarding containers-based virtualization there is the work [Xavier et al. 2013], but we noticed that they focused only on evaluation and comparing performance with hypervisor-based virtualization of high performance computing environments (HPC) and not regular cloud systems. They also presented that it seems to be a consensus that today's hypervisor-based virtualization systems perform well for CPU intensive applications, but exhibit a high overhead when handling I/O intensive applications, especially the network intensive ones. They do not study energy aspects as we propose.

As presented in [Soltesz et al. 2007], container-based systems provide up to 2x the performance of hypervisor-based systems for server-type workloads and scale further

while preserving performance. In general, we notice a lack of studies on the container-based virtualization energy efficiency area, and that LXC support for OpenStack is very limited. This might be due to the fact that containers technology has been integrated on the Linux Kernel recently and by consequence is a topic full of opportunities.

## 3. Energy Efficient VM Allocation

Allocation of VMs can be divided in two problems, the first one is the placement of a new VM on a host while the second one is the optimization of the current VMs allocation. In this paper we focus on the first problem.

Most of the studies we found observes the CPU utilization and makes their decision based on it. This makes sense since there is a relationship between the total power consumption by a server and its CPU utilization as observed by [Fan et al. 2007]. Basically their model proposes that power consumption by a server grows linearly with the growth of the CPU utilization. It is interesting to notice that in data centers, as observed by [Barroso and Holzle 2007], the mean value of the CPU utilization is 36.44% with 95% confidence interval.

CPU utilization based models are able to provide an accurate prediction for CPU-intensive applications, however they tend to be inaccurate for other types of prediction like I/O and memory intensive applications as concluded by [Dhiman et al. 2010]. For this reason our proposal is to include other resource usage indicators to try to increase the accuracy of the predictions.

We notice that the available cloud computing simulators like CloudSim [Rajkumar Buyya 2013] make their decision only based on CPU observation, hence we developed a VM placement simulation framework in Python called pyCloudSim [1][Vigliotti and Batista 2014] which considers other resources like network, memory and I/O usage. It would be easy to customize the code to include new resource indicators on future works.

Energy efficient VM allocation takes advantage of the fact that idle hosts can be suspended to low-power modes to reduce the overall energy consumption. Later the hosts can be reactivated remotely over the network by using Wake-on-LAN when an increase of resources use is detected. It is important to carefully evaluate where consolidation of the VMs is going to take place, we do not want to migrate VMs to an overloaded host. It is very important to always keep in mind the Quality of Service (QoS) so we do not fall into Service Level Agreements (SLA) violations. This is more like the second type of problem (optimization of the current VMs allocation).

The process to allocate virtual machines considered in our work is based on the work proposed in [Beloglazov and Buyya 2012]. This process consists in performing periodical dynamic consolidation of VMs by packing them on fewer physical machines as possible to conserve energy in virtualized data centers. The reason besides this idea is that computers are more efficient when they are operating near 100% of their capacities. This approach can be divided into the following sub-problems.

- **Information retrieval**: as stated before, most current VMs consolidation approaches are limited to CPU utilization analysis. We can include other resources

[1] `https://github.com/vonpupp/pyCloudSim/tree/v0.1`

like network, I/O and memory.

- **Underload detection**: it's important to detect hosts that can be considered underloaded because they will be operating with a bad energy efficiency. In this case all the VMs from this host should be migrated to another host (if possible) and later the "old host"should be switched to a low-power mode.
- **Overload detection**: similar to the previous step, to ensure Quality of Service (QoS) requirements, if a host is considered overloaded, some VMs should be migrated to another host, either active or in low-power mode hosts. The last ones need to be reactivated.
- **VMs selection**: select which VMs to migrate from an overloaded host.
- **VMs migration**: migrate the selected VMs to other hosts (active or suspended).

## 4. Proposed Approaches

This section presents the two proposed approaches to allocate VMs considering the energy consumption of physical machines modeled with CPU and other resources: network interfaces, memory and I/O. We propose two different approaches to solve the problem, the first one is based on a Knapsack strategy while the second one is based on an Evolutionary Computation (EC) strategy.

We can define the problem as follows. Given sets of:

- Physical hosts $H = \{H_1, \ldots, H_n\}$ of size $n$
- Available resources $\vec{C}_p = (C_{p,1}, \ldots C_{p,d})$ of size $d$ (dimensions)
- Virtual Machines $V = \{V_1, \ldots, V_m\}$ of size $m$
- Used resources $R = \{R_1, \ldots, R_d\}$ of size $d$ (dimensions)

Each physical host $H_i$ has a $d$-dimensional available resource capacity as a vector $\vec{C}_i = (C_{i,1}, \ldots C_{i,d})$, i.e CPU, memory, network, etc. All the virtual machines resources usage $vm_j \in V$ are represented by a $d$-dimensional resource demand vector $\vec{r}_j = (r_{j,1}, \ldots, r_{j,d}) \in [0, 1]^d$.

The objective of our approaches is to minimize the physical hosts used, which can also be seen as maximize the number of VMs per host constrained to the resources usage. Moreover, the approaches are not naive in terms of SLA so we also aim to avoid a high rate of blocked virtual machine allocations, which can happen if there are no available physical machine to attend the demand of new virtual machine requests.

### 4.1. The Knapsack Approach

In this first approach we model the VMs placement problem as a Multidimensional Bin Packing (MDBP) approach. In this approach the following decision variables need to be found:

Physical host allocation variable $h_p = \begin{cases} 1 & \text{if the physical host } p \text{ is used by some VM} \\ 0 & \text{otherwise} \end{cases}$

Virtual machine allocation variable $x_{i,p} = \begin{cases} 1 & \text{if the virtual machine } i \text{ is assigned to the host } p \\ 0 & \text{otherwise} \end{cases}$

The goal is to minimize the number of physical hosts,

$$\text{Minimize} \sum_{p=1}^{n} h_p \tag{1}$$

Subject to the following constraints:

$$\sum_{i=1}^{m} r_{i,k} \times x_{i,p} \leq C_{p,k} \times h_p, \forall p \in \{1, \ldots, n\}, \forall k \in R \quad (2)$$

$$\sum_{i=1}^{n} x_{i,p} = 1, \forall i \in \{1, \ldots, m\} \quad (3)$$

Where constraint (2) ensures that the capacity of each physical host is not exceeded and constraint for each resource (3) guarantees that each virtual machine is assigned to exactly one physical host.

MDBP is a variation of the Multiple-Choice Multidimensional Knapsack Problem (MMKP) which is a Non-deterministic Polynomial-time hard (NP-Hard) problem. This type of problem needs to be solved by heuristic algorithms. We first intended to solve MDBP problems limited to three dimensions (i.e. CPU, network and I/O), but we noticed that the algorithm did not scale well due to its complex nature, therefore we reduced the problem to solve the standard knapsack problem $n$-times (one per host). This strategy showed to be far more scalable than the first one, since we could run experiments with more than 280 VMs and 100 hosts in just a few seconds, while the MDBP did not scale for more than 10 hosts without taking a considerable amount of time. We named this approach as: *Iterated-KSP*, which is how we will refer to it for the rest of this work.

A Python library called `OpenOpt` [Kroshko 2013] has been used for the development of the Iterated-KSP strategy.

In Algorithm 1, a list of constraints is built (line 2) for each $d$-dimension. The summation of each resource used by the VMs within a host must be less than 100%. The constraints also include assigning a weight on each VM which will be the criteria to be maximized by the algorithm. In the simulation we used equal weights for all the VMs. Nevertheless, according to different strategies, this could be changed. The placement decisions for that host is performed using the `KSP` function (line 3), i.e. maximize the number of VMs to be placed on that host. As the host is initially without any VM, it is filled with its full computing capacity for each $d$-dimension without exceeding any dimension.

**Algorithm 1** Iterated-KSP Strategy

**Input:** Number of physical hosts: $\mathscr{P}$
**Input:** Number of virtual machines: $\mathscr{V}$

1: **function** ITERATED-KSP-SCENARIO(...)
2: constraints ← build constraints for every $d$-dimension to be $\leq 100$
3: placement ← maximize VMs using `KSP` with $\mathscr{P}$ *and* $\mathscr{V}$
4: **end function**

### 4.2. The Evolutionary Computation Approach

Similar to the Iterated-KSP strategy, we used a Python library called `Inspyred` to develop the evolutionary computation approach (which will be called Iterated-EC in the rest

of the paper). `Inspyred` is a library of biologically-inspired algorithms including evolutionary computation, swarm intelligence, and neural networks. Often these terms are also known as computational intelligence.

The `Inspyred` library grew out of insights from [De Jong 2006]. The goal of the library is to separate problem-specific computation from algorithm-specific computation in a clean way so as to make algorithms as general as possible across a range of different problems. Any bio-inspired algorithm has at least two aspects that are entirely problem-specific: how solutions to the problem look like and how such solutions are evaluated. These components will certainly change from problem to problem [Garrett 2013].

**Algorithm 2** Iterated-EC Strategy

**Input:** Number of physical hosts: $\mathscr{P}$
**Input:** Number of virtual machines: $\mathscr{V}$

1: **function** ITERATED-EC-SCENARIO(...)
2: ea ← inspyred.ec.EvolutionaryComputation
3: ea.selector ← inspyred.ec.selectors.tournament_selection
4: ea.variator ← list of inspyred.ec.variators.n_point_crossover, inspyred.ec.variators.bit_flip_mutation
5: ea.replacer ← inspyred.ec.replacers.generational_replacement
6: ea.terminator ← inspyred.ec.terminators.evaluation_termination
7: constraints ← build constraints for every $d$-dimension to be $\leq 100$
8: placement ← maximize VMs using ea.`evolve` with constraints, $\mathscr{P}$, $\mathscr{V}$, $\mathscr{E}$ $\mathscr{G}$
9: **end function**

The Algorithm 2 is called from the Algorithm 3 on every host to determine the VMs to be placed within. Similar to the Iterated-KSP algorithm, it is assumed that another algorithm already built the physical hosts and VMs lists from a given trace workload. The difference from Iterated-KSP is that now we need to remap the VMs data into a structure compatible with `Inspyred`.

A list of constraints is built (line 7) for each $d$-dimension. The summation of each resource used by the VMs within a host must be less than 100%. As we stated in the beginning of this section there are algorithm-specific components (lines: 3, 4, 5 and 6), and problem-specific components (line 8), which are $\mathscr{E}$ (the evaluator) and $\mathscr{G}$ (the generator). Similar to the Iterated-KSP algorithm the objective is to maximize the number of VMs per physical host. This is done with the ea.`evolve` (line 8).

The generator function ($\mathscr{G}$) generates possible solutions where each bit corresponds to including/excluding the VM at that host, giving a 1% chance of each bit being one, since we want the vast majority of the candidates to be zeroes. The fitness evaluation function ($\mathscr{E}$) is measured by adding up the number of VMs $d$-dimensions and subtracting the total amount of constraints that goes over 100. Other generation and evaluation functions may be used. Further details can be found on Section 7.

## 5. The Simulation Framework and Experiment Methodology

The simulation framework is based on the Algorithm 3 that iterates over the available (unplaced) physical hosts scenarios $\langle PMS \rangle$ (line 2) and later over the VMs scenarios

$\langle VMS \rangle$ (line 3) to determine a placement using a given strategy $\mathcal{S}$ for that scenario of physical hosts and VMs. Depending on the strategy $\mathcal{S}$, Algorithm 1 or Algorithm 2 will be used on every host to determine the VMs to be placed within.

The placement will be performed by evaluating the next suitable physical host that can handle the workload of a given VM, if any. Iterated-KSP and Iterated-EC strategies are energy-aware so hosts that are underloaded are automatically suspended with the consequent energy savings, and reactivated when required using the Wake-on-LAN mechanism. This is the opposite case compared with the Energy-Unaware strategy that does not suspend any host.

Each VM resource usage is represented by a vector as we introduced in Section 4. On each simulation, the CPU is represented directly by the traces we analyzed. To represent the other resources (memory, network, I/O) we used the same traces but with one fourth shifting of the used trace.

**Algorithm 3** Virtual Machines placement framework

**Input:** A Strategy: $\mathcal{S}$
**Input:** List of physical hosts scenarios: $\langle PMS \rangle$
**Input:** List of virtual machines scenarios: $\langle VMS \rangle$
**Output:** A placement solution

```
1: function SIMULATE-STRATEGY(...)
2:     for 𝒫 ∈ ⟨PMS⟩ do
3:         for 𝒱 ∈ ⟨VMS⟩ do
4:             Solution-scenario ← Simulate S with 𝒫 and 𝒱
5:         end for
6:     end for
7: end function
```

The algorithms strategies that can be used as input to Algorithm 3 are:

- Energy-Unaware: It is a simple reference strategy that when evaluating the $i$-th hosts sequentially, picks a random VM and if does not breaks the SLA then it is placed within that host, otherwise another VM is picked to check if again it is possible to place it until no more VMs or hosts are left.
- Iterated-KSP: As described on Algorithm 1
- Iterated-EC: As described on Algorithm 2

Consumed energy estimations are done linearly with the growth of the CPU utilization as proposed in [Fan et al. 2007] without loss of generality since our focus is to efficiently place the VMs regarding their resources use.

It is important to conduct experiments using workload traces from real systems rather than artificially generated ones, as this would help to reproduce a realistic scenario. So we analyzed more than 11,776 24-hour long traces provided as part of the CoMon project, a monitoring infrastructure for PlanetLab. 10 days of workload traces [2] collected during March and April 2011 have been randomly chosen as the data set. The traces

[2] `https://github.com/vonpupp/planetlab-workload-traces`

| Proposed | | Nearest | | | |
|---|---|---|---|---|---|
| Std | Mean | Std | Mean | No. | Trace filename |
| 15 | 25 | 15.5204 | 25.0694 | 1 | 146-179_surfsnel_dsl_internl_net_root |
| 15 | 35 | 15.9337 | 34.7465 | 2 | host4-plb_loria_fr_uw_oneswarm |
| 15 | 45 | 15.1083 | 44.7083 | 3 | plgmu4_ite_gmu_edu_rnp_dcc_ufjf |
| 30 | 25 | 30.1624 | 23.7673 | 4 | planetlab1_fct_ualg_pt_root |
| 30 | 35 | 30.4136 | 33.1076 | 5 | host3-plb_loria_fr_inria_omftest |
| 30 | 45 | 30.5568 | 46.5520 | 6 | planetlab1_georgetown_edu_nus_proxaudio |
| 45 | 25 | 35.0424 | 25.3125 | 7 | planetlab1_dojima_wide_ad_jp_princeton_contdist |
| 45 | 35 | 38.5349 | 35.4791 | 8 | planetlab-20110409-filtered_planetlab1_s3_kth_se_sics_peerialism |
| 45 | 45 | 43.5875 | 47.8680 | 9 | planetlab-wifi-01_ipv6_lip6_fr_inria_omftest |

**Table 1. Trace scenarios with the selected traces**

include data on the CPU utilization collected every 5 minutes from more than a thousand VMs deployed on servers located in more than 500 places around the world.

Later we analyzed each trace individually to obtain statistical indicators like minimum and maximum CPU usage, mean and standard deviation. We noticed that the overall standard deviation range was from 0.2634 to 43.5875, so we divided it into tree equal parts (15, 30, ≈45). The overall mean range was from 0.5173 to 95.9756. This time instead of dividing it into three equal parts we took $35 \pm 10\%$. The reason of using 35% as a base value is due to the observation in [Barroso and Holzle 2007], which states that the mean value of the CPU utilization is 36.44% with 95% confidence interval. The traces analysis is summarized on Table 1. Notice the fifth column (trace number), which is a trace identifier that we will be referring to on the rest of the paper.

The number of physical hosts is in the interval $[10, 100]$ with increments by 10. The number of VMs is in the interval $[16, 288]$ with increments by 16.

A simulation is made of a triplet of (trace, algorithm, hosts) with VMs varying on the interval $[16, 288]$.

## 6. Performance Evaluation

We repeated each simulation 30 times to check if there was a clear tendency, and later reduced the data to three cases:

- Best case: representing the best case among the 30 repetitions
- Worst case: representing the worst case among the 30 repetitions
- Average case: representing the average (mean) case among the 30 repetitions

The hardware used on the experiments was a VM with four dedicated cores and 10 Gigabytes of RAM hosted by an octo-core Intel(R) Core(TM) i7-2700K CPU @ 3.50GHz with 16 Gigabytes RAM host system.

The experiments resulted in more than 8,000 scenarios that have been later summarized into more than 1,900 graphs showing several aspects of each placement. We are going to present the most representative set of figures and our conclusions on the rest of this section. To allow the reproduction of our experiments, all the code, data, documentation and figures related to the experiments are publicly available at [3].

Figure 1 shows the average case of energy consumption for the three algorithms simulating the Trace 1 using 100 hosts, with VMs ranging from 16 to 288. Similarly,

[3] `https://github.com/vonpupp/sbrc-2014-simulation`

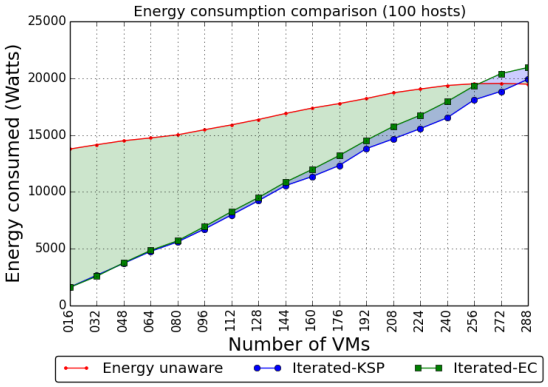



**Figure 1. Energy consumption comparison - Trace 1 (100 hosts) - Average case**

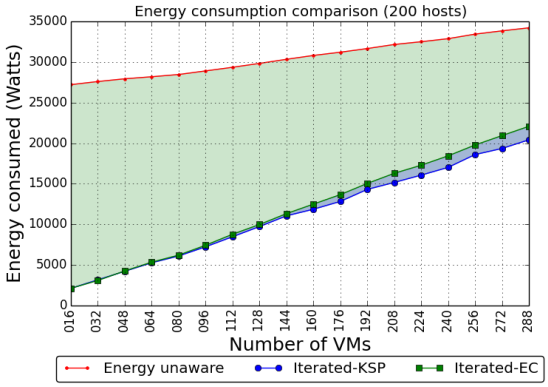



**Figure 2. Energy consumption comparison - Trace 1 (200 hosts) - Average case**

Figure 3 shows results of simulation of Trace 2 and Figure 4 shows results of simulation of Trace 3. The highlighted green area represents the energy savings from the Iterated-EC strategy (line with squares) in relation to the energy unaware strategy (line with dots), while the highlighted blue area represents the extra energy savings comparing the Iterated-EC strategy with the Iterated-KSP (line with circles). The lower the value, the better the strategy, hence the more energy savings. It is interesting to notice that near the intersection of the curves ($\sim$224 VMs on Figure 3 and $\sim$160 VMs on Figure 4) there is a saturation point where both strategies converge, this means that there are more VMs placed compared to the energy unaware strategy thus an optimal placement is achieved, this is also the reason why there is more energy consumption after the intersection of the curves compared with the energy unaware and therefore more VMs placed and a better use of the computation resources. After the saturation point the Iterated-EC tend to stabilize while the Iterated-KSP continues to optimize new VMs when there is no more resources for new VMs since resources are nearly exhausted.

As we can see there is a significant savings of energy with our proposed placement strategies, obviously the lower the overall cluster load (the less the VMs per physical host) the higher the energy savings and consequently the more physical suspended hosts. This can be observed by comparing Figures 1 and 2 which uses Trace 1 with 100 and 200 hosts respectively.

An observed tendency is that when the mean of resources usage increases (first three traces scenarios) a sooner saturation of VMs is achieved (fewer VMs can be placed). We also notice that when the standard deviation decreases the three algorithms tend to perform in a very similar way due to the fact that all VMs will have very similar resource needs.

Figure 5 shows the average case of unplaced VMs for the three algorithms simulating the Trace 2 using 100 hosts. The first strategy to fail to place a new VM is the energy unaware, even when this strategy places the VMs while there are resources available on the physical machines. As expected, it is less efficient than both Iterated-EC and

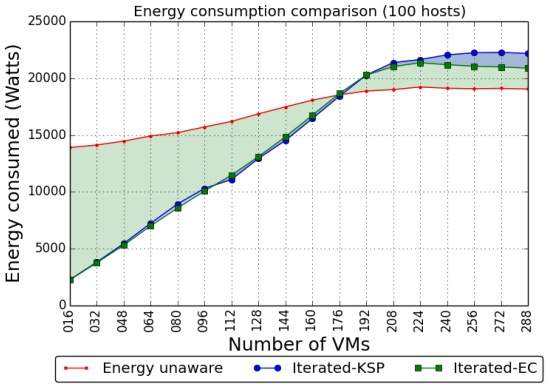



**Figure 3. Energy consumption comparison - Trace 2 - Average case**

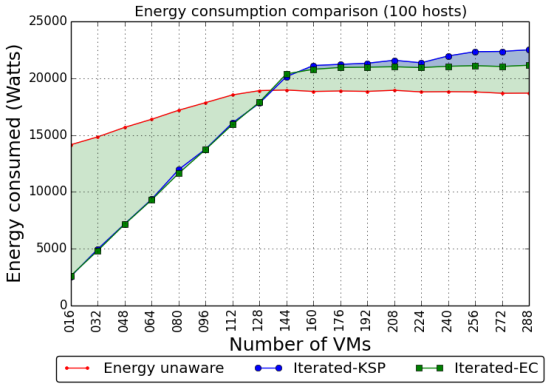



**Figure 4. Energy consumption comparison - Trace 3 - Average case**

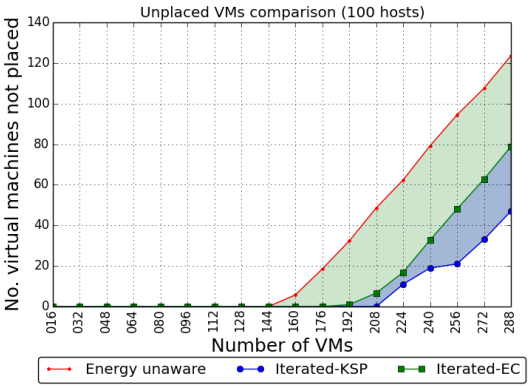



**Figure 5. Unplaced VMs comparison - Trace 2 - Average case**

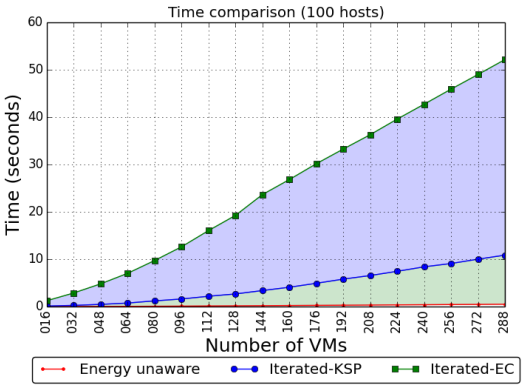



**Figure 6. Exec. time comparison - Trace 3 - Average case**

Iterated-KSP strategies. Iterated-KSP is slightly better than Iterated-EC by placing more VMs.

Figure 6 shows the execution time of the three strategies. The time taken by the Iterated-EC strategy is higher than the time taken by the Iterated-KSP strategy. This is due to the dependencies optimization of `OpenOpt` and our specific heuristic used on the Iterated-EC strategy.

Figure 7 shows the suspended physical machines for the three algorithms when simulating Trace 2 using 100 hosts. Both Iterated-EC and Iterated-KSP strategies are able to consume less energy due to the fact that they are able to place more VMs per physical host than the energy-unaware strategy.

We computed the 95% confidence intervals of the 30 experiments and we noticed that there is no significant difference. The energy unaware strategy is the only one who exhibits a broader confidence range due to the random behavior. The Iterated-EC strategy

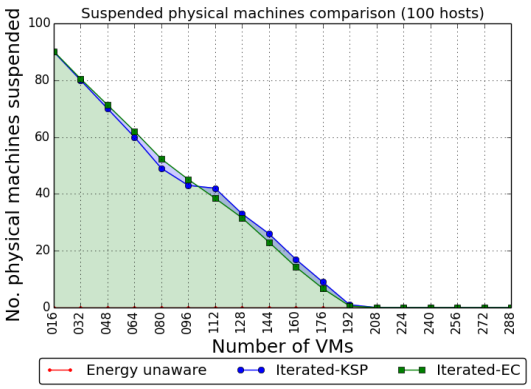



**Figure 7. Suspended physical hosts comparison - Trace 2 - Average case**

has insignificant variances and the Iterated-KSP strategy showed to be the most deterministic. Both showed predictable behavior.

## 7. Conclusion and Future Directions

As shown by the results, the proposed algorithms achieved significant energy savings. The Evolutionary Computation had energy savings starting from 35.46% for a workload of 288 VMs and up to 92.20% for a workload of 16 VMs with 200 hosts. The Knapsack based algorithm had energy savings starting from 40.33% for a workload of 288 VMs and up to 92.21% for a workload of 16 VMs. Concluding that the lower the overall cloud VMs workload the higher the energy savings compared to the energy unaware. We noticed that the Knapsack based approach is 7.55% better than the Evolutionary Computation approach (average case) which can be translated into a difference of 1667.70 Watts.

Both algorithms optimize hardware utilization on full placements compared to the energy unaware. The Knapsack based algorithm optimizes hardware by 6.20% to 20.40% however it is not stable. Evolutionary Computation ranges from 7.49% to 13.14% with a trend to be stable $\approx$ 11%. The Knapsack based algorithm is 11% to 15% faster than the Evolutionary Computation algorithm. The execution time difference tend to increase with the number of hosts and VMs at a rate of $\approx$ 5 seconds per 100 hosts.

There are other methods that are likely candidates to solve the problem of VMs placement that would be interesting to explore, among them we can cite: Greedy algorithms with different heuristics, Particle Swarm Optimization, Genetic Algorithms, Ant Colonies Optimization, Convex Hulls, Simulated Annealing, Tabu Search or Multicommodity flow. Following a *host-iterated* strategy should ease the integration process into our framework.

A specific approach has been proposed for the generation and evaluation functions on Algorithm 2, but there might be others that could be outperform the proposed ones. Also it would be possible to fine tune the parameters of the Iterated-EC algorithm to optimize the consumed time on computations. Another proposal is to run in parallel the Iterated-EC placement algorithm. This could be easily performed since the `inspyred`

installation process is straight forward.

Regarding the simulation framework, multithreading improvements can be done to the simulator so simulations can be delegated to different cores. It is important to only delegate full simulation scenarios which are thread-save, this is, the triplet described on Section 5.

It worth researching about making decisions on the distribution of resources among the physical hosts based on policies, i.e: it would be possible to have a physical servers pool for networking intensive VMs, another one for I/O intensive, etc. Another approach would be the opposite, try to distribute equally the resources $d$-dimensions across the physical hosts to optimize their use.

## Acknowledgments

We would like to thank Aaron Garrett for his help and support using `Inspyred`. We also would like to thank Elaine Watanabe and Danilo Bellini for their positive feedback on how to improve this work.

## References

[Barroso and Holzle 2007] Barroso, L. and Holzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.

[Beloglazov and Buyya 2012] Beloglazov, A. and Buyya, R. (2012). OpenStack neat: A framework for dynamic consolidation of virtual machines in OpenStack clouds–A blueprint. Technical report, Technical Report CLOUDS-TR-2012-4, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne.

[Beloglazov and Buyya 2013] Beloglazov, A. and Buyya, R. (2013). Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366–1379.

[Beloglazov et al. 2010] Beloglazov, A., Buyya, R., Lee, Y. C., and Zomaya, A. (2010). A taxonomy and survey of energy-efficient data centers and cloud computing systems. arXiv e-print 1007.0066.

[Brian et al. 2008] Brian, H., Brunschwiler, T., Dill, H., Christ, H., Falsafi, B., Fischer, M., Grivas, S. G., Giovanoli, C., Gisi, R. E., and Gutmann, R. (2008). Cloud computing. *Communications of the ACM*, 51(7):9–11.

[Community project, supported by Parallels, Inc. 2013] Community project, supported by Parallels, Inc. (2013). OpenVZ linux containers wiki. Accessed: 2013-09-05.

[De Jong 2006] De Jong, K. (2006). *Evolutionary Computation: A Unified Approach*. Bradford Book. Mit Press.

[Dhiman et al. 2010] Dhiman, G., Mihic, K., and Rosing, T. (2010). A system for online power prediction in virtualized environments using gaussian mixture models. In *2010 47th ACM/IEEE Design Automation Conference (DAC)*, pages 807–812.

[Fan et al. 2007] Fan, X., Weber, W.-D., and Barroso, L. A. (2007). Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, page 13–23, New York, NY, USA. ACM.

[Fettweis and Zimmermann 2008] Fettweis, G. and Zimmermann, E. (2008). ICT energy consumption-trends and challenges. In *Proceedings of the 11th International Symposium on Wireless Personal Multimedia Communications*, volume 2, page 6.

[Garrett 2013] Garrett, A. (2013). Inspyred inspired intelligence initiative. Accessed: 2013-11-13.

[Jackson and Lameter 2013] Jackson, P. and Lameter, C. (2013). CGROUPS kernel website.

[Kroshko 2013] Kroshko, D. L. (2013). OpenOpt website. Accessed: 2013-11-13.

[Linux Foundation Collaborative Project 2013] Linux Foundation Collaborative Project (2013). The xen project, the powerful open source industry standard for virtualization. Accessed: 2013-09-04.

[Linux-VServer Development Group 2013] Linux-VServer Development Group (2013). Linux-VServer. Accessed: 2013-09-05.

[LXC Development Group 2013] LXC Development Group (2013). LXC linux containers website. Accessed: 2013-09-02.

[Mell and Grance 2011] Mell, P. and Grance, T. (2011). The NIST definition of cloud computing (draft). *NIST special publication*, 800(145):7.

[Rajkumar Buyya 2013] Rajkumar Buyya, Rodrigo N. Calheiros, N. G. (2013). CloudSim website. Accessed: 2013-12-04.

[Redhat Emerging Technologies 2013] Redhat Emerging Technologies (2013). KVM (for kernel-based virtual machine) - main page. Accessed: 2013-09-04.

[Rich Brown 2007] Rich Brown (2007). Report to congress on server and data center energy efficiency:Public law 109-431.

[Soltesz et al. 2007] 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. In *ACM SIGOPS Operating Systems Review*, volume 41, page 275–287.

[The Open Compute Project Foundation 2013] The Open Compute Project Foundation (2013). Open compute project, "Energy efficiency". Accessed: 2013-10-06.

[Vigliotti and Batista 2014] Vigliotti, A. D. L. F. and Batista, D. M. (2014). pyCloudSim Github repository. Accessed: 2014-03-21.

[VMware, Inc. 2013] VMware, Inc. (2013). VMware virtualization for desktop & server, application, public & hybrid clouds - united states. Accessed: 2013-09-04.

[Xavier et al. 2013] Xavier, M., Neves, M., Rossi, F., Ferreto, T., Lange, T., and De Rose, C. (2013). Performance evaluation of container-based virtualization for high performance computing environments. In *2013 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 233–240.

# Algoritmos para economia de energia no escalonamento de workflows em nuvens computacionais

**Elaine N. Watanabe**[1], **Pedro P. V. Campos**[1], **Kelly R. Braghetto**[1], **Daniel M. Batista**[1]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)
Rua do Matão, 1010 – 05508-090 – São Paulo – SP – Brasil

{elainew,pedrovc,kellyrb,batista}@ime.usp.br

***Abstract.*** *Cloud computing is one of the platforms able to provide green computing solutions nowadays. Moreover, it is able to provide a high performance environment for scientific applications. However, if there is dependency between tasks, there may be a low utilization of the cloud. To address the problem, this paper presents new task schedulers for clouds aiming to achieve energy savings. Results of experiments comparing the proposed schedulers with other existing ones show that it is possible to obtain energy savings of up to 22.7%, with no penalty in the makespan. Moreover it is observed that the efficiency of the scheduling is dependent on how tasks are interconnected in the workflow.*

***Resumo.*** *A Computação em Nuvem é uma das plataformas capazes de prover soluções computacionais "verdes" na atualidade. Além disso, ela pode fornecer um ambiente de alto desempenho para aplicações científicas. Entretanto, caso haja dependência entre as tarefas, pode haver uma baixa utilização da nuvem. Para lidar com esse problema, este artigo apresenta novos escalonadores de tarefas para nuvens visando à economia de energia. Resultados de experimentos comparando os escalonadores propostos com outros existentes mostram que é possível obter economias de até 22,7% no consumo de energia, sem haver penalização no makespan. Além disso, comprova-se que a eficiência do escalonamento é dependente do modo como as tarefas estão interligadas no workflow.*

## 1. Introdução

A Computação Verde (*Green Computing*) tem como objetivo minimizar o impacto ambiental causado pelo uso da tecnologia por meio, por exemplo, do aumento da eficiência energética dos recursos computacionais e da diminuição da emissão de gás carbônico dos *data centers*, responsáveis por 2% do $CO_2$ emitido no mundo [Beloglazov and Buyya 2010]. Uma utilização ineficiente dos recursos computacionais pode resultar também em gastos extras com a infraestrutura necessária para manter um conjunto de servidores [Barroso and Hölzle 2007]. Estima-se que em 2014, custos com refrigeração e energia elétrica representarão 75% do custo total dos *data centers*, enquanto que o equipamento que essa infraestrutura suporta representará 25% [Belady 2007]. Por esses motivos, tecnologias como a Computação em Nuvem, que visa explorar o máximo dos recursos computacionais minimizando, por exemplo, seu consumo energético, são importantes na atualidade.

Além disso, plataformas de Computação em Nuvem são capazes de fornecer um ambiente de alto desempenho, como é observado no experimento realizado que criou

uma instância de aglomerado com 26.496 núcleos, usando máquinas do tipo *c3.8xlarge* da Amazon EC2, que obteve a posição 64 no rank do TOP500 em novembro de 2013[1]. O desempenho dessa instância foi equivalente ao de uma máquina com 484,2 TeraFLOPS. Outra característica importante é a provisão da infraestrutura e serviços sob demanda. Isso possibilita que as aplicações em execução acessem um ambiente elástico, cuja capacidade de processamento pode aumentar ou diminuir automaticamente por meio do uso de "máquinas virtuais" (do inglês *Virtual Machine* – VM). Uma vez que um servidor físico (*host*) pode executar diversas VMs ao mesmo tempo, esse mecanismo permite uma melhor utilização dos recursos disponíveis, resultando em custos menores para o usuário final [Cordeiro et al. 2013]. Assim, a possibilidade de acesso a um ambiente de alto desempenho a um custo reduzido comparado ao de supercomputadores, aliada às características de disponibilização de recursos sob demanda, tornam a Computação em Nuvem uma plataforma vantajosa para a execução de aplicações científicas, em especial, de fluxos de trabalhos (workflows) científicos.

Um *workflow científico* consiste na descrição de um experimento científico por meio de um conjunto de tarefas interligadas a serem executadas em computadores. Essas tarefas realizam ações tais como processamento e análise de grandes quantidades de dados, importantes características da ciência moderna. Entretanto, para que um workflow seja executado em uma nuvem computacional de forma viável, é essencial que sejam garantidos requisitos como a entrega de um resultado antes de certo tempo ou o controle do orçamento a ser gasto baseado em um limite definido pelo usuário. Nesse contexto, o processo chamado de *escalonamento* tem forte influência no comportamento e desempenho da execução do workflow por ser o responsável por determinar o momento do tempo em que cada tarefa será executada e quais recursos serão usados para isso. Esse tipo de processo visa, em geral, otimizar objetivos como o de minimizar o tempo total necessário para a conclusão das tarefas (chamado *makespan*). Além disso, deve garantir, principalmente na execução de workflows, que os dados compartilhados entre as tarefas sejam transferidos sempre que necessário. Portanto, um escalonador que não considerar as dependências entre as tarefas pode acabar reduzindo a exploração dos recursos computacionais compartilhados, por exemplo, não aproveitando 100% da potência máxima de um computador, um dos objetivos da Computação em Nuvem, resultando em um consumo energético desnecessário.

Assim, este trabalho tem como objetivo avaliar o escalonamento de workflows em nuvens computacionais visando à economia de energia. Para isso, são propostos 3 novos algoritmos, baseados no algoritmo de escalonamento HEFT [Topcuoglu et al. 2002], que avaliam a alocação de recursos com base em seu consumo energético. Experimentos foram realizados com a ferramenta CloudSim-DVFS [Guérout et al. 2013], utilizada para simulação energética de workflows em nuvens computacionais, e mostraram economias de energias de até 22,7% sem penalização no *makespan* da aplicação simulada. Tem-se como contribuição neste artigo a proposta de novos mecanismos de escalonamento de workflows ciente de energia, assim como um estudo comparativo com outros escalonadores existentes.

O artigo está estruturado da seguinte forma: na Seção 2 os trabalhos relacionados são discutidos e na Seção 3 são descritos os conceitos básicos necessários para o

[1]http://www.top500.org/system/178321

entendimento do trabalho. A Seção 4 apresenta os novos algoritmos de escalonamento propostos, enquanto que os resultados dos experimentos realizados com esses algoritmos são mostrados na Seção 5. As conclusões e trabalhos futuros são discutidos na Seção 6.

## 2. Trabalhos relacionados

O uso de ambientes de Computação em Nuvem para execução de workflows é abordado em [Vöckler et al. 2011], que descreve a experiência de uso e análise de testes ao usar o mesmo modelo de execução usado em computação em grade. Entretanto, características como custo financeiro e aquisição de recursos sob demanda diferem as nuvens dos ambientes de grades computacionais e aglomerados, sendo necessárias soluções de escalonamento adaptativas. Em [Bittencourt et al. 2012], são comparados vários algoritmos de escalonamento de workflows e sua aplicabilidade no escalonamento em nuvens. Dentre os mais populares, com mais de mil citações segundo o Google Acadêmico[2], destaca-se o HEFT (*Heterogeneous Earliest-Finish-Time*) [Topcuoglu et al. 2002], que será descrito na Seção 3 e servirá como base para os novos algoritmos propostos neste artigo.

Em relação à eficiência energética de nuvens computacionais, é destacada em [Beloglazov et al. 2012] a importância de soluções "verdes". O artigo apresenta um levantamento de projetos de pesquisa que abordam esse problema, destacando desafios de pesquisa em aberto, sob a perspectiva tanto dos provedores de recursos quanto dos consumidores. Propõe, também, princípios de arquitetura e políticas de alocação de recursos que visam ao gerenciamento energeticamente eficiente de nuvens computacionais. Utilizando a ferramenta CloudSim[3], um simulador de nuvens computacionais, tais políticas, que empregavam a consolidação dinâmica de VMs, foram avaliadas e os resultados demonstraram grande potencial para a melhoria da eficiência energética para cenários com cargas de trabalho dinâmicas. Entretanto, diferentemente deste trabalho, o artigo não aborda o problema da execução de um fluxo de cargas de trabalho. Inclusive, a ferramenta utilizada (CloudSim) permitia apenas a simulação de uma única carga.

Em [Aksanli et al. 2012], é descrito um estudo comparativo de simuladores de *data center* para avaliar o suporte a simulações energéticas, detalhando um estudo de caso com o simulador GENSim, o único simulador de *data center* capaz de estimar o impacto da alocação de *service jobs* e de *batch jobs* em um único servidor. O objetivo desse estudo foi demonstrar que a predição de energia verde pode melhorar a eficiência energética global de um *data center*. Pelo fato do GENSim não apresentar suporte à simulação de VMs, ele não foi selecionado para a realização dos nossos experimentos.

Em [Kliazovich et al. 2012], é realizada uma comparação entre simuladores de nuvens existentes e é proposto o GreenCloud, que se diferencia do CloudSim por ser capaz de avaliar os componentes da rede, e não apenas os servidores. O GreenCloud é uma extensão do simulador de redes ns-2 e é capaz de capturar, durante a distribuição de carga de trabalho na rede, detalhes tanto do consumo energético de componentes de um *data center* quanto dos padrões de comunicação nos pacotes de dados em configurações realísticas de ambiente. É destacada a importância de se considerar o consumo de energia de enlaces de comunicação, roteadores e outros elementos, visto que eles representam mais de 30% do total de energia consumida pelos *data centers*. Também é demonstrada

[2] `http://scholar.google.com`
[3] `http://www.cloudbus.org/cloudsim/`

a aplicabilidade e o impacto do uso de diferentes esquemas de gerenciamento de energia, como o dimensionamento de voltagem ou o desligamento dinâmico, aplicados tanto aos componentes computacionais quanto aos de rede. Entretanto, como o simulador não possui código aberto, não foi escolhido para avaliar os algoritmos propostos neste trabalho.

A fim de habilitar a ferramenta a realizar simulações de execuções de workflows, em [Weiwei and Deelman 2012] foi apresentado o WorkflowSim – uma extensão do simulador CloudSim. O WorkflowSim implementa uma camada para gerenciar a simulação da execução de diversas cargas de trabalho dependentes. Além disso, outras funcionalidades são fornecidas, como o agrupamento de tarefas, que é uma técnica que une pequenas tarefas em grandes "jobs" para reduzir sobrecargas. Os experimentos realizados utilizaram diversos workflows reais descritos na linguagem DAX, usada pelo sistema gerenciador de workflows Pegasus[4], permitindo que novos algoritmos de escalonamento sejam validados em cenários realísticos. O WorkflowSim possui também mecanismos para simulação de falhas na execução das tarefas e implementa políticas para o tratamento dessas. Entretanto, ele não utiliza o pacote de simulação energética do CloudSim, impossibilitando o uso com simulações que levem em consideração o consumo de energia, como as apresentadas neste artigo.

Este trabalho utiliza o simulador CloudSim-DVFS [Guérout et al. 2013], por ele possibilitar a simulação em nuvens computacionais de workflows descritos no mesmo padrão usado pelo Pegasus, avaliando o tempo de execução e o consumo energético. Além disso, como seu código é livre, foi possível incluir nele os novos algoritmos de escalonamento de tarefas apresentados neste artigo. Mais informações sobre o CloudSim-DVFS serão mostradas nas próximas seções do texto.

## 3. Conceitos básicos

Nesta seção serão apresentados os conceitos básicos empregados na construção dos algoritmos de escalonamento propostos neste trabalho. Esses algoritmos se baseiam no HEFT. Entretanto, eles têm como objetivo gerar escalonamentos de workflows em nuvens computacionais que resultem em economia de energia quando comparados aos escalonamentos gerados pelo HEFT.

### 3.1. Modelo energético

Um modelo energético pode ser dividido em duas categorias, conforme descrito em [Rivoire et al. 2008]: modelos completos (*comprehensive models*) e modelos de alto nível chamados de caixa-preta (*black-box*). O primeiro tipo permite obter grande acurácia do consumo energético das CPUs, entretanto, é dependente dos detalhes da microarquitetura dos processadores. O segundo tipo de modelo descreve de forma genérica o consumo energético, não sendo tão preciso. Porém, isso permite ampliar sua portabilidade, não necessitando de detalhes específicos da arquitetura do processador utilizado. Nos experimentos descritos neste trabalho, um modelo de alto nível é utilizado, fornecido pelo simulador utilizado, o CloudSim-DVFS [Guérout et al. 2013]. O modelo utilizado, chamado de modelo linear energético para frequências fixas, é descrito na Equação 1.

$$P_{TOT} = (1 - \alpha)P_{CPUOciosa} + (\alpha)P_{CPUCargaMaxima} \quad (1)$$

[4]`http://pegasus.isi.edu`

onde $\alpha$ é o uso de CPU e $P_{CPUOciosa}$ e $P_{CPUCargaMaxima}$ são os valores da energia consumida pela CPU com 0% e 100% de utilização, respectivamente.

### 3.2. Representação de workflows como grafos orientados

Diversos workflows científicos de diferentes áreas da ciência podem ser descritos como Grafos Dirigidos Acíclicos (*Directed Acyclic Graph – DAG*) [Deelman et al. 2009]. Para isso, o DAG representa o conjunto de tarefas da aplicação como vértices no grafo e demonstra a dependência entre elas por meio de arcos, indicando a precedência entre as tarefas. Tantos os vértices quanto os arcos podem ser rotulados com valores, que em geral indicam, para os vértices, o custo de computação e, para os arcos, o custo de comunicação, como é representado na Figura 1 . Essa representação é a base para o escalonamento, que deve ser feito de forma a respeitar as restrições de precedência, ao mesmo tempo que busca otimizar uma função objetivo.

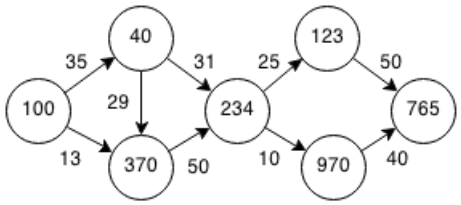



**Figura 1. Grafo dirigido acíclico. Na representação de workflows, os vértices representam as tarefas, enquanto os arcos representam dependência e precedência entre as tarefas. Normalmente os vértices são rotulados com o custo computacional; e os arcos, com o custo de comunicação.**

As dependências podem ser representadas quanto ao fluxo de dados, indicando que uma tarefa não pode ser executada antes que seus dados de entrada estejam disponíveis, ou quanto ao fluxo de controle, que define que uma tarefa só pode ser executada quando as tarefas das quais ela depende tenham sido concluídas. O caminho mais longo de um vértice inicial até um vértice final é chamado *caminho crítico*, em que o tempo mínimo de execução é obtido a partir da duração das tarefas nesse caminho. As demais tarefas, que se encontram fora desse caminho, são chamadas tarefas não-críticas.

### 3.3. Algoritmo de escalonamento HEFT

O algoritmo *Heterogeneous Earliest Finish Time* (HEFT) é utilizado para o escalonamento de aplicações modeladas como DAGs para um número limitado de computadores heterogêneos. Devido à sua simplicidade e baixa complexidade, é considerado um dos algoritmos mais populares para o escalonamento estático de workflows [Bittencourt and Madeira 2012].

O HEFT é um algoritmo baseado em listas de tarefas. Ele faz uso de dados históricos para prever o desempenho das tarefas nos recursos computacionais e, a partir do desempenho previsto, realizar o escalonamento. Ele associa tarefas a prioridades atribuindo um peso a cada vértice e arco no DAG, ordenando-as em uma lista, de modo que as tarefas de maior prioridade estão no início da lista e são escalonadas primeiro. O peso para cada tarefa consiste no custo médio de execução da tarefa em cada recurso computacional, enquanto o peso de cada arco compõe a média dos custos de comunicação entre as tarefas, ou seja, custos de transferência de dados. A atribuição das prioridades

utiliza a Equação 2 para ordenar as tarefas, calculando de maneira recursiva o $rank$ para cada tarefa, navegando pelo DAG de baixo para cima (da tarefa final para a tarefa inicial).

$$rank(t_i) = \overline{w_i} + \max_{t_j \in suc(t_i)} (\overline{c_{i,j}} + rank(t_j)) \quad (2)$$

onde $\overline{w_i}$ é a média do tempo de execução da tarefa $t_i$ sobre todos os recursos, $suc(t_i)$ são todas as tarefas sucessoras imediatas de $t_i$, $\overline{c_{i,j}}$ é a média dos custos de comunicação entre $t_i$ e $t_j$. Basicamente, $rank$ é o comprimento do caminho crítico de uma tarefa $t_i$ até a tarefa final do fluxo, incluindo o custo de computação da tarefa $t_i$. No caso de mais de um caminho possível até a saída, é escolhido o de peso máximo. Após essa etapa de construção da lista (fase de priorização), a tarefa pronta com peso mais alto é selecionada e associada ao recurso que tem o menor tempo de execução esperado para concluir essa tarefa (fase de seleção). O processo se repete até que todas as tarefas estejam escalonadas.

### 3.4. Simulador CloudSim-DVFS

O CloudSim-DVFS [Guérout et al. 2013] é uma extensão do simulador CloudSim, disponibilizada para download em outubro de 2013. Essa ferramenta permite simulações energéticas tanto de uma carga de trabalho quanto de workflows, além de fornecer uma nova versão da simulação da técnica de dimensionamento dinâmico de voltagem e frequência (*Dynamic voltage and frequency scaling* – DVFS). Essa técnica permite mudar dinamicamente a voltagem e a frequência de um processador de acordo com a carga de trabalho, resultando em um consumo de energia menor.

Para a execução de uma única carga de trabalho no simulador, há os mesmos modos descritos no kernel do Linux para definir o desempenho do processador: (i) *Performance*, que utiliza a CPU na frequência máxima; (ii) *PowerSave*, que utiliza a CPU na menor frequência; (iii) *User-Space*, que permite que o usuário defina uma frequência para a CPU dentre uma lista disponível; (iv) *Conservative* que utiliza um limite superior e um inferior para decidir mudanças na frequência da CPU – caso a carga de trabalho esteja acima do limite superior, a frequência é aumentada; no caso contrário, ela é reduzida – ; e (v) *OnDemand*, que utiliza um único limite – caso a carga de trabalho da CPU o ultrapasse, a frequência da CPU é aumentada para a máxima; caso a carga esteja abaixo desse limite por um determinado período de tempo, ela é reduzida.

A ferramenta utiliza para simulação do DVFS uma lista de tipos de VMs que podem ser instanciadas baseadas nos valores fornecidos pelo modelo energético adotado. Por padrão, é utilizado o modelo energético com os valores de consumo de energia dos *hosts* do Grid'5000 Reims[5], descritos na Tabela 1. Além disso, é alocada uma tarefa por VM.

Na simulação de workflows, o módulo permite o uso do escalonador DVFS em três modos: (i) *Performance*, onde a VM sempre estará configurada com a frequência máxima disponível; (ii) *OnDemand*, que permite que o sistema mude dinamicamente a frequência da CPU, atribuindo a frequência máxima quando a CPU está sendo utilizada e a mínima quando a CPU encontra-se no estado ocioso; e (iii) *Optimal*, que considera a existência de caminhos críticos e não-críticos no workflow em execução, atribuindo, assim, VMs com frequências máximas para tarefas no caminho crítico e, para tarefas nos

[5]`https://www.grid5000.fr/mediawiki/index.php/Reims:Hardware`

**Tabela 1. Frequências disponíveis no Grid'5000 REIMS**

| Frequências disponíveis (GHz) | CPU | 0.8 | 1.0 | 1.2 | 1.5 | 1.7 |
|---|---|---|---|---|---|---|
| Consumo(W) | Ociosa | 140 | 146 | 153 | 159 | 167 |
| | Carga Máxima | 228 | 238 | 249 | 260 | 272 |

demais caminhos, são atribuídas VMs com frequências ótimas, ou seja, frequências cujo menor consumo energético foi obtido considerando a razão entre o tempo restante para conclusão da tarefa (*slack-time*) e o tempo de execução necessário para a tarefa.

## 4. Algoritmos propostos

O escalonamento em nuvens computacionais pode ser dividido na alocação de tarefas em máquinas virtuais (VMs) e na alocação de máquinas virtuais em servidores físicos. Este trabalho foca no primeiro caso, propondo escalonar as tarefas de maneira estática com o uso de duas versões estendidas do HEFT, visando à economia de energia. Como não se tem previamente um número fixo de VMs, ambas as abordagens propõem a alocação de VMs dinamicamente baseando-se em uma lista com tipos de VMs fornecidas, com diferentes opções de processadores, mesmo método utilizado em [Guérout et al. 2013].

O primeiro algoritmo proposto baseia-se na variante Lookahead do HEFT [Bittencourt et al. 2010], em que o escalonamento de uma tarefa em uma VM leva em consideração também informações sobre o impacto dessa decisão nas tarefas dependentes da que está sendo alocada. No algoritmo proposto neste trabalho, chamado de PowerHEFTLookahead, a seleção da VM foi modificada, considerando a estimativa de consumo energético ao invés do tempo de conclusão da tarefa.

O segundo algoritmo proposto, chamado de HEFT-DynamicAllocationVM (HEFT-DAVM), modifica a seleção de VMs do HEFT baseando-se no tempo mínimo de início de uma tarefa de acordo com suas tarefas antecessoras e compara esse tempo mínimo com o menor tempo de início dessa mesma tarefa em cada VM disponível em uma dada lista, alocando sempre uma nova VM caso uma sobrecarga de tarefas seja detectada.

Além disso, a partir de uma abordagem descrita em [Weiwei and Deelman 2012], propomos o uso de um algoritmo simples para agrupamento de tarefas em VMs, objetivando uma melhor utilização dos servidores físicos, uma vez que o modelo adotado pelo simulador aloca apenas uma tarefa por VM. E, a fim de comparação do consumo energético com essa abordagem, a técnica de agrupamento é aplicada tanto ao algoritmo proposto em [Guérout et al. 2013] quanto ao algoritmo proposto HEFT-DAVM.

Desse modo, temos os seguintes algoritmos propostos:

- **PowerHEFTLookahead (Algoritmo 1):** inicialmente, há uma lista $V$ de VMs, com apenas uma VM alocada – a mais rápida disponível na lista $O$ de tipos de VMs que podem ser instanciadas. As tarefas são ordenadas conforme os critérios definidos pelo $rank$, descrito na Seção 3.3, e são alocadas em ordem decrescente, como é observado nas linhas 1-5 do algoritmo. Nas linhas 9-11, para cada tarefa $t$, o algoritmo EscalonarPowerHEFT é chamado. Este algoritmo, que não é apresentado por falta de espaço, força a alocação da tarefa $t$ em uma VM disponível e, em seguida, as tarefas sucessoras de $t$ são escalonadas seguindo os critérios do

HEFT original (apresentado na Seção 3.3), retornando o consumo energético parcial dessa alocação, com base no tempo de processamento da VM escolhida e no modelo linear descrito Seção 3.1. Assim, o Algoritmo 1 memoriza a VM que maximiza a economia de energia. Nas linhas 12-19 do Algoritmo 1, é verificado se seria mais vantajoso energeticamente alocar mais uma VM para o processamento, verificando cada opção de VM disponível. A linha 20 escalona a tarefa na VM que minimiza a energia consumida e a lista de VMs $V$ e o $rank$ são atualizados se necessário.

- **HEFT-DAVM (Algoritmo 2):** a alocação de apenas uma VM e a criação de uma lista ordenada de tarefas conforme o $rank$ definido pela Equação 2 é idêntica ao Algoritmo 1, diferenciando-se na seleção e alocação de novas VMs. Para cada tarefa a ser escalonada, na linha 7 é calculado o tempo mínimo de início de uma tarefa $t$, baseado no tempo de execução de suas tarefas antecessoras. Em seguida, nas linhas 8-11, o algoritmo HEFT é aplicado, selecionando a VM que tem o tempo mais cedo de conclusão e definindo o tempo de início efetivo de $t$ nessa VM, que depende das tarefas já alocadas nessa máquina. Sempre que for detectada uma sobrecarga nas máquinas, indicada pelo tempo mínimo de início e o tempo de início efetivo, é alocada uma nova VM, a mais lenta disponível (segundo a Tabela 1, a de 0.8 GHz). Espera-se que essa escolha garanta uma economia de energia sem prejudicar o *makespan*. Além disso, visando a redução de outras sobrecargas, como em casos em que uma VM do tipo mais lenta executa um caminho crítico, o que resulta em um *makespan* maior e, consequentemente, em um consumo de energia desnecessário dependendo do número de VMs que ficaram aguardando as tarefas desta serem concluídas, todas as VMs que foram selecionadas para executar mais de uma tarefa são "promovidas" a VMs do tipo mais rápida nas linhas 15-18, visando obter uma eficiência energética melhor. Esse modelo assume também que as VMs em que o escalonamento é executado pertencem a uma nuvem privada homogênea.
- **Task Clustering**: uma vez que todas as tarefas foram alocadas, uma heurística é aplicada. Tarefas que serão processadas em VMs do tipo mais lenta são agrupadas em uma nova VM, do tipo mais rápida. Essa técnica visa minimizar o número de VMs utilizadas sem impactar negativamente o *makespan*.

## 5. Análise de desempenho

A seguir serão descritos a metodologia adotada para as simulações, os resultados obtidos e a análise dos resultados.

### 5.1. Metodologia

Para avaliar os escalonadores propostos, foram utilizados três workflows reais descritos na linguagem de modelagem DAX (o Sipht, o CyberShake e o Montage) com estruturas distintas e com diversos números de tarefas. Esses workflows são fornecidos pelo sistema Pegasus[6] e, por retratarem diferentes características importantes para workflows, já foram usados como estudo de caso em vários trabalhos relacionados à área de gerenciamento de workflows científicos. Suas execuções foram simuladas no CloudSim-DVFS.

[6] `http://pegasus.isi.edu/workflow_gallery/`

**Algoritmo 1:** PowerHEFTLookahead()

1: //V é o conjunto de VMs usadas ao escalonar
2: //$VmMaisRápida$ no modelo descrito na Tabela 1 é a máquina de 1.7 GHz
3: $V$ = $\{VmMaisRápida\}$
4: $O$ = os tipos de VMs que podem ser instanciadas
5: Ordene o conjunto de tarefas de modo decrescente segundo o critério $rank$
6: **enquanto** há tarefas não escalonadas **faça**
7: $t$ = a tarefa não escalonada de maior $rank$
8: // Vamos tentar escalonar t em uma VM existente
9: **para** cada $v$ em $V$ **faça**
10: $energia$ = EscalonarPowerHEFT(t, v) //Retorna consumo energético do HEFT tradicional
11: **fim para**
12: **para** cada $o$ em $O$ **faça**
13: n = Simular criação de uma VM do tipo $o$;
14: $V = V \cup \{n\}$
15: Atualize os valores de $rank$
16: $t$ = a tarefa não escalonada de maior $rank$
17: $energia$ = EscalonarPowerHEFT(t,n)
18: Retorne $V$ e o escalonamento para os valores do começo do laço
19: **fim para**
20: Escalone $t$ na VM que minimiza a energia consumida
21: Atualize $V$ e $rank$ caso necessário
22: **fim enquanto**

**Algoritmo 2:** HEFT-DynamicAllocationVM()

1: Aloque uma VM do tipo {VMMaisRápida}
2: Defina a média dos custos computacionais das tarefas e dos custos de comunicação
3: Calcule $rank$ para todas as tarefas
4: Ordene as tarefas em uma lista de escalonamento utilizando
uma ordem decrescente de valores de $rank$
5: **enquanto** há tarefas não escalonadas na lista **faça**
6: Selecione a primeira tarefa, $t$ da lista de escalonamento
7: Calcule o tempo mínimo para execução da tarefa $t$
com base nas tarefas das quais $t$ dependa
8: **para** cada VM $m$ no conjunto de VM $(m \in P)$ **faça**
9: Calcule o tempo de início da tarefa e conclusão da tarefa $t$
considerando que ela execute em $m$
10: **fim para**
11: Defina o tempo mais cedo de conclusão da tarefa $t$
e o tempo de início na VM em que esse tempo foi obtido
12: **se** o tempo de início for maior que o tempo mínimo para execução de $t$ **então**
13: Aloque uma nova VM do tipo {VMMaisLenta}
14: **senão**
15: **se** a VM escolhida não é do tipo {VMMaisRápida} **então**
16: Aloque uma nova VM do tipo {VMMaisRápida}
17: Migre todas as tarefas da VM antiga
18: Defina a tarefa $t$ para executar na nova VM
19: **senão**
20: Defina a tarefa $t$ para ser executar na VM que
minimiza o tempo de conclusão dessa tarefa
21: **fim se**
22: **fim se**
23: **fim enquanto**

Os escalonadores avaliados foram: (i) modo Performance (PERF.); (ii) DVFS modo Optimal (OPT.); (iii) DVFS modo OnDemand (ONDEM.); (iv) HEFT; (v) PowerHEFTLookahead (POWERHEFT); (vi) HEFT-DAVM (DAVM); (vii) HEFT-DAVM-Task Clustering (DAVM-TC); (viii) DVFS modo Optimal com Task Clustering (OPT-TC). O número de VMs utilizadas para o HEFT foi gerado aleatoriamente, escolhendo-se um dos cinco tipos de VM adotados para as simulações, limitando o número de VMs ao número de tarefas de cada workflow. Os valores apresentados para esse caso são a média e o desvio padrão baseados em trinta simulações. As métricas utilizadas são o *makespan* (o tempo de término da última tarefa de cada workflow) em segundos e o consumo energético total em Watt-hora, fornecidos pelo simulador.

### 5.1.1. Workflows utilizados

A seguir são descritos os workflows utilizados nos experimentos:

- **Sipht:** utilizado em [Guérout et al. 2013] para validar o CloudSim-DVFS, esse workflow tem como objetivo automatizar a codificação de genes, sendo um projeto de bioinformática de Harvard, cujo exemplo de uma instância é apresentado na Figura 2a.
- **CyberShake:** esse workflow é utilizado pelo Centro de Terremoto do Sul da Califórnia para caracterizar os riscos de terremotos em uma região. Sua estrutura é representada na Figura 2b.
- **Montage:** é uma aplicação criada pela NASA/IPAC, cuja estrutura é ilustrada na Figura 2c, que une diversas imagens de entrada para criar um mosaico personalizado do céu.

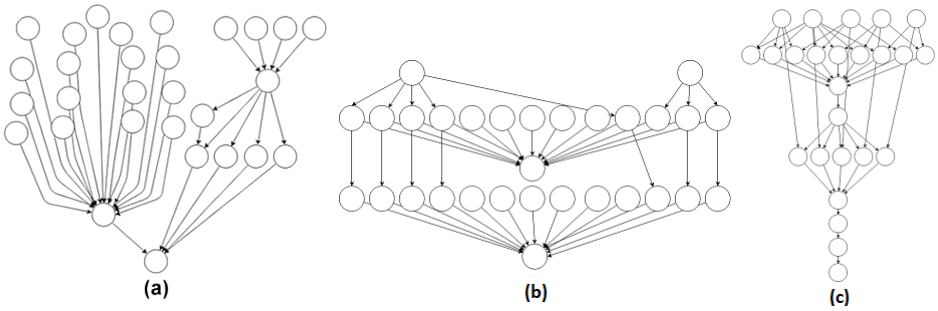



**Figura 2. Workflows usados nos experimentos: (a) Sipht, (b) Cybershake e (c) Montage.**

### 5.1.2. Configurações do CloudSim-DVFS

Todas as configurações são definidas no arquivo `simulation.properties`, disponível no simulador. As configurações utilizadas nos experimentos foram as seguintes:

- ***Data center*:** foram utilizadas as mesmas configurações apresentadas em [Guérout et al. 2013]. Ou seja, o *data center* foi configurado com 500 servidores com o mesmo modelo energético do Grid'5000 REIMS. Esse modelo está incorporado ao pacote power do CloudSim na classe

`PowerModelSpecPower_REIMS`. Para os experimentos, foi considerado que cada servidor possuía 8 núcleos com 1000 MIPS cada.

- **Tipos de VMs utilizadas**: foram utilizados cinco tipos de VMs, com requisitos de 941.2 MIPS, 1176.4 MIPS, 1411.8 MIPS e 2000 MIPS, descritas na classe `VMOffersSimple` do pacote `workflow` do simulador. Cada VM instanciada possuía um único núcleo.
- **Escalonadores:** os escalonadores são alternados no arquivo `simulation.properties`, adicionando-se a referência para a variável `scheduling.policy`. O escalonador HEFT foi implementado no simulador a fim de comparar seu desempenho com as adaptações propostas. A técnica de agrupamento de tarefas (*Task Clustering*) foi aplicada ao HEFT-DAVM e ao OPT estendendo cada classe.
- **Workflows:** a configuração do workflow utilizado é definida na propriedade `dag.file`. Para os experimentos realizados, foram adotados os modelos: (i) Sipht_30.xml (30 tarefas), Sipht_60.xml (60 tarefas), Sipht_100.xml (100 tarefas), (ii) Montage_25.xml (25 tarefas), Montage_50.xml (50 tarefas), Montage_100.xml(100 tarefas) e (iii) CyberShake_30.xml (30 tarefas), CyberShake_50.xml (50 tarefas), CyberShake_100.xml (100 tarefas).

Todos os códigos, dados e documentação dos experimentos realizados estão disponíveis em `https://github.com/pedropaulovc/CloudSim_DVFS`. O objetivo de manter essas informações públicas é permitir a reprodução dos experimentos futuramente por outros pesquisadores.

### 5.2. Resultados e discussões

A Figura 3 ilustra os resultados obtidos para o DAG Sipht. Adotando a mesma comparação feita em [Guérout et al. 2013] utilizando o modo Performance como referência, é possível observar que todos os escalonadores obtiveram valores menores de consumo energético para um DAG com 30 tarefas. Ao realizar o mesmo experimento com mais tarefas, é possível observar que a abordagem de agrupamento de tarefas, DAVM-TC e OPT-TC, mostrou-se vantajosa, superando a eficiência energética e o *makespan* obtidos pelas versões sem essa técnica (DAVM e OPT). É importante observar que apesar do modo ONDEM ter sido o mais eficiente energeticamente e em relação ao *makespan*, os algoritmos propostos (POWERHEFT, DAVM e DAVM-TC) apresentaram reduções no consumo de energia em todos os cenários quando comparados com o algoritmo HEFT original que foi usado como base para as suas construções. Além disso, em termos de *makespan*, com exceção do algoritmo POWERHEFT, não houve perdas significativas. Em média os *makespans* dos algoritmos DAVM e DAVM-TC ficaram apenas 2,85% acima dos valores de *makespan* obtidos com o HEFT original. Os bons resultados obtidos com os métodos que utilizam DVFS apontam que uma direção interessante seria integrar os mecanismos de DVFS aos nossos algoritmos propostos. Os resultados obtidos com a simulação do DAG CyberShake foram similares àqueles encontrados com o DAG Sipht.

Os experimentos com o DAG Montage, um DAG caracterizado por ter caminhos mais longos entre a tarefa de entrada e a tarefa de saída, diferiram dos resultados anteriores. O escalonador HEFT e as nossas propostas baseadas nele (com exceção do POWERHEFT) mostraram-se mais eficientes, até mesmo que o modo ONDEM, tanto energeticamente quanto em relação ao *makespan*. As médias dos ganhos de economia

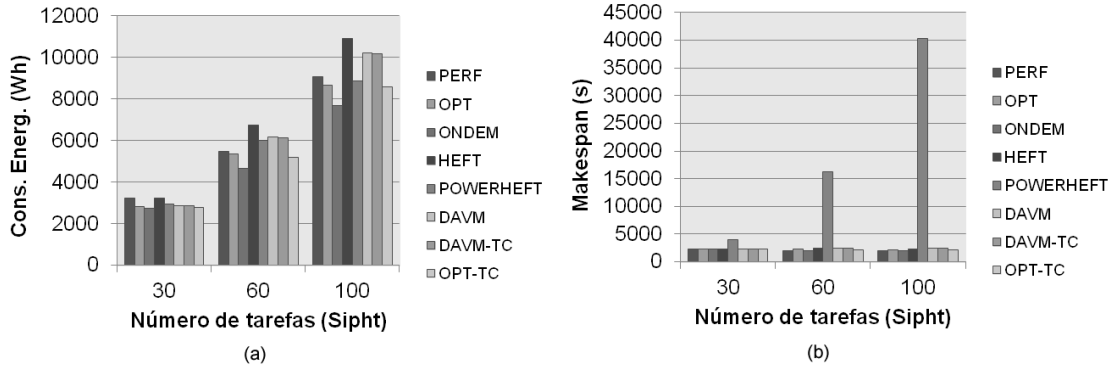



**Figura 3. Comparativo do (a) consumo energético em Watt-hora e (b) *makespan* em segundos do DAG Sipht utilizando diferentes políticas de escalonamento de tarefas.**

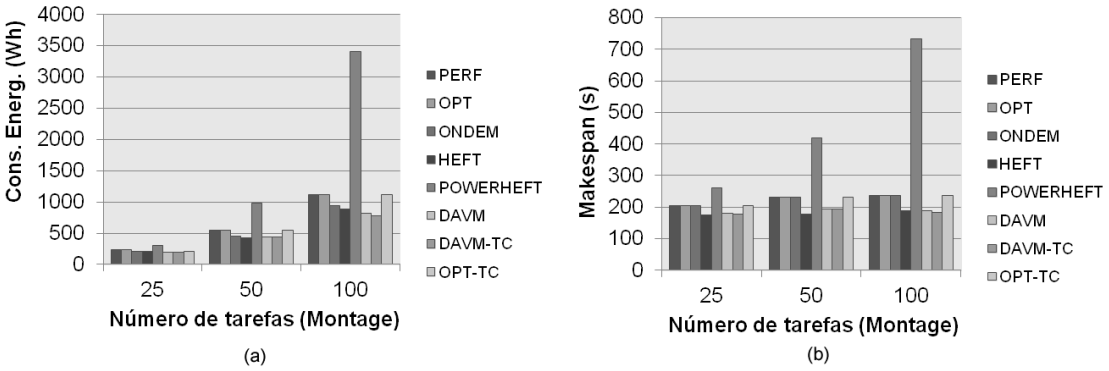



**Figura 4. Comparativo do (a) consumo energético em Watt-hora e do (b) *makespan* em segundos do DAG Montage utilizando diferentes políticas de escalonamento de tarefas.**

energética obtidos, superiores ao modo PERF, foram: 15,5% (ONDEM), 18,9% (HEFT), 21,1% (DAVM), 22,7% (DAVM-TC) e 5% (OPT-TC). Em relação ao *makespan*, apenas o HEFT, o DAVM e o DAVM-TC apresentaram reduções em relação ao PERF: 19,3%, 15,9% e 17,2%, respectivamente. A técnica de agrupamento de tarefas (TC) não obteve desempenho significativo nos resultados mostrados pela Figura 4.

### 5.3. Discussão e análises

Do ponto de vista energético, o algoritmo POWERHEFT obteve um ganho médio superior ao modo PERF apenas para o workflow Sipht, contudo, foi inferior aos ganhos médios obtidos pelos algoritmos DVFS nos modos Opt e OnDemand. A técnica de agrupamento de tarefas mostrou-se interessante tanto em relação ao consumo energético quanto ao *makespan*, obtendo, na maioria dos casos, um ganho superior ao do algoritmo original, como no caso do DAVM-TC em relação ao DAVM e do OPT-TC em relação ao OPT. O modo ONDEM obteve melhor desempenho na maioria dos casos, como é descrito por [Guérout et al. 2013]. Entretanto, foi possível observar que nem sempre ele é a melhor escolha, uma vez que na execução do workflow Montage, os melhores ganhos médios obtidos em relação à economia de energia foram dos algoritmos DAVM e DAVM-TC, propostos neste trabalho, apresentando ganhos consideráveis em relação também ao

*makespan*. Neste mesmo caso, o modo OPT não obteve um ganho significativo uma vez que o Montage não possui caminhos não-críticos, não permitindo a otimização proposta pelo algoritmo.

Assim, como pode ser observado pelos resultados das simulações, os algoritmos propostos conseguem na maioria dos casos melhorar o consumo de energia em relação ao escalonador HEFT original, sem perdas significativas em termos de *makespan*, e em alguns casos conseguem apresentar os melhores ganhos quando comparados com todos os outros escalonadores avaliados. É importante observar que o tamanho do caminho entre as tarefas de entrada e de saída dos DAGs tem influência no desempenho dos escalonadores.

## 6. Considerações finais e trabalhos futuros

Nuvens computacionais exibem um grande potencial para execução de workflows. Contudo, se o escalonamento de suas tarefas nesse ambiente não considerar o consumo energético, pode resultar na baixa utilização dos recursos disponíveis e, consequentemente, em um maior impacto ambiental, uma vez que os *data centers* já são responsáveis por 2% das emissões de $CO_2$ no mundo.

Neste artigo, foram propostos três algoritmos para escalonamento de workflows baseados no HEFT, visando a eficiência energética sem impacto significativo no *makespan* das aplicações. O uso de um simulador permitiu avaliar os escalonadores com workflows reais de maneira controlável e facilmente reproduzível. Os resultados obtidos a partir das simulações com diversos workflows, comparados a escalonadores existentes na ferramenta utilizada, permitiram avaliar o desempenho dessas novas abordagens, conseguindo-se obter a redução do consumo energético na maioria dos casos. Foi possível também avaliar os ganhos obtidos com o agrupamento de tarefas, que se mostrou vantajosa tanto em relação ao consumo energético quanto ao tempo de execução (*makespan*). Além disso, os experimentos comprovaram que a eficiência dos escalonadores para workflows depende da estrutura de interligações das tarefas.

Como trabalho futuro, podem ser incorporados: (i) a simulação desses mesmos experimentos com diferentes servidores e modelos energéticos; (ii) a proposta de novas abordagens com o HEFT; (iii) a validação dos resultados em ambientes reais; (iv) a integração dos modos de operação do DVFS com os nossos algoritmos propostos; (v) a implementação no simulador CloudSim-DVFS de simulações de falhas e de diferentes cargas de trabalho por servidor, como descritos no WorkflowSim; (vi) a avaliação de escalonadores dinâmicos ao invés de estáticos; e (vii) a incorporação de estimativa de consumo energético de outros componentes dos servidores, como no uso de servidores para armazenamento de dados.

## Agradecimentos

Os autores agradecem à CAPES e à FAPESP (processo número: 2011/24114-0) pelo apoio financeiro, ao Rodrigo N. Calheiros e ao Tom Guérout por disponibilizarem o código do simulador CloudSim-DVFS, ao Weiwei Chen pelo suporte ao WorkflowSim e ao Daniel A. Cordeiro pelos esclarecimentos sobre algoritmos escalonadores de tarefas.

## Referências

Aksanli, B., Venkatesh, J., and Rosing, T. (2012). Using datacenter simulation to evaluate green energy integration. *Computer*, 45(9):56–64.

Barroso, L. and Hölzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.

Belady, C. (2007). In the data center, power and cooling costs more than the IT equipment it supports. http://www.electronics-cooling.com/2007/02/in-the-data-center-power-and-cooling-costs-more-than-the-it-equipment-it-supports/ [Online; acessado em 07 de outubro de 2013].

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. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 826–831.

Bittencourt, L. F. and Madeira, E. (2012). Escalonamento de workflows em condições de incerteza. *Anais do WCGA 2012 - X Workshop em Clouds,Grids e Aplicações*.

Bittencourt, L. F., Madeira, E. R., and Da Fonseca, N. L. (2012). Scheduling in hybrid clouds. *Communications Magazine, IEEE*, 50(9):42–47.

Bittencourt, L. F., Sakellariou, R., and Madeira, E. R. (2010). Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In *Parallel, Distributed and Network-Based Processing (PDP), 18th Euromicro International Conference on 2010*, pages 27–34. IEEE.

Cordeiro, D., Braghetto, K. R., Goldman, A., and Kon, F. (2013). Da ciência à e-ciência: paradigmas da descoberta do conhecimento. *Computação em nuvem*, (97):71–80A.

Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2009). Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540.

Guérout, T., Monteil, T., Da Costa, G., Neves Calheiros, R., Buyya, R., and Alexandru, M. (2013). Energy-aware simulation with DVFS. *Simulation Modelling Practice and Theory*, 39(0):76 – 91.

Kliazovich, D., Bouvry, P., and Khan, S. U. (2012). GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283.

Rivoire, S., Ranganathan, P., and Kozyrakis, C. (2008). A comparison of high-level full-system power models. *HotPower*, 8:3–3.

Topcuoglu, H., Hariri, S., and Wu, M.-y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *Parallel and Distributed Systems, IEEE Transactions on*, 13(3):260–274.

Vöckler, J.-S., Juve, G., Deelman, E., Rynge, M., and Berriman, B. (2011). Experiences using cloud computing for a scientific workflow application. In *Proceedings of the 2nd international workshop on Scientific cloud computing*, pages 15–24. ACM.

Weiwei, C. and Deelman, E. (2012). WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In *IEEE 8th International Conference on E-Science (e-Science)*, pages 1–8.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 2*

## *Redes e Sistemas P2P*

# Sistemas de Recomendação de Conteúdo e Seus Impactos no Custo e Desempenho de Redes Par-a-Par

**Diogo Munaro**[1], **Carla Delgado**[1], **Daniel S. Menasché**[1]

[1]Departamento de Ciência da Computação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ – Brazil

***Abstract.*** *Recommendation systems and the performance of computer network systems have fundamental implications over each other. While recommendation systems impact system performance, the latter can be used to guide the former. In this paper, we study the interconnections between recommendation systems and the performance of the network. We propose an analytical model to capture the revenue and the cost to a content provider as a function of the quality of its recommendations and the cost to serve the content. The model is then used to suggest heuristics on how to recommend content accounting for service costs and user preferences.*

***Resumo.*** *Sistemas de recomendação de conteúdo e o desempenho dos sistemas de redes usados para prover tais conteúdos têm implicações fundamentais uns sobre os outros. Os sistemas de recomendação impactam a disponibilidade de conteúdo na rede, e por consequência o desempenho da disseminação de conteúdo. Por outro lado, a disponibilidade de conteúdo pode ser usada para guiar o sistema de conteúdo, tendo em vista que conteúdos mais populares irão estar associados a maior QoS e por consequência maior satisfação do usuário. Nesse artigo, é proposto um modelo para estudar os tradeoffs envolvidos na recomendação de conteúdo e nos custos de redes Par-a-Par (P2P). Usando o modelo, foram derivadas heurísticas para recomendar conteúdo que levem em conta não só as preferências dos usuários mas também os custos para servi-los.*

## 1. Introdução

Sistemas de recomendação e de distribuição de conteúdo estão cada vez mais presentes no dia-a-dia de milhões de usuários na Internet. Embora tais sistemas tenham impacto direto no desempenho do sistema de distribuição de conteúdo e na própria disponibilidade do conteúdo, a relação entre os dois é ainda muito pouco compreendida. Entender esta relação torna-se cada vez mais importante, dada a prevalência das redes par-a-par (P2P).

A tecnologia P2P é aplicada em vários ambientes, desde AVoD (Áudio e Video sob Demanda) [Wu and Lui 2012, Zhou et al. 2012] até ambientes de colaboração e troca de arquivos [Guarnieri et al. 2013, Guarnieri et al. 2014, Zhang et al. 2012, Trestian et al. 2014]. Para todos esses ramos da tecnologia P2P já existe uma série de algoritmos de recomendação visando maior comodidade do usuário em encontrar um melhor conteúdo de acordo com seu perfil [Hecht et al. 2012]. Mesmo com a comprovação de que os mecanismos de recomendação possuem grande influência na decisão de consumo - principalmente multimídia (em média 30% das visualizações de cada vídeo do

YouTube é oriunda de recomendação [Zhou et al. 2010]) - nenhum desses algoritmos de recomendação foi até então usado para influenciar a QoS fornecida pela rede.

Nesse trabalho são propostas heurísticas para recomendação de conteúdo que possam melhorar a disseminação de conteúdo da rede P2P, além de priorizar os aspectos comuns de *precision and recall* para manter a qualidade das recomendações. Para tal, é avaliado como o sistema de recomendação (SR) pode fazer uso de métricas de qualidade de serviço (QoS) para sugerir conteúdo visando maximizar a satisfação do usuário. Reciprocamente, também foi estudado como o SR pode ser usado para aumentar a QoS experimentada por usuários que requerem conteúdo que, *a priori*, seria pouco replicado.

O objetivo desse trabalho é responder duas grandes questões:

1. A satisfação do usuário ao consumir um conteúdo recomendado está relacionada ao conteúdo em si e também à qualidade com que este lhe é servido. Entrando no contexto multimídia, a exigência da QoS de alta qualidade aumenta, por ter a necessidade de transmissão sob demanda sequencial do conteúdo. Tendo isso como base, **como o sistema de recomendação de conteúdo pode fazer uso das métricas de redes para aumentar a satisfação do usuário?**
2. Classicamente, assume-se que a demanda de tráfego por cada conteúdo na rede é uma variável exógena (fora do controle do sistema). Porém, se considerarmos a recomendação como parte do sistema, essa prerrogativa não é mais válida. Sendo assim, **como o administrador de rede pode fazer uso do sistema de recomendação de conteúdo para aumentar o desempenho da rede e diminuir os custos de distribuição?**

Para responder essas perguntas, foi proposto um modelo analítico relacionando as recomendações de conteúdo e a disseminação de conteúdo pela rede. A partir deste modelo, heurísticas para a confecção de um sistema de recomendação foram propostas. Dados do banco de avaliações do Movielens foram utilizados para testar o modelo. O custo de disseminação de conteúdo está sendo utilizado como uma alternativa para uma medição simplificada da QoS.

As principais contribuições do artigo são:

(a) A modelagem dos *tradeoffs* envolvidos na recomendação de conteúdo levando em conta atributos de desempenho e custo da rede;
(b) O uso do modelo proposto para derivar heurísticas a fim de recomendar conteúdos de forma a satisfazer os usuários e ao mesmo tempo reduzir os custos da rede.

O restante deste artigo está organizado da seguinte forma. Na seção 2 são introduzidos os conceitos básicos de SR e redes P2P, além dos trabalhos relacionados. Na seção 3 é apresentado o modelo proposto que unifica sistemas de recomendação e de distribuição de conteúdo em redes. Na seção 4 o modelo é parametrizado. Em seguida, o modelo é usado para propor, na seção 5, heurísticas para *design* de um sistema de recomendação. Na seção 6 é apresentada uma metodologia alternativa, baseada em algoritmos genéticos, para determinar parâmetros para um sistema de recomendação. Resultados experimentais são apresentados na seção 7 e a seção 8 conclui o trabalho.

## 2. Trabalhos Relacionados

Os pilares do presente trabalho são redes P2P [Mendonça and Leão 2012] e sistemas de recomendação de conteúdo [Mei et al. 2007, Chakoo et al. 2008]. Embora a literatura

nessas duas áreas seja vasta, não é de nosso conhecimento nenhum trabalho sobre os possíveis benefícios de uma abordagem integrada para estudo desses dois tópicos.

Os mecanismos de recomendação de conteúdo visam automatizar a filtragem de dados buscados por um usuário a partir dos dados desse mesmo usuário ou de outros similares que o sistema já conheça. Automatizar essa detecção de preferências nunca se mostrou uma tarefa simples, pois as pessoas possuem necessidades e desejos inesperados, que - mesmo demorando - podem mudar com o tempo [Chakoo et al. 2008, Adomavicius and Tuzhilin 2005]. Os mecanismos de recomendação mais populares são o Filtro Baseado em Conteúdo e o Filtro Colaborativo, que normalmente são usados combinados com outras técnicas [Adomavicius and Tuzhilin 2005].

Redes P2P possuem arquitetura distribuída e são bastante vantajosas em termos de economia de recursos por dividir a carga utilizando seus *peers*. No escopo P2P, alguns trabalhos utilizam AVoD P2P [Zhou et al. 2012, Wu and Lui 2012] com o protocolo BitTorrent. O BitTorrent é um dos protocolos mais usados para transferência de arquivos, principalmente arquivos multimídia [Mendonça and Leão 2012].

Arquivos multimídia estão em foco também na área de recomendação [Mei et al. 2007, Chakoo et al. 2008], principalmente depois dos prêmios oferecidos pelo Netflix para melhorar a recomendação do seu sistema AVoD. Um exemplo recente de recomendação multimídia em sistemas P2P é o Radiommender [Hecht et al. 2012], que propôs um SR de músicas distribuído em que o *peer* possui sua rádio personalizada. Mesmo com toda essa inovação, nenhum desses trabalhos tentou melhorar a QoS da rede utilizando recomendação. A recomendação geralmente é voltada somente para a interação entre usuários e conteúdo, não levando em conta a rede.

**A proposta aqui apresentada de recomendação de conteúdo visa mitigar os custos dos servidores, recomendando conteúdo de tal forma a satisfazer o usuário e ao mesmo tempo reduzir a sobrecarga nos servidores.** Uma versão preliminar deste trabalho foi apresentada em [Vieira et al. 2013]. Este trabalho estende [Vieira et al. 2013] de várias formas, incluindo: *(a)* a proposição de um modelo analítico para estudar sistemas de recomendação e suas conexões com sistemas de distribuição de conteúdo; *(b)* o uso do modelo para derivar heurísticas de recomendação de conteúdo e *(c)* a aplicação de algoritmos genéticos para busca do sistema de recomendação ótimo.

## 3. Modelo

Nesta seção é apresentado o modelo analítico para estudo dos *tradeoffs* envolvidos na recomendação de conteúdo levando em conta os custos e desempenho da rede. Primeiro, foram apresentados os parâmetros do modelo, as variáveis de controle e a metodologia para parametrização do modelo em função de *traces*. Em seguida, são introduzidas as funções de custo e recompensa, e é formulado o problema central tratado neste artigo.

### 3.1. Parâmetros e Variáveis de Controle

**Parâmetros do sistema** Os parâmetros do sistema são dados coletados a partir de *traces* de um sistema já existente. A seguir, são descritos os quatro principais parâmetros considerados neste artigo. Consideram-se $N$ arquivos (conteúdos) disseminados por meio de uma rede. Seja $\lambda_i$ ($i = 1, 2, \ldots, N$) a taxa de chegada de visualizações para o conteúdo $i$. Essa taxa de chegada é também referenciada como a popularidade do arquivo $i$ (vide

Tabela 1). Seja $r_{ki}$ a probabilidade de um usuário que assistiu o conteúdo $k$ assistir o conteúdo $i$. O parâmetro $r_{ki}$ depende da relação entre os conteúdos $k$ e $i$, e pode ser parametrizado via *traces* como explicado na seção 4.

**Variáveis de controle** Considera-se $p_{ki}$ a probabilidade do sistema de recomendação de conteúdo recomendar o conteúdo $i$ para um usuário que requisitou e acessou previamente o conteúdo $k$, $k = 1, \ldots, N$, $i = 1, \ldots, N$. Deste ponto em diante, assumimos $p_{ii} = 0$. O principal objetivo deste artigo consiste na determinação de $p_{ki}$ de tal forma a satisfazer o usuário e ao mesmo tempo levando em conta as características de rede. Assume-se $p_{ki}$ como variável de controle, pois é possível manipular essa probabilidade para mudar a recomendação padrão que o usuário recebe.

Seja $\gamma_i$ a taxa de chegada perturbada para o conteúdo $i$, ou seja, a taxa levando em conta o impacto do sistema de recomendação. A taxa $\gamma_i$ é dada, em função dos parâmetros do sistema, por

$$\gamma_i = \sum_{k=1}^{N} \lambda_k p_{ki} \tag{1}$$

Note que ao considerar a taxa de chegada perturbada ao conteúdo $i$, $\gamma_i$, leva-se em conta apenas as requisições provenientes de recomendação. Em particular, os usuários não trazem nenhum arquivo ao entrarem no sistema. O histórico de requisições não corresponde aos arquivos que o usuário disponibiliza para o sistema. O usuário contribui apenas para a popularidade do arquivo $i$ a ele recomendado. O modelo pode facilmente ser adaptado para o cenário em que os usuários também trazem os arquivos por eles previamente solicitados, adicionando-se $\lambda_i$ ao lado direito de (1).

### 3.2. Recompensas do Sistema de Recomendação de Conteúdo

A recompensa desse modelo representa o quão satisfeito o usuário fica ao aceitar uma recomendação. Essa recomendação pode ser obtida pelo próprio sistema de recomendação, conforme descrito na seção 3.1.

Dadas as recompensas individuais dos pares de filmes, a recompensa total do sistema de recomendação de conteúdo, $R$, é definida como

$$R = \sum_{k=1}^{N} \sum_{i=1}^{N} \lambda_k p_{ki} r_{ki} \tag{2}$$

### 3.3. Custos do Sistema de Distribuição de Conteúdo

Nessa seção caracterizamos o custo de disseminação de um conteúdo por um provedor, conforme proposto inicialmente por [Menasche et al. 2009].

Um *swarm* $k$ de um sistema P2P é modelado como um sistema auto-escalável. Em sistemas P2P, arquivos pouco populares dependem do servidor para sua disseminação, enquanto que arquivos muito populares são auto-sustentáveis entre os *peers*, e não requerem sobrecarga no servidor para sua disseminação (vide seção 2). Sendo $\mu$ a capacidade dos *peers* e a taxa de chegada $\lambda_k$, assume-se, como caso extremo, que o servidor só irá servir *peers* caso haja exatamente um *peer* no *swarm*.

| Variável | Descrição |
|---|---|
| $N$ | número de arquivos |
| $\lambda_i$ | popularidade do conteúdo $i$ normalizada |
| $\Lambda$ | taxa agregada de chegada de usuários ao sistema |
| $r_{ki}$ | recompensa por recomendar-se $i$ para requisitante de $k$ |
| $p_{ki}$ | probabilidade de recomendar-se conteúdo $i$ para requisitante do conteúdo $k$ |
| $R$ | recompensa total |
| $C$ | custo total |
| $P$ | custo total perturbado pelo sistema de recomendação |

**Tabela 1. Tabela de notação: há 4 parâmetros no modelo proposto, 1 variável de controle referente ao sistema de recomendação e 3 métricas de interesse.**

Seja $M_k$ o número de usuários no *swarm* $k$. A probabilidade de haver exatamente um usuário no *swarm* $k$, modelado como uma fila $M/M/\infty$, é dada por $P(M_k = 1) = (\lambda_k/\mu)e^{-\lambda_k/\mu}$. O custo para servir cada *swarm* é proporcional a fração de tempo em que o servidor fica ocupado.

$$C_k(\lambda_k) = (\lambda_k/\mu)e^{-\lambda_k/\mu}, \qquad k = 1, \ldots, N \tag{3}$$

O custo total do sistema, para todos os *swarms* (conteúdos) sem influência do sistema de recomendação é

$$C = \sum_{k=1}^{N} (\lambda_k/\mu)e^{-\lambda_k/\mu} \tag{4}$$

O custo varia em função de $\lambda_k$ e $\mu$. No restante deste artigo adota-se $\mu = 1$. Por sua vez, o custo total perturbado pelo sistema de recomendação é

$$P = \sum_{k=1}^{N} \gamma_k e^{-\gamma_k} \tag{5}$$

O custo total perturbado é obtido a partir de (3), substituindo-se $\lambda_k$ por $\gamma_k$, vide (5). Esta transformação captura o fato de que o SR afeta a taxa de chegada dos usuário ao *swarm* $k$, que deixa de ser $\lambda_k$ e passa a ser $\gamma_k$, vide (1).

### 3.4. Formulação do Problema

O problema consiste em recomendar os melhores conteúdos de acordo com os interesses dos usuários e com o menor custo de disseminação possível. Nesse modelo, isso corresponde a determinar $p_{ki}$ ($k = 1, \ldots, N$, e $i = 1, \ldots, N$), de tal forma a minimizar os custos e maximizar as recompensas por recomendação. A seguir, serão apresentadas estratégias para lidar com esta tensão entre objetivos conflitantes, usando heurísticas e algoritmos genéticos.

## 4. Parametrização do Sistema de Recomendação de Conteúdo via Traces

Será apresentada uma abordagem simples para parametrizar $r_{ki}$, ou seja, o quanto que um usuário que gostou do filme $k$ irá gostar do filme $i$ (vide seção 3.1).

Para tal, assume-se que tem-se em mãos um *trace* indicando, para cada usuário, os filmes que este já assistiu e o número de estrelas que o usuário atribuiu a cada filme. Tal informação está disponível sobre os usuários do sistema Movielens [Movielens 2013], que contém não apenas as avaliações dos usuários mas também os instantes em que elas ocorreram. O *trace* do Movielens possui mais de 100,000 avaliações de filmes, sendo que neste artigo amostramos de forma uniforme e aleatória 1,000 delas, resultando em 551 filmes.

Considera-se um *trace* no qual $U$ usuários deram notas (*ratings*) a $N$ filmes. Cada *rating* é um número de estrelas, entre 1 e 5. Seja $Nrating_i$ o número de *ratings* dados ao filme $i$. Na amostra de 551 filmes considerada, $Nrating_i$ variou de 1 a 7, $i = 1, 2, \ldots, 551$. O número de filmes para os quais $Nrating_i$ é igual a 1, 2, ..., 7 foi 301, 143, 58, 22, 17, 4 e 6 respectivamente.

Normalizando o número de avaliações a um filme $k$ pelo total de avaliações, obtêm-se a taxa de chegada (popularidade) do filme $k$, $\lambda_k$,

$$\lambda_k = \Lambda \left( \mathsf{Nrating}_k / \sum_{i=1}^{N} \mathsf{Nrating}_i \right), \quad k = 1, \ldots, N \tag{6}$$

sendo $\Lambda$ a taxa agregada de chegada de usuários ao sistema. Na seção 7, será investigado o efeito de $\Lambda$ nas diferentes métricas de interesse.

Seja $rating_{uki}$ o *rating* do usuário $u$ ao filme $i$, para um usuário $u$ que tenha feito o trajeto do conteúdo $k$ para o conteúdo $i$, ou seja, que avaliou o conteúdo $k$ e posteriormente o conteúdo $i$. Caso o usuário $u$ não tenha feito o trajeto do conteúdo $k$ para o $i$, $rating_{uki} = 0$. Seja $Nrating_{ki}$ a quantidade de usuários que fizeram o trajeto do conteúdo $k$ para o conteúdo $i$.

A partir do *trace* fornecido, calculam-se os valores de $r_{ki}$ tomando como inspiração o modelo do PageRank [Page et al. 1999],

$$r_{ki} = \left[ \left( \frac{\sum_{u=1}^{U} \mathsf{rating}_{uki}}{\mathsf{Nrating}_{ki}} \right) \Big/ \left( \sum_{i=1}^{N} \frac{\sum_{u=1}^{U} \mathsf{rating}_{uki}}{\mathsf{Nrating}_{ki}} \right) \right] \beta + \frac{(1-\beta)1_{k \neq i}}{N-1}, \quad (k,i) \in \{1, \ldots, N\}^2 \tag{7}$$

onde $1_{ki \neq i} = 1$ caso $k \neq i$ e 0 caso contrário.

O primeiro termo entre parêntesis na eq. (7) corresponde ao *rating* médio dado ao conteúdo $i$ pelos usuários $u$ que assistiram $k$ e depois $i$, enquanto o segundo termo é um fator de normalização. O segundo termo é o somatório, para todos os arquivos $i$, $i = 1, \ldots, N$, do *rating* médio dado ao conteúdo $i$ pelos usuários que assistiram $k$. Note que $0 \leq r_{ki} \leq 1$.

Assim como no PageRank, foi assumido um fator de atenuação $\beta$. Nesse artigo, consideramos $\beta = 1$. Esse fator reflete a probabilidade do usuário continuar vendo e avaliando conteúdos, ao invés de simplesmente escolher assistir um filme aleatoriamente. Em trabalhos futuros iremos considerar $\beta < 1$ para capturar o fato de que, mesmo que nenhum usuário tenha assistido um determinado par de filmes, a recomendação mútua possa ser positiva.

## 5. Heurísticas para Recomendação de Conteúdo

Nesta seção são propostas heurísticas para a determinação de $p_{ki}$, $(k,i) \in \{1,\ldots,N\}^2$. As heurísticas são motivadas pelo modelo apresentado na última seção. Sejam $\hat{\lambda}$ e $\hat{r}$ dois limiares (*thresholds*) de popularidade e recompensa, respectivamente. Considera-se que um conteúdo $k$ é popular se $\lambda_k > \hat{\lambda}$ e se a recomendação do conteúdo $i$ para um usuário que requisitou $k$ gera alta recompensa se $r_{ki} > \hat{r}$. Então, as heurísticas são baseadas nas seguintes duas regras simples:

1. **se os usuários que assistem $k$ gostam de $i$ (alta recompensa de recomendação), e $i$ é popular (baixo custo de disseminação), recomendar** $i$: se $\hat{\lambda} < \lambda_i$ e $\hat{r} < r_{ki}$, então o conteúdo $i$ tem alta demanda e há alta recompensa em se recomendar o conteúdo $i$ para usuários que requisitam $k$, logo favorecemos valores de $p_{ki}$ mais elevados;
2. **se os usuários que assistem $k$ não gostam de $i$ (baixa recompensa de recomendação), e $i$ é popular (já possui baixo custo de disseminação, logo não precisa de *boost*), não recomendar** $i$: se $\hat{\lambda} < \lambda_i$ e $r_{ki} < \hat{r}$, então $i$ tem alta demanda (seu custo de disseminação é baixo) e há baixa recompensa e baixa redução de custos ao se recomendar o conteúdo $i$ para usuários que requisitam $k$, então favorecemos valores de $p_{ki}$ mais moderados;

Dado um conteúdo $k$, seja $n_k^+$ o número de conteúdos classificados como de alta prioridade de recomendação (regra 1), $n_k^-$ o número de conteúdos classificados como de baixa prioridade de recomendação (regra 2), e $n_k^o$ os demais, $n_k^- + n_k^+ + n_k^o = N$. Analogamente, dado um conteúdo $k$, sejam $\mathcal{N}_k^+$, $\mathcal{N}_k^-$ e $\mathcal{N}_k^o$ os respectivos conjuntos de conteúdos de prioridade de recomendação alta, baixa e neutra.

- $\mathcal{N}_k^+ = \{j \in \mathbb{N} \mid \hat{\lambda} < \lambda_j \wedge \hat{r} < r_{kj}\}, \quad n_k^+ = |\mathcal{N}_k^+|$
- $\mathcal{N}_k^- = \{j \in \mathbb{N} \mid \hat{\lambda} < \lambda_j \wedge \hat{r} > r_{kj}\}, \quad n_k^- = |\mathcal{N}_k^-|$
- $\mathcal{N}_k^o = \{j \in \mathbb{N} \mid j \notin \mathcal{N}_k^+ \cup \mathcal{N}_k^-\}, \quad n_k^o = |\mathcal{N}_k^o|$

A seguir, consideramos três diferentes heurísticas para recomendação de conteúdo construídas usando os conjuntos $\mathcal{N}_k^+$, $\mathcal{N}_k^-$ e $\mathcal{N}_k^o$.

### 5.1. Heurística 1: recomendação uniforme de conteúdos de prioridade alta e neutra

Segundo esta heurística, os conteúdos de prioridade alta e neutra são recomendados de forma aleatória uniformemente. Assim,

$$p_{ki} = \begin{cases} 1/(n_k^+ + n_k^o), & \text{se } i \in \mathcal{N}_k^+ \cup \mathcal{N}_k^o \\ 0, & \text{caso contrário} \end{cases} \tag{8}$$

### 5.2. Heurística 2: recomendação de conteúdos de prioridade alta e neutra, favorecendo os primeiros

Segundo esta heurística, os conteúdos de prioridade alta e neutra são recomendados de forma aleatória uniformemente, mas os primeiros são favorecidos. Para tal, foi introduzido o parâmetro $\epsilon < 1/(n_k^+ + n_k^o)$. Seja $\epsilon$ a probabilidade de se redirecionar uma recomendação para um conteúdo de alta prioridade e que, de acordo com a heurística 5.1, seria feita para um conteúdo de prioridade neutra. Assim,

$$p_{ki} = \begin{cases} 1/(n_k^+ + n_k^o) + \epsilon n_k^o/n_k^+, & \text{se } i \in \mathcal{N}_k^+ \\ 1/(n_k^+ + n_k^o) - \epsilon, & \text{se } i \in \mathcal{N}_k^o \\ 0, & \text{caso contrário} \end{cases} \tag{9}$$

### 5.3. Heurística 3: recomendação de conteúdos de prioridade alta e neutra, favorecendo os primeiros de acordo com suas recompensas

Esta última heurística é similar à heurística 5.2. Entretanto, ao invés de redirecionar-se uniformemente as recomendações que, de acordo com a heurística 5.1, seriam feitas para um conteúdo de prioridade neutra, faz-se esse redirecionamento adotando as recompensas $r_{ki}$ como pesos. Assim,

$$p_{ki} = \begin{cases} 1/(n_k^+ + n_k^o) + \epsilon n_k^o r_{ki}/(\sum_{i=1}^N r_{ki}), & \text{se } i \in \mathcal{N}_k^+ \\ 1/(n_k^+ + n_k^o) - \epsilon, & \text{se } i \in \mathcal{N}_k^o \\ 0, & \text{caso contrário} \end{cases} \tag{10}$$

### 5.4. Setando limiares de popularidade e recompensa

Os limiares de popularidade e recompensa são usados para determinar se um determinado conteúdo é popular e se está associado a alta recompensa por recomendações, respectivamente. O limiar $\hat{\lambda}$ é definido como sendo a média das popularidades,

$$\hat{\lambda} = \frac{\sum_{i=1}^N \lambda_i}{N} \tag{11}$$

Em seguida, é determinado o limiar de recompensa. Sejam $\overline{r}$ e $\overline{\overline{r}}$ o valor mínimo e máximo de todos os $r_{ki}$, $\overline{r} = \min_{(k,i)} r_{k,i}$ e $\overline{\overline{r}} = \max_{(k,i)} r_{k,i}$. O limiar de recompensa $\hat{r}$ é definido como a média aritmética entre a recompensa mínima e a máxima,

$$\hat{r} = (\overline{\overline{r}} + \overline{r})/2 \tag{12}$$

## 6. Algoritmos Genéticos

A seguir, é proposta uma metodologia envolvendo o uso de algoritmos genéticos para buscar o sistema de recomendação de conteúdo ótimo. Em cada cenário, são fixados valores para $r_{ki}$ e $\lambda_k$, $i = 1, \ldots, N$, $k = 1, \ldots, N$, e executado o algoritmo genético para determinação dos $p_{ki}$ ótimos. Foi utilizado o programa MatLab® 2009 com a extensão de algoritmo genético (AG).

Note que, enquanto nas seções anteriores assume-se $\sum_{i=1}^N p_{ki} = 1$, nessa seção considera-se o caso em que $\sum_{i=1}^N p_{ki} \leq 1$. Dessa forma, usuários que não receberem recomendação de conteúdo vão embora do sistema imediatamente após consumirem o conteúdo de interesse. Isto ocorre, por exemplo, se a recomendação de certo conteúdo geraria custo tão alto que é vantajoso não recomendar ao usuário nenhum conteúdo.

O nível de *fitness* é dado pela recompensa total menos o custo total normalizado. O custo total normalizado é determinado de tal forma a penalizar mais as alocações nas quais o custo modificado esteja muito distante do custo de referência. Esta ideia é inspirada na maximização da recompensa sob a restrição de que o custo não ultrapasse significativamente o custo de referência. Neste trabalho, considera-se uma função degrau para o custo normalizado que penaliza mais fortemente custos que sejam muito acima do valor de referência, como indicado a seguir

$$\overline{C} = \begin{cases} 0, & \text{se } P - C < -1 \\ 10.000, & \text{se } -1 \leq P - C < 1 \\ 1.000.000, & \text{se } 1 \leq P - C < 10 \\ 100.000.000, & \text{caso contrário} \end{cases} \tag{13}$$

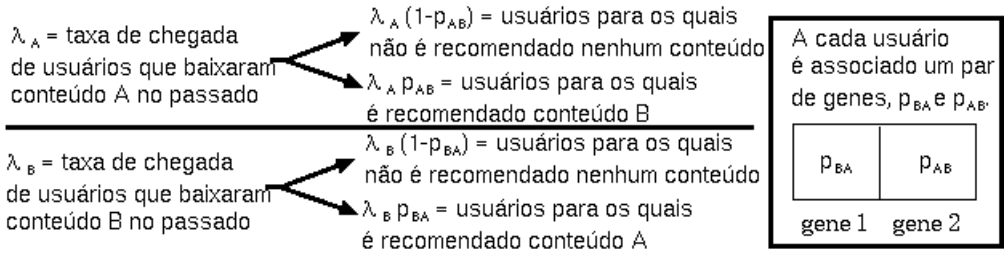



**Figura 1. Na análise via algoritmos genéticos mais simples, consideram-se apenas dois arquivos. O objetivo é obter $p_{AB}$ e $p_{BA}$.**

A função de *fitness* $F$ a ser maximizada é a recompensa $R$ dada por (2) menos o custo normalizado dado por (13),

$$F = R - \overline{C} \tag{14}$$

Nos resultados experimentais, o objetivo é maximizar (14).

### 6.1. Cenário com Dois Arquivos

Na primeira análise via algoritmos genéticos, considera-se em cada cenário apenas dois arquivos, $A$ e $B$, para estudar a interação dos $p_{ki}$ com o AG. Com isso, são obtidos resultados no ambiente mais simples possível para depois ampliar a base de estudo. Dados um perfil de recompensas e popularidades, ou seja, dadas as recompensas $r_{AB}, r_{BA}$ e as taxas de chegada $\lambda_A$ e $\lambda_B$, o objetivo é obter $p_{AB}$ e $p_{BA}$. Para tal, considera-se uma população de 200 indivíduos com dois genes $p_{AB}^{(l)}$ e $p_{BA}^{(l)}$, $A, B \in \{1, \ldots, N\}$, em cada indivíduo $l$ da população, $l = 1, \ldots, 200$. Foi avaliada então a evolução da população por 50 gerações e utilizados os seguintes parâmetros no AG: a) 2 indivíduos em elitismo; b) a seleção é proporcional utilizando o método do torneio entre os 4 melhores indivíduos; c) *crossover* em um ponto único com taxa de mutação de 1% (vide Figura 1).

Após executar-se o algoritmo genético por uma geração, obtêm-se um novo conjunto de amostras de $p_{AB}^{(l)}$ e $p_{BA}^{(l)}$, para $l = 1, \ldots, 200$. Dado o perfil de popularidades e recompensas, obtêm-se o custo modificado pelo SR, $P$, o custo de referência, $C$ e a recompensa total, $R$, dados por (5), (4) e (2), respectivamente. O nível de *fitness* da população é dado por (14). Repete-se o algoritmo por 50 gerações.

### 6.2. MovieLens

O intuito agora é usar o AG para analisar o *trace* do Movielens, a fim de encontrar os valores de $p_{ki}$ que maximizem a função de *fitness* e comparar o custo e a recompensa resultantes com aquelas obtidas usando as heurísticas propostas. Para isso, foram utilizados os mesmos parâmetros da seção 6, envolvendo 551 filmes amostrados do *trace* de 1,000 avaliações do Movielens. Note que temos agora 87 genes a serem otimizados, correspondendo aos 87 valores de $p_{ki}$ para os quais $r_{ki} > 0$ (assumimos que $r_{ki} = 0 \Rightarrow p_{ki} = 0$). Repete-se o algoritmo por 100 gerações.

## 7. Resultados

Nessa seção são apresentados os resultados experimentais. Os objetivos são a) avaliar a qualidade das heurísticas propostas e b) estudar o comportamento do algoritmo genético na determinação de parâmetros para um sistema de recomendação de conteúdo.

### 7.1. Heurísticas

Nesta seção, são apresentados os resultados obtidos com as heurísticas da seção 5. Como referência, considera-se o conjunto de $N = 551$ arquivos introduzido na seção 4. Tanto a taxa de chegada dos arquivos quanto as recompensas são setadas usando um *trace* do Movielens [Movielens 2013], de acordo com a metodologia descrita na seção 4.

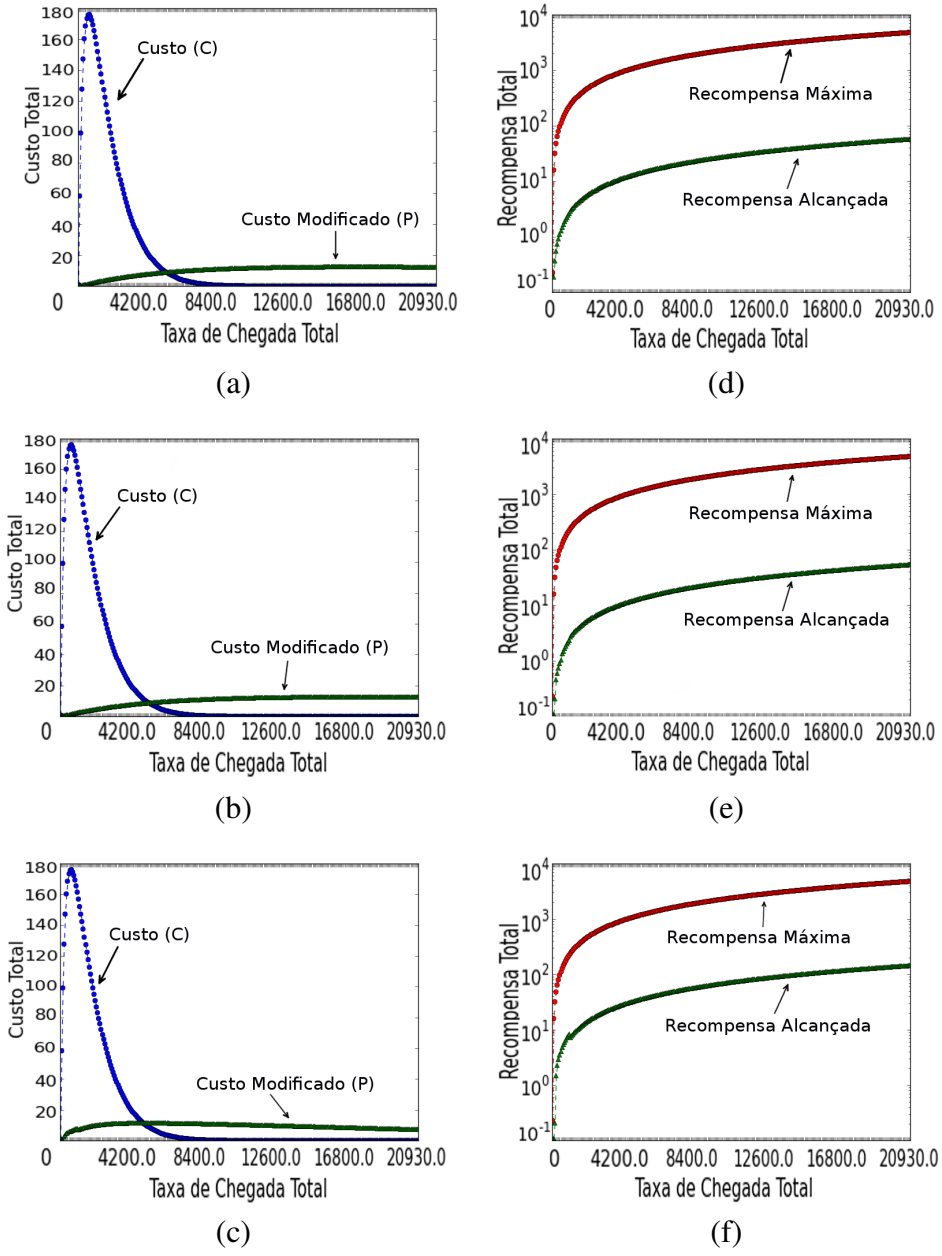



**Figura 2. Tradeoffs envolvidos no sistema de recomendação (custos e recompensas). Resultados com heurísticas 1, 2 e 3, nas linhas 1, 2 e 3, respectivamente. Na primeira e segunda colunas estão as curvas de custos (em azul, sem sistema de recomendação e em verde com sistema de recomendação) e recompensas (em vermelho, máximo alcançável, com sistema de recomendação que não leva em conta a rede, e em verde, valor obtido com recomendação que leva em conta a rede). As heurísticas apresentaram padrões de custos similares, sendo que a terceira destacou-se pela sua maior recompensa.**

A seguir, comparam-se os custos e recompensas de referência *versus* aqueles obtidos com as heurísticas propostas na seção 5. As curvas na Figura 2 ilustram como os custos e recompensas variam em função do somatório das taxas de chegada $\Lambda$. Nessas curvas foi considerado $\Lambda$ variando de 1 até 20930, em incrementos de 70 (vide eq. (6)).

As curvas verdes de custos (Figuras 2(a)-(c)) correspondem aos valores obtidos com o sistema de recomendação, e as azuis sem ele. Note que, ao utilizar o sistema de recomendação, o pico do custo diminui significativamente para qualquer uma das heurísticas adotadas.

Nos gráficos de recompensas (Figuras 2(d)-(f)), as curvas vermelhas correspondem à recompensas obtidas caso o sistema de recomendação não levasse em conta os fatores relativos à rede. Nesse cenário clássico, recomenda-se para o usuário que assistiu $k$ o conteúdo $i$ tal que $r_{ki}$ seja maximizado, ou seja, aumentando a satisfação ($r_{ki}$) do usuário que assistiu $k$ e agora pode assistir $i$. As Figuras 2(a)-(c) quantificam a diminuição do custo ao adotar-se o sistema de recomendação proposto (curva verde) contra o cenário clássico (curva azul). Note que a heurística 3 gerou recompensas maiores que as heurísticas 1 e 2, com redução similar de custos, sendo assim considerada superior no cenário em questão.

### 7.2. Algoritmo Genético

A seguir, é analisado o comportamento do algoritmo genético em um cenário simplificado com dois arquivos (Seção 7.2.1) e usando o trace do Movielens (Seção 7.2.2). No primeiro caso serviu para uma análise qualitativa de comparação das heurísticas com os resultados obtidos via AG, verificando em que situações as heurísticas são corroboradas pelos resultados obtidos via AG, enquanto que no segundo caso foi feita uma comparação quantitativa dos resultados obtidos com as heurísticas na seção anterior contra aqueles obtidos usando o AG.

#### 7.2.1. Cenário com Dois Arquivos

A seguir, são apresentados os resultados obtidos com o algoritmo genético proposto na seção 6. Consideramos um cenário de 2 arquivos, $A$ e $B$. Para esses arquivos, o algoritmo genético tenta otimizar os valores de $p_{AB}$ e $p_{BA}$ a cada rodada, buscando em uma população de 200 indivíduos. Cada indivíduo possui inicialmente um $p_{AB}$ e um $p_{BA}$ escolhidos de forma uniforme e aleatória de 0 a 1, sendo que esses parâmetros $p_{AB}$ e $p_{BA}$ associados a cada indivíduo são chamados de **genes**. A intenção do AG é confrontar esses 200 indivíduos e encontrar as melhores combinações de $p_{AB}$ e $p_{BA}$ que tenham um custo total baixo e uma recompensa total alta.

Os custos totais de disseminação dos arquivos, originais e perturbados pelo mecanismo de recomendação, são dados pelas equações (4) e (5), respectivamente. Às recompensas $r_{AB}$ e $r_{BA}$ são atribuídos valores 0,1 (baixa), 0,5 (média) ou 0,9 (alta), e às taxas de chegada $\lambda_A$ e $\lambda_B$ são atribuidos valores de 5 a 30, de acordo com os objetivos do experimento em questão. Os cenários analisados são listados na Tabela 2. Nela, aparecem quatro grandes grupos de parâmetros que categorizamos como:

- *Grupo de parâmetros G1:* conteúdos com alta recompensa e alta taxa de chegada;

- *Grupo de parâmetros G2:* conteúdos com baixa recompensa e alta taxa de chegada;
- *Grupo de parâmetros G3:* conteúdos com baixa recompensa e baixa taxa de chegada;
- *Grupo de parâmetros G4:* conteúdos com taxa de chegada intermediária, estipulada como 15.

| *cenário* | $r_{AB}$ | $r_{BA}$ | $\lambda_A$ | $\lambda_B$ | $p_{AB}$ | $p_{BA}$ | *score* |
|---|---|---|---|---|---|---|---|
| **Grupo G1: Conteúdo com alta recompensa e alta taxa de chegada** | | | | | | | |
| 1 | 0,9 | 0,9 | 30 | 30 | 0,993292 | 0,998423 | -9704 |
| **Grupo G2: Conteúdo com baixa recompensa e alta taxa de chegada** | | | | | | | |
| 2 | 0,1 | 0,1 | 30 | 30 | 0,985521 | 0,972787 | -9967 |
| **Grupo G3: Conteúdo com baixa recompensa e baixa taxa de chegada** | | | | | | | |
| 3 | 0,1 | 0,1 | 5 | 5 | 0,01 | 0,01 | -9999 |
| **Grupo G4: Conteúdo com $\lambda$ intermediário estipulado como 15.** | | | | | | | |
| 4 | 0,1 | 0,1 | 15 | 15 | 0,990565 | 0,988124 | -999983 |
| 5 | 0,5 | 0,5 | 15 | 15 | 0,995851 | 0,99 | -999918 |
| 6 | 0,9 | 0,9 | 15 | 15 | 0,993053 | 0,994235 | -999852 |

**Tabela 2. Alguns resultados obtidos com o AG. A tabela está dividida respectivamente entre os quatro grupos de parâmetros estudados. Cada cenário possui uma taxa de chegada $\lambda$, um $r_{ki}$ e um $p_{ki}$ para os arquivos $A$ e $B$, além do melhor $score$ (combinação entre custo e recompensa) encontrado pelo AG.**

No grupo G1 da Tabela 2, é possível visualizar que os $p_{AB}$ e $p_{BA}$ estão bem próximos de 1. Sendo assim, o AG está informando que é interessante recomendar esses arquivos, por terem grande quantidade de usuários interessados neles (alto $\lambda$) e por serem muito desejados por esses usuários (alto $r_{AB}$ e $r_{BA}$). Esse grupo corrobora com a regra 1 da seção 5 para criação das heurísticas de $p_{ki}$: se $\hat{\lambda} < \lambda_i$ e $\hat{r} < r_{ki}$, então recebe $p_{ki}$ mais elevado.

Já no grupo G2, o AG aparentemente determinou uma solução que vai de encontro ao estabelecido na regra 2 da seção 5: se $\hat{\lambda} < \lambda_i$ e $r_{ki} < \hat{r}$, então recebe $p_{ki}$ mais moderados. Nesse grupo, existem dois arquivos muito populares (alto $\lambda$), mas não recomendáveis um ao outro (baixo $r_{AB}$ e $r_{BA}$). Mesmo assim, o AG determinou altos valores de $p_{AB}$ e $p_{BA}$ (próximos de 1). Isso pode ser explicado pelo fato de $\lambda$ muito alto estar contribuindo muito mais na função de custo do que o $r_{ki}$ baixo na recompensa. Por ter um $\lambda$ muito elevado, naturalmente as funções de custo das equações (3) e (5) tendem a diminuir muito ao recomendarmos os arquivos, mesmo que eles não sejam fortemente complementares. Assim, na ausência de outras possibilidades o AG favorece a recomendação em prol da diminuição dos custos de rede.

No grupo G3 da Tabela 2, o AG corrobora com a regra 2 das heurísticas. Para um conteúdo pouco popular (baixo $\lambda$) e com pouco interesse via recomendação por parte dos usuários (baixo $r_{AB}$ e $r_{BA}$), não parece ser interessante recomendar o conteúdo para qualquer usuário. Com isso, ficou estabelecido pelo AG que conteúdos desse tipo receberiam valores próximos a 0 de $p_{AB}$ e $p_{BA}$.

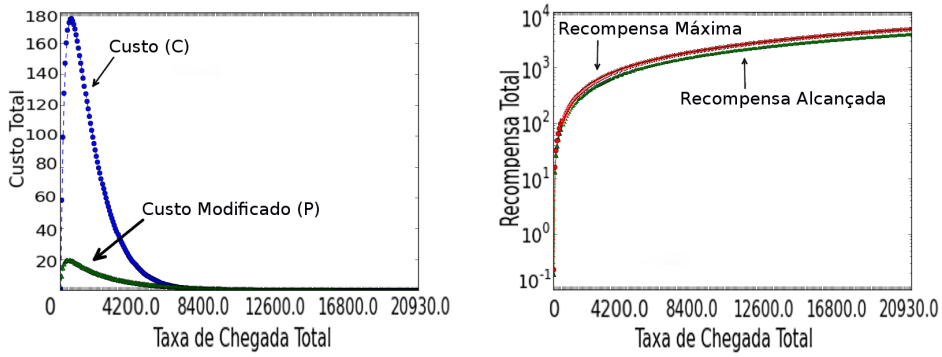
No grupo G4 da Tabela 2, com $\lambda = 15$, foram considerados três cenários. Para os três cenários, o *score* é menor que o dos grupos 1, 2 e 3. Quando $\lambda = 15$, os custos

são muito elevados e a recompensa baixa. Isto se deve ao fato de que o AG não foi capaz de determinar os parâmetros ótimos neste grupo. Trabalhos futuros consistem em avaliar políticas alternativas para estas situações.

A partir de uma análise detalhada de todos os cenários é possível determinar valores adequados de $p_{ki}$ para diferentes ecossistemas de variáveis. Isso simplifica a criação de um SR que leve em conta as características de cada grupo de parâmetros e, assim, pré-estabelecer valores de $p_{ki}$ adequados para diferentes situações.

### 7.2.2. Inferindo $p_{ki}$ de Movielens com Algoritmo Genético

Depois de explorar um ambiente mais simples com dois arquivos, foram obtidos alguns resultados com o *trace* do Movielens e o AG. A Figura 3, ilustra os resultados do AG que podem ser comparados com as heurísticas da Figura 2. Em (a) e (b) é possível verificar os resultados de custo e recompensa, respectivamente, para os $p_{ki}$ determinados pelo AG. O AG conseguiu diminuir o custo total do sistema e aumentar a recompensa até bem perto do valor máximo de recompensa total. Esses resultados ilustraram maior recompensa e menor custo que os resultados obtidos via melhor heurística (heurística 3, vide Seção 5.3).




(a) (b)

**Figura 3. Tradeoffs envolvidos no sistema de recomendação com AG (custos e recompensas). Os $p_{ki}$ gerados pelo AG reduziram o custo e aumentaram a recompensa comparados a Figura 2.**

## 8. Conclusão

Neste trabalho foram discutidos alguns dos potenciais benefícios de se estabelecer uma visão unificada sobre redes P2P e recomendação de conteúdo. As ideias aqui apresentadas podem ter importante impacto em outras redes, como em ICNs (*Information Centric Networks*) para armazenamento do melhor conteúdo, ou até mesmo na determinação do melhor conteúdo a ser guardado no *cache* de servidores *proxy* para diminuir a quantidade de *cache misses*. A otimização dos parâmetros do sistema de recomendação foi mais eficiente quando ao utilizar o AG em comparação com heurísticas pré-estabelecidas, dando margem para o estabelecimento de melhores heurísticas em trabalhos futuros.

*Agradecimentos:* Agradecemos a Gabriel Mendonça por comentários valiosos na preparação final do artigo.

## Referências

Adomavicius, G. and Tuzhilin, A. (2005). Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734 – 749.

Chakoo, N., Gupta, R., and Hiremath, J. (2008). Towards Better Content Visibility in Video Recommender Systems. *2008 Japan-China Workshop Frontier Comp. Sci. Tech.*

Guarnieri, T., da Silva, A. P. C., and Vieira, A. B. (2013). Impact of dynamic social relationship in a file sharing system. In *Workshop on Dynamic Networks*.

Guarnieri, T. A., da Silva, A. P. C., Almeida, J. M., and Vieira, A. B. (2014). Influência do facebook em enxames bittorrent. In *32o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2014*.

Hecht, F., Bocek, T., Bar, N., Erdin, R., Kuster, B., Zeeshan, M., and Stiller, B. (2012). Radiommender: P2p on-line radio with a distributed recommender system. In *P2P Computing, 2012 IEEE 12th*, pages 73–74.

Mei, T., Yang, B., Hua, X., and Yang, L. (2007). VideoReach: an online video recommendation system. *SIGIR '07. ACM*, pages 767–768.

Menasche, D. S., Neglia, G., Towsley, D., and Zilberstein, S. (2009). Strategic reasoning about bundling in swarming systems. In *GameNets' 09*, pages 611–620. IEEE.

Mendonça, G. G. and Leão, R. M. M. (2012). BTStream — Um ambiente para desenvolvimento e teste de aplicações de streaming P2P. In *XXX SBRC, Salão de Ferramentas*.

Movielens (2013). Movielens - movie recommendations. `http://movielens.umn.edu/`. [Online; accessed 11-March-2013].

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Trestian, I., Huguenin, K., Erramilli, V., Traverso, S., Papagiannaki, K., and Laoutais, N. (2014). Social-aware replication in geo-diverse online systems. *IEEE Transactions on Parallel and Distributed Systems*, 99(PrePrints):1.

Vieira, D. M., Delgado, C., and Menasché, D. S. (2013). Como influenciar seus pares? sistemas de recomendação de conteúdo para redes p2p. In *WP2P+/SBRC (resumo extendido)*, pages 67–72.

Wu, W. and Lui, J.-S. (2012). Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation. In *INFOCOM*, pages 1206–1214.

Zhang, H., Liu, B., Weng, X., and Yu, C. (2012). Can online social friends help to improve data swarming performance? In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–7.

Zhou, R., Khemmarat, S., and Gao, L. (2010). The impact of YouTube recommendation system on video views. *IMC '10*, page 404.

Zhou, Y., Fu, T., and Chiu, D. (2012). A unifying model and analysis of p2p vod replication and scheduling. *INFOCOM, 2012 Proceedings IEEE*.

# Sobre a Capacidade de Serviço de Sistemas P2P

**Edmundo de Souza e Silva**[1], **Rosa M.M. Leão**[1], **Daniel S. Menasché**[1], **Don Towsley**[2]

[1]Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ – Brazil

[2]University of Massachusetts at Amherst (UMass)
Amherst, MA – USA

***Abstract.*** *One of the most fundamental problems in the realm of peer-to-peer systems consists of determining their service capacity. In this paper, we first propose a new Markovian model to compute the throughput of peer-to-peer systems. Then, we present a simple approximate model for obtaining the system throughput for large peer populations. From the models, we obtain novel insights on the behavior of p2p swarming systems which motivate new mechanisms for publishers and peers to improve the overall performance. In particular, we show that if publishers adopt the most deprived peer selection, and peers reduce their service rate when they have all the file blocks but one, the system's capacity can significantly increase.*

***Resumo.*** *Um dos problemas fundamentais no domínio de redes par-a-par consiste na determinação da capacidade de serviço desses sistemas. Neste artigo, propomos inicialmente um novo modelo Markoviano para calcular a vazão de sistemas par-a-par. Propomos ainda um modelo aproximado simples para calcular a vazão do sistema quando o número de pares é grande. Os resultados dos modelos motivaram a elaboração de novos mecanismos para seleção de pares por parte dos servidores e também dos pares, de tal forma a aumentar o desempenho geral do sistema. Em particular, mostramos que se os servidores adotarem a política de servir primeiro os pares com menor número de blocos, e se os pares reduzirem suas taxas de serviço quando tiverem obtido todo o arquivo exceto por um dos blocos, a capacidade de serviço do sistema aumenta significativamente.*

## 1. Introdução

Sistemas par-a-par (*peer-to-peer* ou P2P) têm tido muito sucesso na disseminação de conteúdo na Internet e importantes empresas têm adotado essa arquitetura. Por exemplo, a *Blizzard entertainment* distribui arquivos grandes via o *Blizzard downloader* que é baseado no *BitTorrent Opensource*. O Linux Ubuntu também é disponível para *download* via BitTorrent. A *Amazon Simple Storage Service* (Amazon S3) é uma infraestrutura de armazenamento e distribuição de dados e também suporta o protocolo BitTorrent para distribuir arquivos. Propostas recentes sugerem o uso de P2P para distribuição de grandes arquivos em redes orientadas a conteúdo [Jacobson et al. 2009], dando origem a uma *Network Swarm Architecture*.

Arquiteturas P2P têm sido estudadas por mais de uma década e inúmeros modelos para entender o seu desempenho foram propostos. Por exemplo, modelos para determinar o impacto dos mecanismos de incentivo e das taxas de *upload* e *download* sobre a

performance; do efeito do chamado *free riding* [Xia and Muppala 2010]; da forma com que arquivos são disseminados; e modelos para estudar a equidade (*fairness*) do sistema e ainda a disponibilidade de conteúdo no *swarm* [Murai et al. 2012, de Souza e Silva et al. 2013]. Além disso, estudos têm sido realizados sobre as estratégias de distribuição de blocos (*pieces*) e escolha dos *peers* para transmissão [Massoulie and Twigg 2008].

Apesar dos inúmeros artigos na área, apenas recentemente a escalabilidade do sistema começou a ser melhor equacionada. Em outras palavras, sistemas P2P eram ditos escaláveis uma vez que cada novo usuário no sistema a ele acrescenta novos recursos e, portanto, a capacidade total disponível deveria aumentar proporcionalmente aos recursos incorporados. Consequentemente, é esperado que a vazão do sistema cresça linearmente e sem limites com o aumento da população. Entretanto, limitações intrínsecas à arquitetura P2P têm sido observadas. Recentemente Hajek e Zhou [Hajek and Zhu 2010] mostraram que se a taxa de chegada $\lambda$ de *peers* no sistema for maior que a capacidade $U$ do servidor de conteúdo *dedicada* ao *swarm*, a população de usuários cresce sem limites quando *peers* e blocos são escolhidos de forma aleatória.

Menasché *et al* [Menasché et al. 2012] avaliaram o impacto de diferentes estratégias e parâmetros do sistema sobre a sua vazão (ou, analogamente, sobre a estabilidade do sistema). Os resultados indicaram que, quando o servidor de conteúdo dá prioridade aos usuários menos favorecidos (aqueles que possuem menor número de blocos do arquivo sendo distribuído), e aos blocos mais raros, enquanto que os *peers* selecionam pares aleatoriamente para trocar informação, o limite que pode ser atingido para a vazão do sistema é significantemente maior do que o reportado em [Hajek and Zhu 2010]. A partir de um modelo Markoviano bastante simples, estudamos a vazão atingida por diferentes estratégias para o servidor e *peers* de um *swarm*.

Neste trabalho propomos modelos mais elaborados que o usado em [Menasché et al. 2012] que nos permitem realizar estudos detalhados sobre a vazão e os limites de estabilidade do sistema. Identificamos duas regiões dentro das quais os sistemas P2P podem operar. Na primeira região, quando $\lambda < \lambda_c$, o sistema é estável. Na segunda, quando $\lambda_c < \lambda < \lambda_d$, o sistema é instável, mas sua vazão ainda pode aumentar em função da taxa de chegada dos *peers*. A identificação dessas regiões permitiram obter modelos eficientes, capazes de resolver sistemas com um grande número de usuários. Nossas principais contribuições incluem:

1. A extensão do modelo Markoviano de [Menasché et al. 2012], levando em conta estruturas especiais da cadeia resultante para resolver sistemas com mais blocos e usuários.
2. A elaboração de um novo modelo de filas para se obter limites máximos para a vazão do sistema de acordo com diferentes estratégias de serviço.
3. A partir dos resultados dos modelos,
    - uma nova proposta de política muito simples adotada pelos *peers* que permite aumentar drasticamente a vazão do sistema e;
    - uma proposta de política para o *tracker* que também traz ganhos significativos de desempenho.

Inicialmente, na seção 2, discutimos brevemente o problema de estabilidade e apresentamos uma extensão do limite da vazão obtido em [Menasché et al. 2012]. Na seção 3 descreveremos os modelos que usaremos para obter os resultados deste artigo. Os

resultados obtidos são discutidos na seção 4, onde propomos também uma nova política a ser adotada pelos *peers*, que permite um significativo aumento da vazão. Discutimos os trabalhos relacionados em 5 e na seção 6 apresentamos nossas conclusões.

## 2. Escalabilidade de sistemas *Peer-to-Peer*

Em sistemas P2P, cada novo *peer* incorporado a um *swarm* traz novos recursos (capacidade de transmissão e memória), pois funciona tanto como cliente quanto como provedor de conteúdo. Por conseguinte, é esperado que o número de clientes atendidos por unidade de tempo (vazão ou *throughput*) aumente com o número de *peers*. Entretanto, esse aumento não é ilimitado. A Figura 1, obtida com um dos modelos que desenvolvemos, ilustra essa questão. Na Figura, a medida que a população do *swarm* aumenta a vazão ($\lambda$)

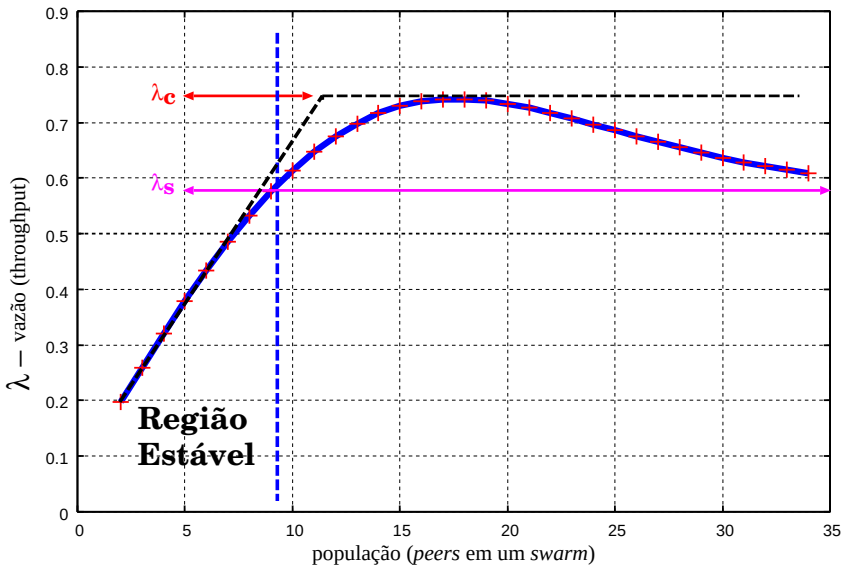



**Figura 1. Vazão limitada.**

aumenta quase que linearmente. Entretanto, a partir de uma certa população isso não é mais verdade. Sucintamente, embora a quantidade de recursos aumente, eles nem sempre são utilizados integralmente. Por exemplo, quando um *peer* não tem nenhum bloco diferente daqueles que os demais *peers* possuem, este *peer* não pode fazer uso de sua capacidade de *upload*.

Ainda com referência a Figura 1, podemos identificar dois regimes de operação. O primeiro é um regime *estável*, onde os *peers* que chegam ao sistema fazem o *download* completo do conteúdo em um tempo finito. A esse regime é associada uma taxa limite $\lambda_s$ de forma que, quando $\lambda < \lambda_s$, o sistema é estável. Para muitos sistemas com recursos limitados, $\lambda_s$ é a máxima vazão que pode ser alcançada. Entretanto, esse não é o caso em sistemas P2P. Neste caso, o limite para a vazão é $\lambda_c > \lambda_s$. Na próxima seção indicaremos como obter esses limites através dos modelos que desenvolvemos.

Inicialmente explicaremos de forma genérica o que acontece. Considere um sistema P2P no qual *peers* saem do sistema imediatamente após concluírem o *download*. O arquivo a ser distribuído é inicialmente dividido em $K$ blocos de igual tamanho. Seja $U$ a capacidade do servidor dedicada a um *swarm*, em blocos por unidade de tempo. Caso

a taxa de chegada dos *peers* seja bem alta (em relação a $U$), o sistema alcançará, com alta probabilidade, um estado no qual todos os *peers* possuem o arquivo completo exceto um dos seus blocos (e.g., bloco $X$). Nesse cenário, apenas o servidor possui o bloco $X$ e assim que um *peer* recebe o bloco faltante do servidor, ele deixa o sistema, sem que tenha ajudado a servir outros *peers*. Por outro lado, novos *peers* que chegam ao sistema são rapidamente servidos por seus pares e conseguem obter todos os blocos exceto $X$. Em consequência, o sistema passa a se comportar aproximadamente como cliente-servidor: *peers* partem do sistema a taxa $U$, já que não cooperam uns com os outros para obter o bloco faltante. Apesar da quantidade de recursos crescer com o tamanho da população, tais recursos não são plenamente utilizados. A Figura 2, também obtida por nosso modelo analítico, serve para esclarecer o comportamento descrito acima. Cada linha vertical nas quatro sub-figuras indica o número de *peers* em três subconjuntos: aqueles que não contém nenhum bloco (portanto, recém-chegados ao *swarm*), aqueles que contém o bloco mais raro no *swarm* e aqueles que contém todos os blocos menos o mais raro. As Figuras 2a, 2b, 2c e 2d são *snapshots* em quatro instantes de tempo: 1, 5, 10 e 20 unidades. É fácil notar que apesar de, em $t = 1$, poucos *peers* no *swarm* possuírem todos menos o bloco mais raro (linha vertical vermelha) o sistema evolui para a condição em que a maioria contém todos exceto o bloco raro.

O modelo de filas proposto neste artigo para estimar a vazão leva em consideração a observação acima, isto é, com alta probabilidade a maioria dos *peers* eventualmente consegue obter todos os blocos exceto um (Figura 2). Mostraremos na seção 3 como

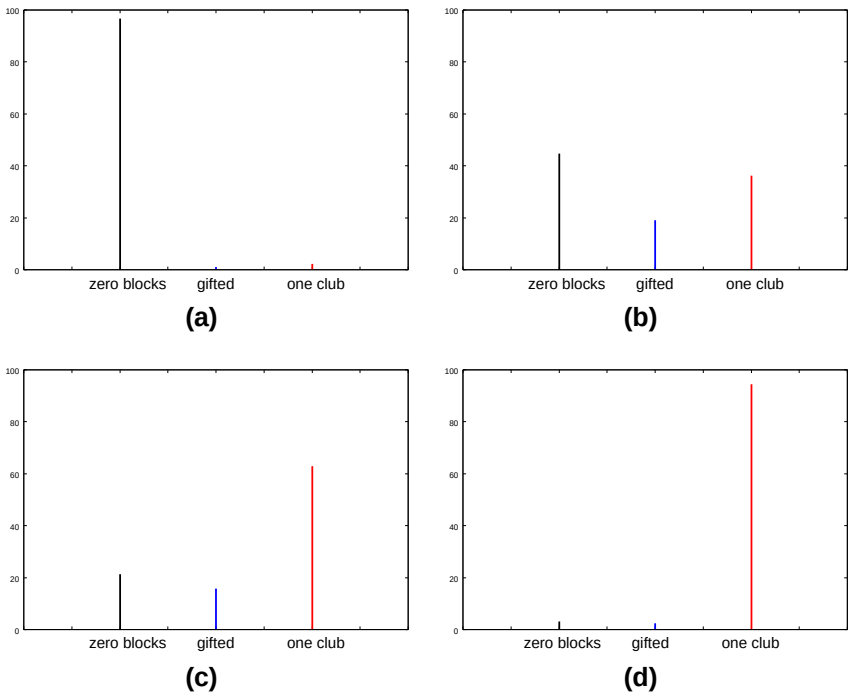



**Figura 2. Evolução no tempo de *peers* em um *swarm*.**

obter o limite $\lambda_c$ para a vazão do sistema. Essa taxa limite depende das políticas adotadas pelo servidor e *peers*.

## 3. Modelos

Sistemas P2P admitem diferentes políticas de seleção de *peers* e blocos tanto pelo servidor quanto para os próprios *peers* vizinhos. Em [Menasché et al. 2012] várias destas políticas foram estudadas usando simulação e um modelo Markoviano muito simples. Foi realizado um estudo para determinar o impacto de diferentes parâmetros no desempenho do sistema. Um dos resultados em [Menasché et al. 2012] mostrou que a vazão limite ($\lambda_c$ na Figura 1) é proporcional a $KU$ quando o servidor adota a política de servir prioritariamente os *peers* recém chegados no sistema. Pelo seu melhor desempenho, esta será a política na qual focamos este trabalho. Entretanto, ressaltamos que, como será mostrado na seção 4, adotando-se ligeiras modificações desta política para o servidor e ainda outra para os *peers*, a vazão do sistema aumenta significativamente.

Nesta seção apresentamos os modelos usados para obter os resultados deste trabalho. Inicialmente obtemos a taxa limite $\lambda_c$, generalizando o resultado de [Menasché et al. 2012]. Em seguida comentaremos rapidamente sobre o modelo de Markov que utilizamos para obter a vazão em função do número de *peers* em um *swarm*. Finalmente, baseado no modelo Markoviano, construímos um modelo de filas para determinar $\lambda_s$.

### 3.1. Obtendo $\lambda_c$

Consideremos um servidor que serve **prioritariamente *peers* com menor número de blocos**. Seja $\mu$ a capacidade de *upload* dos *peers*. Além disso, vamos supor que os *peers* podem alterar a sua taxa de serviço da seguinte forma: quando um *peer* obtém $K-1$ blocos ele pode reduzir a sua capacidade de servir outros *peers* para $\mu' \leq \mu$. A motivação para essa política será esclarecida no que se segue, ao mostrarmos que esse simples ajuste na taxa de serviço dos *peers* com $K-1$ blocos pode aumentar significativamente a vazão do sistema. Seja $\lambda$ a taxa de chegada de *peers*, e vamos supor que ela é grande o suficiente de forma que há sempre novos *peers* sem blocos. Portanto, devido à política do servidor, a maior parte da sua capacidade é dedicada aos *peers* recentes no *swarm*.

**Proposição 3.1** *Suponha que as políticas de serviço sejam as seguintes: (a) o servidor prioriza os peers com menor número de blocos e, dentre esses, serve os blocos mais raros; (b) cada peer escolhe um outro de forma aleatória e, dentre o selecionado, escolhe um bloco aleatório para servir dentre os úteis (blocos úteis são os que faltam no peer escolhido e que o peer transmissor possui). Nestas condições, a vazão do sistema $\lambda_c$ é limitada por*

$$\lambda_c \leq ((K-2)\mu/\mu' + 2)U \tag{1}$$

*onde $K$ é o número de blocos do conteúdo sendo distribuído.*

*Argumento:* O argumento é uma extensão do apresentado em [Menasché et al. 2012], de forma a permitir mudança na taxa de serviço dos *peers*, conforme descrito acima. Assuma que $\lambda_c$ é suficientemente grande de forma que, como mostrado na Figura 2, o sistema irá evoluir para o estado no qual uma grande quantidade de *peers* terá todos menos um bloco, sendo esse evidentemente o mais raro. Estes *peers* são chamados de *one club*. (Veja também a Figura 3). Pela política de atendimento do servidor, os *peers* recém chegados recebem o bloco mais raro deste (a taxa $U$). Estes *peers* são chamados de *gifted* pois passam a possuir o bloco mais raro. Como os *peers* adotam a política de seleção uniforme e aleatória de pares, e como o conjunto do *one club* tem tamanho relativamente

grande, o restante dos *peers* recém chegados recebe blocos populares (todos exceto o mais raro) de *peers* no *one club* (a taxa $\mu'$). Os *gifted peers* transmitirão o bloco mais raro, com alta probabilidade, aos membros do *one club*. Isso ocorre porque há um número grande de *peers* no *one club* e os *peers* selecionam uns aos outros de forma uniforme e aleatória. As demais transmissões podem ser negligenciadas (e.g., seta (a) na Figura 3), pois a população do *one club* é relativamente alta.

Os *gifted peers* levam $(K-2)/\mu'$ unidades de tempo adicionais para receberem os próximos $(K-2)$ pedaços do *one club*. Pelo resultado de Little, existem em média $U(K-2)/\mu'$ *peers gifted*, com número de blocos inferior ou igual a $K-2$, cada um servindo o *one club* a taxa $\mu$. Esses *gifted peers*, ao receberem o $(K-1)$-ésimo bloco, passam a servir o bloco raro a taxa $\mu'$, devido à política de serviço adotada pelos *peers*. Logo, a taxa agregada com a qual o *one club* é servido é de $U(K-2)\mu/\mu' + U$ (seta (c) e (d)) na Figura 3). Assim sendo, os *peers* obtém todos os blocos e saem do sistema a taxa $U(K-2)\mu/\mu' + U$ partindo do *one club*, e os *peers gifted* entram e saem do sistema a taxa $U$. Logo, a vazão do sistema é $((K-2)\mu/\mu' + 2)U$. □

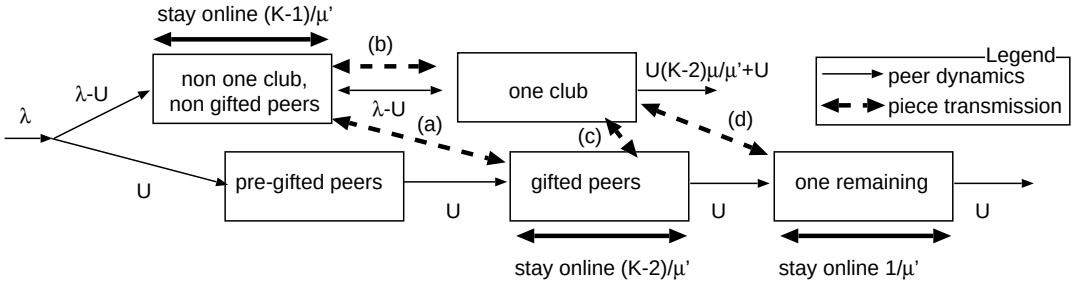



**Figura 3. Diagrama usado para determinação da região de estabilidade.**

É importante observar que o resultado acima **é contra-intuitivo** pois, reduzindo-se a taxa de *upload* dos *peers* que detém todo o conteúdo exceto por um bloco, a vazão $\lambda_c$ do sistema aumenta inversamente a essa taxa! A redução de $\mu'$, traz economia de banda para os *peers*, aumenta a vazão do sistema e não necessita de nenhum mecanismo de incentivo. Os modelos mais elaborados que seguem irão confirmar esse resultado, inclusive para a taxa limite de estabilidade $\lambda_s$.

### 3.2. Modelo Markoviano

Elaboramos um modelo Markoviano para o cálculo da vazão em função da população do *swarm*, e para diferentes políticas de atendimento. Por limitações de espaço, descreveremos apenas algumas das características gerais do modelo. Em primeiro lugar supomos que o sistema é *fechado*, i.e., assim que um *peer* deixa o *swarm* um novo é imediatamente admitido e seja $N$ o número de *peers* no sistema. Modelos fechados são úteis para determinar gargalos em função da população do sistema. Esta suposição permite o estudo do desempenho de um *swarm* para uma população aproximadamente constante.

Nosso modelo implementa as características fundamentais das políticas estudadas. Uma escolha ingênua para as variáveis de estado do sistema consiste em usar, para cada *peer*, a *assinatura* dos blocos que ele possui, isto é, um vetor indicando quais blocos o *peer* possui. Obviamente essa escolha causa rapidamente uma explosão de estados, inviabilizando a solução mesmo para valores pequenos de $K$ (blocos) e $N$ (população do

*swarm*). Entretanto, é interessante notar que o modelo possui simetrias que podem ser utilizadas para aglutinar estados (*lump states*) [de Souza e Silva and Muntz 1992].

Essas simetrias no modelo permitem fazer uma escolha viável para as variáveis de estado, da seguinte forma. Seja $\Omega$ o espaço de estados do modelo. Cada estado $\boldsymbol{\sigma} \in \Omega$ consiste em um vetor $\boldsymbol{\sigma} = \langle \sigma_1, \ldots \sigma_M \rangle$ onde o elemento $\sigma_i$ do vetor é igual ao número de *peers* com assinatura $i$ e $M = 2^{K-1}$ é o número total de assinaturas.

Pode-se verificar que há outras simetrias que nos permitem reduções adicionais (de uma ordem de grandeza) na cardinalidade de $\Omega$, nos exemplos aqui considerados. Para dar um exemplo simples, imagine um *swarm* com 3 blocos. Cada um dos blocos pode se tornar o mais raro. Suponha que $R$ *peers* tenham todos os blocos exceto o primeiro, e $N - R$ *peers* tenham o bloco mais raro. É fácil verificar que esse estado tem a mesma probabilidade que cada um dos estados onde o segundo ou o terceiro bloco é o mais raro. Por conseguinte, esses 3 estados podem ser aglutinados em um único. O nosso modelo Markoviano, usado nos casos onde a população varia, faz uso das aglutinações possíveis. Ressaltamos que aglutinação é uma operação "exata" e que não há perda de informação nessa operação.

### 3.3. Modelo de filas

O modelo Markoviano acima permite que a vazão seja obtida em função da população e portanto, a partir dele, pode-se, para $N$ relativamente grande, obter a vazão $\lambda_s$ que delimita a região de estabilidade (Figura 1). Entretanto, mesmo aproveitando as simetrias existentes no modelo para reduzir o espaço de estados, a solução analítica torna-se inviável para valores grandes de $K$ e $N$. Portanto, é importante desenvolver um modelo mais simplificado de forma a obter $\lambda_s$ com um baixo custo computacional. Esse é o principal objetivo do novo modelo de filas que desenvolvemos abaixo. O modelo captura as principais características do sistema e, pela sua simplicidade, permite também uma melhor compreensão dos efeitos fundamentais dos parâmetros do sistemas sobre o seu desempenho.

O modelo consiste em quatro filas que capturam os *peers* em determinadas situações. Uma suposição fundamental é que o sistema encontra-se perto da saturação e, como já comentado anteriormente, neste caso o número de *peers* no *one club* é muito grande.

A Figura 4 ilustra o modelo, que será brevemente descrito no que se segue. Os novos *peers*, ao chegarem ao sistema com taxa $\lambda^{(0)}$, entram na fila $F_0$. Esses *peers* não possuem nenhum bloco e, portanto, podem ser servidos tanto pelo servidor (que prioritariamente serve os *peers* com menos blocos) quanto pelos demais *peers*. Como o *one club* é relativamente grande, um bloco proveniente de *peers* para um recém chegado será fornecido por um membro do *one club*. Coletivamente, membros do *one club* suprem blocos a taxa $\mu'$ para os recém chegados.

Seja $n_0$ o número de usuários na fila $F_0$. A taxa de serviço de $F_0$ é então $n_0\mu' + U$. Com probabilidade $U/(\mu' n_0 + U)$ um *peer* nesta fila recebe um bloco do servidor e com probabilidade complementar, do *one club*. Caso um recém chegado obtenha o bloco do servidor, este bloco será necessariamente o mais raro devido à política de atendimento do servidor. Este *peer* será encaminhado à fila $F_G$ de *gifted peers* (os que detém o bloco mais

raro). Caso o recém chegado obtenha um bloco do *one club*, será encaminhado para a fila $F_1$.

Como os *peers* em $F_1$ possuem apenas um único bloco sendo que esse não é o mais raro, podem novamente ser servidos pelo servidor, somente se a fila $F_0$ está vazia, pois o servidor dá preferência aos *peers* sem blocos. Similarmente a $F_0$, os *peers* em $F_1$ podem ser servidos pelo *one club*.

Os *gifted peers* em $F_G$ são servidos apenas pelo *one club*, pois o servidor dá preferência aos que não possuem o bloco mais raro. Essa é então uma simples fila $M/M/\infty$ com taxa de serviço $\mu' n_G$ (onde $n_G$ é o número de *peers* em $F_G$). Uma vez obtido mais um bloco, os *peers* em $F_G$ encaminham-se para a fila $F_{G+}$ para obter os blocos restantes. Um *peer* em $F_1$, caso obtenha um bloco do servidor, é encaminhado à fila $F_{G+}$ de *gifted peers* com o bloco raro e um ou mais blocos adicionais. De outra forma, ele se juntará ao *one club*. Isso acontece porque assumimos que a probabilidade de *peers* com 2 blocos obterem o bloco raro do servidor é muito baixa pois esse dá preferência aos *peers* com menor número de blocos.

Para resolver a rede de filas, supomos que as probabilidades de encaminhamento entre as filas, que são dependentes do estado das filas, são obtidas do valor médio da população de cada fila como indicado na Figura 4. A solução da rede de filas é obtida como se segue.

Suponha que o valor da taxa de chegada ($\lambda^{(0)}$) é conhecido. A fila $F_0$ pode ser resolvida a partir dos parâmetros $U$ e $\mu'$, dados do problema. A partir da solução em estado estacionário, calculamos o valor médio $E[n_0]$ do número de usuários em $F_0$. Utilizando a suposição de independência das probabilidades de encaminhamento em relação ao estado da fila $F_0$, as taxas de chegada às filas $F_G$ e $F_1$ podem ser obtidas. Analogamente, as outras taxas de chegada nas demais filas são obtidas e, finalmente, a taxa de saída $\gamma^{(0)}$ do sistema. Note que a taxa de serviço da fila $F_1$ é dependente do estado da fila $F_0$, e usamos o valor estacionário $P[n_0 = 0]$ como aproximação. (Na Figura 4, a notação $I_{n_0=0}$ é a função indicadora tal que $I_{n_0=0} = 1$ se $n_0 = 0$ e $I_{n_0=0} = 0$ caso contrário.)

Após resolver a rede de filas, e calculada a taxa de saída $\gamma^{(0)}$ do sistema a partir de $\lambda^{(0)}$, fazemos $\lambda^{(1)} = \gamma^{(0)}$, pois estamos resolvendo um sistema fechado, onde uma saída de *peer* corresponde a uma chegada ao sistema. Iteramos da forma $\lambda^{(n)} = \gamma^{(n-1)}$ até que a convergência seja atingida. Os detalhes são omitidos por limitações de espaço.

## 4. Resultados

Nesta seção apresentamos resultados numéricos obtidos com o modelo Markoviano e com a aproximação proposta. Nossos objetivos são: (a) mostrar como a vazão varia em função de diferentes parâmetros do modelo; (b) motivar novas políticas de atendimento do servidor e dos *peers*. O cenário de referência consiste em $K = 3$, $\mu = 1.00$ e $U = 1$. Variamos então os parâmetros de acordo com os objetivos do experimento.

### 4.1. Validando a aproximação

A Figura 5 tem o objetivo de validar a aproximação da vazão $\lambda_s$ obtida com o modelo de quatro filas. Para tal, consideramos um arquivo de 3 blocos, $U = 0.3$ e variamos $\mu$ (com $\mu' = \mu$). Resolvemos o modelo Markoviano também via simulação (para população variando de 2 a 70 usuários) e via solução analítica usando um método de solução de cadeias

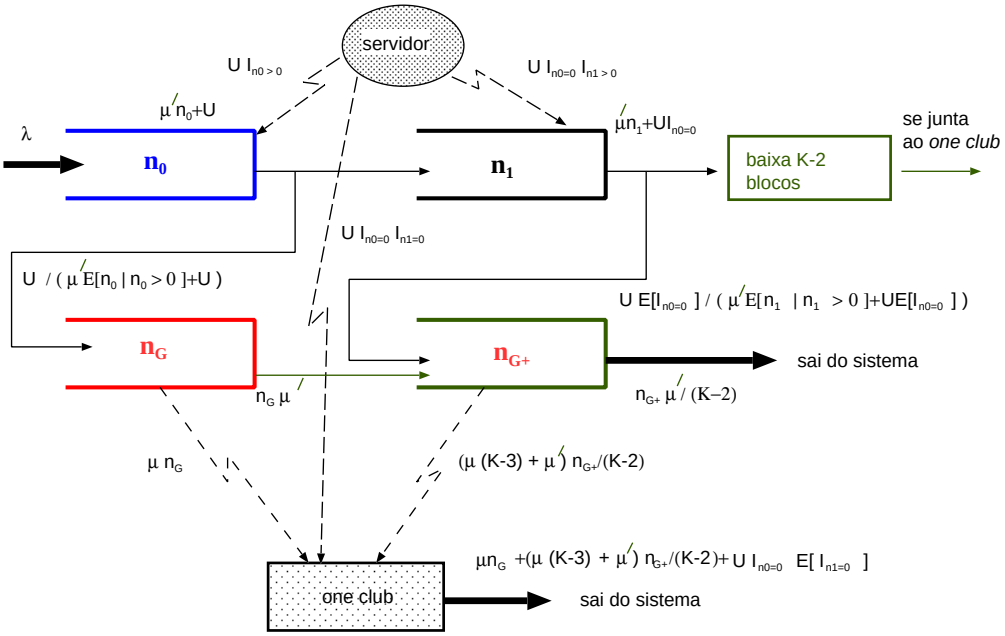



**Figura 4. Modelo de filas.**

de Markov (para população variando de 2 a 30 usuários). Não é surpreendente que as soluções obtidas analiticamente estejam dentro do intervalo de confiança da simulação. Para todos os valores de $U$ e $\mu$, podemos notar que na medida em que a população aumenta, a vazão primeiro aumenta, atingindo um pico e depois diminui, tendendo a vazão $\lambda_s$. Este valor é bem aproximado pelo modelo de redes filas apresentado na seção 3, que captura as propriedades essenciais do sistema.

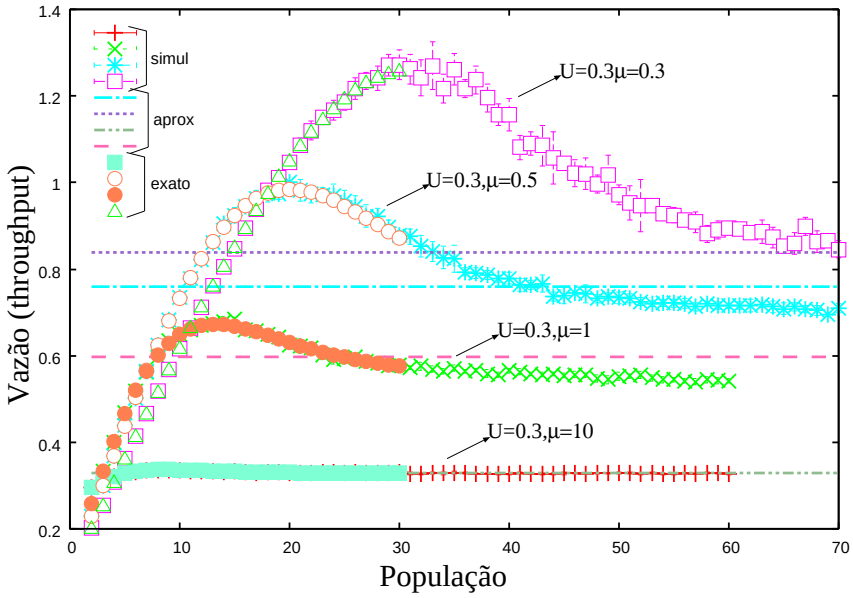
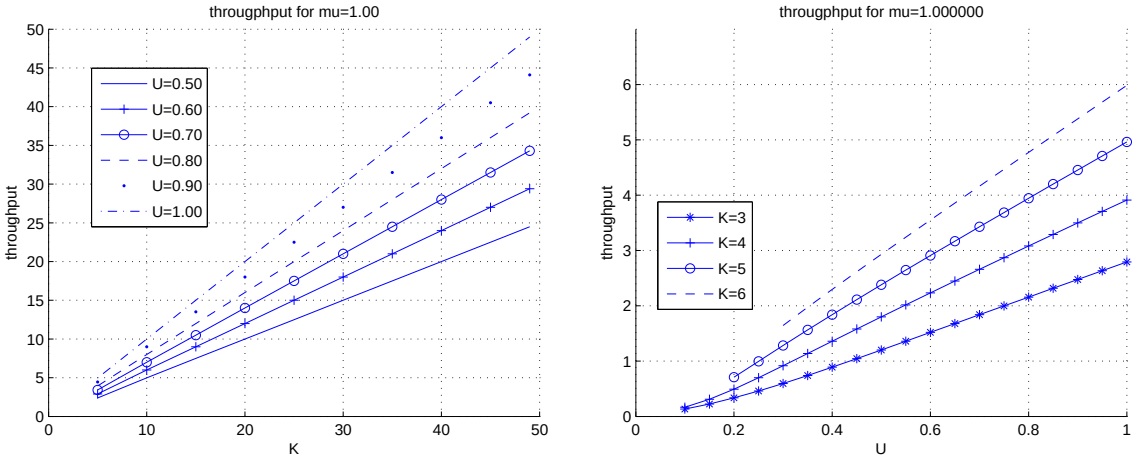



**Figura 5. Validando a aproximação, $K = 3, \mu = \mu'$.**

### 4.2. Variando a capacidade do servidor dedicada ao *swarm*

A Figura 6 ilustra como a vazão varia em função de $K$ e $U$. A figura indica que o modelo captura o fato de que a vazão varia de forma aproximadamente linear em função de $K$ e $U$, para $\mu$ constante. Isto ocorre pois o servidor, na maioria de suas transmissões, envia pacotes para os *peers* que não possuem nenhum bloco. Cada *gifted peer* contribui para a saída de, na média, $K-1$ *peers* do *one club*, antes dos *gifted* saírem do sistema. Dessa forma, cada transmissão do servidor contribui para a saída de $K$ *peers* do sistema, e como o servidor transmite blocos a taxa $U$ blocos por unidade de tempo, a vazão do sistema cresce linearmente tanto com $K$ quanto com $U$.




(a) Vazão em função de $K$ (b) Vazão em função de $U$

**Figura 6. A vazão varia linearmente com $U$ e $K$.**

### 4.3. Serviço modificado dos *peers* com $(K-1)$ blocos

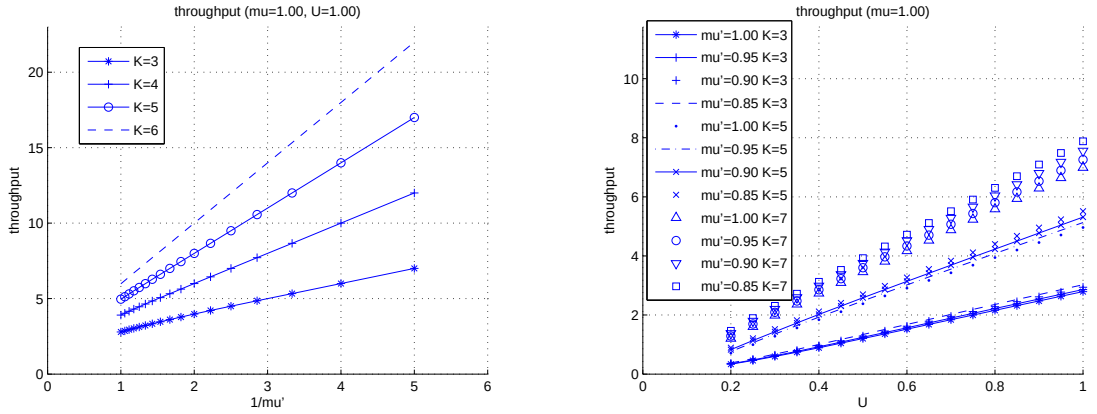
Nesta seção, consideramos a vazão do sistema em cenários nos quais, quando *peers* que possuem todos os blocos menos o bloco mais raro servem a taxa $\mu'$, onde $\mu' < \mu$. Esta política é motivada pelo fato de que queremos diminuir a taxa com que *peers* do *one club* servem *peers* que possuem o bloco mais raro para que estes últimos permaneçam mais tempo no sistema e possam servir o bloco mais raro. De fato, como mostram as Figuras 7 e 8, o aumento da vazão ocorre quando os *peers* que possuem todos os blocos menos um servem com uma taxa reduzida ($\mu'$). Na medida em que $\mu'$ diminui, a vazão aumenta. É importante notar que o aumento da vazão é significativo, a partir de uma simples modificação de política de fácil implementação, sendo naturalmente compatível com os incentivos dos *peers* em economizar banda. Enfatizamos que esse resultado é contra-intuitivo e foi obtido a partir dos modelos que construímos.

O aumento de vazão com a diminuição da taxa de serviço dos *peers* com $K-1$ blocos não é ilimitado quando $\mu' \to 0$. A partir de um valor pequeno de $\mu'$ a vazão volta a diminuir (Figura 8(a)). Em particular, se $K=2$ e $\mu'=0$, a vazão é igual a $U/2$, dado que todos os blocos serão servidos apenas pelo servidor (o sistema se degenera em um sistema cliente-servidor).

A Figura 8(b) mostra a diferença entre o cenário onde todos os *peers* servem com taxa $\mu = 1.0$ e o cenário onde os *peers* que possuem todos os blocos menos um servem

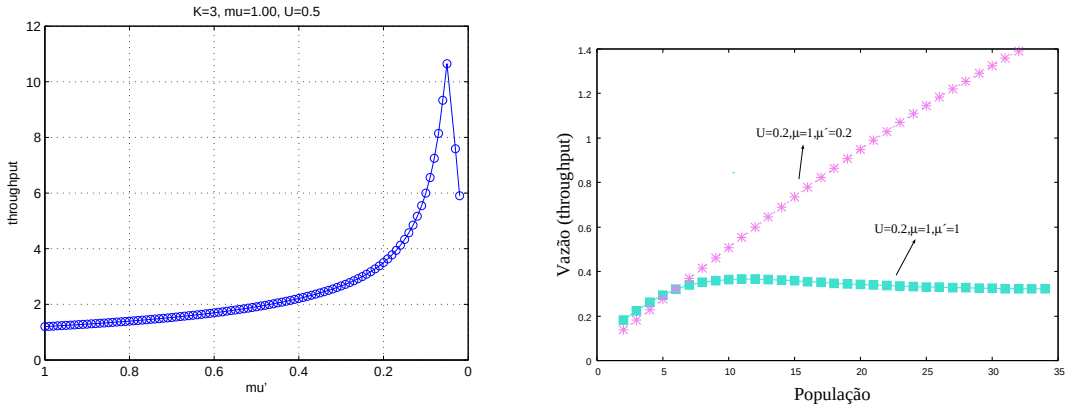
com taxa $\mu' = 0.2$, isto é, $1/5$ da taxa de *upload*. Pode-se notar que a vazão alcançada para populações maiores que 10 usuários é significativamente maior no caso de uma taxa de serviço diferenciada, neste exemplo $\mu = 1.0$ e $\mu' = 0.2$.

Como regra geral, é bastante vantajoso reduzir a taxa de *upload* $\mu'$ como indicado acima de forma a aumentar significativamente a região de estabilidade do sistema. Analogamente, adotando-se uma taxa diferenciada $\mu'$, pode-se reduzir a taxa $U$ que o servidor dedica a um *swarm* e, mesmo assim, manter o sistema na região de estabilidade.




**(a)** **(b)**

**Figura 7. Vazão quando os *peers* que possuem todos os blocos menos um modificam suas taxas de serviço.**




**(a)** **(b)**

**Figura 8. Vazão quando os *peers* que possuem todos os blocos menos um modificam suas taxas de serviço.**

## 4.4. Protegendo *peers* recém chegados

Nesta seção, ilustramos a importância de "proteger" *peers* recém chegados de tal forma que uma fração deles receba, garantidamente, blocos do servidor, antes de outros blocos mais populares. Consequentemente esses recém chegados obterão o bloco raro e integrarão o conjunto dos *gifted peers*. Por "proteção" entendemos que o *tracker* não anuncia

a chegada de um novo *peer* para os demais caso tenha possibilidade de servi-lo. A Figura 9 mostra a vazão do sistema para esse cenário e para os parâmetros $K = 3$, $U = 0.1$ e $\mu = \mu' = 10$. A razão do aumento de vazão é resumidamente explicada a seguir.

Suponha o atendimento sem a modificação acima descrita. Caso a taxa do servidor seja bem pequena em relação a taxa de *upload* dos *peers*, os membros do *one club* acabam por suprir os blocos dos novos *peers*, e esses entram no *one club* antes do servidor servir o bloco raro. Neste caso a vazão do sistema converge para $U = 0.1$ (curva azul).

Considere o cenário acima. Um *peer* recém chegado é protegido pelo *tracker* do restante da população, não o anunciando para os outros membros até que ele termine de receber o primeiro bloco do servidor (curva vermelha). Nesse caso, a vazão $\lambda_s$ é de aproximadamente 0.25, alcançando o pico de $KU = 0.3$ previsto pela Proposição 3.1.

A política de proteção dos *peers* recém chegados é interessante principalmente quando $\mu >> U$. Neste caso, os *peers* estarão servindo os blocos mais populares aos recém chegados com uma taxa muito maior que o servidor tem para servir o bloco mais raro. A introdução da "proteção" descrita favorece a vazão. Note que quando a capacidade do servidor para o *swarm* é comparável ou maior que a de *upload* dos *peers*, isto é, $\mu \leq U$ essa política não se faz necessária e a vazão $\lambda_s$ pode ser atingida normalmente (vide Figura 6(a)). Entretanto, a mudança de política permite que o servidor dedique menos capacidade a um *swarm*, e possa portanto atender a um maior número de *swarms*.

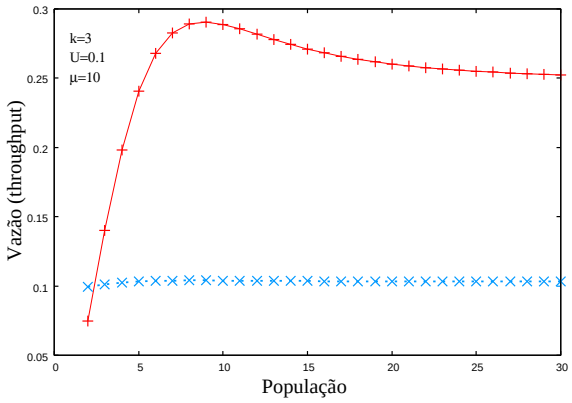



**Figura 9. Protegendo *peers* recém chegados, a vazão cresce substancialmente.**

## 5. Trabalhos relacionados

A literatura sobre sistemas P2P é vasta [Nunez-Queija and Prabhu 2008, Massoulie and Vojnovic 2006, Qiu and Sang 2008, Leskela et al. 2010, Massoulie and Twigg 2008, Norros et al. 2011, Reittu 2009, Rocha et al. 2009, Menasché et al. 2012, Mathieu and Reynier 2006] e sobre a estabilidade dos mesmos vem crescendo desde a publicação do artigo seminal de Hajek e Zhu [Hajek and Zhu 2010]. Entretanto, não é do nosso conhecimento nenhum trabalho anterior que tenha considerado os problemas de estabilidade levantados por Hajek e Zhu [Hajek and Zhu 2010, Zhu and Hajek 2011], e em particular o *missing piece syndrome*, no cálculo da vazão de sistemas P2P. Como indicamos no presente artigo, levar em conta estas características chaves dos sistemas P2P é fundamental na determinação da capacidade em atender os usuários (e.g., vazão).

A análise de sistemas P2P usando modelos de fluido assume que o arquivo pode ser dividido em infinitos pedaços [Massoulie and Twigg 2008, Massoulie and Vojnovic 2006]. Como indicado por Hajek e Zhu [Hajek and Zhu 2010], essa aproximação nem sempre é satisfatória. Desde então, alguns trabalhos analisam políticas de *peers* e servidores levando em conta o fato de que o arquivo possui uma quantidade finita de pedaços [Oguz et al. 2012, Zhu and Hajek 2013, Oguz et al. 2012]. Entretanto, nenhum desses trabalhos considerou a vazão de tais sistemas.

A política de servir primeiro o *peer* com menos blocos (*most deprived peer policy*) foi inicialmente proposta por Massoulie et al. [Massoulie and Vojnovic 2006]. Trabalhos anteriores consideraram que *peers* não tinham informação sobre o número de réplicas de cada pedaço no sistema [Nunez-Queija and Prabhu 2008, Reittu 2009, Norros et al. 2011, Leskela et al. 2010]. Neste artigo, por outro lado, assumimos que o servidor tem tal informação. Acreditamos que, na prática, o servidor possa obter essa informação, mesmo que de forma aproximada, por meio de um *tracker*.

As implicações das políticas de serviço dos servidores no contexto de estabilidade e vazão de sistemas P2P foi primeiro estudada em [Menasché et al. 2011] e [Menasché et al. 2012]. O presente artigo estende [Menasché et al. 2011, Menasché et al. 2012] de várias formas, incluindo: (1) a proposta de modelos Markovianos mais elaborados e um modelo de filas para estimar a vazão assintótica do sistema para populações grandes; (2) novas política para *peers* e o servidor que proporcionam aumento da vazão do sistema.

## 6. Conclusão

Devido a sua escalabilidade, robustez e eficiência, sistemas P2P são responsáveis por uma significativa fatia do tráfego da Internet, e constituem um dos pilares de algumas das novas arquiteturas de redes orientadas a conteúdo [Jacobson et al. 2009]. Embora estes sistemas sejam extremamente populares, suas limitações fundamentais ainda não são bem compreendidas. Neste artigo, apresentamos novos resultados que permitem quantificar a vazão de sistemas P2P. Os modelos que desenvolvemos contribuem para o entendimento da capacidade de serviço dos sistemas P2P. Em particular, mostramos como minorar o efeito da síndrome do bloco faltante que ocorre quando a taxa de chegada dos *peers* é alta em relação a capacidade do servidor dedicada ao *swarm*. Propomos também novas políticas de atendimento do servidor e dos *peers* que levam a um significativo aumento da eficiência do sistema.

## Referências

de Souza e Silva, E., Leão, R., Menasché, D., and Rocha, A. (2013). On the interplay between content popularity and performance in P2P systems. In *Quantitative Evaluation of Systems*, pages 3–21. Springer.

de Souza e Silva, E. and Muntz, R. (1992). *Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação*. VIII Escola de Computação - SBC.

Hajek, B. and Zhu, J. (2010). The missing piece syndrome in peer-to-peer communication. In *IEEE ISIT*.

Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and Braynard, R. (2009). Networking named content. In *Proc. of the 5th Int. Conf. on Emerging Networking Experiments and Technologies*, pages 1–12. ACM.

Leskela, L., Robert, P., and Simatos, F. (2010). Stability properties of linear file sharing networks. In *Advances in Applied Probability*, volume 42(3), pages 834–854.

Massoulie, L. and Twigg, A. (2008). Rate-optimal schemes for peer-to-peer live streaming. *Performance Evaluation*, 65(11-12):804–822.

Massoulie, L. and Vojnovic, M. (2006). Coupon replication systems. In *SIGMETRICS'06*.

Mathieu, F. and Reynier, J. (2006). Missing piece issue and upload strategies in flash-crowds and p2p-assisted filesharing. In *AICT/ICIW*.

Menasché, D. S., de A. Rocha, A. A., de Souza e Silva, E., Towsley, D., and Leão, R. M. M. (2011). Implications of peer selection strategies by publishers on the performance of p2p swarming systems. *ACM SIGMETRICS Performance Evaluation Review*, 39(3):55–57.

Menasché, D. S., Rocha, A., de Souza e Silva, E., Leão, R., and Towsley, D. (2012). Stability of peer-to-peer swarming systems. In *SBRC*, pages 161–174.

Murai, F., Rocha, A., Figueiredo, D., and de Souza e Silva, E. (2012). Heterogeneous download times in a homogeneous bittorrent swarm. *Computer Networks*, 56:1983–2000.

Norros, I., Reittu, H., and Eirola, T. (2011). On the stability of two-chunk file-sharing systems. *Queueing Systems*, 67(3):183–206.

Nunez-Queija, R. and Prabhu, B. (2008). Scaling laws for file dissemination in p2p networks with random contacts. In *IWQoS*, pages 75–79.

Oguz, B., Anantharam, V., and Norros, I. (2012). Stable, distributed p2p protocols based on random peer sampling. In *50th Allerton Conf. on Comm., Control and Comput.*, pages 915–919. IEEE.

Qiu, D. and Sang, W. (2008). Global stability of peer-to-peer file sharing systems. In *Computer Communications*, volume 31(2).

Reittu, H. (2009). A stable random-contact algorithm for peer-to-peer file sharing. In *IWS0S*.

Rocha, A. A., Menasché, D. S., Towsley, D. F., and Venkataramani, A. (2009). On P2P systems for enterprise content delivery. In *SBRC*, pages 379–392.

Xia, R. and Muppala, J. (2010). A survey of bittorrent performance. *IEEE Communications Surveys & Tutorials*, 12(2):140–158.

Zhu, J. and Hajek, B. (2011). Stability of peer to peer systems. In *PODC*.

Zhu, J. and Hajek, B. (2013). Tree dynamics for peer-to-peer streaming. *arXiv preprint arXiv:1308.1971*.

# Mecanismos de Incentivo para um Sistema P2P IPTV

**Daniel A. G. Manzato, Nelson L. S. da Fonseca**

[1]Instituto de Computação – Universidade Estadual de Campinas
Campinas – SP – Brasil

{dmanzato,nfonseca}@ic.unicamp.br

***Resumo.*** *Sistemas baseados em redes P2P dependem amplamente da cooperação dos usuários para se tornarem eficientes e escaláveis. Tal problema se agrava com a transmissão de conteúdo com restrições temporais, como é o caso dos sistemas P2P IPTV. Devido ao comportamento racional dos usuários, que tendem a cooperar somente com a obtenção de benefícios em troca, mecanismos de incentivo se fazem fundamentais neste tipo de sistema. Este trabalho propõe mecanismos de incentivo para um sistema P2P IPTV avaliado anteriormente pelos autores.*

***Abstract.*** *P2P systems rely broadly on user cooperation to achieve performance and scalability, a problem which is aggravated with the transmission of time sensitive content, as it is the case of P2P IPTV systems. Since peers act rationally, cooperating only when they obtain benefits in exchange, incentive mechanisms constitute an essential component of such systems. This paper proposes incentive mechanisms to a P2P IPTV system evaluated previously by the authors.*

## 1. Introdução

Sistemas baseados em redes P2P buscam eficiência e escalabilidade através da utilização dos recursos dos usuários, cuja disponibilidade varia em função da ocorrência de comportamentos cooperativos. A dependência de tais comportamentos torna-se ainda mais acentuada em sistemas que entregam conteúdo com restrições temporais, como é o caso dos fluxos de vídeo em sistemas P2P IPTV. Como os usuários desses sistemas comportam-se racionalmente, disponibilizando seus recursos somente com a obtenção de benefícios em troca, o emprego de mecanismos de incentivo constitui um requisito fundamental.

Neste trabalho, propõem-se mecanismos de incentivo para a arquitetura *iPeer TV*, um sistema P2P IPTV avaliado anteriormente pelos autores [Manzato and da Fonseca 2013a, Manzato and da Fonseca 2011, Manzato and da Fonseca 2008b]. Os mecanismos propostos têm como objetivo aliviar a demanda de banda de rede nos servidores da camada de infraestrutura da arquitetura, através da utilização das bandas de saída dos usuários para a distribuição dos fluxos de vídeo. Apesar dos mecanismos propostos serem baseados em padrões de incentivo comuns, a principal contribuição do trabalho está no estudo da arquitetura *iPeer TV* com incentivos. Em trabalhos futuros, mecanismos mais elaborados serão propostos e então comparados aos resultados atuais.

O restante deste artigo está organizado da seguinte forma. Na Seção 2, a arquitetura *iPeer TV* é revisada, enquanto que na Seção 3, os mecanismos de incentivo são propostos. A Seção 4 descreve os experimentos de simulação realizados, enquanto que a Seção 5 avalia os mecanismos propostos. Trabalhos relacionados são apresentados na Seção 6, enquanto que a Seção 7 tece considerações finais.

## 2. Arquitetura *iPeer TV*

A Figura 1 ilustra a arquitetura *iPeer TV*, que emprega redes P2P, múltiplas árvores de distribuição e provê serviços de troca rápida de canais [Manzato and da Fonseca 2008b]. A abordagem de distribuição de conteúdo por múltiplas árvores é empregada com o intuito de se reduzir o tempo de início de reprodução dos fluxos, já que elimina os custos de escalonamento de transmissão envolvidos com as estruturas em *mesh*.

Componentes chamados SPs (do Inglês, *stream providers*) produzem conteúdo televisivo de diferentes canais e geram múltiplos subfluxos (ou descritores) para cada canal, através da codificação MDC (do Inglês, *multiple description coding*) [Goyal 2001]. Apesar do número de descritores poder variar para cada canal, suas larguras de banda são as mesmas após a codificação. Desse modo, um canal com alta qualidade apresenta um número maior de descritores do que um canal com baixa qualidade.

Os SPs são conectados a componentes chamados DSNs (do Inglês, *dedicated super nodes*), que constituem a camada de infraestrutura da arquitetura. Os DSNs são responsáveis por propagar e disponibilizar todos os descritores de todos os canais, bem como coordenar a admissão de novos usuários no sistema. Além disso, eles compensam o deficit de recursos (em termos de banda de rede de admissão) causado pelos usuários menos capacitados. A propagação de conteúdo entre os DSNs ocorre através de *multicast*, buscando alcançar o menor número possível de saltos para que a latência de recepção seja mínima.

Os usuários são classificados em duas categorias, de acordo com suas capacidades: TSNs (do Inglês, *temporary super nodes*) e RNs (do Inglês, *regular nodes*). Os TSNs possuem melhor capacidade e cooperam com as tarefas de gerenciamento de árvores e distribuição de conteúdo; eles são admitidos pelos DSNs. Os RNs, por outro lado, possuem capacidade inferior, são admitidos pelos TSNs, através de múltiplas árvores de distribuição, e cooperam nas árvores em que são nós interiores. Cada RN é admitido como nó interior em uma única árvore, sendo nó folha nas demais. Como cada TSN serve apenas um único canal em qualidade máxima, a seleção de canal de um RN determinará abaixo de qual TSN ele estará admitido.

Além de servir um único canal em qualidade máxima, cada TSN também provê descritores de navegação de outros canais, habilitando, assim, serviços de troca rápida de canais aos RNs. Em cada TSN, dois conjuntos de árvores são utilizados: um para a transmissão dos descritores do canal assistido e outro para a transmissão dos descritores de navegação [Manzato and da Fonseca 2013b, Manzato and da Fonseca 2010a].

## 3. Mecanismos de Incentivo Propostos

Nesta seção, propõem-se dois mecanismos de incentivo com cinco variações para a arquitetura *iPeer TV*.

### 3.1. Visão Geral e Aplicabilidade dos Padrões de Incentivo

Padrões de incentivo são definidos na literatura como forma de mapear as características e comportamentos comuns dos diversos mecanismos de incentivo existentes. Uma abordagem interessante é apresentada em [Manzato and da Fonseca 2010b, Obreiter and Nimis 2003], que descreve os padrões de incentivo com base em estudos econômicos da sociedade real. De acordo com essa abordagem, os padrões de incentivo podem ser

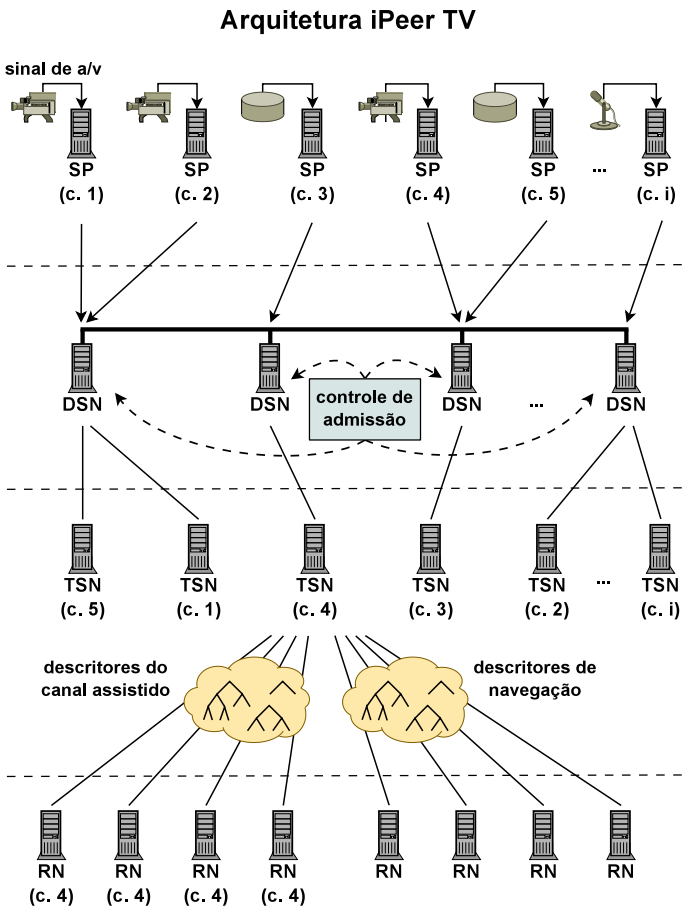



**Figura 1. Arquitetura *iPeer TV*, que emprega redes P2P, múltiplas árvores e provê serviços de troca rápida de canais [Manzato and da Fonseca 2008b].**

amplamente classificados como sendo baseados em confiança ou em comércio. Nos baseados em confiança, as entidades cooperam quando confiam nos consumidores de seus serviços. Tal confiança pode ser estática, i.e., não varia em função do tempo, ou dinâmica, i.e., varia de acordo com o comportamento das entidades. No primeiro caso, há o padrão de incentivo coletivismo (do Inglês, *the collective pattern*), no qual as entidades ajudam-se mutuamente por compartilharem dos mesmos ideais, que são comuns ao grupo. No segundo caso, há o padrão de incentivo comunidade (do Inglês, *the community pattern*), no qual interesses individuais são levados em consideração e as confiabilidades das entidades são representadas através de reputações variáveis, sendo proporcionais aos montantes de cooperação já realizados. Nesses sistemas de reputação, é comum uma entidade cooperar somente quando o consumidor de seus serviços apresenta uma reputação maior ou igual à sua.

Nos padrões de incentivo baseados em comércio, as entidades cooperam para receber uma remuneração explícita. Essa remuneração pode ser redimida imediatamente, na forma de troca direta de serviços, ou posteriormente, na forma de uma promessa de retorno de serviço que será honrada no futuro pelo próprio consumidor ou por uma entidade terceira em seu favor. No primeiro caso, há o padrão de incentivo escambo (do Inglês, *the barter trade pattern*), no qual as entidades cooperam enquanto usufruem da cooperação de outras entidades. Esse padrão apresenta diversas vantagens sobre os demais: as entida-

des não precisam revelar suas identidades (anonimato); a remuneração é garantida mesmo quando um dos participantes se desconecta abruptamente (persistência); um grande número de participantes não compromete a eficiência do mecanismo (escalabilidade); e nenhuma entidade terceira, tal como um banco, é requerida para que a remuneração aconteça (localidade) [Manzato and da Fonseca 2010b]. No entanto, dado que as entidades podem preferir não redimir suas remunerações de maneira imediata, esse padrão pode não ser apropriado para algumas aplicações. No segundo caso, há os padrões de incentivo promissória própria, promissória de terceiro, sistema bancário e notas bancárias (do Inglês, *the bearer notes, bearer bills, banking, and banknotes patterns*). Todos eles promovem um desacoplamento temporal entre cooperação e remuneração, com a principal diferença sendo a forma de promessa utilizada. O mais comum é o padrão sistema bancário, no qual contas individuais são mantidas em um sistema distribuído ou centralizado que assume o papel de um banco.

O padrão de incentivo coletivismo é, evidentemente, o menos apropriado para a arquitetura *iPeer TV*, uma vez que, neste tipo de aplicação, os usuários não apresentam qualquer tipo de senso coletivo no grupo. O padrão comunidade também não seria uma boa escolha, já que apresenta escalabilidade limitada devido à dificuldade envolvida em se propagarem reputações em um grupo com muitos participantes [Manzato and da Fonseca 2010b], como tipicamente é o caso de um sistema IPTV. Na verdade, esse padrão poderia ser empregado para apenas um subconjunto dos participantes do grupo, tais como os TSNs da arquitetura *iPeer TV* [Manzato and da Fonseca 2008b]. Entretanto, uma vez que se busca, neste trabalho, um mecanismo de incentivo primário com o objetivo de aumentar a cooperação de todos os usuários do sistema, opta-se por não empregar nenhum padrão de incentivo baseado em confiança.

O padrão de incentivo escambo é uma opção interessante para a arquitetura *iPeer TV* não só pelas suas vantagens sobre os demais padrões, mas também pelo fato dele ser apropriado para sistemas P2P de fluxo de mídia com múltiplas árvores de distribuição [Manzato and da Fonseca 2008a]; nesse tipo de sistema, os usuários repassam os descritores recebidos a outros participantes do grupo, enquanto estiverem conectados. Adicionalmente, esse padrão já comprovou sua eficiência em outros sistemas que permitem remuneração síncrona [Cohen 2003]. Sendo assim, ele é escolhido para este trabalho.

Considerando-se os padrões de incentivo baseados em comércio e com remuneração postergada, o promissória própria e o promissória de terceiro não apresentam a mesma persistência do padrão escambo. Isso significa que uma entidade poderia usufruir de cooperação alheia, emitir uma promissória própria como pagamento e então se desconectar da rede, sem nunca honrar a promessa feita. De forma análoga, uma entidade poderia emitir uma promissória de terceiro cujo devedor também se desconectasse prematuramente, sem a devida execução da promessa realizada. Em um sistema dinâmico tal como o *iPeer TV*, no qual as chegadas e partidas dos usuários são frequentes, a característica de persistência deve ser um requisito a ser considerado em qualquer projeto de mecanismo de incentivo. Os padrões de incentivo sistema bancário e notas bancárias são capazes de promover tal persistência, através do emprego de uma entidade terceira confiável (tipicamente um banco), que assume os papeis de devedor das promissórias ou de emissor das notas bancárias, respectivamente. No entanto, o padrão notas bancárias demanda mecanismos adicionais para se validar a autenticidade das notas pré-emitidas [Manzato

and da Fonseca 2010b], o que aumentaria a complexidade do sistema, bem como deixaria uma brecha para fraudes. Sendo assim, o padrão sistema bancário é escolhido como opção alternativa ao padrão escambo.

Nas próximas subseções, os padrões escambo e sistema bancário são considerados para a derivação de dois mecanismos de incentivo distintos para a arquitetura *iPeer TV*.

### 3.2. Mecanismo de Incentivo *Esc*

O primeiro mecanismo de incentivo proposto para a arquitetura *iPeer TV* denomina-se *Esc* e é baseado no padrão escambo. Neste mecanismo, a taxa de consumo de cada usuário é limitada proporcionalmente pela sua taxa de cooperação, sendo que a relação entre elas é determinada por uma razão. Seja $cooperacao$ a taxa de cooperação e $consumo$ a taxa de consumo (ambas em Kbps). A razão entre consumo e cooperação, $RCC$, pode ser expressa da seguinte forma:

$$RCC = \frac{consumo}{cooperacao} \tag{1}$$

Na arquitetura *iPeer TV*, a taxa de consumo de um usuário é determinada pelo número total de descritores recebidos, incluindo os dos canais assistido, antigo, adjacentes e populares [Manzato and da Fonseca 2013a]. Como cada árvore de distribuição transporta um único descritor no sistema, a taxa de consumo de um RN pode ser calculada pelo número total de árvores nas quais ele está admitido, incluindo as de seus TSNs primário e secundário [Manzato and da Fonseca 2013a]; a taxa de consumo de um TSN pode ser calculada pelo número total de descritores que ele está recebendo diretamente do DSN que o admitiu. Já a taxa de cooperação de um usuário é determinada pelo número total de nós filhos para os quais ele está repassando descritores. No caso de um RN, esse repasse ocorre nas árvores de distribuição em que ele foi admitido como nó interior; no caso de um TSN, o conjunto de nós filhos inclui todos os RNs admitidos diretamente por ele.

O ajuste das taxas de consumo e de cooperação é realizado através da redução do maior valor medido entre eles. O Algoritmo 1 é proposto para a realização desse ajuste.

**Algoritmo 1.** Ajuste das taxas de consumo e de cooperação para o mecanismo *Esc*.

**Require:** $RCC > 0$
  **for all** $usuarios\ conectados$ **do**
    **if** $consumo > RCC * cooperacao$ **then**
      $consumo \leftarrow RCC * cooperacao$
    **else**
      $cooperacao \leftarrow consumo/RCC$
    **end if**
  **end for**

Três instâncias deste mecanismo de incentivo são propostas, variando-se a razão entre consumo e cooperação de 0.5 a 2. O limite superior de 2 foi estabelecido para que pelo menos a metade da demanda total de banda de rede do sistema fosse suprida pelos usuários, já que valores maiores degenerariam a arquitetura para cliente-servidor. O limite inferior de 0.5 para a razão foi estabelecido para se conter o desperdício de recursos, com uma banda de rede total excedente não ultrapassando 100% da demanda total. Sendo assim, no mecanismo *Esc2*, a razão é fixada em 2 com o objetivo de se permitir um

consumo em uma taxa que seja até o dobro daquela de cooperação; no mecanismo *Esc05*, a razão é fixada em 0.5 com o intuito de se permitir uma taxa de consumo de apenas a metade daquela cooperada; e no mecanismo *Esc1*, a razão é fixada em 1 para se permitir uma taxa de consumo que seja equivalente àquela de cooperação.

### 3.3. Mecanismo de Incentivo *Rem*

O segundo mecanismo de incentivo proposto para a arquitetura *iPeer TV* é denominado *Rem* e se baseia no padrão sistema bancário. Neste mecanismo, não se mantém nenhuma relação entre as taxas de consumo e de cooperação dos usuários. Ao invés disso, cada usuário recebe uma conta individual que acumula, periodicamente, a diferença entre os montantes de dados consumidos e cooperados durante o período considerado. Por exemplo, se um usuário se mantiver consumindo em uma taxa equivalente à de sua cooperação, o seu saldo permanecerá inalterado; se ele consumir em uma taxa maior que à de cooperação, créditos serão deduzidos periodicamente de sua conta, que poderá apresentar, eventualmente, um saldo negativo; por outro lado, se ele consumir em uma taxa menor que à de cooperação, créditos serão adicionados periodicamente em sua conta, que tenderá a apresentar um saldo positivo. Seja $cooperacao$ a taxa de cooperação, $consumo$ a taxa de consumo (ambas em Kbps) e $DP$ a duração do período considerado para os cálculos periódicos (em segundos). O montante de créditos relativo ao período, $creditos$, a ser somado na conta de um usuário, pode ser expresso da seguinte forma:

$$creditos = \frac{(cooperacao - consumo) * DP}{8} \quad (2)$$

Nota-se que, quando a taxa de consumo for maior que a de cooperação, o montante de créditos calculado para o período será negativo, produzindo um débito na conta do usuário. Permitir ou não saldos negativos é uma opção de sistema, assim como o estabelecimento de um valor limite para esses saldos. Quando um limite negativo pré-estabelecido é atingido, a taxa de consumo do respectivo usuário deve ser ajustada para um valor menor ou igual ao de sua taxa de cooperação, de forma a se limitar o deficit de recursos no sistema. Neste caso, o usuário tem a opção de passar a cooperar em uma taxa maior novamente para voltar a acumular créditos e, com isso, ter a sua taxa de consumo regular restabelecida.

O Algoritmo 2 é proposto para o cálculo periódico dos saldos dos usuários. É importante ressaltar que $taxa\_reduzida$ corresponde a um valor utilizado para o ajuste da taxa de consumo quando o saldo do usuário se torna negativo; $valor\_limite$ estabelece um valor mínimo requerido para o saldo do usuário antes que a sua taxa de consumo possa ser restabelecida em $taxa\_regular$; e $saldo\_negativo$ é uma propriedade de sistema que indica se saldos negativos são ou não permitidos.

Duas instâncias deste mecanismo de incentivo são propostas. No mecanismo *RemP*, saldos negativos não são permitidos com o objetivo de se limitar o deficit de recursos no sistema. No mecanismo *RemN*, saldos negativos são permitidos com o intuito de se aumentar o desempenho do sistema.

## 4. Experimentos de Simulação

Um simulador de eventos discretos para sistemas P2P IPTV foi desenvolvido com o objetivo de se avaliar o desempenho dos mecanismos propostos na arquitetura *iPeer TV*. É de conhecimento dos autores que nenhum dos simuladores existentes implementando as

**Algoritmo 2.** Cálculo periódico dos saldos dos usuários para o mecanismo *Rem*.

**Require:** $taxa\_reduzida \leq cooperacao$
  **for all** $usuarios\ conectados$ **do**
    $creditos \leftarrow (cooperacao - consumo) * DP/8$
    **if** $(saldo + creditos) < 0$ **and not** $saldo\_negativo$ **then**
      $saldo \leftarrow 0$
      $consumo \leftarrow taxa\_reduzida$
    **else**
      $saldo \leftarrow saldo + creditos$
      **if** $consumo = taxa\_reduzida$ **and** $saldo > valor\_limite$ **then**
        $consumo \leftarrow taxa\_regular$
      **end if**
    **end if**
  **end for**

operações fundamentais de serviços de IPTV estava disponível para uso na ocasião do desenvolvimento deste trabalho. Apesar de uma ferramenta ter sido desenvolvida em [Qiu et al. 2009] para a geração de tráfego sintético, ela também não estava disponível para uso. Sendo assim, o simulador desenvolvido demandou a implementação das operações fundamentais de serviços de IPTV, o que foi feito a partir dos modelos estatísticos apresentados em [Cha et al. 2008] e [Qiu et al. 2009]. A sua validação considerou as diferentes características do tráfego sintético gerado, o corretismo de processamento de cada tipo de evento e os efeitos resultantes no sistema, tais como número total de usuários em função do tempo, padrões de tráfego diários, taxas de chegada, partida e troca de canais, popularidades e probabilidades de transição entre os canais, e distribuição das diferentes classes de conexão entre os usuários.

As taxas de chegada foram modeladas através de um processo não estacionário composto por sequências de processos de chegada Poisson estacionários, cada uma delas tendo a duração de 15 minutos [Veloso et al. 2002]. As taxas de cada sequência variaram entre 5 e 2000 chegadas por minuto e foram definidas, para cada período estacionário, de forma a refletir o número total de usuários de um sistema real [Cha et al. 2008]. Com isso, reproduziram-se os padrões diários de uso de um sistema real, que apresentam dois grandes picos por volta de 3PM e 10PM, tendo um pico menor por volta de 8AM [Cha et al. 2008]. Os tempos de sessão seguiram uma distribuição Lognormal com os parâmetros $\mu = 6.351$ e $\sigma = 2.01$ [Veloso et al. 2002], tendo em média 4320 segundos (1.2 horas) [Cha et al. 2008]. Para modelar os tempos de permanência em canais, usou-se uma amostragem de valores reais disponibilizada pelos autores de [Cha et al. 2008], a partir da qual se construiu um histograma. A média e a mediana desses valores são, respectivamente, 14.8 minutos e 8 segundos. O tempo total simulado foi de 7 dias, gerando mais de 62 milhões de eventos de troca de canais que foram analisados.

O número de canais utilizado foi 105 [Cha et al. 2008]. As popularidades e as probabilidades de transição entre eles foram derivadas de informações e dados obtidos de [Cha et al. 2008] e [Qiu et al. 2009]. Diferenciaram-se as trocas de canais que resultaram em permanência ou somente em passagem pelos canais selecionados; para essa separação, usou-se um limite de tempo de 60 segundos [Cha et al. 2008]. Para simular a

**Tabela 1. (a) Cenários considerados, variando-se a largura de banda dos fluxos e o número de árvores de distribuição; (b) Experimentos sem incentivo, variando-se o nível de altruísmo ($NA$); (c) Mecanismos avaliados, variando-se a razão entre consumo e cooperação ($RCC$) e a opção de saldos negativos ($SN$).**

(a)

| Cenário | Fluxo (Kbps) | Árvs. (#) |
|---|---|---|
| S300T16 | 300 | 16 |
| S300T32 | 300 | 32 |
| S1024T16 | 1024 | 16 |
| S1024T32 | 1024 | 32 |
| S2048T16 | 2048 | 16 |
| S2048T32 | 2048 | 32 |

(b)

| Experimento | Variação |
|---|---|
| Alt0 | $NA = 0\%$ |
| Alt30 | $NA = 30\%$ |
| Alt50 | $NA = 50\%$ |
| Alt70 | $NA = 70\%$ |

(c)

| Mecanismo | Variação |
|---|---|
| Esc05 | $RCC = 0.5$ |
| Esc1 | $RCC = 1.0$ |
| Esc2 | $RCC = 2.0$ |
| RemP | $SN = false$ |
| RemN | $SN = true$ |

heterogeneidade de banda de rede dos usuários, criaram-se diferentes classes de conexão, definidas de acordo com estatísticas de padrões de acesso da Internet Brasileira [Teleco 2014]. As classes são: (i) *Provedor 1:* ADSL 1 Mbps, ADSL 2 Mbps, ADSL 5 Mbps, ADSL 10 Mbps, ADSL 15 Mbps e ADSL 20 Mbps; (ii) *Provedor 2:* HFC 1 Mbps, HFC 10 Mbps, HFC 20 Mbps e HFC 100 Mbps; (iii) *Provedor 3:* ADSL 1 Mbps, ADSL 2 Mbps, ADSL 4 Mbps e ADSL 10 Mbps; e (iv) *Provedor 4:* DSL 5 Mbps, DSL 10 Mbps, DSL 15 Mbps, DSL 35 Mbps, DSL 50 Mbps e DSL 100 Mbps.

A largura de banda dos fluxos variou entre 300, 1024 e 2048 Kbps [Hei et al. 2007]. O número de árvores de distribuição utilizadas variou entre 16 e 32 [Padmanabhan et al. 2003]. Para modelar os tempos de início de reprodução dos fluxos, compostos pelo período necessário para admissão nas árvores de distribuição e *bufferização* dos quadros de vídeo, usou-se uma distribuição uniforme no intervalo $[5, 20]$ segundos [Liu et al. 2008, Hei et al. 2007].

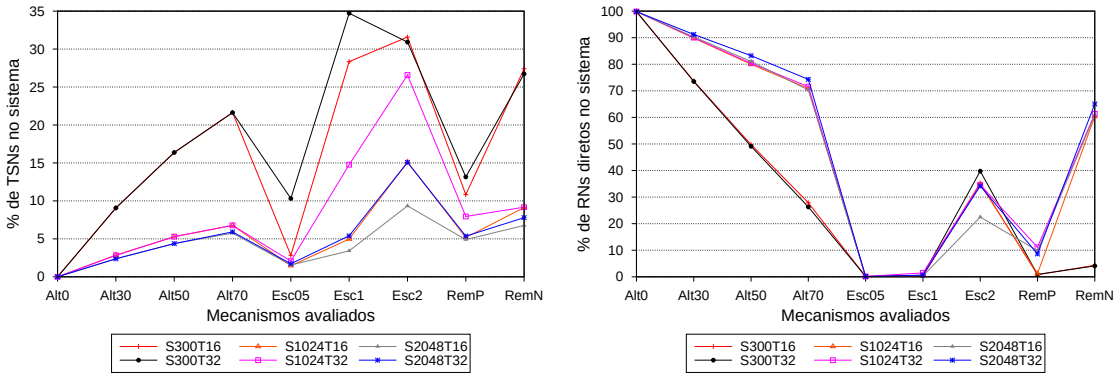
As seguintes métricas foram coletadas nas simulações: (i) *TSNs no sistema*, i.e., percentagem de usuários atuando como TSNs (com pelo menos um nó filho admitido); (ii) *RNs diretos no sistema*, i.e., percentagem de usuários admitidos diretamente por DSNs (devido à exaustão de TSNs); (iii) *Banda de rede total dos servidores*, i.e., capacidade de admissão máxima requerida de todos os DSNs conjuntamente (somatório de suas bandas de saída); (iv) *Falhas de admissão em árvores*, i.e., percentagem de falhas nas operações de admissão em árvores; (v) *Trocas de canais imediatas*, i.e., percentagem de eventos de troca de canais sem latência; e (vi) *Qualidade de fluxo*, i.e., número médio de descritores recebidos para o canal assistido pelos usuários durante suas sessões.

## 5. Avaliação dos Mecanismos Propostos

Diferentes cenários foram considerados variando-se a largura de banda dos fluxos dos canais existentes e o número de árvores de distribuição utilizadas. O objetivo era investigar o desempenho dos mecanismos em diferentes situações de contenção de banda de rede. A Tabela 1 mostra as configurações dos diferentes cenários e dos experimentos realizados. Como os intervalos de confiança são pequenos, eles foram omitidos dos gráficos por questões de clareza.

### 5.1. Sistema sem Mecanismo de Incentivo

Em cada cenário, realizaram-se experimentos com o sistema sem nenhum mecanismo de incentivo empregado, variando-se o nível de altruísmo entre 0% e 70%, i.e., a percentagem dos usuários que cooperam *voluntariamente* com suas bandas de saída. As Figuras 2(a) e 2(b) mostram, respectivamente, as percentagens de TSNs e de RNs diretos no sistema. No experimento *Alt0*, nenhuma cooperação ocorreu e o sistema degenerou para uma arquitetura cliente-servidor, com todas as admissões sendo efetuadas pelos DSNs. De fato, nesse caso especifico, a percentagem de TSNs é zero (Figura 2(a)) e a de RNs diretos é 100% (Figura 2(b)). Conforme o nível de altruísmo aumenta, nos experimentos *Alt30*, *Alt50* e *Alt70*, nota-se que a percentagem de TSNs também aumenta (Figura 2(a)), juntamente com uma diminuição da percentagem de RNs diretos (Figura 2(b)). Esse fato evidencia um número maior de usuários sendo admitidos por outros usuários, ao invés de serem admitidos pelos servidores. Adicionalmente, percebe-se que esse efeito é mais acentuado nos cenários com fluxo pequeno (300 Kbps), já que eles demandam menos das bandas de saída limitadas dos usuários.

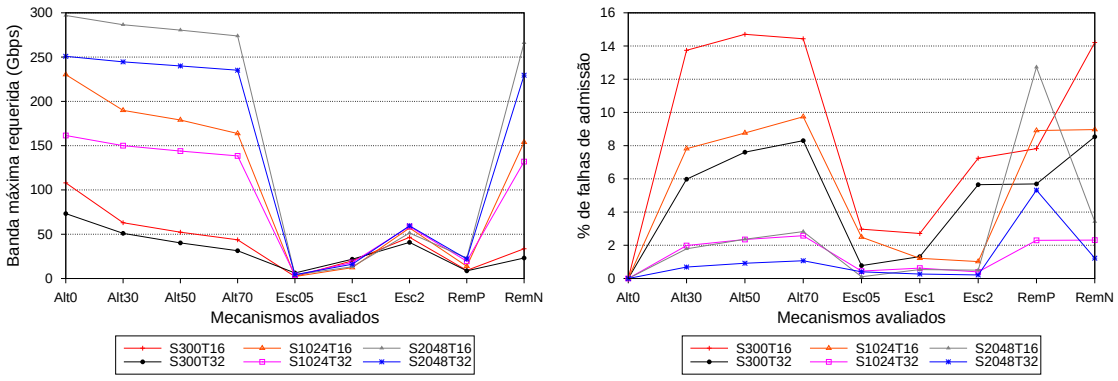



(a) (b)

**Figura 2. (a) Percentagem de usuários atuando como TSNs; (b) Percentagem de usuários admitidos diretamente por DSNs.**

Como consequência do aumento do nível de altruísmo, a demanda de banda de rede nos DSNs diminui. A Figura 3(a) ilustra essa demanda com os resultados da métrica banda de rede total dos servidores. De fato, o experimento *Alt0* apresentou os maiores valores para essa métrica em todos os cenários, já que a arquitetura degenerou para cliente-servidor, enquanto que os experimentos *Alt30*, *Alt50* e *Alt70* tiveram valores decrescentes. Uma outra consequência do aumento do nível de altruísmo é que, dado que um número maior de usuários deixa de ser admitido diretamente pelos servidores, o número de falhas de admissão em árvores, decorrentes da exaustão da banda de saída dos TSNs, aumenta. A Figura 3(b) mostra a percentagem dessas falhas de admissão. Como pode ser visto, o experimento *Alt0* não apresentou nenhuma falha, já que todas as admissões foram realizadas pelos servidores, enquanto que os experimentos *Alt30*, *Alt50* e *Alt70* resultaram em números crescentes de falhas.

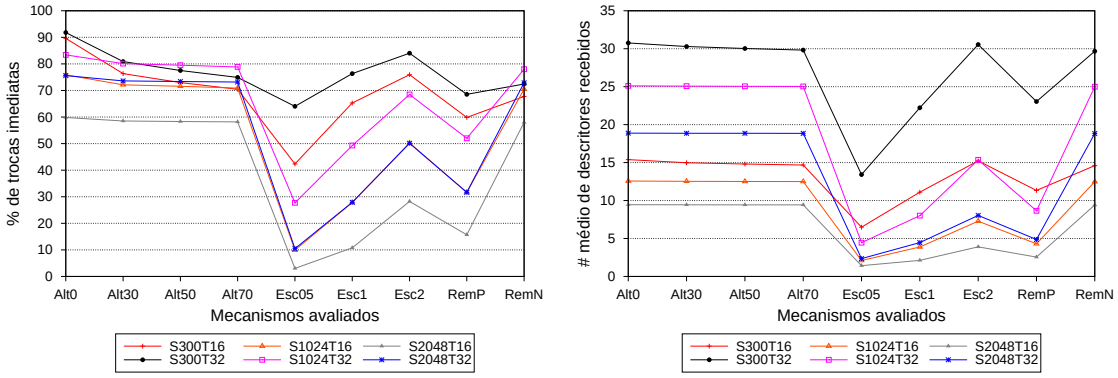
Como as falhas de admissão em árvores provocam indisponibilidade temporária tanto dos descritores do canal assistido quanto dos de navegação, tem-se que a qualidade do fluxo e o desempenho das troca de canais sofrem uma redução proporcional ao nível de altruísmo envolvido. As Figuras 4(a) e 4(b) ilustram, respectivamente, a percentagem




(a) (b)

**Figura 3. (a) Capacidade de admissão máxima requerida de todos os DSNs conjuntamente; (b) Percentagem de falhas nas operações de admissão em árvores.**

de trocas de canais imediatas e a qualidade de fluxo. Como pode ser visto, o experimento *Alt0* produziu, em todos os cenários, os maiores valores para essas duas métricas, já que não ocasionou nenhuma falha de admissão em árvores, enquanto que os experimentos *Alt30*, *Alt50* e *Alt70* apresentaram valores decrescentes para ambas as métricas.




(a) (b)

**Figura 4. (a) Percentagem de eventos de troca de canais sem latência; (b) Número médio de descritores recebidos para o canal assistido pelos usuários durante suas sessões.**

## 5.2. Sistema com os Mecanismos Propostos

Comparando-se o desempenho das três variações do mecanismo de incentivo *Esc*, o *Esc2* produziu resultados consideravelmente melhores do que o *Esc1* e o *Esc05* para as métricas trocas de canais imediatas (Figura 4(a)) e qualidade de fluxo (Figura 4(b)). Isso ocorreu pelo fato da maioria dos enlaces dos usuários ser assimétrica e com uma baixa capacidade de *upload*, já que as razões entre consumo e cooperação de 1 (*Esc1*) e 0.5 (*Esc05*) restringem as capacidades de *download* em apenas 100% e 50%, respectivamente, de tal capacidade de *upload* limitada. Mesmo com os DSNs estando aptos a compensar o deficit de recursos causado pela baixa capacidade de *upload* dos usuários, os consumos ficam limitados pelo mecanismo de incentivo, resultando em um número reduzido de descritores recebidos para os canais assistido e de navegação. Por outro lado, quando uma razão entre consumo e cooperação de 2 é empregada (*Esc2*), os usuários podem consumir até o

dobro de suas capacidades de *upload*, o que melhora seus desempenhos de troca de canais bem como suas qualidade de fluxo. Consequentemente, o mecanismo *Esc2* demanda mais banda de rede dos servidores; apesar disso, como essa demanda adicional é limitada (em torno de 50 Gbps, Figura 3(a)), ele pode ser considerado a melhor opção dentre as três variações do mecanismo de incentivo *Esc*.

Analisando-se agora o desempenho das duas variações do mecanismo de incentivo *Rem*, o *RemN* produziu resultados significativamente melhores do que o *RemP* para as métricas trocas de canais imediatas (Figura 4(a)) e qualidade de fluxo (Figura 4(b)). Na verdade, esse resultado era esperado, uma vez que o objetivo do mecanismo *RemP* era limitar a demanda de banda de rede nos servidores através da cessação do consumo dos usuários quando seus saldos se tornassem negativos. No entanto, uma descoberta interessante foi que o mecanismo *RemP* atuou de maneira excessivamente econômica, sendo que, com o emprego de somente um pouco mais de banda de rede nos servidores (28-43 Gbps, Figura 3(a)), seria possível substituí-lo pelo mecanismo *Esc2*, já que este superou o *RemP* nas métricas trocas de canais imediatas (Figura 4(a)) e qualidade de fluxo (Figura 4(b)). Sendo assim, levando-se em consideração o baixo desempenho do mecanismo *RemP*, pode-se considerar o *RemN* como a melhor opção dentre as duas variações do mecanismo de incentivo *Rem*.

De acordo com as Figuras 4(a) e 4(b), o mecanismo *Esc2* superou o *RemN* nos cenários com fluxo pequeno (300 Kbps), enquanto que o inverso ocorreu nos cenários com fluxos grandes (1024 e 2048 Kbps). Isso foi consequência do fato de que as percentagens de RNs diretos nos cenários com fluxo pequeno são maiores para o mecanismo *Esc2* (35-40% vs. 5%, Figura 2(b)), enquanto que as mesmas nos cenários com fluxos grandes são maiores para o *RemN* (60-65% vs. 22-35%); e esse fato ocorre porque, nos cenários com fluxo pequeno, o mecanismo *Esc2* limita a cooperação dos usuários devido à razão entre consumo e cooperação mantida por ele, aliada a um baixo consumo. Devido a esse fator, há uma maior percentagem de usuários (em comparação ao *RemN*) que não pode ser admitida pelos TSNs existentes e, por isso, acaba sendo admitida diretamente pelos DSNs. Já nos cenários com fluxos grandes, o mecanismo *Esc2* limita o consumo dos usuários devido à mesma razão entre consumo e cooperação, mas agora aliada a uma baixa cooperação resultante dos enlaces assimétricos e com capacidade de *upload* limitada dos usuários. Consequentemente, a maioria dos usuários não pode receber todos os descritores dos canais assistido e de navegação; por esse motivo, como demandam menos banda de rede, eles podem ser admitidos pelos TSNs existentes, o que reduz a percentagem de RNs diretos no sistema (em comparação ao *RemN*).

## 6. Trabalhos Relacionados

Diversos trabalhos foram propostos na área de incentivos à cooperação. Como o número desses trabalhos é vasto, priorizam-se, nesta seção, mecanismos de incentivo que foram projetados para sistemas P2P de fluxo de mídia, já que a arquitetura *iPeer TV* recai nesta categoria. Uma visão geral dos padrões de incentivo existentes pode ser encontrada em [Manzato and da Fonseca 2010b, Obreiter and Nimis 2003]. Em [Manzato and da Fonseca 2008a], um mecanismo foi proposto para o sistema CoopNet [Padmanabhan et al. 2003], buscando tratar tanto o problema da baixa cooperação quanto o das rupturas nas árvores de distribuição causadas pelas constantes conexões e desconexões dos usuários. Para isso, definiu-se um incentivo secundário, baseado em reputação, que estimula

os usuários a permanecerem mais tempo conectados no sistema: quanto maior o tempo de sessão de um usuário, maior será a sua reputação e, com isso, menor será a cooperação exigida dele com base em sua taxa de consumo. O incentivo primário, que trabalha em conjunto com o secundário, implementa uma variação do padrão escambo, fazendo com que as taxas de consumo dos usuários sejam proporcionais às suas taxas de cooperação, de maneira similar ao mecanismo apresentado na Subseção 3.2, porém levando em consideração as reputações individuais dos usuários que determinam diferentes razões entre consumo e cooperação, dentro de um limite preestabelecido. Assim, com o estímulo de cooperar menos, os usuários tendem a permanecer mais tempo conectados, reduzindo as rupturas nas árvores e, com isso, aumentando as qualidades de fluxo gerais obtidas.

Um outro mecanismo empregando o padrão escambo foi proposto em [hua Chu et al. 2004]. Enquanto o mecanismo de [Manzato and da Fonseca 2008a] utiliza a banda excedente proveniente dos usuários conectados há menos tempo para permitir que aqueles outros conectados há mais tempo cooperem menos, o mecanismo de [hua Chu et al. 2004] utiliza a banda excedente proveniente dos usuários melhor capacitados para permitir que os outros com enlaces limitados recebam uma qualidade de fluxo superior àquela determinada por suas cooperações. Para isso, utilizou-se, em [hua Chu et al. 2004], um modelo de tributação linear extraído da literatura de finanças públicas.

Um terceiro mecanismo empregando o padrão escambo foi apresentado em [Liu et al. 2007]; ele difere dos anteriores, bem como dos mecanismos propostos no presente trabalho, no tipo de codificação utilizada: no lugar de MDC, emprega-se a codificação em camadas. Os autores de [Liu et al. 2007] argumentam que esse tipo de codificação é mais promissor do que a MDC, devido à redundância de dados inserida por esta última; por outro lado, os autores também reconhecem os desafios envolvidos na utilização da codificação em camadas, tais como importâncias desiguais e interdependências entre as diferentes camadas, bem como eventual indisponibilidade de camadas mais altas para usuários habilitados a receber a maior qualidade do sistema.

Em [Tan and Jarvis 2006], um mecanismo baseado no padrão sistema bancário foi proposto para se promover diferenciação de serviço. Os usuários ganham pontos ao repassarem dados do fluxo de vídeo a outros participantes, de maneira similar ao mecanismo apresentado na Subseção 3.3. No entanto, enquanto esse último utiliza os créditos acumulados para permitir taxas de consumo maiores que as de cooperação, o mecanismo de [Tan and Jarvis 2006] os utiliza para permitir que os usuários disputem pelos melhores nós superiores que servirão seus fluxos. Um algoritmo distribuído para a seleção de nós superiores foi proposto, com várias estratégias sendo consideradas. Adicionalmente, o mecanismo estimula cooperações de usuários que não estejam recebendo nenhum vídeo, de forma que os pontos acumulados possam ser utilizados por eles em serviços futuros.

O mecanismo proposto em [Ngan et al. 2004] é baseado no padrão comunidade (reputação) e fornece uma maneira dos usuários identificarem aqueles que se recusaram a cooperar no passado, de forma que esses últimos possam ser retaliados no protocolo de admissão. A única premissa exigida é a reconstrução periódica das estruturas de distribuição, de maneira que um nó superior com comportamento não cooperativo possa se tornar nó inferior daqueles a quem recusou cooperação no passado.

Um outro mecanismo que envolve benefícios provenientes da topologia de rede

sobreposta foi apresentado em [Piatek et al. 2010], tendo como objetivo remunerar os usuários mais cooperativos com uma melhor qualidade de serviço. Para isso, a rede sobreposta é periodicamente reestruturada, de forma que os usuários que cooperam mais possam ser aproximados da origem dos fluxos de vídeo, obtendo, assim, uma qualidade mais previsível. O trabalho também propõe um protocolo de segurança baseado em chaves públicas/privadas e mecanismos de validação central e distribuído para combater diferentes tipos de ataques, bem como validar a cooperação mensurada dos usuários.

## 7. Conclusão

Neste trabalho, dois novos mecanismos de incentivo foram propostos para a arquitetura *iPeer TV*. Eles foram projetados com o objetivo de reduzir a demanda de banda de rede nos servidores da camada de infraestrutura da arquitetura, através da utilização das bandas de saída dos usuários para a distribuição dos fluxos de vídeo. O mecanismo *Esc2* foi o melhor avaliado para os cenários com fluxo pequeno, tendo promovido uma cooperação que resultou em 62% de usuários sendo admitidos por outros usuários, ao invés de serem admitidos diretamente pelos servidores, em comparação ao sistema sem incentivo e sem altruísmo. Consequentemente, a demanda de banda de rede nos servidores foi reduzida em 51%, envolvendo um custo de utilização do mecanismo que reduziu o desempenho das trocas de canais em 12% e a qualidade de fluxo em 1%, em média. Esse custo foi causado pelas limitações de banda de rede dos usuários e a natural instabilidade dos participantes de uma rede P2P.

Já o mecanismo *RemN* foi a melhor opção para os cenários com fluxos grandes, tendo estimulado uma cooperação que resultou em 38% de usuários sendo admitidos por outros usuários, ao invés de serem admitidos diretamente pelos servidores, em comparação ao sistema sem incentivo e sem altruísmo. Com isso, a demanda de banda de rede nos servidores foi reduzida em 18%, com um custo de utilização do mecanismo reduzindo o desempenho das trocas de canais em 5% e a qualidade de fluxo em 0.4%, em média.

De acordo com os resultados obtidos, pode-se concluir que os mecanismos de incentivo propostos são capazes de melhorar os níveis de cooperação na arquitetura *iPeer TV* com um baixo custo, apesar dos benefícios dessa cooperação serem mais significativos nos cenários com fluxos pequenos, já que estes são mais compatíveis com as capacidades de *upload* limitadas dos usuários.

## Referências

Cha, M. et al. (2008). Watching television over an ip network. In *Proc. 8th ACM SIGCOMM conf. on Internet measurement (IMC '08)*, pages 71–84, New York, NY, USA.

Cohen, B. (2003). Incentives build robustness in bittorrent. In *Proc. 1st Wksp. on the Economics of Peer-to-Peer Systems*.

Goyal, V. K. (2001). Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine*, 18(5):74–93.

Hei, X. et al. (2007). A measurement study of a large-scale P2P IPTV system. *IEEE Trans. on Multimedia*, 9(8):1672–1687.

hua Chu, Y., Chuang, J., and Zhang, H. (2004). A case for taxation in peer-to-peer streaming broadcast. In *Proc. ACM SIGCOMM Wksp. on Practice and theory of incentives in networked systems (PINS '04)*, pages 205–212, New York, NY, USA. ACM Press.

Liu, J. et al. (2008). Opportunities and challenges of peer-to-peer internet video broadcast. *Proceedings of the IEEE*, 96(1):11–24.

Liu, Z. et al. (2007). Using layered video to provide incentives in p2p live streaming. In *Proc. ACM Wksp. on P2P Streaming and IP-TV*, pages 311–316, New York, NY, USA.

Manzato, D. A. G. and da Fonseca, N. L. S. (2008a). Incentive mechanism for the coopnet network. *Peer-to-Peer Networking and Applications (P2PNA)*, 1(1):29–44.

Manzato, D. A. G. and da Fonseca, N. L. S. (2008b). Peer-to-peer IPTV services. In *Proc. 2nd IEEE Wksp. on Enabling the Future Service-Oriented Internet*.

Manzato, D. A. G. and da Fonseca, N. L. S. (2010a). A channel switching scheme for IPTV systems. In *Proc. IEEE Global Commun. Conf. (GLOBECOM 2010)*, pages 1–6.

Manzato, D. A. G. and da Fonseca, N. L. S. (2010b). *Incentive Mechanisms for Cooperation in Peer-to-Peer Networks*, volume 5 of *Handbook of Peer-to-Peer Networking*, chapter 2, pages 631–660. Springer US.

Manzato, D. A. G. and da Fonseca, N. L. S. (2011). A comparison of channel switching schemes for IPTV systems. In *Proc. IEEE Int'l. Conf. on Commun. (ICC 2011)*, pages 1–6.

Manzato, D. A. G. and da Fonseca, N. L. S. (2013a). iPeer TV: a P2P IPTV architecture with fast channel switching. In *Proc. IEEE Int'l. Conf. on Commun. (ICC 2013)*, pages 1–6.

Manzato, D. A. G. and da Fonseca, N. L. S. (2013b). Providing fast channel switching in P2P IPTV systems. *IEEE Journal on Selected Areas in Commun.*, 31(9):326–337.

Ngan, T.-W. J., Wallach, D. S., and Druschel, P. (2004). Incentives-compatible peer-to-peer multicast. In *Proc. 2nd Wksp. on Economics of Peer-to-Peer Systems*.

Obreiter, P. and Nimis, J. (2003). A taxonomy of incentive patterns – the design space of incentives for cooperation. In *Proc. 2nd Int'l. Wksp. on Agents and P2P Comp. (AP2PC'03), Springer LNCS 2872*, pages 89–100.

Padmanabhan, V. N., Wang, H. J., and Chou, P. A. (2003). Resilient peer-to-peer streaming. In *Proc. 11th IEEE Int'l. Conf. on Network Protocols (ICNP)*, pages 16–27.

Piatek, M. et al. (2010). Contracts: Practical contribution incentives for p2p live streaming. In *Proc. 7th USENIX Symp. on Networked Systems Design and Implementation (NSDI '10)*, pages 81–94, Berkeley, CA, USA. USENIX Association.

Qiu, T. et al. (2009). Modeling user activities in a large IPTV system. In *Proc. 9th ACM SIGCOMM conf. on Internet measurement (IMC '09)*, pages 430–441, NY, USA.

Tan, G. and Jarvis, S. A. (2006). A payment-based incentive and service differentiation mechanism for peer-to-peer streaming broadcast. In *Proc. 14th IEEE Int'l. Wksp. on Quality of Service (IWQoS '06)*, pages 41–50.

Teleco (2014). Teleco. http://www.teleco.com.br/. (Access Date).

Veloso, E. et al. (2002). A hierarchical characterization of a live streaming media workload. In *Proc. 2nd ACM SIGCOMM Wksp. on Internet measurment (IMW '02)*, pages 117–130. ACM Press.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 3*

## *Redes Tolerantes a Atrasos e Interrupções*

# Uma Proposta de Solução para o Problema de Nós Preferidos em Protocolos Oportunistas com Base Social

**Nelson M. Junior, Carlos Alberto V. Campos, Sidney C. Lucena**

Departamento de Informática Aplicada (DIA)
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Rio de Janeiro – RJ – Brasil

{nelson.junior, beto, sidney}@uniriotec.br

***Abstract.*** *Opportunistic networks are formed by self-organizing wireless nodes, where data delivery is not guaranteed, due to intermittent end-to-end connectivity. In order to infer future contact opportunities, information extracted from the context of users has been used to forward messages for opportunistic networks. Yet one open issue in this approach is the constant usage of repeated nodes considered as the most likely to deliver the messages, causing overload and a faster consume of battery resources. Thus, this paper proposes a new social-based forwarding protocol that has efficient energy consumption. The proposed protocol obtained better results on message distribution and reducing energy consumption of preferred nodes in comparison to Epidemic, PROPHET and BUBBLE Rap protocols.*

***Resumo.*** *Redes oportunistas são formadas por redes de nós sem fio auto-organizáveis, onde a entrega de dados não é garantida, devido a intermitente a conectividade fim-a-fim. Para inferir futuras oportunidades de contato, a informação extraída do contexto dos usuários tem sido utilizada no encaminhamento de mensagens para redes oportunistas. Nesse contexto, uma questão em aberto é o uso constante de repetidos nós tidos como os mais propensos a entregar a mensagem, o que provoca sobrecarga e aumento no consumo de bateria destes nós. Assim, este artigo propõe um protocolo de encaminhamento oportunista com base social que possui consumo de energia eficiente. O protocolo proposto obteve melhores resultados na distribuição de mensagem e redução do consumo de energia em nós preferidos em comparação aos protocolos: Epidêmico, PROPHET e BUBBLE Rap.*

## 1. Introdução

A modelagem das redes de computadores cabeadas que se baseiam em conexões fim-a-fim, estabelecidas em virtude do conhecimento dos nós fixos que formam a rede não é suficiente o bastante para representar o novo contexto que vem surgindo nos tempos atuais: a mobilidade dos dispositivos. Assim, como lidar então com situações em que, devido a essa intensa mobilidade dos nós, as conexões muitas vezes se perdem e, portanto, nem sempre é possível ter uma troca direta de dados? Nesse cenário, em que as conexões intermitentes são uma característica recorrente, os protocolos desenvolvidos para redes cabeadas que dependem de uma continuidade de conexão com outro computador para que ocorra a comunicação entre eles, não podem tolerar limitações espaço-temporais, pois os nós móveis podem sofrer conexões e desconexões muito

frequentes em virtude de inúmeros fatores como distância, queda de energia, movimentação aleatória, etc, fora o fator tempo, durante o qual a estrutura de uma mesma rede passa a alterar inúmeras vezes. Logo, para tais redes, a mobilidade é um desafio a superar. Como consequência dessas restrições e visando caracterizar uma nova representação para cenários onde ocorrem longos períodos de desconexões, o foco dos estudos e pesquisas voltaram-se para o campo das redes ad hoc móveis [Hoebeke et al. 2004], [Conti e Giordano 2007] como uma forma de atender às necessidades atuais de mobilidade dos dispositivos.

Dentre os cenários estudados na literatura, o encaminhamento oportunista de dados, em que a mobilidade do usuário pode ser uma oportunidade para a troca de dados, tirando vantagem da existência de um nó próximo, que é utilizado como um intermediário na entrega da mensagem da origem ao destino, tem recebido muita atenção nos últimos anos como uma abordagem para redes sem fio desconexas [Pelusi et al. 2006]. Sendo assim, neste contexto, a mobilidade dos dispositivos destas redes é vista como uma oportunidade para a comunicação e não como um desafio.

Uma vez que os dispositivos móveis são carregados por seres humanos, as estruturas sociais humanas tornaram-se recentemente o núcleo de soluções para transmissão de dados em redes oportunistas [Conti et al. 2010]. A comunicação nestas redes passa a ser fortemente impactada pela mobilidade humana, que é resultado do comportamento social do usuário [Zyba et al. 2011].

O cenário móvel também é crítico para uma rede de oportunista, visto que o recurso de bateria de seus dispositivos é limitado. Para protocolos que levam em conta os dispositivos cujos proprietários possuem um comportamento mais social no seu dia-a-dia, um dispositivo com mais contatos ativos ao longo do dia vai ser encarado como tendo real potencial para entregar a mensagem para o destino. No entanto, isto tem um custo, uma vez que esse dispositivo pode ficar sobrecarregado com tantas requisições. O trânsito frequente de informações por um determinado nó resulta em um gargalo que é prejudicial a este nó, porque reduz a vida útil de sua bateria em uma taxa mais alta.

Este artigo apresenta o *Social-based Energy-Efficient Routing Protocol* (SOCLEER), um protocolo de encaminhamento oportunista com base social desenvolvido através de uma modificação no consagrado protocolo BUBBLE Rap [Hui et al. 2008]. A alteração proposta visa aliviar a frequente repetição de nós intermediários tidos como preferidos, mediante a redistribuição da carga de participação desses nós no envio de mensagens a outros de características sociais semelhantes e, assim, reduzir seu consumo de energia.

Como contribuições do trabalho neste artigo, citam-se as seguintes: (i) desenvolvimento de um novo protocolo oportunista com base social denominado de SOCLEER, resultante de uma modificação no mecanismo decisor do consagrado protocolo oportunista BUBBLE Rap para diminuir a ocorrência de nós preferidos e consequentemente reduzir o consumo de bateria; (ii) implementação do protocolo SOCLEER no simulador ONE (*Opportunistic Network Environment*); (iii) utilização de *trace* de mobilidade realística (*RollerNet*) para realização das simulações, coletado em um evento de grande aglomeração de pessoas com dispositivos móveis; (iv) comparação dos resultados obtidos com protocolos oportunistas consagrados. Nos resultados obtidos, o SOCLEER apresentou uma redução no consumo de energia dos nós mais

preferencialmente requisitados, como consequência de uma melhor distribuição das mensagens encaminhadas pelos nós intermediários da rede.

O restante deste artigo está estruturado da seguinte forma. A Seção 2 faz uma revisão dos trabalhos relacionados. A Seção 3 descreve o problema dos nós preferidos. A Seção 4 apresenta o protocolo proposto e descreve suas característica, bem como sua implementação no simulador. A Seção 5 apresenta uma avaliação de desempenho realizada. Finalmente, a Seção 6 descreve as conclusões e trabalhos futuros sugeridos.

## 2. Trabalhos Relacionados

Qual a melhor forma de se enviar uma mensagem de um nó de origem de modo que ela chegue ao nó de destino em uma rede desconexa? Vários protocolos têm sido propostos em redes oportunistas para resolver este problema, cada qual apresentando evoluções e melhorias de acordo com o cenário estudado, mas essa ainda é uma questão em aberto.

Explorar o comportamento social humano em redes móveis oportunistas tem recebido recentemente uma atenção considerável por pesquisadores como forte fator a ser utilizado nas decisões de encaminhamento de mensagens, visto que, a mobilidade humana, não é imprevisível como se pensava, mas ao contrário, apresenta rastros de previsibilidade em função do dia-a-dia rotineiro das pessoas, o qual é influenciado pelo âmbito social na qual elas estão inseridas. Protocolos oportunistas têm sido propostos, como *BUBBLE Rap* [Hui et al. 2008] SimBetTS [Daly e Haahr 2009], e *PeopleRank* [Mtibaa et al. 2010], nos quais as decisões de encaminhamento são geralmente feitas usando-se o conhecimento coletado localmente a partir de dispositivos móveis sobre o comportamento social de seus usuários, associadas a métricas sociais derivadas de contatos entre dispositivos (por exemplo, *betweenness* e *degree centrality*, *tie-strength*) e algoritmos de detecção de comunidades para prever futuras oportunidades de contato.

Em [Zyba et al. 2011], experimentos usando *traces* de mobilidade reais foram feitos a fim de estudar as propriedades fundamentais das interações humanas. Toda a análise realizada foi focada principalmente na disseminação epidêmica de mensagens [Vahdat e Becker 2000]. Com base nos resultados dos experimentos, observou-se que, sob certas circunstâncias que parecem ser comuns em situações da vida real, a eficácia da disseminação depende predominantemente no número de usuários ao invés de seu comportamento social, em contradição com algumas observações previamente estabelecidas em disseminação de dados em redes oportunistas.

Com base neste experimento, a modelagem de protocolos de roteamento oportunistas, levando em consideração apenas o contexto social para tomar decisões de roteamento não são suficientes para disseminar as informações da melhor maneira. Também é importante considerar a questão da densidade populacional, uma vez que a eficácia da disseminação de conteúdo não é apenas uma consequência do status social dos proprietários dos dispositivos, mas também uma consequência do número e densidade de dispositivos. Protocolos como os citados anteriormente (*BUBBLE Rap*, SimBetTS e *PeopleRank*) que utilizam métricas baseadas no comportamento social dos usuários dos dispositivos para tomar decisões de encaminhamento, roteiam sobre "laços fortes" entre os usuários dos dispositivos móveis, inferidas ou do comportamento do contato ou de amizades declaradas. Mas o estudo em [Zyba et al. 2011] estende esses esforços anteriores, explorando o papel e o potencial de dispositivos não-sociais anteriormente ignorados para a comunicação e disseminação de dados entre dispositivos

móveis. Considerando também este trabalho, esta pesquisa objetiva explorar não somente o caráter social dos dispositivos móveis, mas também a densidade populacional destes para encontrar uma melhor forma de encaminhar a mensagem em redes oportunistas. Isso pode ser justificado pelo uso de um *trace* de conectividade realística e bastante denso a ser descrito na Seção 5.1.

## 3. O Problema dos Nós Preferidos

A análise do comportamento social humano tem sido o fator de incentivo para muitas pesquisas na área de redes oportunistas. As redes sociais, que de certa forma traduzem as relações humanas, deixam um rastro das informações sobre o ambiente em que os usuários estão inseridos: o contexto. A mobilidade diária dos seres humanos faz com que novos protocolos oportunistas desenvolvidos se valham das informações registradas, por exemplo, no Facebook ou no Twitter de *smartphones*, *tablets* ou quaisquer outros dispositivos. Estas informações reunidas geram o referido contexto, que é o mecanismo utilizado por esses novos protocolos na tomada de decisão de encaminhamento de uma mensagem. O que pode resultar numa alta taxa de entrega sem réplicas redundantes e desnecessárias, a fim de melhorar o desempenho da rede.

Um problema identificado em [Xu et al. 2010] mostra que mecanismos de encaminhamento que tendem a utilizar apenas os nós mais preferidos, característica dos protocolos baseados em contexto social, podem sofrer com a constante seleção de nós com energia ou armazenamento insuficientes: nós que são mais conectados em uma rede social (isto é, nós com mais "amigos") podem ser selecionados com mais frequência para o encaminhamento de mensagens, podendo causar carga desleal [Pujol et al. 2010] no sistema. Nas simulações feitas em [Xu et al. 2010] foi demonstrado que 63% do tráfego é encaminhado através de apenas 10% dos nós quando se utiliza protocolos que dependem somente dos nós mais preferidos para encaminhamento.

Como visto em [Pujol et al. 2010], uma das características de tais redes sociais é que elas apresentam uma distribuição de conectividade de "*fat-tail*", onde poucos nós possuem muitas conexões, enquanto a maioria tem muito poucas. Uma vez que as mensagens são enviadas através de contatos, é inevitável que a maioria dos nós mais conectados transportem a maioria do tráfego, produzindo com isso uma distribuição de carga injusta, em que tais nós são então sobrecarregados em detrimento dos nós que, por terem menos contatos, não são tão requisitados para encaminhar a mensagem por não serem tão sociais. Assim, considera-se, portanto, que nós preferidos são aqueles mais conectados em uma rede social que, em virtude dessa característica, são os mais requisitados como retransmissores por protocolos oportunistas com base social.

Neste contexto, tem-se a seguinte questão. Como diminuir o custo da utilização de nós preferidos em uma rede oportunista, que é representado pelo rápido consumo de seus recursos de energia em consequência da tomada de decisão para o encaminhamento de dados entre os nós que se dá em função de um comportamento social dos usuários dos dispositivos? Pode ser vantajoso então, considerar também um cenário denso de dispositivos com o intuito de se valer de mais opções de nós com características sociais semelhantes aos preferidos para efeito de redistribuição de carga entre os nós ao invés de utilizar apenas nós que são obviamente os mais apropriados [Bigwood e Henderson 2011]. Para economizar energia em ambientes móveis, tais protocolos de roteamento devem, portanto, minimizar o encaminhamento desnecessário de mensagem.

A investigação da questão acima é o cerne dessa pesquisa, que culmina na próxima seção deste artigo, em que é idealizada e proposta a alteração no *BUBBLE Rap* que é um consagrado protocolo de roteamento de rede oportunista baseado em contexto social. O objetivo da alteração é fazer uma distribuição de carga mais justa no que diz respeito a participação dos nós no envio de mensagens e, em consequência disso, uma redução no consumo de energia dos nós mais conectados.

## 4. O Protocolo SOCLEER

A principal ideia por trás do *BUBBLE Rap* é que ele funciona sob a hipótese de uma estrutura específica de relações sociais dos usuários, explorando suas propriedades para a seleção de rotas. Especificamente, ele se concentra em dois aspectos de uma rede social: comunidade e centralidade. A sociedade humana está estruturada em comunidades, e dentro de uma comunidade alguns são mais populares do que outros: eles têm uma alta centralidade. As mensagens são enviadas da comunidade de origem para os nós de maior ranking (ou seja, usuários mais sociais) até que um contato dentro da comunidade do destino seja encontrado. Por disseminação, as mensagens são armazenadas nos nós mais populares, que têm mais chances de entrar em contato com a comunidade do destino [Hui et al. 2008].

O que se percebeu em simulações efetuadas com o *BUBBLE Rap* para este artigo é que a métrica social de centralidade não era guardada quando calculada durante os contatos entre os nós com o objetivo de encaminhar a mensagem até o destino. Como o teste desta métrica é feita em tempo de execução de nó a nó, obviamente que, em um grafo formado no momento da conexão, os nós conectados ao nó detentor da mensagem que possuem maior centralidade que este (conexões ativas) vão sempre receber a mensagem para o encaminhamento futuro, considerando-se que dentre tais nós conectados não seja identificado o destino, nem um nó que esteja na comunidade do destino, situações nas quais a centralidade não é considerada para efeito de entrega da mensagem. Esta é uma abordagem gulosa do próprio protocolo e configura o problema apresentado, na Seção 3, pois um nó com mais conexões é mais sociável, ou seja, ele possui mais "amigos" conectados na rede. Tendo, portanto, centralidade de grau maior que os demais, o nó mais conectado vai ser sempre mais utilizado em detrimento dos outros com centralidade próxima a este, porém menor. Referente a esta questão que está presente não só no *BUBBLE Rap*, mas também nos demais protocolos oportunistas com base social (tal como descrito em [Xu et al. 2010]), e considerando um cenário de densa população, isto é, uma série de dispositivos móveis formando a rede, desenvolveu-se a ideia de um protocolo oportunista que pudesse efetuar uma redistribuição da carga de participação dos nós no envio de mensagens para outros nós intermediários que não estão sendo muito demandados. Com essa alteração pretende-se reduzir a sobrecarga dos nós preferidos e, ao mesmo tempo, diminuir o consumo de energia desses nós. As modificações efetuadas no mecanismo decisor do protocolo *BUBBLE Rap* baseiam-se num mecanismo de sorteio simples (randômico) e resultaram no protocolo SOCLEER.

A mecânica da alteração que distingue o SOCLEER do *BUBBLE Rap* consiste no seguinte. Ao invés da mensagem ser enviada imediatamente do nó atual ao nó conectado que possua maior centralidade do que a sua, durante o tempo *t* em que o grafo de contatos é formado, como faz o *BUBBLE Rap*, as centralidades são calculadas e armazenadas em uma lista de centralidades. Isso vale tanto para a centralidade global, quanto para a local, que continuam sendo utilizadas da mesma forma que o algoritmo

original. Então, essa lista é ordenada decrescentemente do nó com maior centralidade para o nó de menor centralidade, onde então um método criado para efetuar o sorteio é invocado. Este método efetua um sorteio de *n* nós, sendo o mínimo de 1 e o máximo de 10, dentre os maiores em centralidade contidos na lista para encaminhar a mensagem.

Foi escolhido o valor máximo de 10 nós para sorteio, pois para o *trace RollerNet*, verificou-se em modo de *debug* do código durante as simulações que, as listas montadas (seja local ou global) não ultrapassavam na média esse valor. Mas, para outros *traces* de mobilidade esse valor máximo deve ser alterado. Cabe ressaltar que nem todos os nós que compõem a lista são elegíveis ao sorteio, mas apenas aqueles dentro de uma faixa delimitadora, podendo ser variável, que vai do nó de maior centralidade até aquele dentro do índice que delimita a faixa. Em outras palavras, se são sorteados, por exemplo, 3 nós dentre 5 de maior centralidade em uma lista contendo 8 nós, participarão dos sorteio, após a ordenação decrescente da lista, os nós nas posições de índice [0], [1], [2], [3] e [4]. Na Seção 6.3 são descritos os cenários de sorteio usados.

A idealização do sorteio simples como mecanismo decisor para retransmissão de conteúdo tem por objetivo reduzir as réplicas de informação na rede e, por conseguinte, reduzir a carga de participação dos nós no envio de mensagens, pois somente os nós sorteados segundo a regra estabelecida funcionarão como retransmissores. Com isso, espera-se que o consumo de bateria dos nós reduza, pois nem todos estarão gerando réplicas. Assim, o gasto com o uso de CPU para recepção, armazenamento e retransmissão tende a diminuir. Além disso, a alternância de opções dentre as melhores do contexto social, motivada pelo sorteio, pode fazer com que um nó mais conectado não seja sempre o mais escolhido para se encaminhar a mensagem.

Tirando o fato da criação das listas, da seleção dos nós candidatos que farão parte delas para efeito do sorteio segundo suas respectivas centralidades (global ou local) e do sorteio em si dos nós aos quais a mensagem será encaminhada, o mecanismo decisor do SOCLEER é semelhante ao do *BUBBLE Rap* para efeitos de detecção de comunidade e cálculo das respectivas centralidades.

## 5. Avaliação do Protocolo Proposto

O SOCLEER foi codificado em linguagem de programação Java. O código-fonte foi desenvolvido com base na implementação do protocolo *BUBBLE Rap* desenvolvida na mesma linguagem por P.J. Dillon da University of Pittsburgh (veja detalhes em *http://people.cs.pitt.edu/~pdillon*). O pseudocódigo encontra-se descrito na Figura 1.

### 5.1. *Dataset*

O método de pesquisa utilizado pelo presente trabalho foi a simulação. Experimentos reais envolvendo disseminação oportunista de dados são muito custosos e complexos. Além disso, dependendo do tipo de experimento a ser realizado, os requisitos necessários podem variar, mas, no geral, há a necessidade de pessoas, antenas, computadores, *softwares* configurados previamente para o experimento, e é claro, dispositivos móveis. Enfim, todo um aparato se faz necessário para a realização de experimentos. Porém, para esse tipo de experimento, existem bases de dados de *traces* reais de mobilidade, como a disponível no CRAWDAD. Desta base de dados foram extraídos os *traces* de mobilidade utilizados na avaliação do algoritmo proposto.

```
Início
    Para cada nó encontrado faça
        Se (o nó conectado é o destino)
            Entrega a mensagem ao nó
        Senão
            Se (o nó encontrado está na comunidade do destino e o nó atual não)
                Entrega a mensagem ao nó
            Senão
                Se (o nó encontrado não está na comunidade do destino e o nó atual está)
                    Não entrega a mensagem
                Senão
                    Se (ambos estão na comunidade do destino)
                        Se (a centralidade local do nó encontrado > centralidade local do nó atual)
                            Armazena a centralidade local do(s) nó(s) encontrado(s) em uma lista local
                        Senão
                            Não armazena a centralidade na lista
                    Senão
                        Se (ambos não estão na comunidade do destino)
                            Se (a centralidade global do nó conectado > centralidade global do nó atual)
                                Armazena a centralidade global do(s) nó(s) encontrado(s) em uma lista global
                            Senão
                                Não armazena a centralidade na lista
    Fim Para
    Ordena lista ( )
    Se (lista global ou local <> null & > 0)
        Se (lista global ou local = 1)
            Entrega a mensagem ao nó
        Senão
            Efetua o sorteio de n nós na lista delimitada por faixa
            Entrega a mensagem aos nós sorteados
Fim
```

**Figura 1. Pseudocódigo do SOCLEER**

Da base CRAWDAD (veja detalhes em *www.crawdad.cs.dartmouth.edu*) foi escolhido o *RollerNet* [Tournoux et al. 2011], que é um dataset contendo *traces* de 62 dispositivos iMotes que foram distribuídos para um grupo de pessoas a fim de coletar qualquer varredura oportunista de outros dispositivos Bluetooth de grupos de patinadores (incluindo os iMotes distribuídos), compostos por uma multidão de cerca de 2.500 pessoas que participaram de uma turnê de patinação pela cidade de Paris na França pelo período de 3h. Segundo relatado nos experimentos de [Tournoux et al. 2011], o *RollerNet* é o primeiro *trace* de conectividade altamente denso e móvel.

## 5.2. Cenário de Simulação

Para avaliar o SOCLEER foi utilizado a ferramenta ONE que é um simulador de rede feito em Java e voltado para redes oportunistas. Nos cenários analisados, a carga de 1.000, 10.000, 20.000 e 30.000 mensagens de 50KB foram criadas aleatoriamente durante o tempo de simulação de 9999 seg., que se refere à duração do *trace RollerNet*. Para avaliar o desempenho da rede em função das cargas de mensagens geradas sem a influência da restrição do tamanho do buffer, o mesmo foi configurado em 2GB, que é suficiente para suportar as mensagens. Cada rodada de simulação foi repetida 10 vezes e nos resultados plotados utilizou-se o intervalo de confiança de 95% da tabela *T-Student*.

Os valores utilizados para a carga máxima de energia da bateria, perda de energia por scan da rede e energia gastas por envio e recebimento de mensagens foram tomadas com base na maioria dos smartphones atuais. Por exemplo, um smartphone HTC One® tem uma carga máxima de bateria de 2.300 mAh. No entanto, o cenário simulado utilizando o *trace RollerNet* não reflete uma realidade total no que diz respeito

ao consumo de energia dos dispositivos. Além disso, foram desconsiderados alguns fatores que influenciam o consumo de bateria, como brilho da tela, acesso a Internet (seja por celular ou por Wi-Fi), envio e recepção de chamadas telefônicas, a utilização de aplicativos embarcados nos dispositivos, além da própria complexidade para efetivação do mecanismo de sorteio do SOCLEER no consumo de energia.

Tanto para o *BUBBLE Rap*, assim como para o SOCLEER, definiu-se nas simulações o valor 5 como o número máximo de distribuições de mensagens (*k*). No cálculo das centralidades através do algoritmo *C-Window* [Hui et al. 2008] foi definido o valor de 10 min. como janela de tempo de espera antes de se recalcular novos valores.

### 5.3. Métricas e Protocolos Oportunistas

Para avaliar o desempenho/eficiência do SOCLEER foram usadas as seguintes métricas:

**Participação dos nós no envio de mensagens**: Quantidade média de vezes em que um nó é requisitado para encaminhar uma mensagem (neste cálculo, somente as mensagens entregues foram incluídas) [Villares et al. 2013].

**Consumo de bateria dos dispositivos móveis:** Quantidade média de energia gasta pelos nós durante o envio e recepção de mensagens e a cada scan na rede.

**Fração de mensagens entregues:** Quantidade de mensagens entregues, dividido pela quantidade de mensagens criadas na origem.

**Overhead (Sobrecarga) de mensagens:** Quantidade da diferença entre mensagens retransmitidas e mensagens entregues, dividido pelo número de mensagens entregues.

**Latência:** Tempo que a mensagem leva desde sua criação até a chegada ao destino

**Número médio de saltos**: Número médio de saltos realizados pelas mensagens desde a origem até o destino.

Uma comparação do SOCLEER foi realizada com os seguintes protocolos:

Epidêmico: os nós de encaminham mensagens para todos os nós encontrados que ainda não possuem uma cópia [Vahdat e Becker 2000]. A transmissão das mensagens funciona utilizando o conceito de epidemia, onde todos os vizinhos são "contaminados" por um nó "infectado". Em outras palavras, para que uma mensagem seja entregue a um destino específico, o protocolo a torna conhecida por grande parte da rede.

PROPHET: é um protocolo de encaminhamento probabilístico [Lindgren et al. 2004]. Ele calcula a previsibilidade de entrega em cada nó para cada destino usando o histórico de encontros e a transitividade. A mensagem é encaminhada para um nó se ele tem maior previsibilidade de entrega do que o nó atual para esse destino particular.

*BUBBLE Rap*: Um protocolo de encaminhamento com base social desenvolvido por [Hui et al. 2008]. Quando um nó tem uma mensagem para um destino e este destino não faz parte da sua comunidade local, o nó encaminha a mensagem para nós mais globalmente centrais aqueles que se estimam possuírem um maior valor de centralidade global. A intenção aqui é que os nós no centro da rede social são mais propensos a contactar o destino. Assim, a mensagem "borbulha" na rede social para nós mais centrais, até que um nó é encontrado que relate o destino em sua comunidade local. Neste ponto, a mensagem é encaminhada apenas dentro dos nós da comunidade local e propagada para os nós mais localmente centrais ou até ser entregue o destino.

## 5.4. Resultados Obtidos

Inicialmente, para 1.000 mensagens geradas foi configurada para o protocolo *BUBBLE Rap* (Figura 1) a mesma carga de mensagens em diferentes cenários de simulação para o protocolo SOCLEER (Figura 2). Esta variação nos cenários era para testar, em relação as métricas avaliadas, a maior eficácia do método de sorteio de nós como intermediários para o envio da mensagem dentre os nós de maiores centralidades guardados na lista. Os cenários de sorteio foram definidos da seguinte forma (6 cenários): (i) - sorteio de 1 nó dentre 2 de maior centralidade; (ii) - sorteio de 2 nós dentre 3 de maior centralidade; (iii) - sorteio de 3 nós dentre 5 de maior centralidade; (iv) - sorteio de 3 nós dentre todos os elementos da lista; (v) - sorteio de 5 nós dentre todos os elementos da lista e; (vi) - sorteio de 10 nós dentre todos os elementos da lista.

Após a extração, plotagem e análise gráfica dos resultados para os cenários simulados acima, percebeu-se que alguns cenários tiveram melhor desempenho nas métricas de participação do nó no envio de mensagens e de consumo de bateria quando comparados com os resultados do *BUBBLE Rap*. Assim, os cenários de simulação para o SOCLEER com carga de 10.000 mensagens passaram a ser os seguintes (3 cenários): (i) - sorteio de 1 nó dentre 2 de maior centralidade; (ii) - sorteio de 2 nós dentre 3 de maior centralidade e; (iii) - sorteio de 10 nós dentre todos os elementos da lista.

Após nova extração, plotagem gráfica e análise dos resultados para os cenários acima, percebeu-se que o cenário de simulação do SOCLEER que obteve os melhores resultados com relação ao consumo de bateria dos nós, tanto os preferidos como todos os outros nós envolvidos, foi o com **sorteio de 1 nó dentre 2 de maior centralidade**. Para uma análise mais profunda desta métrica, houve a necessidade de aumentar novamente a carga de mensagens, desta vez para 20.000 mensagens. Nas simulações com os protocolos Epidêmico, PROPHET e *BUBBLE Rap*, foi utilizada a mesma regra de carga de mensagens, ou seja, 1.000, 10.000 e 20.000. Os resultados dessas simulações e as mudanças de cenários serão explicados mais detalhadamente a seguir.

Para verificar a eficiente do protocolo SOCLEER quanto a uma melhor distribuição da participação dos nós no encaminhamento das mensagens em relação ao *BUBBLE Rap*, entendeu-se com base em [Xu et al. 2010] que seria necessário definir uma margem de corte percentual dessa participação dos nós no envio de mensagens para se determinar tal ocorrência de nós preferidos com o objetivo de comparar os resultados das simulações do SOCLEER com o *BUBBLE Rap*, e, consequentemente, o consumo de bateria dos nós. Portanto, foi considerado que os nós preferidos estão em um nível de corte onde ocorre a participação média de $n$ nós no envio de mensagens de até no mínimo $p$% da média total das mensagens criadas em cada simulação. Para a análise da energia, via simulação, foi utilizado o modelo de consumo de energia proposto e implementado por Silva e publicado no trabalho [Silva et al. 2012].

- *Análise da participação do nó no envio de mensagens para o cenário com 1.000 mensagens criadas*

Comparando-se as Figuras 2 e 3, pode-se verificar que a margem de 6% de corte como patamar para se determinar a ocorrência de nós preferidos, o SOCLEER teve a melhor redistribuição absoluta na participação dos nós no envio de mensagens em comparação com o *BUBBLE Rap*, o que representa um decréscimo de 14,52% na ocorrência, não importando o cenário de sorteio. Foram observadas 15 ocorrências em 62 para o SOCLEER contra 24 em 62 para o protocolo *BUBBLE Rap*.

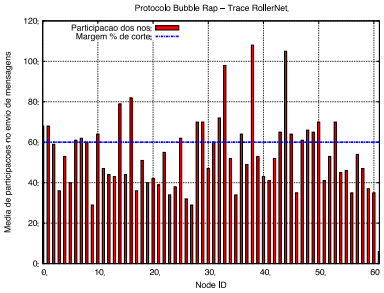



**Figura 2. Participação dos nós no envio de mensagens simulando o *BUBBLE Rap* para um corte de 6%**

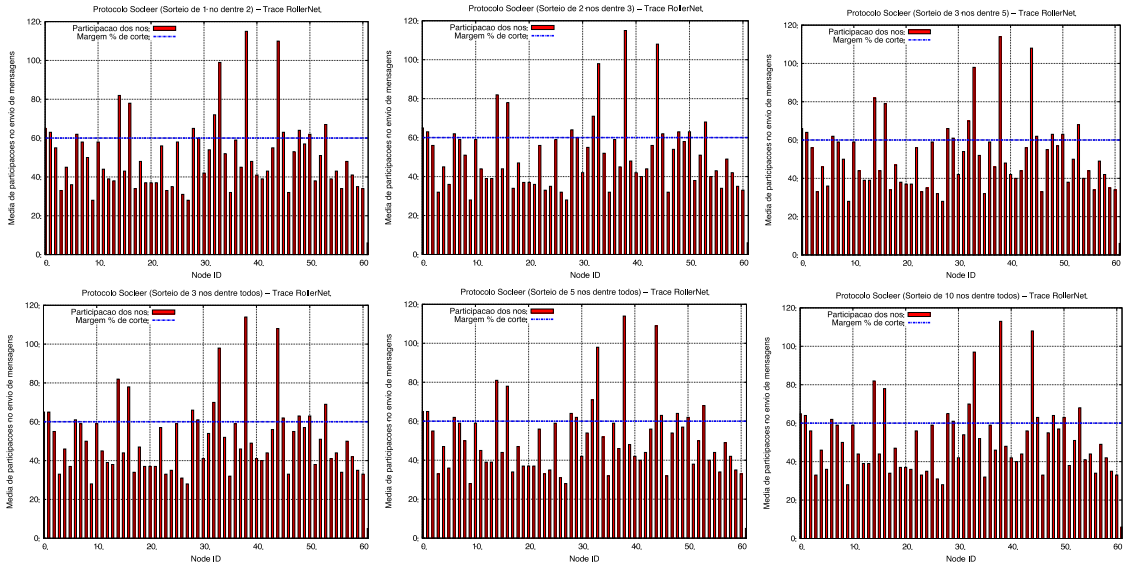



**Figura 3. Participação dos nós no envio de mensagens simulando o SOCLEER para um corte de 6%, variando o número de nós do sorteio com maior centralidade.**

Na Figura 4, pode-se observar que o consumo de bateria do SOCLEER foi menor que o dos protocolos Epidêmico e PROPHET, e muito semelhante ao BUBBLE Rap. Portanto, para o cenário de 1.000 mensagens, que foi considerado de baixa carga, o mecanismo de sorteio do SOCCLER não fez diferença em relação ao BUBBLE Rap.

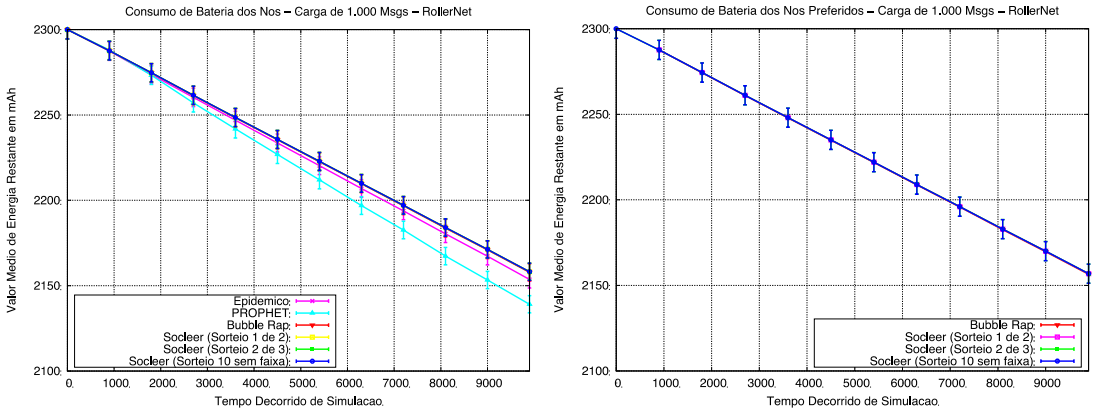



**Figura 4. O valor médio da energia restante para todos os nós (a) e para os nós preferidos em (b).**

- *Análise da participação do nó no envio de mensagens para o cenário com 10.000 mensagens criadas*

Novas simulações foram feitas, onde então foi realizada nova carga de 10.000 mensagens geradas para análise do *BUBBLE Rap* (Figura 5). Além disso, os cenários de simulação para SOCLEER (Figura 6) tornaram-se as seguintes variações: (a) sorteio de 1 nó dentre 2 de maior centralidade, (b) sorteio de 2 nós dentre 3 de maior centralidade, e (c) sorteio de 10 nós dentre todos os elementos da lista.

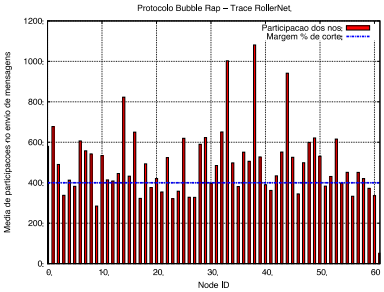



**Figura 5. Participação dos nós no envio de mensagens simulando o *BUBBLE Rap* para um corte de 4%**

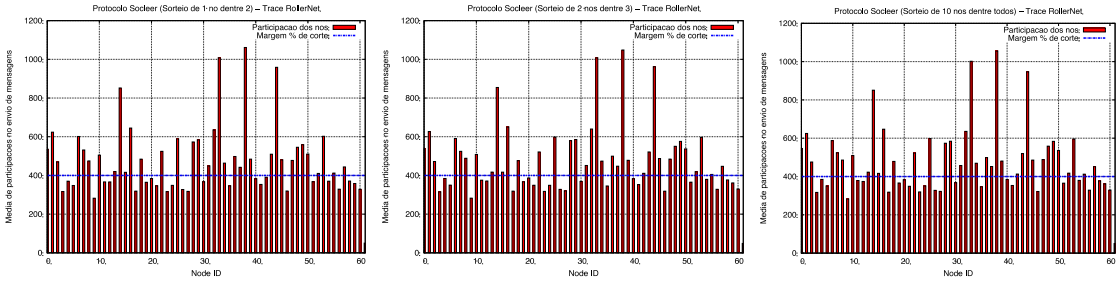



**Figura 6. Participação dos nós no envio de mensagens simulando o SOCLEER para um corte de 4%**

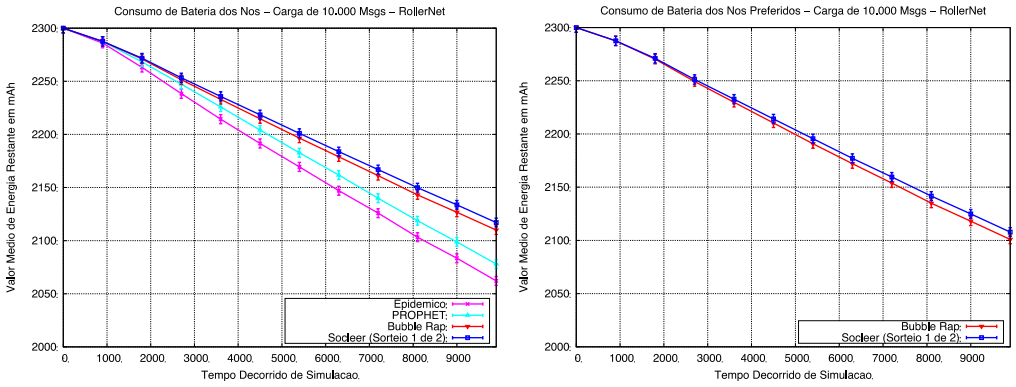



**Figura 7. O valor médio da energia restante para todos os nós (a) e para os nós preferidos em (b).**

Depois de observar os gráficos das Figuras 5 e 6, a configuração do SOCLEER que obteve os melhores resultados em matéria de consumo de bateria dos nós, tanto os preferidos como todos os outros, foi o sorteio de 1 nó dentre 2 de maior centralidade. Esta configuração reduziu o consumo da bateria de SOCLEER em comparação com os outros protocolos, como evidenciado na Figura 7.

- *Análise da participação do nó no envio de mensagens para o cenário com 20.000 mensagens criadas*

Para resultados mais precisos sobre esta métrica, houve a necessidade de aumentar a carga de mensagens novamente para 20.000 mensagens. Pode-se verificar

nos gráficos da Figura 8 que o SOCLEER mostrou apenas uma pequena diferença na participação dos nós no envio de mensagens para esta carga de dados. No entanto, para o consumo da bateria (Figura 9), para o cenário com 20.000 mensagens criadas, o protocolo SOCLEER continuou apresentando uma maior energia residual para todos nós e para os nós preferidos do que em outros protocolos analisados.

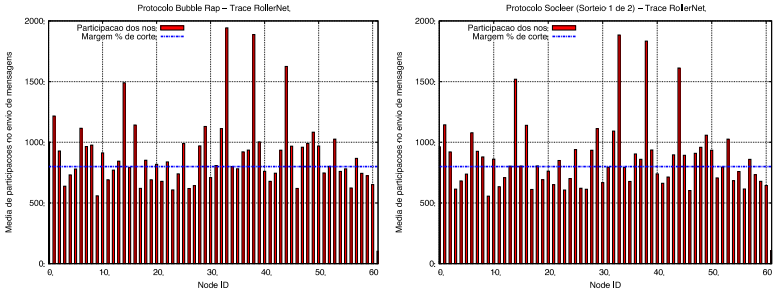



**Figura 8. Participação dos nós no envio de mensagens para um corte de 4% simulando o *BUBBLE Rap* (a) e o SOCLEER (b)**

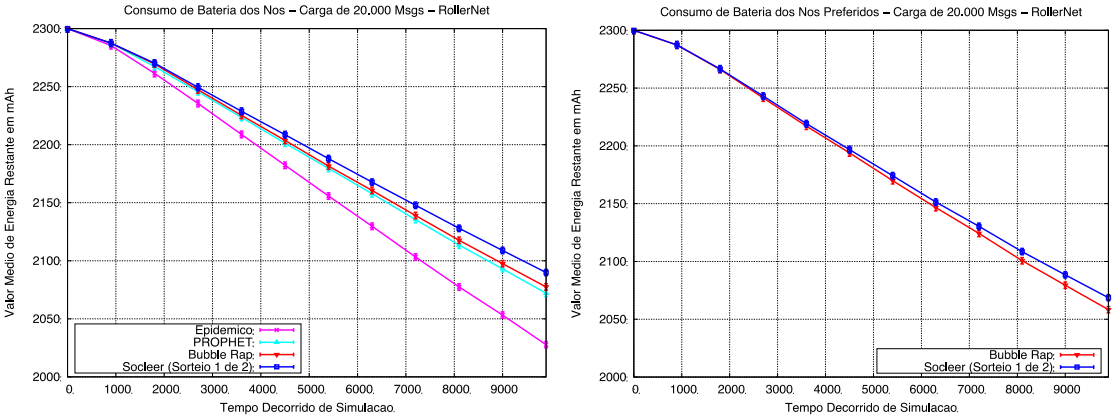



**Figura 9. O valor médio da energia restante para todos os nós (a) e para os nós preferidos em (b)**

Além da avaliação da ocorrência de nós preferidos e a redução do consumo de energia, neste trabalho, o protocolo SOCLEER também foi avaliado em relação à fração de mensagens entregues, latência, *overhead* e saltos por rota. Os parâmetros de simulação foram descritos na Seção 5.2, no entanto, além da carga de mensagens criadas (1.000, 10.000 e 20.000), também foi analisada uma carga de 30.000 mensagens para ser verificar a ocorrência de uma possível estabilização nos resultados.

Na Figura 10-a, pode-se verificar que para o cenário de carga baixa (1.000 mensagens) o SOCLEER apresentou uma baixa fração de mensagens entregues, mas à medida em que a carga de mensagens na rede aumenta, seu desempenho melhora em comparação aos outros protocolos, especialmente para a carga de 30.000 mensagens. Como o mecanismo de sorteio elege sempre os nós mais centrais para retransmissão da mensagem e, levando-se em conta o cenário específico do RollerNet de alta densidade de dispositivos na rede, pode-se dizer que o SOCLEER usou essa característica para efetuar uma melhor distribuição de mensagens em comparação aos outros protocolos.

Para a métrica de latência (Figura 10-b), o SOCLEER levou mais tempo do que os outros protocolos para entregar mensagens em cenários de baixa carga, mas quando a carga de mensagens aumentou (20.000 mensagens), pôde superar os outros protocolos gerando o menor atraso para o cenário de alta carga de mensagens. Em decorrência do

sorteio, o número de dispositivos que recebem uma réplica da mensagem diminui o que intensifica o atraso na entrega considerando-se que há poucas mensagens criadas e poucos nós na rede para retransmití-las. Porém, como o trace é denso, esse atraso diminui na medida em que o número de mensagens criadas aumenta, compensando assim a redução dos dispositivos habilitados para a retransmissão mediante o sorteio.

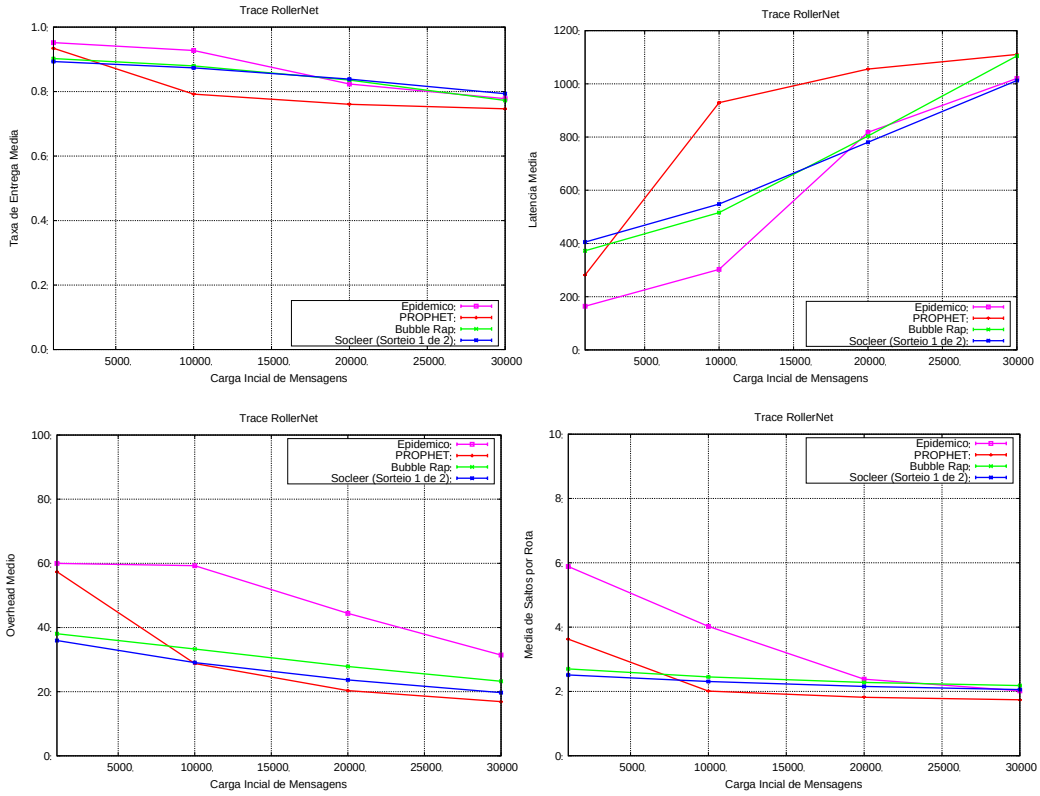



**Figura 10. Avaliação da fração de mensagens entregues (a), latência (b), overhead (c) e saltos por rota (d) para os protocolos analisados**

Para o overhead (Figura 10-c) e a média de saltos por rota (Figura 10-d), o SOCLLER apresentou o melhor resultado que todos os outros protocolos, com exceção do PROPHET para cenários de alta carga (20.000 e 30.000 mensagens).

## 6. Conclusões e Direções Futuras

Protocolos de encaminhamento oportunistas têm sido desenvolvidos, como *BUBBLE Rap*, SimBetTS e *PeopleRank* levando-se em consideração algum contexto do comportamento social humano como mecanismo decisor para a entrega das mensagens aos seus respectivos destinos. No entanto, estes protocolos têm sofrido com o problema dos nós preferidos, aqueles que são considerados mais sociais dentro deste contexto e, por isso, consequentemente, os mais solicitados como melhores rotas através da rede.

Neste artigo foi proposto o SOCLEER, um protocolo com base social resultante de uma mudança no mecanismo decisor do *BUBBLE Rap*, a fim de fazer uma redistribuição da carga na participação dos nós no encaminhamento, aliviando a ocorrência de tais nós preferidos e economizando energia desses nós. Os resultados obtidos evidenciam o potencial de economia de energia do SOCLLER e um melhor desempenho, em relação os outros protocolos, para cenários de alta carga. Entretanto, essas observações são específicas para o cenário do *RollerNet*.

Como trabalhos futuros pretende-se avaliar o desempenho do SOCLEER com outros protocolos de base social usando outros *traces* de mobilidade, bem como investigar outras métricas de centralidade para o mecanismo decisor do SOCLEER.

## Referências

Hui, P., Crowcroft, J. and Yoneki, E. "BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks". MobiHoc, 2008.

Pelusi, L., Passarella, A. and Conti, M. "Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks," IEEE Commun. Mag., v. 44, n. 11, 2006.

Hoebeke, J., Moerman, I., Dhoedt, B. and Demeester, P. "An Overview of Mobile Ad Hoc Networks: Applications and Challenges". Journal of the Communication Network, v. 3, pp. 60-66, Jul. 2004.

Conti, M. and Giordano, S. "Multihop Ad Hoc Networking: The Reality". IEEE Communications Magazine, v. 45, n. 4, pp. 88-95, Apr. 2007.

Conti, M., Giordano, S., May, M. and Passarella, A. "From Opportunistic Networks to Opportunistic Computing". IEEE Communic. Magazine, v. 48, n. 9, pp. 126-139, 2010.

Zyba, G., Voelker, G. M., Ioannidis S. and Diot, C. "Dissemination in Opportunistic Mobile Ad-hoc Networks: The Power of the Crowd". INFOCOM, 2011.

Daly, E. M. and Haahr, M. "Social Network Analysis for Information Flow in Disconnected Delay-Tolerant MANETs". IEEE Transactions on Mobile Computing, v. 8, pp. 606–621, May 2009.

Mtibaa, A., May, M., Diot, C. et al. "PeopleRank: Social Opportunistic Forwarding". INFOCOM Mini Conference, 2010.

Xu, K., Li, V. O. K. and Chung, J. "Exploring Centrality for Message Forwarding in Opportunistic Networks", IEEE WCNC, Apr. 2010.

Bigwood, G. and Henderson, T. "Bootstrapping Opportunistic Networks Using Social Roles", Workshop AOC, 2011.

Pujol, J., Toledo, A. and Rodriguez, P. "Fair Routing in Delay Tolerant Networks. In: Proceedings of INFOCOM '09. IEEE, 2009.

Keränen, A., Ott, J. and Kärkkäinen, T. "The ONE Simulator for DTN Protocol Evaluation". Simutools, 2009.

Vahdat, A. and Becker, D. "Epidemic Routing for Partially-Connected Ad Hoc Networks" CS-200006, Duke University, USA 2000.

Lindgren, A., Doria, A. and Schelén, O. "Probabilistic Routing in Intermittently Connected Networks". SAPIR, 2004.

Tournoux, P.-U., Leguay, J., Benbadis, F. et al. "Density-Aware Routing in Highly Dynamic DTNs: The RollerNet Case". IEEE Trans. on Mobile Comp, v. 10, n. 12, 2011.

Silva, D. R., Costa, A. and Macedo, J. "Energy Impact Analysis on DTN Routing Protocols". ExtremeCom, 2012.

Villares, T. L., Campos, C. A. V. and Viana, A. C. "A Influência de Nós Especiais na Entrega de Mensagens nas Redes Tolerantes a Atrasos e Interrupções". III WRA'13, Simpósio Brasileiro de Redes de Computadores - SBRC, Brasília, DF, Brasil, 2013.

# Uma Avaliação do Ataque de Falsificação de Reconhecimentos Positivos em Redes Tolerantes a Atrasos e Desconexões

**Juliano F. Naves**[1,2], **Igor M. Moraes**[1]

[1]Laboratório MídiaCom, PGC-TCC / Instituto de Computação
Universidade Federal Fluminense

[2]Instituto Federal de Rondônia, *campus* Ji-Paraná

{jfischer,igor}@ic.uff.br

***Resumo.*** *Os protocolos de roteamento propostos para redes tolerantes a atrasos e desconexões (Delay/Disruption Tolerant Networks - DTNs) comumente utilizam a técnica de envio de reconhecimentos positivos de mensagens. O objetivo é liberar espaço nos* buffers *dos nós e evitar que mensagens sejam descartadas desnecessariamente devido à sobrecarga desses* buffers*. No entanto, os protocolos que utilizam tal técnica estão sujeitos ao ataque de falsificação de reconhecimentos. Neste ataque, nós maliciosos forjam reconhecimentos para que nós legítimos descartem mensagens ainda não entregues. Assim, o desempenho da rede é reduzido. Neste trabalho, analisa-se uma contramedida para este ataque, na qual um nó ignora reconhecimentos para mensagens que não foram recebidas anteriormente por ele. Três cenários reais de mobilidade e um padrão de movimentação bem conhecido são utilizados em simulações e os resultados mostram que quando aproximadamente 12% dos nós da rede são maliciosos, a taxa de entrega de mensagens é reduzida de até 61 pontos percentuais.*

***Abstract.*** *The routing protocols proposed for Delay and Disruption Tolerant Networks (DTNs) often employ acknowledgements. The goal is to increase the available space in buffers and also to avoid messages from being dropped unnecessarily due to buffer overflow. Protocols that employ such technique, however, are prone to suffer with from the acknowledgement counterfeiting attack. In this attack, malicious nodes counterfeit acknowledgements in order to force legitimate nodes to drop messages that are not already delivered to destination. Thus, the network performance is degraded. In this work, we analyze a countermeasure for this attack, in which nodes ignore acknowledgements for messages that were not previously received by this node. Three real mobility traces and a well-known mobility model are used in simulations and results show that when approximately 12% of the nodes in the network are malicious, the message delivery rate decreases up to 61 percentage points.*

## 1. Introdução

Na arquitetura TCP/IP, para que haja comunicação entre um par de nós, assume-se que sempre existe um caminho fim-a-fim entre a origem e o destino de uma mensagem. No entanto, tal suposição pode não ser apropriada para modelos existentes de redes sem-fio, que são caracterizados pela grande variação das condições do meio de

transmissão e pela mobilidade dos nós. Desconexões frequentes da rede são resultantes destas características, ou seja, um caminho fim-a-fim pode nem sempre estar disponível ou até mesmo não existir entre os nós que desejam se comunicar. Nestes cenários, a utilização da arquitetura TCP/IP é pouco eficiente e, por isso, faz-se necessário o desenvolvimento de uma nova arquitetura, específica para essas redes, que são chamadas de Redes Tolerantes a Atrasos e Desconexões (*Delay/Disruption Tolerant Networks* - DTN) [McMahon e Farrell 2009, Oliveira et al. 2007]. As redes DTN, são baseadas no paradigma armazena-carrega-e-encaminha (*store-carry-and-forward paradigm*). Nesse paradigma os nós da rede são dotados de *buffers* e podem armazenar persistentemente uma mensagem, caso não haja um caminho fim-a-fim entre origem e destino, até que uma oportunidade de encaminhamento apropriada surja. Essas oportunidades de encaminhamento são chamadas de contato.

Vários protocolos de roteamento, com diferentes características, foram propostos para operarem em redes DTN [Cao e Sun 2013]. É comum que estes protocolos façam uso da replicação das mensagens como forma de aumentar a probabilidade de entrega de mensagens. No entanto, tal medida pode exaurir rapidamente os recursos da rede, levando à sobrecarga dos *buffers* dos nós [Naves et al. 2012]. Para evitar essa sobrecarga, diferentes protocolos baseados em replicação enviam reconhecimentos positivos[1](*acknowledgements*) para sinalizar a entrega de uma mensagem ao destino. Uma forma de implementação de reconhecimentos é através da disseminação na rede de pequenos pacotes contendo identificadores de mensagens que já chegaram ao destino. Ao receber um reconhecimento, um nó é capaz de remover do seu *buffer* as mensagens identificadas por esse pacote e, assim, liberar espaço em *buffer* para evitar descartes desnecessários de mensagens que ainda não foram entregues. Devido ao pequeno tamanho, estes reconhecimentos resultam em pouca sobrecarga e sua efetividade já foi comprovada por outros trabalhos [Balasubramanian et al. 2007, Burgess et al. 2006].

Apesar de comprovadamente eficiente, a técnica de reconhecimentos positivos pode ser utilizada para degradar o desempenho da rede, uma vez que essas mensagens podem ser falsificadas. Esse é ataque de falsificação de reconhecimentos positivos (*ack counterfeiting*) [Burgess et al. 2007]. A ideia é que um nó malicioso pode enviar reconhecimentos com identificadores de mensagens ainda não entregues para o destino. Assim, tais mensagens tem sua probabilidade de entrega reduzida, uma vez que serão removidas dos *buffers* dos nós que receberem esses reconhecimentos falsos. Para lidar com o ataque de falsificação de reconhecimentos, Burgess *et al.* propõem uma contramedida em que os nós somente identificam um reconhecimento como legítimo, se já tiverem recebido anteriormente a mensagem para o qual este reconhecimento foi gerado [Burgess et al. 2007]. Esta contramedida é baseada no fato em que, segundo os autores, as mensagens propagam-se dos nós próximos a fonte para os nós próximos ao destino e que os reconhecimentos destas mensagens seguem o caminho inverso. Os autores demonstraram através de simulação que a contramedida proposta é eficiente, considerando o padrão de mobilidade dos nós da rede UMASS DieselNet.

A suposição de Burgess *et al.*, no entanto, pode não ser verdadeira em outros cenários de mobilidade, nos quais a mensagem de reconhecimento de uma determinada mensagem percorra um caminho diferente do caminho percorrido pela própria mensagem

[1]Deste ponto em diante, reconhecimentos positivos e reconhecimentos são usados como sinônimos.

até o destino. O objetivo deste trabalho, portanto, é avaliar a contramedida de Burgess *et al.* em diferentes cenários de mobilidade. Para a avaliação, são utilizados três registros reais de mobilidade: Rollernet, Infocom05 e DieselNet. Além disso, utilizou-se o padrão de movimentação bem conhecido, o *Random Waypoint*. Os resultados contestam a eficiência desta contramedida, evidenciando a necessidade de projeto de técnicas de defesa mais eficientes.

O restante deste trabalho é organizado como a seguir. Na Seção 2, os trabalhos relacionados são descritos. Na Seção 3, é descrita a motivação para o presente trabalho. A Seção 4 apresenta o ambiente de avaliação das propostas, assim como os registros de mobilidade reais e o padrão de movimentação utilizados nas simulações. Os resultados obtidos através das simulações são apresentados na Seção 5. Na Seção 6, alguns aspectos adicionais são discutidos. Finalmente, as conclusões alcançadas no decorrer do presente trabalho são apresentadas, bem como possíveis extensões a este trabalho e prováveis trabalhos futuros.

## 2. Trabalhos Relacionados

A segurança continua sendo um dos principais desafios dos vários existentes em DTNs [Khabbaz et al. 2012]. A ausência de autenticação dos nós, por exemplo, permite que eles se comportem de maneira maliciosa, atacando a rede de diferentes maneiras. Um destes ataques é conhecido como falsificação de reconhecimentos positivos [Burgess et al. 2007]. Neste ataque, nós maliciosos falsificam reconhecimentos com o objetivo de remover mensagens que ainda não foram entregues ao destino dos *buffers* dos nós, impactando negativamente no desempenho da rede.

Burgess *et al.* avaliaram o ataque de falsificação de reconhecimentos como pouco efetivo, desde que uma contramedida simples seja aplicada. Nesta contramedida, os nós aceitam um reconhecimento como legítimo somente quando já receberam a mensagem indicada pelo reconhecimento em algum momento prévio, tendo esta mensagem sido reencaminhada ou não [Burgess et al. 2007]. Este raciocínio é baseado na premissa de que as mensagens trafegam na rede da origem para o destino e que os reconhecimentos tendem a fazer o caminho inverso. Essa premissa, no entanto, pode não ser verdadeira em DTNs. A mobilidade é vista como um conceito chave nessas redes [Hui et al. 2005]. Isto porque ela desempenha um papel fundamental, que é proporcionar contatos oportunistas entre os nós. Estes contatos, mesmo que não programados, podem ser utilizados para troca das mensagens entre os nós. Porém, a alta mobilidade pode resultar em cenários nos quais os caminhos percorridos pelos reconhecimentos são diferentes dos caminhos percorridos pela própria mensagem.

Outro fato que pode tornar a contramedida de Burgess *et al.* ineficiente é que nós maliciosos podem estar posicionados ou se movimentando próximos às fontes das mensagens. Neste caso, os reconhecimentos falsificados são disseminados quando o número de réplicas da mensagem na rede é baixo, aumentado a probabilidade de eliminar a mensagem em questão da rede. Além disso, mesmo que as premissas assumidas por Burgess *et al.* sejam verdadeiras, ou seja, mesmo que a mensagem e o reconhecimento percorram o mesmo caminho, a contramedida não levará ao efeito desejado. Isto porque nada impede que o reconhecimento falso percorra o mesmo caminho de volta, sendo assumido como legítimo pela fonte da mensagem. Essa possibilidade de ataque

inclusive é apontada por Choo *et al.*, que discutem sobre a possibilidade de um nó malicioso, ao saber da existência de uma mensagem, inundar a rede com reconhecimentos para esta mensagem [Choo et al. 2010].Os autores também propõem uma variação do ataque de falsificação de reconhecimentos denominada *Identity Impersonation Attack*, que atua como uma das componentes de um ataque à rede. Neste mecanismo, os atacantes tentam se passar por diferentes identidades, para atuar como destino para diferentes mensagens, enganando nós intermediários ou as próprias fontes das mensagens. Além disso, ao saber da existência de uma mensagem, os atacantes inundam a rede com reconhecimentos forjados. A outra componente do ataque investigado por Choo *et al.* é denominada *non-deliverable packet flooding* e consiste em enviar mensagens para nós que não existem com o objetivo de causar contenção de recursos. O problema do posicionamento dos nós maliciosos próximos à fonte é examinado na Seção 3.

O presente trabalho avalia a contramedida de Burgess *et al.* com o objetivo de verificar sua eficiência em diferentes cenários de mobilidade, para determinar se novas técnicas devem ser desenvolvidas para combater o ataque de falsificação de reconhecimentos em DTNs.

## 3. O Problema do Nó Malicioso Intermediário

Devido alta mobilidade dos nós em DTNs, um reconhecimento pode percorrer um caminho diferente do que a mensagem para qual ele foi gerado percorreu, e assim, reduzir a eficiência da contramedida proposta por Burgess *et al.*. Outro comportamento não previsto pela contramedida em questão e que, neste trabalho, é chamado de problema do nó malicioso intermediário é ilustrado na Figura 1. Nessa figura, existem dois nós legítimos $A$ e $B$, que utilizam a contramedida de Burgess *et al.* , além de um nó malicioso $C$. Neste cenário, ocorrem quatro contatos que são representados por setas. A ordem cronológica dos contatos é indicada por números posicionados acima das setas. O primeiro contato é entre os nós $A$ e $B$, no qual o nó $A$ envia a mensagem $m_1$ para o nó $B$, cujo destino não é nenhum dos três nós ilustrados no exemplo. Em seguida, ocorre um outro contato entre o nó $B$ e o nó malicioso $C$. Durante este contato, o nó $B$ envia a mesma mensagem $m_1$ que recebeu de $A$ para $C$. O terceiro contato é novamente entre o nó legítimo $B$ e o nó malicioso $C$. Neste contato, o nó $C$ envia um reconhecimento falsificado $ack(m_1)$ para o nó $B$, indicando que $m_1$ foi entregue ao destino. O nó $B$ remove essa mensagem de seu *buffer*, visto que o requisito da contramedida é atendido, ou seja, $B$ já recebeu previamente a mensagem para o qual o reconhecimento foi gerado. O último contato é entre os dois nós legítimos e durante este contato, o nó $B$ reencaminha o reconhecimento falsificado $ack(m_1)$ para o nó $A$, que remove a mensagem $m_1$ de seu *buffer*. Portanto, mesmo aplicando a contramedida de Burgess *et al.*, neste pior caso, uma mensagem foi expurgada da rede por um nó malicioso valendo-se do ataque de falsificação de reconhecimentos.

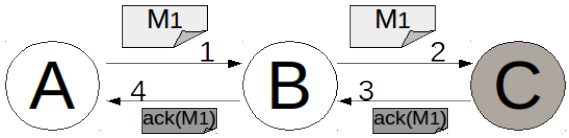



**Figura 1. O problema do nó malicioso intermediário: após quatro contatos, a mensagem $m_1$ é removida dos *buffers* dos nós sem ter sido entregue ao destino.**

Dado o problema exposto e ilustrado pela Figura 1 e considerando-se que estes

eventos podem ser relativamente frequentes e recorrentes, pretende-se avaliar o desempenho da contramedida em diferentes cenários dos utilizados por Burgess *et al*.

## 4. Cenários de Avaliação

Três conjunto reais de mobilidade, são utilizados para avaliar o desempenho da contramedida variando-se a quantidade de nós maliciosos na rede. Também é utilizado um modelo de movimentação sintético, o *Random Waypoint* [Johnson e Maltz 1996]. Esses registros e modelos de movimentação se diferenciam em termos de duração, conectividade e quantidade de nós, de forma a obter resultados para cenários com características bastante diferentes. Assim, é possível avaliar a eficiência da premissa definida por Burgess *et al.* em sua contramedida.

O conjunto de registros Rollernet [Tournoux et al. 2009] é resultado de um experimento no qual foram distribuídos 62 iMotes para voluntários participantes de um circuito de patinação, que acontece regularmente em Paris. A duração do conjunto Rollernet é de aproximadamente 3 horas. No conjunto de registros Infocom05 [Hui et al. 2005], coletado pelo projeto *Haggle*, 41 iMotes foram distribuídos entre os participantes da conferência IEEE Infocom do ano de 2005. Este conjunto tem duração de aproximadamente 3 dias. Por fim, o conjunto de registros DieselNet é resultado de um experimento realizado pela Universidade de Massachusetts, que implantou uma plataforma de testes para DTNs usando seus ônibus [Zhang et al. 2007, Burgess et al. 2006]. Os registros do conjunto DieselNet utilizados neste trabalho, compreendem um conjunto total de 31 nós e foram pré-processados para exclusão dos contatos entre ônibus e pontos de acesso, visando considerar somente as conexões entre os nós móveis da rede.

No modelo de movimentação *Random Waypoint*, os nós movimentam-se para um ponto selecionado aleatoriamente e com um velocidade também selecionada aleatoriamente. Ao chegar no ponto de destino, os nós esperam por um intervalo de tempo aleatório e repetem o procedimento sorteando um novo destino e uma nova velocidade. Nas simulações, os nós são distribuídos por uma área de 3.000 m x 1.300 m, o alcance de transmissão de cada nó é configurado para 50 m, a velocidade de movimentação é de no mínimo 0,5 e no máximo 1,5 m/s e o tempo de espera é de no mínimo 0 e no máximo 120 s. Visa-se com essas configurações aproximar-se o máximo possível de um grupo de pessoas andando em uma determinada área. As características dos conjuntos de dados e do RWM são resumidas na Tabela 1.

O ataque de falsificação de reconhecimentos positivos e a contramedida proposta por Burgess *et al.* foram implementados em um simulador propriamente projetado para validar protocolos de roteamento e aplicações em redes DTN, chamado ONE (*Opportunistic Network Environment*) [Keränen et al. 2009].

Os parâmetros de configuração das simulações também são resumidos na Tabela 1. O padrão de tráfego de mensagens para cada conjunto de registros é o seguinte. Para o Rollernet, 500 mensagens são geradas durante as duas primeiras horas de um tempo total de aproximadamente 3 horas de simulação. Para os conjuntos de dados DieselNet e Infocom05 foram geradas 1.000 mensagens nas primeiras, respectivamente, 143 e 52 horas, aproximadamente, de um tempo de simulação total de aproximadamente 187 e 72 horas. Para o cenário utilizando o modelo de movimentação *Random Waypoint*, optou-se por gerar 1.500 mensagens nas primeiras 20 horas, de um total de 24 horas de tempo

**Tabela 1. Parâmetros das simulações e características dos cenários.**

| Parâmetros/Conjuntos | DieselNet | Infocom05 | Rollernet | RWM |
|---|---|---|---|---|
| Dispositivo | iMote | iMote | iMote | Sintético |
| Duração ($\approx$) | 7.8 dias | 3 dias | 3 horas | 24 horas |
| Número de nós | 26$\sim$31 | 36$\sim$41 | 57$\sim$62 | 40$\sim$45 |
| Número de mensagens | 1.000 | 1.000 | 500 | 1.500 |
| Tamanho das mensagens (MB) | 1 | 1 | 1 | 1 |
| Taxa de Transmissão (Mbps) | 1 | 1 | 1 | 1 |
| TTL | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| *Buffer* | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| Alcance (m) | - | - | - | 50 |
| Área (m x m) | - | - | - | 3.000 x 1.300 |

de simulação. O período de inatividade na geração de mensagens tem como objetivo diminuir a quantidade de mensagens que não chegam ao destino devido ao encerramento abrupto da simulação. O tamanho configurado para as mensagens é de 1,0 MB, visto que redes DTN trabalham com agregados que podem conter várias mensagens. Ainda sobre o padrão de tráfego, em todos os cenários os nós maliciosos não são fonte e nem destino de nenhuma mensagem. Para os cenários DieselNet, Infocom05 e Rollernet, as 10 rodadas de simulação foram obtidas através da geração de 10 arquivos de mensagens distintos.

O número de usuários maliciosos na rede é variado da seguinte forma. O cenário Rollernet possui um total de 62 nós, no entanto, considera-se inicialmente apenas 57 nós presentes na rede. Posteriormente, os nós maliciosos foram adicionados na rede, um a um. Portanto, com 1 nó malicioso, a rede possui 58 nós. Este raciocínio continua até a rede chegar a 62 nós, sendo 5 deles maliciosos, que é o número máximo de nós maliciosos considerados neste trabalho. De forma similar, o cenário DieselNet possui um total de 26 a 31 nós e o cenário Infocom05 possui de 36 a 41 nós. O mesmo raciocínio é utilizado para o cenário RWM, que possui a princípio 40 nós e atinge 45 nós quando são adicionados os 5 nós maliciosos.

Para evitar efeitos negativos da má configuração do TTL, nenhuma mensagem expira durante todo o período de simulação. Além disso, o tamanho do *buffer* foi configurado para comportar todas as mensagens geradas na rede, portanto, em todas as simulações não ocorrem descartes de mensagens devido à sobrecarga dos *buffers*. Uma exceção é feita na Seção 5.3, com o objetivo de avaliar o impacto do tamanho do *buffer* no desempenho da rede.

## 5. Resultados

O objetivo das simulações é avaliar a eficiência da contramedida proposta por Burgess *et al.* quando a rede sofre o ataque de falsificação de reconhecimentos positivos. Os resultados apresentados a seguir foram obtidos através da média de 10 rodadas de simulação distintas. Para todos os pontos das curvas apresentadas, calcula-se um intervalo de confiança para um nível de confiabilidade de 95%, representado por barras verticais. Utilizou-se como métricas para avaliação: a taxa de entrega de mensagens, o atraso de entrega médio das mensagens e o tempo médio em *buffer* das mensagens. Nos gráficos de taxa de entrega, o eixo $X$ representa a quantidade de nós maliciosos na rede, variando de

0 a 5, ou o tamanho do *buffer* variando de 10 MB à 50 MB. O eixo $Y$, representa a taxa de entrega. Para os gráficos de atraso de entrega médio e tempo médio de permanência em *buffer* o eixo $Y$ representa o atraso de entrega médio e o tempo médio de permanência em *buffer*, respectivamente. A unidade de medida utilizada para estas duas métricas é segundos. O eixo $X$, representa a variação de 0 a 5 da quantidade de nós maliciosos na rede.

### 5.1. Impacto do Número de Nós Malicioso na Taxa de Entrega

A taxa de entrega de mensagens é definida como o percentual das mensagens criadas que chegaram ao destino. A Figura 2 ilustra os resultados para todos os quatro cenários de mobilidade avaliados.

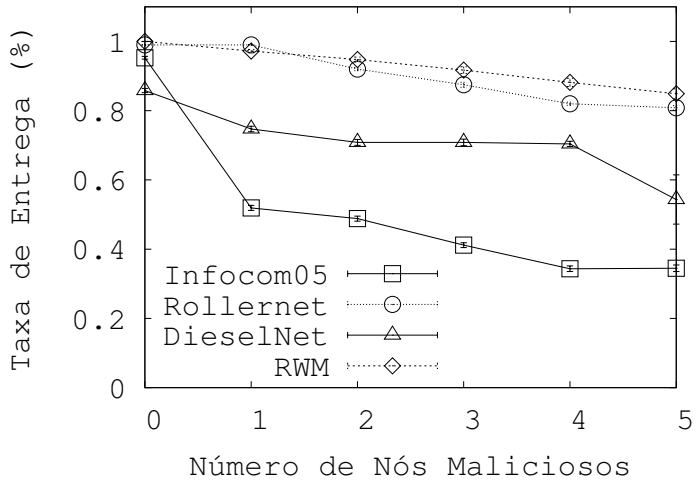



**Figura 2. A taxa de entrega na rede sob ataque.**

É possível observar que, para todos os cenários, a taxa de entrega decresce com o aumento do número de nós maliciosos. Quando a rede não está sob ataque, a taxa de entrega é, no pior caso, superior a 80%. Com 5 nós atacando a rede, no pior caso, a taxa de entrega é inferior a 40%. Para o cenário DieselNet, em particular, a taxa de entrega alcançada pelo cenário cai de aproximadamente 85%, quando não haviam nós maliciosos na rede, até aproximadamente 54%, com a presença de 5 nós maliciosos na rede. Isto representa uma queda de 31 pontos percentuais.

Para o cenário Rollernet, por sua vez, a taxa de entrega é de 99% quando a rede não possui nenhum nó malicioso e permanece inalterada quando um nó malicioso é adicionado à rede. No entanto, a partir da adição do segundo nó malicioso, cada nó adicionado diminui significativamente o desempenho da rede, sendo aproximadamente 92%, 87%, 82% para 2, 3 e 4 nós maliciosos, respectivamente, chegando a aproximadamente 80% com 5 nós maliciosos realizando o ataque contra a rede. É interessante ressaltar que os nós maliciosos formam, mesmo que sem anuência dos mesmos, um conluio contra a rede. Isto acontece em decorrência dos nós maliciosos não aplicarem a contramedida, dessa forma, quando dois nós maliciosos encontram-se, os mesmos acabam trocando os reconhecimentos que têm, sejam eles legítimos ou não. Para o *Random Waypoint*, pode-se observar que a taxa de entrega que é de 100% quando não existem nós maliciosos, cai

para aproximadamente 84% com 5 nós maliciosos atacando a rede. A taxa quando existiam 1, 2, 3 e 4 nós atacando é de aproximadamente 97%, 94%, 91% e 88%. Por último, observa-se que a taxa de entrega para o cenário Infocom05 cai de aproximadamente 95%, quando todos os nós presentes na rede são legítimos, para aproximadamente 34% com a inserção de 5 nós maliciosos na rede. Tratando-se ainda do cenário Infocom05, quando haviam 1, 2, 3 e 4 nós maliciosos na rede, a taxa de entrega obtida é de aproximadamente 52%, 49%, 41% e 34%.

É interessante observar, que mesmo utilizando um cenário com mesmo padrão de mobilidade, DieselNet, obteve-se diferentes resultados e conclusões com relação a efetividade da contramedida proposta por Burgess *et al.*. Esta diferença é atribuída aos demais parâmetros da simulação, tais como, taxa de transmissão, tamanho das mensagens e quantidade das mensagens, que foram configurados com valores diferentes neste trabalho. Isto atesta de que esta contramedida não é ideal em vários cenários.

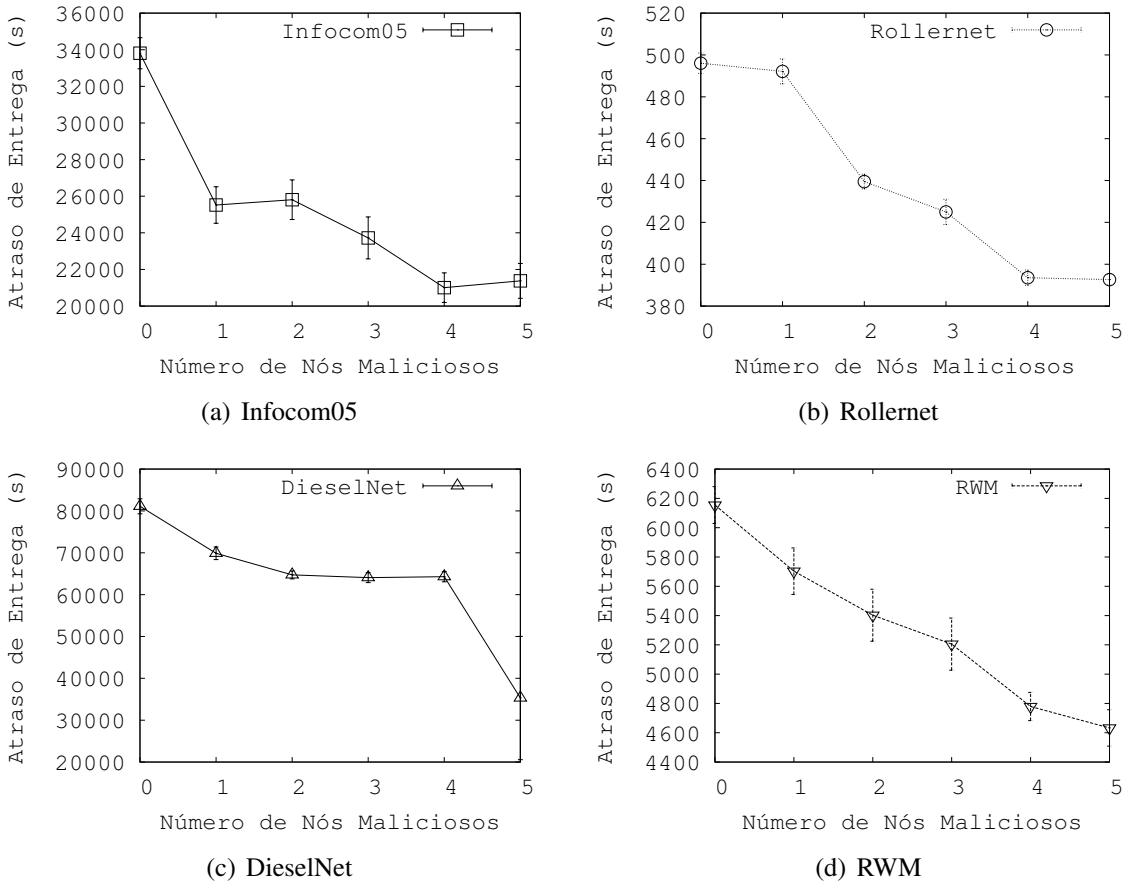
É importante ressaltar que os nós maliciosos formam, mesmo que sem anuência dos mesmos, um conluio contra a rede. Isto acontece em decorrência dos nós maliciosos não aplicarem a contramedida, dessa forma, quando dois nós maliciosos encontram-se, os mesmos acabam trocando os reconhecimentos que têm, sejam eles legítimos ou não. Se esse conluio for feito de comum acordo, nós maliciosos podem consentir em formar um conluio para tentar aumentar a probabilidade de suas mensagens chegarem ao destino, enviando reconhecimentos falsificados das outras mensagens, de forma a expurgá-las da rede. Esta pode ser uma motivação para utilização deste ataque em DTNs implantadas. Neste trabalho, este efeito não foi observado, visto que os nós maliciosos não são fonte e nem destino de mensagens nas simulações.

### 5.2. Impacto do Número de Nós Maliciosos no Atraso de Entrega

O atraso de entrega representa o tempo decorrido desde a criação de uma mensagem até o instante em que ela alcança o destino. Portanto, esta métrica só leva em consideração as mensagens entregues ao destino. A Figura 3 ilustra o atraso de entrega obtido para os 4 cenários avaliados.

Para todos os cenários, o mesmo comportamento é observado. Quanto maior o número de atacantes na rede, menor o atraso. Isto ocorre porque com mais nós maliciosos, mais mensagens são removidas dos *buffers*, o que reduz a carga na rede. Estes resultados indicam também que quanto mais tempo as mensagens permanecem na rede, maior a probabilidade de elas serem afetadas pelo ataque de falsificação de reconhecimentos, sendo algumas vezes indevidamente removidas da rede e não alcançando seu destino final. Ressalta-se que eventualmente reconhecimentos falsificados podem atuar de maneira positiva, removendo mensagens que já chegaram ao destino. Porém, estes reconhecimentos ainda podem ser considerados como falsos, pois foram forjados sem que o nó malicioso atestasse que a mensagem realmente chegou ao destino.

Os resultados também indicam que o ataque de falsificação de reconhecimentos pode ter maior impacto negativo em redes nas quais as mensagens sofram grandes atrasos. Em DTNs, as mensagens podem sofrer atrasos de minutos, horas ou dias, dependendo das especificidades de cada cenário. Portanto, nestes cenários, é necessário que medidas eficientes sejam tomadas contra o ataque de falsificação de reconhecimentos.




(a) Infocom05 (b) Rollernet

(c) DieselNet (d) RWM

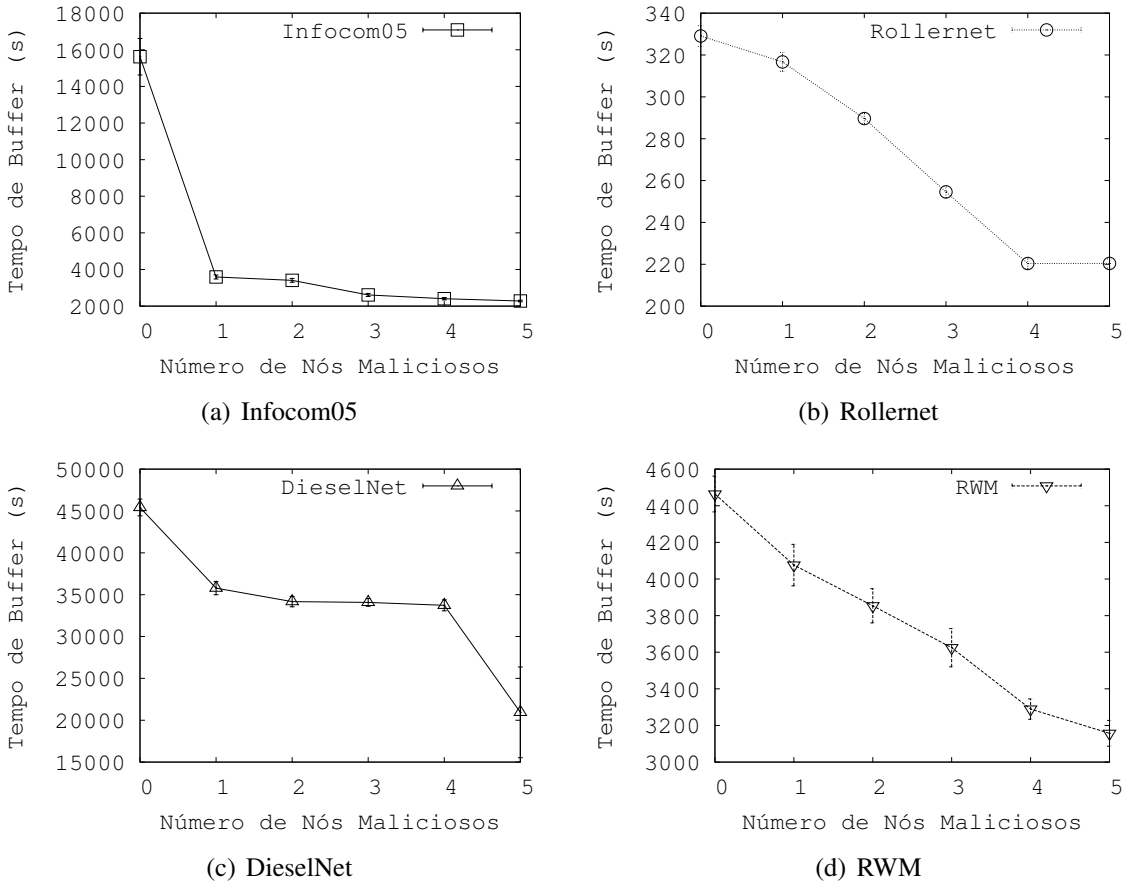
**Figura 3. Atraso de entrega na rede sob ataque.**

### 5.3. Impacto do Número de Nós Maliciosos no Tempo de Permanência em *Buffer*

O tempo de permanência em *buffer* de uma mensagem é computado desde o instante que esta mensagem é criada até quando a mensagem é removida do *buffer* do nó. A remoção pode ocorrer quando o TTL chega ao fim, quando uma mensagem é descartada devido à sobrecarga do *buffer*, quando uma mensagem é encaminhada, ao invés de replicada ou quando um nó recebe um reconhecimento positivo informando que a mensagem chegou ao destino. A Figura 4 mostra o tempo médio de permanência das mensagens em *buffer* para os quatro cenários avaliados. Para todos eles, o tempo médio de permanência em *buffer* diminui conforme aumenta a quantidade de nós maliciosos na rede.

Nas simulações deste trabalho, as mensagens nunca são descartadas devido à expiração do TTL, pois este foi configurado para ter um valor maior que o tempo total de duração da simulação. Além disso, as mensagens também não são removidas dos *buffers* dos nós devido a sobrecarga dos mesmos. Isto porque o tamanho do *buffer* de cada nó é configurado com tamanho suficiente para comportar todas as mensagens geradas na rede. Portanto, restam duas possibilidades para explicar a remoção das mensagens dos *buffers* dos nós: quando as mensagens são encaminhadas ou quando reconhecimentos são espalhados e as mensagens são, então, excluídas por terem chegado ao destino. Portanto, a Figura 4 mostra que conforme mais nós maliciosos ingressam na rede, o tempo médio que as mensagens permanecem em *buffer* diminui. Este fato ocorre devido prin-

cipalmente aos descartes resultantes de reconhecimentos falsificados. Estes resultados corroboram com os resultados apresentados na Seção 5.2, concluindo-se que as mensagens que permanecem mais tempo na rede tem maior probabilidade de serem afetadas pelo ataque de falsificação de reconhecimentos, possivelmente não alcançando o destino.




(a) Infocom05 (b) Rollernet

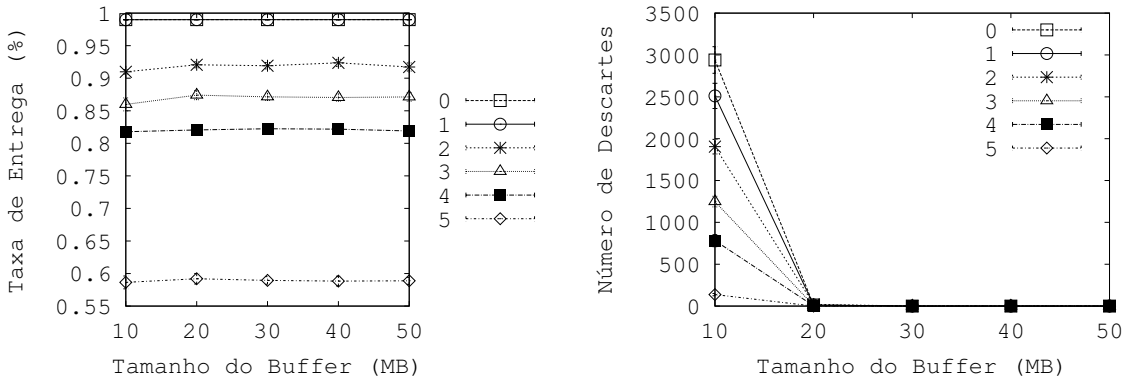
(c) DieselNet (d) RWM

**Figura 4. Tempo médio de permanência das mensagens em *buffer* na rede sob ataque.**

### 5.3.1. Análise do Impacto do Tamanho do *Buffer*

Outro objetivo é verificar o impacto do tamanho do *buffer* no desempenho da rede, com relação a taxa de entrega, durante um ataque de falsificação de reconhecimentos. Para tanto, executa-se simulações para o cenário Rollernet com 5 tamanhos diferentes de *buffer*, variando de 10 MB até 50 MB. Os resultados obtidos são apresentados na Figura 5.

A Figura 5(a) apresenta no eixo $X$ o tamanho do *buffer* e no eixo $Y$ a taxa de entrega, enquanto que as linhas e pontos representam a variação da quantidade de nós maliciosos na rede. A principal conclusão é que o desempenho da rede é mais prejudicado pelo ataque do que pelas restrições do tamanho dos *buffers*. É possível observar que conforme aumenta o tamanho do *buffer* dos nós, há um acréscimo muito pequeno na taxa de entrega alcançada. Este resultado também indica que o protocolo de roteamento MaxProp faz boas escolhas com relação ao gerenciamento de *buffer* e ao encaminhamento de mensagens em cenários com restrições de recurso.




(a) Taxa de Entrega para o Cenário Rollernet (b) Quantidade de descartes para o Cenário Rollernet

**Figura 5. Avaliação do impacto do tamanho do *buffer* quando a rede está sob ataque.**

A Figura 5(b) mostra a quantidade de descartes resultantes de sobrecarga do *buffer* efetuados por todos os nós durante a simulação. Por ela, nota-se a razão da pequena influência do tamanho do *buffer* nos resultados. Observa-se que a quantidade de descartes realizados na rede cai para valores próximos de zero quando o *buffer* é configurado para 20 MB. A partir de 30 MB, a quantidade de descartes efetuados na rede passa a ser 0 para todos os cenários. A Figura 5(b) também corrobora com os resultados apresentados na Figura 5(a) e atesta que o protocolo MaxProp torna mais eficiente a utilização dos *buffers* dos nós utilizando reconhecimentos positivos.

Outra observação interessante extraída da Figura 5(b), é que quando o tamanho do *buffer* é configurado para 10 MB, o aumento de nós maliciosos diminui a quantidade de descartes realizados pela rede. Este é um resultado esperado, pois os reconhecimentos falsificados, ao excluir mensagens, acabam liberando espaço em *buffer* para novas mensagens, evitando a sobrecarga. No entanto, apesar de aparentemente melhorar as condições da rede liberando espaço nos *buffers* dos nós, o ataque de falsificação de reconhecimentos exclui mensagens que ainda não chegaram ao destino, prejudicando a taxa de entrega de mensagens. Finalmente, estima-se que esta diminuição na quantidade de descartes é resultado do aumento de reconhecimentos falsificados, que é resultante do aumento de nós maliciosos na rede.

## 6. Discussões

Neste trabalho, verificou-se o impacto negativo do ataque de falsificação de reconhecimentos na rede, mesmo que os nós adotem a contramedida de Burgess *et al.*. Na avaliação, este ataque foi implementado de forma isolada, sem estar associado ou sendo utilizado em conjunto com qualquer outro tipo de ataque. Na prática, porém, o ataque de falsificação de reconhecimentos pode ser implementado em conjunto com outros ataques, como, por exemplo, o ataque do buraco negro (*black hole attack*). Neste ataque, nós maliciosos anunciam-se como parte do menor caminho para os nós, cujos pacotes ele quer interceptar. Uma vez estabelecida a rota forjada, o nó malicioso pode optar por descartar os pacotes. Assim, tem-se um ataque de negação de serviço ou um ataque do homem no meio (*man-in-the-middle*). Dados os resultados apresentados neste trabalho, é intuitivo pensar que a junção dos ataques de buraco negro e falsificação de reconhecimentos

resultaria em um impacto negativo ainda maior na rede.

As características de conectividade de cada nó também podem ser exploradas pelos nós maliciosos para aumentar a eficiência do ataque. Dentre estas características estão o grau do nó, a centralidade, o número de conexões, o tempo de duração médio de cada contato, o tamanho e demais características da componente conexa da qual o nó em questão faz parte. Um nó malicioso com maior grau, por exemplo, pode enviar reconhecimentos falsos para um número maior de vizinhos. Um nó malicioso com maior centralidade tem mais mensagens sendo encaminhadas por ele e, assim, descobre mais facilmente mensagens que existem na rede. Isto pode comprometer ainda mais a contramedida de Burgess *et al.*. Pretende-se avaliar estas características e sua influência em trabalhos futuros.

Sobre as técnicas de defesa contra o ataque de falsificação de reconhecimentos, uma forma de validar um reconhecimento como verdadeiro, de maneira a neutralizar o ataque, é a utilização de criptografia assimétrica. Desta forma, o nó de destino assinaria o reconhecimento com sua chave privada e os demais nós possuindo a chave pública do nó de destino estariam aptos a certificar que o reconhecimento em questão realmente foi gerado pelo destino da mensagem. Porém, o gerenciamento de chaves em DTNs é reconhecido como um problema difícil [Symington et al. 2011]. Entretanto, existem alternativas como o uso de certificados sociais para o compartilhamento de chaves em DTNs [Oliveira et al. 2013]. Pretende-se avaliar alternativas existentes, de forma a verificar a eficiência dessas alternativas para lidar com o ataque de falsificação de reconhecimentos em DTNs. Ainda com relação as técnicas de defesa contra o ataque de falsificação de reconhecimentos, uma alternativa mais simples é manter a quantidade de reconhecimentos distintos recebidos pela primeira vez de cada nó. Se um nó legítimo recebe uma quantidade muito maior de reconhecimentos pela primeira vez de um determinado nó, do que os seus demais vizinhos, isto pode ser um indicativo de que este nó que envia muitos reconhecimentos pela primeira vez é um nó malicioso.

Por último, é necessária uma avaliação mais detalhada para deliberar até que ponto a utilização de reconhecimentos positivos é benéfica para redes suscetíveis ao ataque de falsificação de reconhecimentos. Basicamente, é preferível utilizar reconhecimentos e ficar suscetível ao ataque de falsificação de reconhecimentos ou evitar este tipo de ataque não utilizando reconhecimentos positivos na rede? Os resultados apresentados na Seção 5.3 motiva-nos a pensar que a utilização de reconhecimentos positivos é benéfica mesmo sob ataques de falsificação de reconhecimentos, no entanto, pretende-se empregar maiores esforços na avaliação deste problema.

## 7. Conclusões e Trabalhos Futuros

Neste trabalho, o impacto negativo do ataque de falsificação de reconhecimentos positivos em DTNs foi avaliado. Avaliou-se também a contramedida proposta por Burgess *et al.* para este ataque. Mesmo com a utilização desta contramedida, constatou-se através de simulações que a presença de um número pequeno de nós maliciosos tem um impacto negativo considerável no desempenho da rede, principalmente, em termos da taxa de entrega de mensagens. Para o cenário Infocom05, por exemplo, 5 nós maliciosos representam aproximadamente 12% dos nós da rede e levaram a uma queda de 61 pontos percentuais na taxa de entrega de mensagens. Além disso, verificou-se que quanto

maior a quantidade de tempo que uma determinada mensagem permanece na rede, maior a probabilidade dessa mensagem ser afetada pelo ataque de falsificação de reconhecimentos. Conclui-se que uma contramedida mais eficiente deve ser projetada para este ataque, que apesar de ser simples de ser implementado, pode reduzir significativamente a taxa de entrega da rede.

Como trabalhos futuros, pretende-se avaliar a contramedida em questão com outros cenários visando reforçar os resultados apresentados neste trabalho. Deseja-se também avaliar o impacto da formação conluios entre os nós maliciosos. A ideia é aumentar a taxa de entrega de mensagens entre os nós maliciosos. Pretende-se estudar ainda, alternativas à contramedida proposta por Burgess *et al.*, incluindo as que utilizam autenticação, com o objetivo de propor alternativas simples para lidar com o ataque de falsificação de reconhecimentos. Pretende-se também estudar e avaliar outros ataques aplicados a DTNs com o objetivo de verificar o desempenho das contramedidas existentes e avaliar o impacto de ataques híbridos.

## Agradecimentos

Esse trabalho é apoiado pela TBE/ANEEL, CAPES, CNPq, FAPERJ, CTIC e Proppi/UFF. Os autores agradecem pelos dados obtidos do arquivo CRAWDAD de Dartmouth College.

## Referências

Balasubramanian, A., Levine, B. e Venkataramani, A. (2007). DTN routing as a resource allocation problem. Em *ACM SIGCOMM*, páginas 373–384.

Burgess, J., Bissias, G. D., Corner, M. D. e Levine, B. N. (2007). Surviving attacks on disruption-tolerant networks without authentication. Em *Mobihoc*, página 61.

Burgess, J., Gallagher, B., Jensen, D. e Levine, B. N. (2006). MaxProp: Routing for vehicle-based disruption-tolerant networks. Em *IEEE INFOCOM*, páginas 1–11.

Cao, Y. e Sun, Z. (2013). Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges. *Communications Surveys & Tutorials, IEEE*, 15(2):654–677.

Choo, F. C., Chan, M. C. e Chang, E.-C. (2010). Robustness of DTN against routing attacks. Em *COMSNETS*, página 148.

Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J. e Diot, C. (2005). Pocket switched networks and human mobility in conference environments. Em *ACM SIGCOMM WDTN Wkshp.*, página 244.

Johnson, D. B. e Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. *Kluwer International Series in Engineering and Computer Science*, página 153.

Keränen, A., Ott, J. e Kärkkäinen, T. (2009). The ONE simulator for DTN protocol evaluation. Em *ICST SIMUTools*, página 55:1—55:10.

Khabbaz, M. J., Assi, C. M. e Fawaz, W. F. (2012). Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. *IEEE Communications Surveys & Tutorials*, 14(2).

McMahon, A. e Farrell, S. (2009). Delay-and disruption-tolerant networking. *IEEE Internet Computing*, 13(6):82–87.

Naves, J. F., Moraes, I. M. e Albuquerque, C. (2012). LPS and LRF: Efficient buffer management policies for delay and disruption tolerant networks. Em *IEEE 37th Conference on Local Computer Networks (LCN)*, páginas 368–375. IEEE.

Oliveira, C. T., Moreira, M. D. D., Rubinstein, M. G., Costa, L. H. M. K. e B.Duarte, O. C. M. (2007). Redes tolerantes a atrasos e desconexões. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores*.

Oliveira, T. R., de Oliveira, S., Macedo, D. F. e Nogueira, J. M. S. (2013). Certificados sociais para segurança em redes veiculares tolerantes a interrupções. Em *Simpósio Brasileiro de Redes de Computadores*.

Symington, S. F., Farrell, S., Weiss, H. e Lovell, P. (2011). Bundle security protocol specification.

Tournoux, P.-U., Leguay, J., Benbadis, F., Conan, V., de Amorim, M. D. e Whitbeck, J. (2009). The accordion phenomenon: Analysis, characterization, and impact on DTN routing. Em *IEEE INFOCOM*, páginas 1116–1124.

Zhang, X., Kurose, J., Levine, B. N., Towsley, D. e Zhang, H. (2007). Study of a bus-based disruption-tolerant network: Mobility modeling and impact on routing. Em *ACM MobiCom*, página 195.

# Uma proposta eficiente de códigos fontanais na distribuição de conteúdo em redes tolerantes a atrasos e interrupções

**Fábio Albini[1], Anelise Munaretto[1], Mauro Fonseca[2], Marcelo Dias de Amorim[3], Francesco De Pellegrini[4]**

[1]Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI), Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, Brasil

[2]Programa de Pós-graduação em Informática Aplicada (PPGIA), Pontifícia Universidade Católica do Paraná (PUC-PR), Curitiba, Brasil

[3]LIP6/CNRS - UPMC Sorbonne Universites, 4 place Jussieu, 75005 Paris, France

[4]CREATE-NET, via alla Cascata 56/D, 38123 Trento, Italy

fabioalbini@yahoo.com

***Abstract.*** *In Delay/Disruption Tolerant Networks (DTNs), content distribution is performed leveraging on the transient and dynamic contacts between moving users. Such a best effort mechanism is prone to unbounded downloading delays and bad user experience. To cope with these occurrences, this paper introduces a novel mechanism to improve content distribution in DTNs. The proposed mechanism is based on the Delay Tolerant Transport Protocol (DTTP)and aims at controlling the diversity in the coded information transmission without increasing the use of network resources or including any feedback messages. In order to validate the proposed mechanism, different scenarios have been evaluated using known DTNs routing protocols. The results show significant gains achieved in terms of delivery ratio in all evaluated scenarios, confirming the efficiency and effectiveness of our protocol independently of the lower layer protocols in use.*

***Resumo.*** *A distribuição de conteúdo em redes tolerantes a atrasos e interrupções (DTNs) é realizada utilizando os contatos transitórios e a dinâmica entre os usuários móveis. Tal comportamento pode ocasionar atrasos indeterminados e má experiência do usuário. Para lidar com esses problemas, este artigo propõe um novo mecanismo para melhorar a distribuição de conteúdo em DTNs. O mecanismo proposto é baseado no Protocolo de Transporte Tolerante a Atrasos com códigos fontanais (DTTP - Delay Tolerant Transport Protocol) e tem por objetivo controlar a diversidade de informações codificadas na transmissão, sem aumentar o uso de recursos da rede ou incluir quaisquer mensagens de retorno (feedback). A fim de validar o mecanismo proposto, diferentes cenários foram avaliados sob diversos protocolos de roteamento para DTNs. Os resultados mostram ganhos significativos em termos de taxa de entrega em todos os cenários avaliados, confirmando a eficiência do mecanismo proposto.*

## 1. Introdução

Os protocolos de Internet formam uma plataforma de comunicação eficiente para interligar dispositivos heterogêneos independentemente da sua localização geográfica. No

entanto, algumas configurações de rede podem determinar a falta de comunicação persistente e confiável, fim-a-fim, entre os nós da rede. Neste contexto, a manutenção de caminhos fim-a-fim com baixa latência e baixa perda de pacotes é um desafio pois a conectividade fim-a-fim é intermitente e pode muitas vezes não existir. Devido a essas características, essas redes são referidas como Redes tolerantes a atrasos e desconexões (DTNs) [Cerf et al. 2007] [Fall 2003] [Oliveira et al. 2008] [Vasilakos et al. 2011]. As DTNs foram desenvolvidos para lidar com os casos quando a conectividade persistente entre origens e destinos não pode ser garantida. Neste tipo de redes, os dados são transportados da fonte ao destino através de nós intermediários (ou seja, nós relays) por meio de um mecanismo de custódia de dados conhecida como *store-carry-and-forward* [Chahin et al. 2011]. Os nós relays participam do roteamento como transportadores de dados. No entanto, é necessário combinar a qualquer protocolo de roteamento um protocolo de transporte adequado para permitir que aplicações realizem a troca de dados. Os mecanismos de distribuição de conteúdo habituais, para garantir transferências de dados confiáveis, são baseados no TCP. O problema é que o TCP requer pelo menos um caminho estável entre a fonte e o destino. Como DTNs são geralmente esparsas e desconexas, um caminho estável entre a origem e o destino não pode ser mantida [Zhang et al. 2009]. Por esta razão, o TCP não é adequado para redes tolerantes a atraso [Fall 2003]. Desta maneira para que ocorra a comunicação utiliza-se um conceito chamado *overlay* onde se implementa uma camada chamada *bundle layer* para que ocorra a comunicação de uma maneira ponto-a-ponto, ao invés de fim-a-fim.

O objetivo é conseguir a distribuição de conteúdo eficiente em DTNs sem a necessidade de um canal de retorno (feedback). Claramente, uma solução possível é usar UDP (User Datagram Protocol). UDP é, contudo, um protocolo de transporte não orientado a conexão, sem confirmação de dados, onde não há garantia de entrega de pacotes e nenhuma informação dos pacotes perdidos. Assim, o UDP pode ser aplicado apenas para aplicações onde essas garantias não são obrigatórias. Uma alternativa à utilização do UDP seria enviar dados sem conexão, mas utilizando uma técnica para assegurar que os dados recebidos estejam completos. Caso os dados recebidos não estiverem corretos, essa técnica deve recuperar a informação faltante sem a necessidade de pacotes de confirmação ou retransmissões. Neste contexto, uma opção pode ser enviar informações redundantes na rede. Os dados não entregues ao destino poderiam ser recuperados usando pacotes redundantes. No entanto, o envio de informações redundantes tem um custo em termos de recursos de sistema. Por tal razão, é necessário introduzir uma política de distribuição, pois é necessário garantir (estatisticamente) que a informação seja recebida pelo menos uma vez. Este problema foi comprovada por Jain et al. ser *NP-hard*. Os autores, em particular, mostram que o sucesso do mecanismo de distribuição é fortemente dependente da distribuição espacial das réplicas através da rede [Jain et al. 2005]. Para contornar este problema de alocação, Altman e De Pellegrini [Altman and De Pellegrini 2009] sugerem o uso de códigos corretores de erros, e.g. códigos fontanais (FC - Fountain Codes) [Mackay 2005]. Baseado nessa ideia, os autores deste trabalho introduziram um protocolo de transporte projetado especificamente para DTNs chamado: *Delay Tolerant Transport Protocol* (DTTP) [Albini et al. 2011a] ou *Protocolo Tolerante a Atrasos* (PTTA) [Albini et al. 2011b]. O protocolo de transporte proposto no trabalho anterior utiliza FC para aumentar a probabilidade de entrega de dados.

A principal contribuição do trabalho proposto neste artigo é a definição e avaliação de um mecanismo de controle capaz de fornecer diversidade na geração de dados e melhorar o equilíbrio entre a distribuição de conteúdo e a utilização de recursos de rede. Este mecanismo é utilizado no DTTP [Albini et al. 2011a] para avaliação da proposta.

O restante deste artigo está organizado da seguinte forma. A seção 2 apresenta os trabalhos mais relevantes da área. A seção 3 descreve brevemente o DTTP e introduz o mecanismo de controle proposto. A seção 4 apresenta uma descrição dos cenários avaliados seguido dos resultados e discussões. Finalmente, a seção 5 conclui o artigo apontando alguns trabalhos futuros.

## 2. Trabalhos Relacionados

O uso de códigos corretores de erros em DTNs tem sido estudado nos últimos anos. Wang et al. comparou o ganho de desempenho com os códigos de correção de erros para protocolos de replicação simples, ou seja, aqueles com envio de várias réplicas da mesma mensagem. A fim de mostrar que os códigos de correção de erros proporcionam benefícios ao desempenho das DTNs, várias simulações foram realizadas em diversos protocolos de roteamento [Wang et al. 2005].

Jain et al. estudou o padrão de encontro não uniforme entre os nós. Além disso, mostra que a probabilidade de entrega de informações é fortemente relacionado com as réplicas alocadas ao longo dos vários caminhos de rede existentes. Os autores estudaram diferentes técnicas de alocação e a complexidade do problema foi provada ser NP-hard [Jain et al. 2005].

O formalismo matemático descrito por Altman e De Pellegrini considera os códigos de correção de erros e códigos fontanais em DTNs. Este é o primeiro trabalho que tem-se conhecimento do que se propõe e estuda o uso de códigos fontanais em DTNs. Os autores limitam sua análise ao armazenamento de apenas um pacote nos buffers dos relays no âmbito de um processo de Poisson [Altman and De Pellegrini 2009].

Dai et al. propôs a integração dos códigos fontanais e o protocolo Optimal Probabilistic Forwarding (OPF). Os dados são codificados e uma regra de encaminhamento é definida para decidir por onde o pacote será enviado [Dai et al. 2010].

Dvir e Vasilakos consideram uma abordagem alternativa chamado roteamento de contrapressão, em que as decisões de roteamento e encaminhamento são feitos tendo por base cada pacote. Informação sobre atrasos de fila, repasse aleatório, e periodicidade de contatos são usados no roteamento de pacotes e nas decisões de encaminhamento. Os resultados da simulação são usados para confirmar as vantagens obtidas [Dvir and Vasilakos 2010].

Spyropoulos et al. dividem as estratégias de encaminhamento existentes em um pequeno número de peças comuns chamadas “módulos encaminhamento”, em seguida, eles mostram como um dado módulo de encaminhamento deve ser aplicado, de acordo com um conjunto de características de rede presentes na aplicação sem fios. Depois disso, eles tentam identificar características genéricas da rede que são realmente relevantes para o processo de roteamento. A idéia foi identificar um conjunto de características úteis para ajudar o usuário a escolher um protocolo de roteamento apropriado tanto para o aplicativo de destino como para a rede [Spyropoulos et al. 2010].

Sun et al. estudaram o desempenho dos protocolos da *convergence layer adapter* (CLA) nas DTNs para transmissão confiável de dados sobre a infra-estrutura de comunicação espacial constituída por taxas assimétricas de canais. Eles também deram uma atenção especial para o recém-desenvolvido *Licklider Transmission Protocol* (LTP) CLA (chamado LTPCL). O desempenho do LTPCL foi comparado com outros CLAs confiáveis, uma combinação de TCP CLA com LTPCL, e avaliados onde havia uma taxa de canais altamente assimétrica nas comunicações cislunar com atraso longo. O LTPCL também foi estudado em um cenário de comunicação do espaço profundo que caracteriza-se por um tempo de atraso e interrupção muito grande no enlace [Sun et al. 2013].

Wang et al. apresentaram uma avaliação experimental do protocolo Bundle (BP) em diferentes protocolos da camada de convergência. Eles usaram um cenário de comunicação cislunar simulado com diferentes atraso de propagação do sinal e perda de dados. Os autores visam a camada de convergência LTP (LTPCL) rodando em cima de UDP/IP (BP/LTPCL/UDP/IP). O desempenho de BP/LTPCL/UDP/IP na transferência de dados realistas é comparada com as duas outras pilhas de protocolo DTN, a saber, a BP/TCPCL/TCP/IP e BP/UDPCL/UDP/IP. Os resultados mostraram que o LTPCL supera o TCPCL para muitos atrasos de enlaces, independentemente da taxa de erro de bit (BER-*Bit Error Rate*). É também afirmado que, em determinada condição, o LTPCL mostrou uma vantagem significativa no goodput sobre o TCPCL em todos os níveis de atraso analisados para o enlace. O LTPCL mostrou também uma vantagem significativa consistente no goodput sobre UDPCL em todos os níveis de atraso e BERs analisados [Wang et al. 2011a].

Wang et al. estudaram os protocolos da camada de convergência DTN em uma infra-estrutura de comunicação cislunar onde há muito atraso, várias taxas assimétricas de canais e variável taxa de perda de dados. O objetivo do trabalho era conhecer qual CLP é o melhor para comunicações cislunar com longo atraso com as taxas de canais altamente assimétricas e se o protocolo de agregação (BP) auxilia a melhorar o goodput na presença de assimetria no canal IPN [Wang et al. 2011b]

Wang et al. estudaram o efeito da agregação de pacotes de dados na comunicação espacial com canais com taxas assimétricas e baixas. Eles também pesquisaram qual é o número de pacotes a serem agregados para as taxas de canal cislunar altamente assimétricos dentro de um bloco visando o melhor desempenho [Wang et al. 2013].

Zeng et al. propuseram não utilizar métricas, tais como atraso, contagem de saltos, e largura de banda para fazer um esquema de roteamento melhor, mas concentrar-se em green communications, como economia de energia, otimizando o desempenho da rede e evitando o aquecimento global. Eles também argumentam que os sistemas de comunicação eficientes são interessantes por causa da poluição, consumo de energia e dissipação de calor. Com isso em mente, os autores apresentam um esquema de escalonamento e encaminhamento direcional (DRSS -Directional Routing and Scheduling Scheme) para DTNs *green vehicle*. O sistema resolve o problema de roteamento e escalonamento como um processo de aprendizagem pelo roteamento geográfico e controle de fluxo na direção ideal. Para tornar as coisas mais rápidas, eles usaram um método híbrido com encaminhamento e replicação de acordo com o tráfego. A validação do sistema proposto foi feita usando uma DTN veicular com um modelo de mobilidade predeterminado. Os resultados mostram que o DRSS, quando equipado com capacidades de

aprendizagem, consegue boa eficiência energética [Zeng et al. 2013].

Um protocolo de transporte para DTNs , chamado *Delay tolerant Transport Protocol* (DTTP), foi proposto por Albini et al. A principal característica deste protocolo é a utilização de códigos fontanais para melhorar a taxa de entrega dos dados. Códigos fontanais tem a flexibilidade de controlar a geração das mensagens codificadas (diversidade de dados) em tempo real, uma vez que os pacotes recém codificados podem ser continuamente enviados pelo nó de origem. No entanto, é necessário definir o tempo de parada para a geração de novos pacotes codificados. Na verdade, se o tempo de parada é muito longo, por conseguinte, uma quantidade significativa de recursos da rede serão desperdiçados. Por outro lado, se for muito curto, a quantidade de redundância pode impor um aumento que não seja substancial na probabilidade de distribuição de conteúdo [Albini et al. 2011a] [Albini et al. 2011b].

O presente artigo apresenta um mecanismo para definir o tempo de parada e a maneira de se propagar as informações redundantes. A restrição geral é representada por um limite imposto na quantidade de dados gerados no nó de origem. Com isso, uma melhoria é alcançada na utilização de recursos de rede e na taxa de entrega de dados de uma forma *cega*, ou seja, sem incluir quaisquer mensagens de feedback ou de controle.

## 3. Proposta

Esta seção começa com uma descrição breve do protocolo DTTP. Em seguida será detalhado o mecanismo cego proposto para limitar a diversidade de informação.

### 3.1. O Delay Tolerant Transport Protocol (DTTP)

DTTP é um protocolo de transporte proposto em [Albini et al. 2011a] e [Albini et al. 2011b] especialmente concebido para DTNs. Um esquema do DTTP simplificado está representado na fig. 1. Este protocolo utiliza técnicas de códigos corretores de erros, mais especificamente códigos fontanais para obter uma taxa de entrega de dados melhor sem um canal de feedback. Seu objetivo é fornecer uma comunicação confiável fim-a-fim e foi projetado para ser independente do protocolo de roteamento. Basicamente, este protocolo foi concebido para transmissão *unicast*. No entanto, também pode ser utilizado para a transmissão de *multicast* ou *broadcast*.

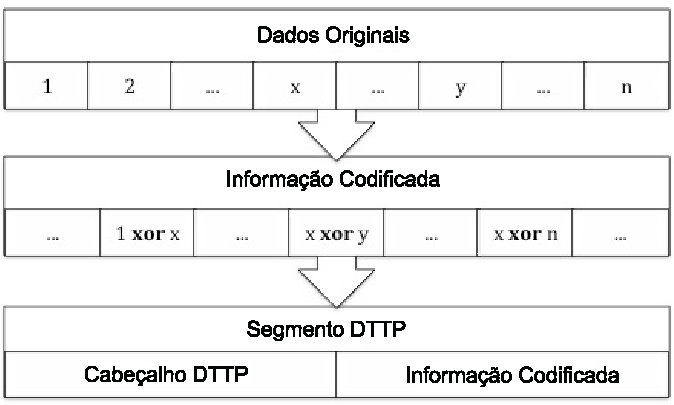



**Figure 1. Funcionamento do DTTP.**

Conforme exibido na fig. 1, cada segmento do DTTP será uma combinação de diversos trechos dos dados originais, o que acaba por gerar a redundância nas informações. Esta redundância permite que caso algum trecho seja perdido, este seja recuperado pela redundância contida em outros segmentos. Para maiores informações vide [Albini et al. 2011a] [Albini et al. 2011b].

### 3.2. Mecanismo proposto

Como foi dito anteriormente, códigos fontanais são utilizados para melhorar a entrega dos dados, independentemente de quais pacotes foram recebidos, só o número de pacotes é relevante. Este deve ser suficiente para que ocorra o processo de descodificação do DTTP. Pode observar-se no entanto, que o custo da manutenção de uma grande quantidade de diversidade de informação através da rede tem efeitos secundários como o aumento da ocupação dos buffers e o aumento da troca de mensagens, devido as operações dos protocolos de roteamento.

Para evitar o efeito colateral da diversidade de informações representado por $\alpha_{DTTP}$, o método adotado é o controle de dois atributos do DTTP: a taxa de geração de segmento $\lambda_{DTTP}$ e o TTL (Time To live) dos pacotes representados por $\psi_{DTTP}$. A escolha desses atributos se deu porque ambos são informações locais e facilmente controláveis pelo nó de origem.

Primeiramente, para definir a referência de carga de rede, avaliou-se o uso da rede quando os dados originais são enviados sem a utilização do DTTP, ou seja, a aplicação envia os pacotes diretamente sobre o protocolo de roteamento usando cabeçalho UDP.

O mecanismo proposto controla os recursos de rede limitando a sobrecarga e o desperdício de recursos. Ele ajusta a taxa de geração de segmentos DTTP $\lambda_{DTTP}$ para moldar a carga da rede e manter o consumo de recursos abaixo do limiar de referência, que foi previamente definido com o envio de dados sem uso DTTP. Para atingir esse objetivo, considera-se o TTL dos pacotes $\psi$ e a taxa de geração de pacotes $\lambda$, (ou seja, a carga da rede). Observa-se que tais valores são locais e conhecidos em cada transmissão DTN.

Eq (1) reproduz a carga da rede, tendo em conta a diversidade:

$$M_{Max} = \begin{cases} (\lambda \times \alpha) \times \psi & \text{se } \psi \leq \Delta t \\ (\lambda \times \alpha) \times \Delta t & \text{caso contrário} \end{cases} \quad (1)$$

onde $M_{Max}$ é o número máximo de pacotes diferentes enviado por um nó fonte da rede, a fim de transmitir um determinado conteúdo, $\lambda$ é a taxa de geração de pacotes, $\psi$ é o Time to Live (TTL, em segundos) do pacote, $\Delta t$ é o tempo de geração de pacotes, e $\alpha$ é a taxa de diversidade de informação, quando o DTTP não é usado $\alpha$=1 .

Diversidade significa aqui o número de diferentes pacotes enviados por uma fonte de aplicação (nó origem) na rede. A descontinuidade da função é devido ao caso em que não há mensagens a serem geradas suficientes enquanto o TTL $\psi$ dura. Nesse caso, o tempo de geração $\Delta t$ é mais curto do que o TTL de uma mensagem. Este é o fator limitante da $M_{Max}$.

Para limitar a carga da rede e, consequentemente, o desperdício de recursos, o mecanismo proposto utilizado na equação 1 representa a carga da rede máxima gerada por

uma fonte de aplicação, ou seja, o número máximo de diferentes pacotes em transporte na rede de uma transmissão. A fim de aproveitar esta equação, os atributos locais $\lambda$, $\psi$, $\Delta t$ e $\alpha$ = 1 são conhecidos e o valor unitário é devido ao fato de que é necessário para enviar pelo menos uma cópia de cada mensagem, a fim de ter a oportunidade de receber todas as mensagens.

Esses são atributos de aplicativo locais sem DTTP. O resultado é apresentado na Equação 2, que une esses parâmetros para descrever a relação entre $\lambda_{DTTP}$, o que representa a taxa de geração de pacotes de DTTP, e o valor correspondente de $\alpha_{DTTP}$, que representa a diversidade de informação quando se utiliza o DTTP. O melhor $\alpha_{DTTP}$ é dependente da topologia da rede. O TTL, usado por pacotes que transportam os segmentos DTTP $\psi_{DTTP}$, está representado na equação 3. A Equação 3 mostra a relação entre $\psi_{DTTP}$, $\psi$ e $\alpha_{DTTP}$:

$$\lambda_{DTTP} = \frac{M_{Max}}{\psi_{DTTP} \times \alpha_{DTTP}} \tag{2}$$

$$\psi_{DTTP} = \frac{\psi}{\alpha_{DTTP}} \tag{3}$$

Outro ponto importante é saber quando parar a geração de segmentos DTTP, porque na abordagem de código fontanal o gerador é continuo. Este tempo de parada, conhecido como $TTL_{fluxo}$, é definido com base no tempo em que o último pacote gerado $\Delta t$ será descartado pelo TTL (tempo de vida). Esta relação é dada por:

$$TTL_{flow} = \Delta t + \psi \tag{4}$$

Para obter o tempo de parada adequado (da geração de segmentos), é necessário utilizar o mesmo $TTL_{fluxo}$ para o caso do DTTP. Usando a Equação 4 sem alterar o $TTL_{fluxo}$, é possível resolver o $\Delta t_{DTTP}$, ou seja, o tempo de parada da geração de segmentos DTTP:

$$\Delta t_{DTTP} = \Delta t + \psi - \psi_{DTTP} \tag{5}$$

Nota-se que, após a expiração do temporizador, nenhum recurso de rede adicional é usado para gerar novos dados codificados.

## 4. Avaliação de Desempenho

Esta seção descreve os experimentos de simulação que realizamos usando o simulador Opportunistic Network Environment (ONE) Simulator [Keränen et al. 2009] para investigar os benefícios do algoritmo proposto.

### 4.1. Cenários de Simulação

Usando ONE, foi possível avaliar o desempenho e a precisão da nossa proposta em cenários de simulação com dois modelos de mobilidade diferentes. O primeiro cenário é o de Helsinki, com limites de movimento impostas pelas ruas, linhas férreas, caminhos de ônibus e calçadas. O segundo cenário usa a mesma área (do mapa Helsink),

mas a mobilidade é gerada de acordo com o modelo de mobilidade Random Way Point (RWP) [Johnson and Maltz 1996][Broch et al. 1998].

Em ambos os casos, há 126 nós. No primeiro cenário os nós são divididos em categorias, incluindo ônibus, carros, pedestres e bondes. Cada um tem predefinidos a velocidade, características do movimento, os limites permitidos de mobilidade e capacidade de armazenamento. O limite de mobilidade permitida é uma característica fundamental do modelo porque os padrões de encontros são afetados por assimetrias neste cenário. Um exemplo é uma rua de mão única, onde as pessoas podem se mover em ambas as direções mas os carros não. Todas estas limitações foram incluídas para se certificar de que o padrão de encontro neste cenário seja assimétrica que é realmente importante para a DTN.

Todos os nós se comunicam através de uma interface Bluetooth com uma taxa de transmissão de 250 Kbps. Tramways, carros e ônibus têm uma interface Wi-Fi adicional com uma taxa de transmissão de 1 Mbps. Estes atributos foram atribuídos para simular um cenário mais realista. No segundo cenário (RWP), os 126 nós têm o mesmo comportamento e todos os nós operam apenas com uma interface Bluetooth.

O objetivo destas simulações é de avaliar o desempenho do protocolo de transporte tolerante a atrasos (DTTP), com a técnica de controle diversidade de dados proposta para mitigar a sobrecarga nos recursos de rede. Para validar os resultados de cada simulação existem vários fluxos de dados entre os nós de tráfego aleatórios (ocorrendo em paralelo). Em seguida, avaliamos a transmissão de dados de um objeto de 10 MB, por exemplo, um pequeno vídeo, usando o DTTP entre dois nós selecionados aleatoriamente em cada simulação. A simulação foi executada até que toda a informação pudesse ser decodificada no receptor ou até atingir o limite de tempo de simulação (cerca de 12 horas). Cem simulações foram realizadas com diferentes sementes aleatórias para cada cenário. Os resultados são a média das 100 simulações com intervalos de confiança de 95 %. O tamanho do pacote foi definido em 500 KB, que é o tamanho padrão usado por aplicativos no simulador ONE [Keränen et al. 2009].

A proposta foi avaliada sob os seguintes protocolos de roteamento: Epidemico [Vahdat and Becker 2000], Prophet [Lindgren et al. 2003] e two hops. Estes protocolos de roteamento foram selecionados porque eles são os protocolos de roteamento mais difundidos para DTNs.

### 4.2. Resultados

Os testes foram divididos em quatro fases. Em primeiro lugar, foi realizada a distribuição de conteúdo de objetos sem usar o protocolo de transporte proposto (DTTP), ou seja, os dados foram fragmentados em pacotes de 500 KB e enviados com UDP. Cada pacote foi enviado apenas uma vez ($\alpha$) = 1, com TTL ($\psi$) = 300 minutos e a taxa de geração de pacotes ($\lambda$) = $\frac{1}{25}$ segundos.

Na segunda fase, a distribuição de conteúdos foi realizada utilizando o protocolo de transporte proposto (DTTP), onde os dados foram codificados e enviados aos protocolos de roteamento mencionados anteriormente. Uma análise dos pacotes recebidos foi feita e a simulação com o tempo de parada da geração das mensagens de maneira ótima ($\alpha$ = ideal) foi realizada. Mas, para saber o tempo de parada da geração ideal, o conhecimento do comportamento da rede é necessário, porque o tempo de parada da geração

ótima é o momento em que o último pacote necessário para decodificar todo o conteúdo foi gerado (feedback instantâneo). Portanto, esta simulação é aqui chamado como o *DTTP ideal*.

A terceira fase, a distribuição de conteúdo também foi realizada utilizando o protocolo de transporte proposto (DTTP), onde o objeto foi segmentado, codificado e depois enviado através dos protocolos de roteamento anteriormente citados. Contudo, nesta fase, não foi utilizado um tempo de parada da geração do segmento, ($\alpha = \infty$), isto é, foi utilizado uma geração contínua de pacotes codificados (diversidade de informação).

Nestas simulações, observou-se que a quantidade de diversidade de informação na rede aumenta muito a quantidade de pacotes na rede, o que leva a um desperdício de recursos devido à geração contínua de pacotes.

Analisando o último pacote necessário para decodificar todo o conteúdo distribuído em função de distribuição cumulativa (CDF) (como mostrado na fig.2), o percentual 99 é cerca de 10 vezes maior do que o conteúdo original. Neste caso, se uma diversidade de informação $\alpha = 10$ é utilizada, se tem o conteúdo distribuído em cerca de 99,9 % de todas as simulações.

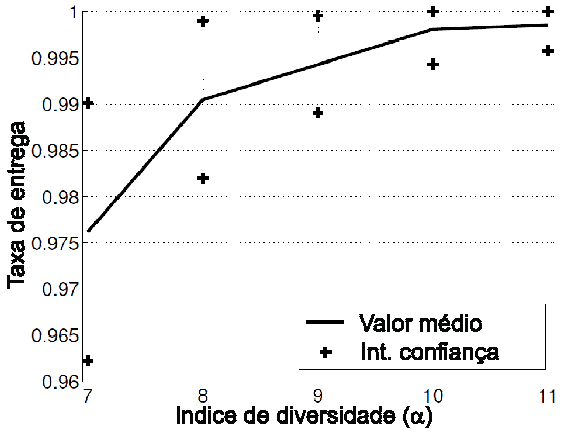



**Figure 2. Função de Distribuição Cumulativa (CDF).**

Na quarta fase, o mecanismo proposto é utilizado para controlar a diversidade de informações e, em seguida, com base na fig 2, $\alpha = 10$ é usada porque representa um limiar coerente após a análise da terceira fase. Portanto, quando se aplica o mecanismo proposto para $\alpha_{DTTP}$ = 10, a seguir parâmetros são calculados como: (1) pela Equação 3 o $\psi_{DTTP}$ = 30 minutos; (2) pela equação 2 o $\lambda_{DTTP} = \frac{1}{900}$ segundos; e (3) pela Equação 3 o $\Delta T_{DTTP}$ = 16700 segundos.

Para alcançar a taxa de entrega ideal usando DTTP, a diversidade de informação $\alpha_{DTTP}$ é mais importante do que o tempo TTL $\psi_{DTTP}$. Como consequência, para alcançar o objetivo de mitigar o custo local de rede, o $\psi_{DTTP}$ é reduzido na mesma proporção em que a diversidade de informação aumenta (Eq. 3).

Em todas as simulações, foram utilizados os cenários DTN descritos na Seção 4.1. Em todos os testes, os parâmetros de entrada são iguais às configurações da rede, ou seja, ambos têm o mesmo tamanho de pacote, o mesmo tipo de semente aleatória, o mesmo número de nós de rede, entre outros.

Os resultados de simulação, com o protocolo epidêmico são apresentados na fig. 3.

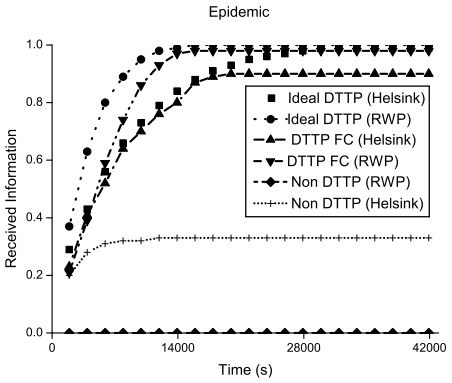



**Figure 3. Sucesso na entrega usando o protocolo de roteamento epidêmico.**

Nesta figura, podemos observar que no cenário Random Way Point usando o protocolo de roteamento Epidêmico e enviando as informações sem DTTP, a quantidade de informação recebida é insignificante. Isto porque a média, com intervalo de confiança de 95 %, era praticamente zero. Ele também mostra que, em quase todas as (cem) simulações, a informação não foi recebida. Este parece ser um valor errado, mas se entendermos o protocolo epidêmico quando a mobilidade segue o modelo RWP, é possível ver o que acontece. Neste cenário, o tráfego paralelo está sendo gerado. É importante ter isso em conta, porque quando o nó de interesse gera seus pacotes, os buffers de quase todos os relays estão usados e podem estar cheios. Isso significa que eles iriam descartar os pacotes de interesse, que muito afeta a taxa de entrega, pois as mensagens são geradas apenas uma vez, ou seja, $\alpha_{DTTP}$ = 1. Em comparação com o caso em que é utilizado DTTP, que alcançou melhores índices médios de entrega, uma vez que pode ser visto na fig. 3. Além disso, o cenário de Helsinki permite atingir uma taxa de entrega de cerca de 90 % (ou seja, um ganho de três vezes), no pior caso, com o protocolo de transporte DTTP, em comparação com 30 % sem DTTP.

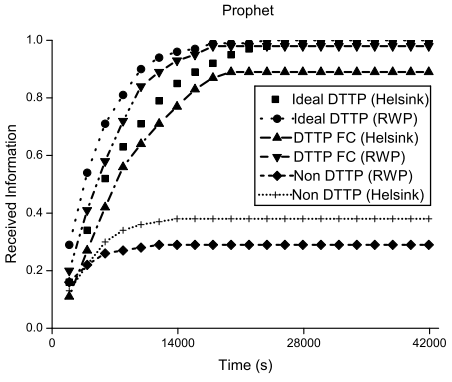



**Figure 4. Sucesso na entrega usando o protocolo de roteamento prophet.**

Fig. 4 mostra que quando o protocolo de encaminhamento Prophet sem DTTP é utilizado, a quantidade de informação recebida é de cerca de 30 % para o cenário RWP e

cerca de 40 % em Helsinki. Entretanto, ao usar DTTP, a taxa de entrega média é de quase 100 %, mais de três vezes maior (que o caso sem DTTP) no cenário RWP e mais de duas vezes em Helsinki.

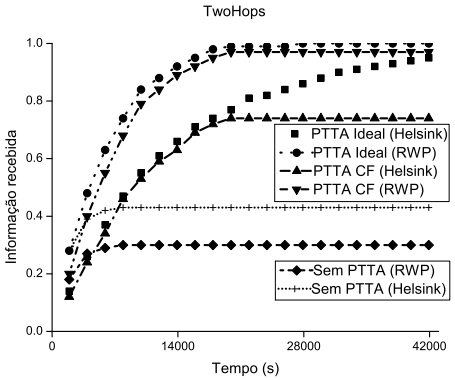



**Figure 5. Sucesso de entrega utilizando o protocolo de roteamento Twohops.**

Na Figura 5 é visível o impacto do número de encontros entre os nós ao se utilizar o protocolo twohops. Como no cenário RWP a quantidade de encontros entre os nós é maior, devido a inexistência de limitações como ruas e calçadas, a taxa de entrega com o PTTA com o controle da fonte quase atingiu 100%, enquanto quando não se utilizou o protocolo ficou próximo dos 30%. Nos gráficos referentes a Helsinki é possível observar que a propagação da mensagem tornou-se mais lenta, isso porque o número de encontros entre os nós é baixo. Além disso, é possível se fazer o paralelo com o resultado matemático obtido, porém como este refere-se ao momento em que todos os nós recebem todas as mensagens, ele é um pouco defasado com relação ao obtido em simulação, onde apenas o destino é monitorado.

Como pode ser visto em todos os resultados, quando o controle na geração de mensagens é usado com DTTP, uma taxa de entrega de dados melhor é atingida (próxima a ideal). Mas ainda não é a ideal, pois para ela ser alcançada, um canal de retorno é necessário ou um grande $\alpha_{DTTP}$ e $\Delta t_{DTTP}$ que leva aos efeitos secundários mencionados anteriormente. Esse canal de retorno é necessário para saber exatamente quando os pacotes necessários foram recebidos no lado do destino e, mesmo assim, com este controle de diversidade de informações não seria tão eficiente, porque a fonte iria manter a geração de informações até que a mensagem de feedback chegasse, sendo que pode nunca ser recebida. Porém, a proposta visa eliminar a necessidade deste canal de retorno, assim, os resultados foram muito promissores. Para provar isso, verificou-se o uso dos recursos da rede através do tráfego nesta.

Fig. 6 mostra o número de pacotes diferentes distribuídos pela rede versus o tempo de simulação. É muito importante notar que o número de pacotes diferentes (utilização de recursos da rede) foi moldado com o mecanismo proposto de geração de diversidade de informação. Em seguida, o número de pacotes diferentes na rede em cenários iniciais, onde DTTP não foi utilizado, é comparado. Com esta comparação, é possível ver que a área dos dois gráficos são muito semelhantes, o que mostra que os recursos de rede utilizados são inferiores para quando utilizou-se o DTTP e os resultados obtidos para as taxas de dados de entrega, como mostrado anteriormente, são ainda melhores. Claramente

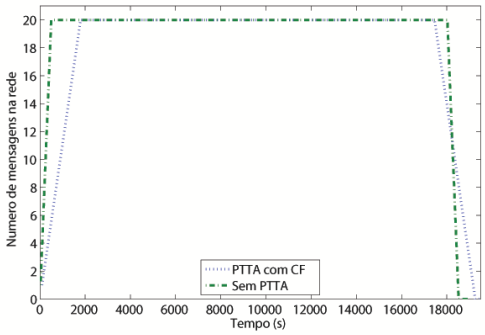



**Figure 6. Número de pacotes diferentes do objeto distribuído por tempo de simulação.**

se obtém uma economia de recursos e um melhor desempenho.

## 5. Conclusões

Neste trabalho, foi proposto um método de controle da fonte com garantia estatística de entrega para o protocolo DTTP a fim de otimizar a distribuição de conteúdo em DTNs. Este método baseia-se no perfil de uso da rede e sobre o controle da diversidade de informação. Avaliou-se o desempenho da proposta e demonstrou-se que ela é capaz de melhorar a utilização dos recursos da rede e exigindo menor sobrecarga pois não utiliza pacotes de controle (*feedback*). Ao mesmo tempo, a probabilidade de distribuição de conteúdo mostrou melhorias significativas em relação ao encaminhamento de mensagens não codificadas em DTN.

Os resultados apresentados foram avaliados em cenários realistas, i.e. onde existe tráfego concorrente na rede gerando um grande número de mensagens em nós intermediários (relays) e preenchimento dos buffers. Isso prova que a proposta alcança, mesmo em cenários reais, um ganho importante quando comparado com outras propostas. Através da avaliação do esquema proposto sob diferentes protocolos de roteamento, também mostrou que a proposta alcança melhores resultados a respeito de transmissão de dados, mostrando a sua independência das camadas inferiores da pilha de protocolos.

Como trabalho futuro, pretende-se realizar testes em algumas condições extremas com relação aos recursos da rede, tais como cenários com altas taxas de descarte de mensagens, muito esparsos, desconexos e buffers menores.

## References

Albini, F. L. P., Munaretto, A., and Fonseca, M. (2011a). Delay tolerant transport protocol - DTTP. *Global Information Infrastructure Symposium (GIIS), 2011*, pages 1 – 6.

Albini, F. L. P., Munaretto, A., and Fonseca, M. (2011b). Protocolo de transporte tolerante a atrasos - PTTA. *XXIX Simpósio Brasileiro de Telecomunições - SBrT2011*.

Altman, E. and De Pellegrini, F. (2009). Forward correction and fountain codes in delay tolerant networks. In *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*, pages 1899–1907. IEEE.

Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., and Jetcheva, J. (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '98, pages 85–97, New York, NY, USA. ACM.

Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H. (2007). RFC 4838, Delay-Tolerant Networking Architecture. *IRTF DTN Research Group*.

Chahin, W., De Pellegrini, F., El-Azouzi, R., and Azad, A. P. (2011). Blind online optimal forwarding in heterogeneous delay tolerant networks. *IFIP - Wireless Days 2011*.

Dai, Y., Yang, P., Chen, G., and Wu, J. (2010). Cfp: Integration of fountain codes and optimal probabilistic forwarding in dtns. In *GLOBECOM*, pages 1–5. IEEE.

Dvir, A. and Vasilakos, A. V. (2010). Backpressure-based routing protocol for dtns. In Kalyanaraman, S., Padmanabhan, V. N., Ramakrishnan, K. K., Shorey, R., and Voelker, G. M., editors, *SIGCOMM*, pages 405–406. ACM.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA. ACM.

Jain, S., Demmer, M., Patra, R., and Fall, K. (2005). Using redundancy to cope with failures in a delay tolerant network. *SIGCOMM Comput. Commun. Rev.*, 35:109–120.

Johnson, D. B. and Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA. ICST.

Lindgren, A., Doria, A., and Schelén, O. (2003). Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7:19–20.

Mackay, D. J. C. (2005). Fountain codes. *Communications, IEE Proceedings-*, 152(6):1062–1068.

Oliveira, C. T. d., Taveira, D. M., Braga, R. B., and Duarte, O. C. M. B. (2008). Uma proposta de roteamento probabilístico para redes tolerantes a atrasos e desconexões. *XXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, SBRC.*, pages 735–748.

Spyropoulos, T., Rais, R. N., Turletti, T., Obraczka, K., and Vasilakos, A. (2010). Routing for disruption tolerant networks: taxonomy and design. *Wirel. Netw.*, 16(8):2349–2370.

Sun, X., Yu, Q., Wang, R., Zhang, Q., Wei, Z., Hu, J., and Vasilakos, A. (2013). Performance of dtn protocols in space communications. *Wireless Networks*, pages 1–19.

Vahdat, A. and Becker, D. (2000). Epidemic Routing for Partially Connected Ad Hoc Networks.

Vasilakos, A., Zhang, Y., and Spyropoulos, T. (2011). *Delay Tolerant Networks: Protocols and Applications*. Wireless Networks and Mobile Communications Series. CRC Press.

Wang, R., Burleigh, S., Parikh, P., Lin, C.-J., and Sun, B. (2011a). Licklider transmission protocol (ltp)-based dtn for cislunar communications. *Networking, IEEE/ACM Transactions on*, 19(2):359–368.

Wang, R., Wei, Z., Dave, V., Ren, B., Zhang, Q., Hou, J., and Zhou, L. (2011b). Which dtn clp is best for long-delay cislunar communications with channel-rate asymmetry? *Wireless Communications, IEEE*, 18(6):10–16.

Wang, R., Wei, Z., Zhang, Q., and Hou, J. (2013). Ltp aggregation of dtn bundles in space communications. *Aerospace and Electronic Systems, IEEE Transactions on*, 49(3):1677–1691.

Wang, Y., Jain, S., Martonosi, M., and Fall, K. (2005). Erasure-coding based routing for opportunistic networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 229–236, New York, NY, USA. ACM.

Zeng, Y., Xiang, K., Li, D., and Vasilakos, A. V. (2013). Directional routing and scheduling for green vehicular delay tolerant networks. *Wireless Networks*, 19(2):161–173.

Zhang, Q., Jin, Z., Zhang, Z., and Shu, Y. (2009). Network coding for applications in the delay tolerant network (dtn). In *Proceedings of the 2009 Fifth International Conference on Mobile Ad-hoc and Sensor Networks*, MSN '09, pages 376–380, Washington, DC, USA. IEEE Computer Society.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 4*

## *Redes Orientadas a Conteúdo*

# Uma Estratégia Rápida e Eficiente de Localização e Encaminhamento em Redes Orientadas a Conteúdo

**João Vitor Torres e Otto Carlos M. B. Duarte**

[1]Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ
Rio de Janeiro, Brazil
Email: {jvitor, otto}@gta.ufrj.br

***Abstract.*** *Content-Centric Networks focus on the communicated data name fundamentally changing the network task of locating and forwarding information. Additionally, the huge amount of content names challenges the scalability of technics used for this task. This article proposes a strategy that consolidates the control plane on a dedicated node apart of switches responsible for the data plane. The proposal shows higher performance compared with others strategies identified in the literature. The comparison uses analytical modeling and simulation to measure convergence delay and efficiency in terms of useful and signaling traffic ratio. The results prove the proposal superior performance with the scalability of the network topology size, the number of content names and the content request profile.*

***Resumo.*** *As Redes Orientadas a Conteúdo focam o nome do dado comunicado alterando de forma fundamental a tarefa de localização e encaminhamento de informação em rede. Adicionalmente, a enorme quantidade de nomes de conteúdo desafia a escalabilidade das técnicas utilizadas nessa tarefa. Este artigo propõe uma estratégia com consolidação do plano de controle em nó separado dos comutadores responsáveis pelo plano de dados. A proposta apresenta desempenho superior em comparação com as demais estratégias identificadas na literatura. A comparação utiliza modelagem analítica e simulação para medir o tempo de convergência e a eficiência em termos da relação entre tráfego útil e de sinalização. Os resultados obtidos comprovam o desempenho superior da proposta com a escalabilidade do tamanho da topologia da rede, do número de nomes de conteúdo e do padrão de solicitação de conteúdo.*

## 1. Introdução

As Redes Orientadas a Conteúdo (*Content Centric Network* - CCN) [Jacobson et al. 2009] mudam drasticamente os princípios de localização e encaminhamento, passando o foco diretamente para o nome do conteúdo e não mais o endereço da máquina, ou hospedeiro, como é hoje na Internet. Isto tem a grande vantagem de permitir que cópias locais do conteúdo sejam armazenadas em diferentes pontos e, consequentemente, mais perto do usuário ao invés de solicitadas repetidamente à fonte. Adicionalmente, pedidos paralelos para o mesmo nome são agregados e uma única solicitação é encaminhada a frente através da rede. Um dos principais desafios da rede orientada a conteúdo é a escalabilidade da localização e do encaminhamento,

Este trabalho recebeu recursos da PETROBRAS, FINEP, FUNTTEL, CNPq, CAPES e FAPERJ.

uma vez que a quantidade de conteúdo é bem maior que a quantidade de hospedeiros. Para tratar a tarefa de localização de forma escalável, a proposta de rede orientada a conteúdo (CCN) utiliza nomeação hierárquica de conteúdos, organizando os nomes em uma estrutura em níveis vinculada à topologia de nós da rede. Tal estrutura hierárquica permite a agregação dos nomes de conteúdo em seus prefixos comuns na direção do nível mais alto da hierarquia. A agregação possibilita a divulgação concisa de sumários de localização usando esquemas de roteamento baseados em anúncios de prefixos de nomes.

O vínculo entre nome e localização herda as limitações do IP em tratar requisitos de mobilidade e hospedagem multidomicílio, requisitos que são ampliados pela distribuição de cópias de conteúdo na rede, muitas vezes fora do caminho até a fonte. O anúncio de rotas para nomes fora da hierarquia desagrega prefixos, aumenta o tráfego de controle e as tabelas de encaminhamento, o que as torna economicamente inviáveis atualmente para redes CCN [Perino and Varvello 2011].

Este artigo propõe uma nova estratégia de localização e encaminhamento de informação em redes CCN baseada na separação dos planos de dados e de controle. A proposta restringe e aprofunda a análise realizada em trabalho anterior [Torres et al. 2013] focando em cenários com apenas um controlador responsável pelo plano de controle. A nova estratégia inunda a rede apenas pontualmente para localização do controlador e não de forma periódica como inicialmente proposto. Além disso, a nova proposta mantém a reutilização da memória das tabelas de encaminhamento dos nós substituindo registros antigos por novos sob demanda. O artigo compara a proposta com estratégias existentes demonstrando desempenho superior em uma composição de cenários. A avaliação define duas métricas para a comparação: o atraso de convergência e a eficiência de sinalização. O trabalho avalia o compromisso de cada estratégia entre estas duas métricas frente a variação de características da rede e do perfil de tráfego.

O restante deste artigo está organizado da forma a seguir. Na Seção 2 os principais trabalhos relacionados são apresentados. As estratégias de localização e encaminhamento estudadas são detalhadas na Seção 3. Na Seção 4 é apresentada a modelagem analítica e os resultados de simulação para o desempenho de cada estratégia em função de parâmetros de topologia e de tráfego. Por fim, na Seção 5 ressalta-se as principais conclusões.

## 2. Trabalhos Relacionados

Os esquemas atuais de roteamento aplicados no CCN constroem as regras de encaminhamento baseadas no OSPF. O OSPF inunda toda a rede com atualizações de prefixos não agregados, impondo fortes limitações de escalabilidade quanto ao número de prefixos distintos e a mobilidade do conteúdo [Wang et al. 2012]. A proposta NLSR [Hoque et al. 2013] usa um esquema de sincronismo de base de dados salto a salto em substituição as inundações do OSPF para propagar os anúncios de prefixos na rede. Apesar de evitar inundações na rede, a eficiência do esquema NLSR em função do número de prefixos e do tamanho da rede ainda não é bem conhecido.

Os esquemas OSPF e NLSR atuam proativamente anunciando para a rede os prefixos de conteúdo disponíveis. As propostas [Chiocchetti et al. 2013, Xie et al. 2013] utilizam abordagem reativa ao adaptar as tabelas de encaminhamento sob realimentação do plano de dados. Estas propostas inundam a rede com interesses sem correspondência na *Forwarding Information Base* - FIB e mediante recepção de conteúdo em resposta

adicionam novas entradas mais específicas na FIB. Apesar de fornecer baixo tempo de convergência, a sobrecarga de sinalização destas propostas não é modelada de forma comparativa com outras abordagens evidenciando o impacto das inundações.

Baid *et al.* utiliza um esquema que mapeia os prefixos de conteúdo em nomes planos únicos e este nomes em endereços topológicos de rede, reduzindo os requisitos de memória e de troca de mensagens de controle [Baid et al. 2012]. De forma análoga, o trabalho [Zhu et al. 2013] utiliza um esquema do tipo DNS para mapear nomes de conteúdo em prefixos de nome vinculados a segmentos de rede de acordo com a movimentação do conteúdo. Estas propostas quebram o vínculo fundamental entre o conteúdo do pacote e o seu nome utilizado para encaminhamento, o que é essencial para a agregação de solicitações para o mesmo conteúdo.

Yi *et al.* ressaltam a adaptabilidade do encaminhamento em função de informações obtidas do plano de dados e discutem se há necessidade de protocolos de roteamento em redes CCN [Yi et al. 2013]. Apesar da reposta afirmativa, os autores argumentam que o requisito de tempo de convergência para CCN é menos restrito quando comparado a redes IP. Contudo, não há consideração sobre os requisitos para o tamanho das tabelas de encaminhamento.

Propostas do tipo redes definidas por *software* (*Software Defined Networks* - SDN) empregam um controlador para instalar, sob demanda, nos nós da rede, regras de encaminhamento de pacotes por fluxo [Mattos et al. 2011, Fernandes et al. 2011]. Estas propostas fazem a separação das funções de roteamento em plano de controle, que calcula as rotas, e plano de dados, que executa o encaminhamento dos pacotes. Um controlador processa as mensagens de controle e, assim, reduz os requisitos de memória e de processamento dos nós comutadores. O conceito de separação de planos das redes SDN é utilizado no presente trabalho para uma proposta de localização e encaminhamento de informação em redes CCN. Ao contrário de propostas anteriores [Salsano et al. 2013], a presente proposta é independente de infraestrutura baseada no protocolo IP e *OpenFlow* [McKeown et al. 2008].

## 3. Estratégias de Localização e Encaminhamento Estudadas

### 3.1. Estratégia Proposta *CRoS-NDN*

Este trabalho propõe a estratégia *Controller based Routing Strategy for Named Data Networking - CRoS-NDN*, a qual separa os planos de controle e de dados. O controlador calcula as métricas de roteamento a partir de um esquema distribuído de obtenção da topologia da rede e do mapa de prefixos associados a cada nó produtor de conteúdo. O esquema utiliza inundação pontualmente para que os nós localizem o controlador e posteriormente registrem informações da rede nele. Os nós consultam o controlador sempre que um interesse não tenha registro compatível na sua FIB local. A partir da resposta do controlador, o nó constrói um interesse especial que desencadeia a instalação de registros na FIB dos nós até o produtor do conteúdo para o nome no interesse enviado pelo consumidor. Caso o tempo de espera de um registro na *Pending Interest Table* - PIT expire, então o registro associado na FIB é removido. A ausência de resposta ao interesse pode ser causada por mudanças na topologia da rede tornando o registro na FIB inválido. O Algoritmo 1 apresenta o resumo em alto nível do esquema utilizado para obtenção de informações pelo controlador e fornecimento de rotas sob demanda para os demais nós.

**Algoritmo 1:** Resumo CRoS-NDN

**Entrada**: nó *i*, controlador *x*;
**repita**
  nó *i* monitora vizinhos e descobre controlador *x*;
  **se** *a lista de vizinhos de i mudou* **então**
    *i* atualiza registro no controlador;
    controlador atualiza lista de vizinhos do nó *i*;
  **fim**
  **se** *i está registrado e existem novos prefixos de produtores* **então**
    *i* registra novos prefixos de produtores de conteúdo;
  **fim**
  **se** *i recebe interesse sem registro compatível na FIB* **então**
    *i* solicita rota ao controlador;
    *i* gera interesse especial com resposta do controlador e o encaminha instalando rota até o produtor;
  **fim**
  **se** *i recebe interesse de instalação de rota para prefixo prefixoA* **então**
    *i* adiciona registro na FIB para *prefixoA* apontando para próximo salto da rota;
  **fim**
**até** *sempre*;

### 3.2. Estratégia *OSPFLike*

Ao contrário da *CRoS-NDN*, a estratégia *OSPFLike* utiliza o procedimento de inundação de forma recorrente para cada prefixo a ser anunciado na rede. O nó produtor inunda a rede anunciando os prefixos de nome para os quais ele possui conteúdo. A inundação de interesses ocorre sem garantia de entrega e precisa ser reforçada periodicamente. O produtor utiliza um prefixo especial adicionado aos nomes anunciados. Todos os nós possuem um registro na FIB correspondente a este prefixo especial e este registro aponta para todas as interfaces do nó como saída permitindo a inundação. Um nó ao receber um interesse com o prefixo especial identifica que se trata de um anúncio e inclui um registro na FIB correspondente ao nome anunciado apontando para interface de entrada deste interesse. Caso o anúncio seja recebido por múltiplas interfaces, utiliza-se a interface com menor número de saltos até o produtor. Para os interesses de consumidores, em caso de ausência de resposta o estouro do tempo de espera do registro na PIT provoca a remoção do registro na FIB utilizado no encaminhamento do respectivo interesse. Os nós são agnósticos à topologia da rede e têm apenas uma visão local de qual interface de saída utilizar para cada prefixo com anúncio recebido.

### 3.3. Estratégia *NLSRLike*

A estratégia *NLSRLike*, baseada em [Hoque et al. 2013], substitui a inundação periódica de prefixos da estratégia *OSPFLike* por um esquema de sincronismo entre as bases de dados dos nós. Esta base de dados, chamada *Link State DataBase* - LSDB, armazena a informação do estado da rede com registros chamados *Link State Advertisements* - LSAs. A estratégia *NLSRLike* utiliza dois tipos de LSA. O primeiro tipo, LSA de vizinhos, arma-

zena a informação de vizinhança de um salto de um nó. O segundo tipo, LSA de prefixo, armazena a associação entre um prefixo de nome e a identificação do nó produtor.

O esquema de sincronismo da base de dados LSDB utiliza troca de *hashes* dos registros LSA para verificar e atualizar a consistência entre nós vizinhos. Cada nó constrói um mapa da topologia da rede e da associação entre prefixos e nós produtores a partir da LSDB. Ao receber um interesse de um consumidor, o nó verifica localmente o identificador do produtor do prefixo associado ao nome solicitado e calcula a interface de saída usando o algoritmo de Dijkstra para determinar o menor caminho até o produtor.

### 3.4. Estratégia *ARPLike*

A estratégia *ARPLike*, inspirada em [Chiocchetti et al. 2013, Xie et al. 2013], troca a abordagem proativa de anúncio de prefixos das estratégias anteriores pela busca reativa utilizando inundação conforme necessário. O nó inunda a rede sempre que não existe um registro específico na FIB para o interesse. Ao receber o dado correspondente a um registro ainda válido na PIT, o nó adiciona ou atualiza o registro na FIB para o respectivo prefixo do dado apontando para a interface de recepção deste dado. Os interesses subsequentes para o mesmo prefixo são encaminhados diretamente utilizando o registro específico na FIB. Caso o tempo de espera de um registro na PIT expire, então o registro associado na FIB é removido. A expiração pode ser causada, por exemplo, por mudanças na topologia da rede tornando o registro na FIB inválido.

### 3.5. Estratégia de Inundação - *Flooding*

A estratégia *Flooding* representa o cenário de pior caso da *ARPLike*, no qual cada novo interesse tem prefixo diferente de todos os anteriores. Na estratégia *Flooding* e na *ARPLike* com tal cenário, o nó encaminha cada interesse recebido em todas as interfaces exceto a de recepção. A agregação de interesses na PIT impede a ocorrência de ciclos.

### 3.6. Estratégia Omnisciente - *Omniscient*

Esta é uma estratégia onde todas as entradas da FIB são calculadas *a priori* sem adicionar sobrecarga de sinalização e não há atraso de convergência. A estratégia *Omniscient* é utilizada como referência na comparação das demais estratégias, pois tem o melhor desempenho possível.

## 4. Análise de Desempenho

As estratégias estudadas empregam diferentes abordagens para a localização e o encaminhamento de conteúdo em rede. As estratégias *OSPFLike* e *NLSRLike* anunciam em inundação e de forma proativa a informação de localização na rede. As estratégias *ARPLike* e *Flooding* inundam a rede de forma reativa buscando o conteúdo sob demanda. A estratégia *CRoS-NDN*, proposta deste trabalho, evita inundações recorrentes da rede ao consolidar a informação de localização em um nó controlador. Esta seção compara o desempenho de cada estratégia utilizando duas métricas: (i) eficiência de sinalização e (ii) atraso de obtenção do conteúdo. Primeiro as duas métricas acima são modeladas através de equações, ver seção 4.1. Em seguida, é apresentada uma análise de tendência para casos especiais de interesse, ver seção 4.2. Finalmente, os resultados são validados através de simulação, ver seção 4.3.

### 4.1. Modelagem analítica

A métrica de eficiência de sinalização (*ES*) é definida em função da fração útil de tráfego (*FU*). *FU* é definida como a razão entre o número de conteúdos distintos recebidos pelo consumidor e o número de interesses trafegados nos enlaces da rede. A equação (1) modela a *FU* de cada estratégia. As variáveis utilizadas no modelo são listadas na Tabela 1. A *ES*, equação (2), é definida em função da *FU* para a estratégia *Omniscient*, caso ideal e referência para comparação com *ES = 1*. A estratégia *Omniscient* propaga tráfego apenas pelo caminho ótimo de acordo com a taxa de interesses dos consumidores *TC*.

A métrica de atraso é composta pela soma de três componentes: *AD* - o atraso de ida e volta direto entre consumidor e produtor de conteúdo; *AA* - o atraso entre o anúncio de um prefixo de conteúdo pelo produtor e o alcance de toda a rede; e *AT* - o atraso entre uma mudança na topologia da rede e o recálculo de rotas. As equações (3), (4) e (5), respectivamente, fornecem o cálculo destes componentes de atraso.

A modelagem de cada estratégia é detalhada individualmente a seguir considerando ambas as métricas. São consideras as distâncias entre consumidor e produtor, entre consumidor e controlador e entre produtor e controlador como *D*, pior caso.

A estratégia *Omniscient* fornece o menor atraso possível composto apenas pela parcela *AD*. As estratégias *Flooding* e *ARPLike* proporcionam atraso igual à *Omniscient*, porém com sobrecarga de interesses de sinalização. No caso *Flooding*, o total de interesses é proporcional ao número de enlaces da rede *E* e a taxa de interesses *TC*. No caso *ARPLike*, a sobrecarga adicional é proporcional ao número de enlaces da rede *E* e à taxa de interesses sem registro específico na FIB, $TC \times FI$.

$$FU_{Omniscient} = \frac{TC}{TC \times D} = \frac{1}{D} \tag{1a}$$

$$FU_{Flooding} = \frac{TC}{TC \times E} = \frac{1}{E} \tag{1b}$$

$$FU_{ARPLike} = \frac{TC}{TC \times (FI \times E + (1 - FI) \times D)} = \frac{1}{(FI \times E + (1 - FI) \times D)} \tag{1c}$$

$$FU_{OSPFLike} = \frac{TC}{NP \times E \times TV + TC \times D} \tag{1d}$$

$$FU_{NLSRLike} = \frac{TC}{2 \times E \times (TV + TA + TT) + TC \times D} \tag{1e}$$

$$FU_{CRoS-NDN} = \frac{TC}{2 \times E \times TV + TT \times E + D \times (N \times TT + TA + TC \times (FI + 1))} \tag{1f}$$

$$ES_{estrategiaX} = \frac{FU_{estrategiaX}}{FU_{Omniscient}} \tag{2}$$

$$AD_1 = 2 \times AE \times D \quad {}^1Omniscient, Flooding, ARPLike, OSPFLike, NLSRLike \tag{3a}$$

$$AD_{CRoS-NDN} = 4 \times AE \times D \tag{3b}$$

**Tabela 1. Parâmetros para Avaliação das Estratégias**

| Tipo | Variável | Descrição |
|---|---|---|
| Entrada | N | Número de nós da rede |
| Entrada | E | Número de enlaces da rede |
| Entrada | D | Diâmetro da rede |
| Entrada | TC | Taxa de interesses do consumidor |
| Entrada | NP | Número total de prefixos de conteúdo |
| Entrada | TA | Taxa de anúncio de prefixos |
| Entrada | FI | Fração de interesses para prefixos ainda não solicitados |
| Entrada | TV | Taxa de monitoração de vizinhança |
| Entrada | TT | Taxa de alterações de topologia |
| Entrada | AE | Atraso de propagação no enlace |
| Saída | FU | Fração útil de tráfego |
| Saída | ES | Eficiência de sinalização |
| Saída | AD | Atraso de ida e volta direto entre consumidor e produtor |
| Saída | AA | Atraso entre anúncio de novo prefixo e alcance em toda rede |
| Saída | AT | Atraso de atualização da topologia |

$$AA_2 = 0 \quad {}^{2}Omniscient, Flooding, ARPLike \tag{4a}$$

$$AA_3 = AE \times D \quad {}^{3}OSPFLike, CRoS - NDN \tag{4b}$$

$$AA_{NLSRLike} = (4 \times AE + \frac{1}{TV}) \times D \tag{4c}$$

Na estratégia *OSPFLike*, a inundação de anúncios com frequência *TV* e com quantidade de interesses proporcional ao total de prefixos anunciados *NP* também gera sobrecarga de sinalização. Além disso, alterações de topologia geram atraso adicional na entrega de conteúdo. Este atraso, *AT*, é inversamente proporciona a taxa *TV* e adicionado ao atraso de propagação de anúncios na rede, *AA*.

Na estratégia *NLSRLike*, a sobrecarga de sinalização é proporcional à taxa de monitoração de vizinhança *TV* e ao número de enlaces *E*. Adicionalmente, o sincronismo da LSDB gera dois novos interesses para cada nova LSA em cada enlace, seja LSA de alteração de topologia ou de novo anúncio de produtor. O sincronismo da LSDB acarreta também atraso adicional na entrega de conteúdo. Este atraso depende da verificação de consistência da LSDB entre vizinhos e que ocorre a intervalos $\frac{1}{TV}$. Após detecção de inconsistência, a atualização de LSA entre dois vizinhos adiciona duas iterações consecutivas de interesse e resposta entre eles, ou seja, adiciona atraso igual a quatro vezes o

**Tabela 2. Análise de Tendência para a Eficiência de Sinalização - ES**

| Cenário | ES |
|---|---|
| $E >> D$ | Flooding $\rightarrow 0$ |
| $E >> D, FI \rightarrow 1$ | ARPLike $\rightarrow 0$ |
| $E >> D, TV = 1, NP \rightarrow TC$ | OSPFLike $\rightarrow 0$ |
| $TC >> E, E >> D, TV = 1, TT = 0, TA \rightarrow TC$ | NLSRLike $\rightarrow 0$ |
| $TC >> E, E >> D, TV = 1, TT = 0, TA \rightarrow TC$ | CRoS-NDN $\rightarrow 1/(2 + FI)$ |
| $TA = 0, TT = 0, FI = 0$ | NLSRLike = CRoS-NDN |

tempo de propagação no enlace entre os vizinhos. A soma destes atrasos multiplicado pelo diâmetro da rede fornece o tempo total de convergência.

De forma análoga à *NLSRLike*, a estratégia *CRoS-NDN* adiciona igual sobrecarga de sinalização para monitoração de vizinhança entre vizinhos. Porém, alterações de topologia e novos anúncios de produtor são encaminhados diretamente ao controlador com número de mensagens adicional proporcional ao diâmetro da rede no pior caso. Cada alteração de topologia acarreta também nova inundação de descoberta do controlador, seguida de renovação de registro dos nós com caminho até o controlador afetado, afetando todos os nós no pior caso. Interesses sem registro na FIB demandam ainda consulta ao controlador para obtenção de rota, adicionando a parcela $D \times TC \times FI$. Em relação ao atraso, consultas ou registro de prefixos no controlador são proporcionais a *D* no pior caso. As alterações de topologia demandam intervalo $\frac{1}{TV}$ para detecção e no máximo $3 \times AE \times D$ adicionais, caso seja necessário localizar novamente o controlador.

$$AT_4 = 0 \quad {}^4Omniscient, Flooding, ARPLike \tag{5a}$$

$$AT_{OSPFLike} = \frac{1}{TV} \tag{5b}$$

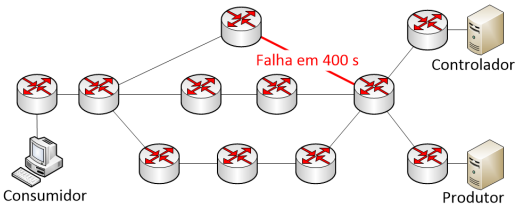
$$AT_{NLSRLike} = (4 \times AE + \frac{1}{TV}) \times D \tag{5c}$$

$$AT_{CRoS-NDN} = 3 \times AE \times D + \frac{1}{TV} \tag{5d}$$

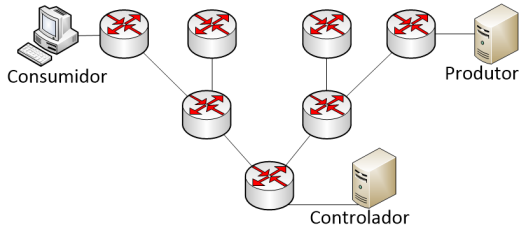
## 4.2. Análise de Tendência

A eficiência de sinalização $ES$ das estratégias em cenários de crescimento da rede e do número de prefixos distintos de conteúdo é de especial interesse, pois indicam sua escalabilidade. A Tabela 2 resume os resultados obtidos para casos limite. Em particular, avalia-se o crescimento da rede com restrição no crescimento do diâmetro evitando aumento ilimitado do atraso fim a fim. A estratégia *ARPLike* degrada com o aumento da proporção de tráfego sem registros específicos na FIB. Quando o número de prefixos é significativamente superior à memória FIB disponível, perfis de tráfego com baixa correlação entre os prefixos solicitados, $FI \rightarrow 1$, provocam inundações recorrentes.

Em cenários com topologia estável, com número de interesses de consumo *TC* e de anúncio de conteúdo *TA* com valor próximo e numericamente muito maior do que a




(a) Topologia com 11 nós, 12 enlaces, diâmetro 4, enlaces de 1 Gbit/s, $AE = 10ms$ e falha de enlace.




(b) Topologia com 7 nós, 6 enlaces, diâmetro 4, enlaces de 10 Mbit/s e $AE = 1ms$.

**Figura 1. A topologia (a) é utilizada na simulação de verificação de convergência após falha de enlace, ver Figura 3. A topologia (b) é utilizada nas simulações que comparam o desempenho sob influência do número de prefixos, da taxa de interesses, do tamanho da topologia e da taxa de monitoração de vizinhança, ver Figuras 4, 5, 6 e 7 respectivamente.**

quantidade de enlaces $E$, a eficiência $ES$ das estratégias *OSPFLike* e *NLSRLike* tende à zero. Na mesma situação, a eficiência da estratégia *CRoS-NDN* converge para a constante um terço no pior caso.

O último cenário analisado mostra que o desempenho das estratégias *NLSRLike* e *CRoS-NDN* são iguais quando não há novas mudanças de topologia ou novos anúncios de prefixo e todos os conteúdos solicitados já possuem registro na FIB.

### 4.3. Simulação

As estratégias estudadas foram implementadas no simulador ndnSIM [Afanasyev et al. 2012]. As Figuras 1 a 7 apresentam as topologias utilizadas para obtenção da evolução temporal da eficiência de sinalização $ES$. Estas curvas permitem verificar também o atraso total, $AD + AA + AT$, associado a cada estratégia. Para suavizar as curvas e permitir legibilidade, a $ES$ foi calculada tomando valores médios do total de pacotes em intervalos de 25 segundos. Todas as curvas têm intervalo de confiança de 95%. O ambiente de simulação acrescenta um salto ao diâmetro total da rede. O salto correspondem ao enlace entre o nó hospedeiro da aplicação e a própria aplicação

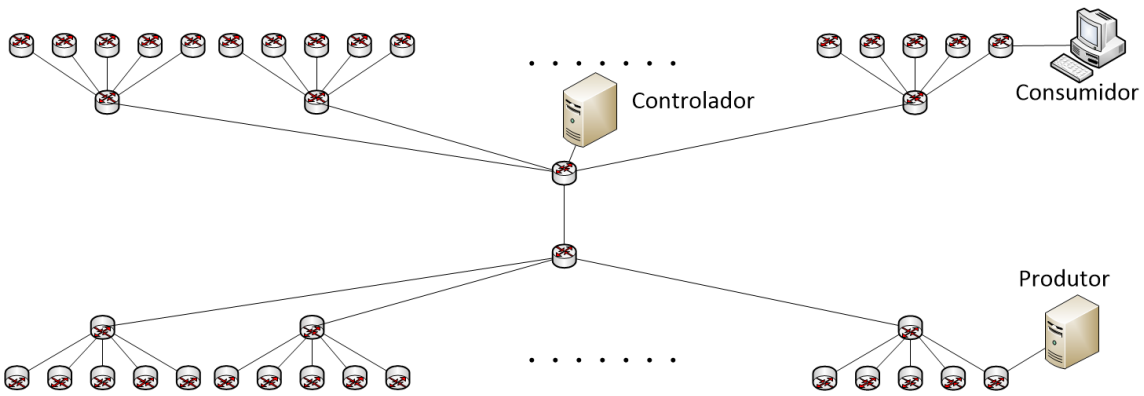



**Figura 2. Topologia utilizada na simulação com comparativo de desempenho em função do tamanho da rede, ver Figura 6. Características: 122 nós, 121 enlaces, diâmetro 5, enlaces de 100 Mbit/s, $AE = 10ms$, consumidor e produtor em extremidades opostas, controlador no centro.**

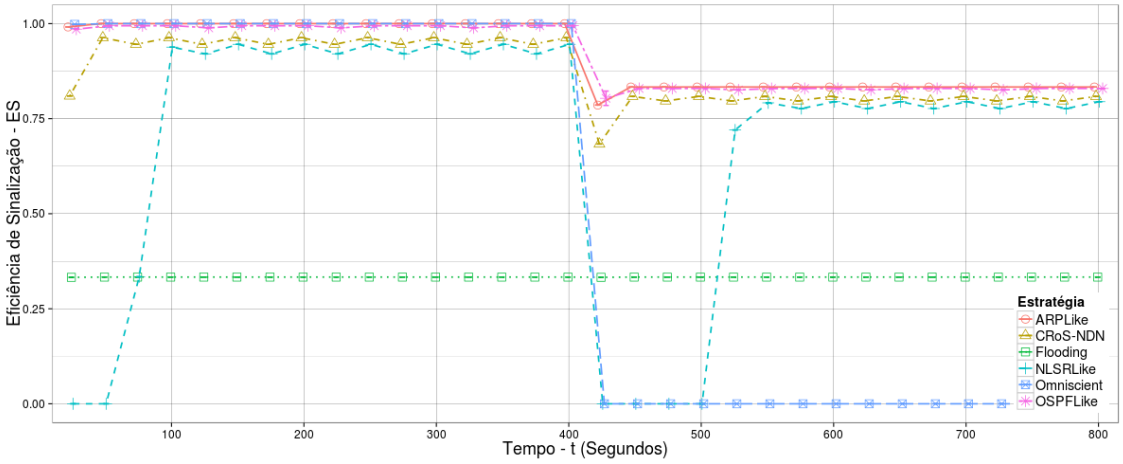
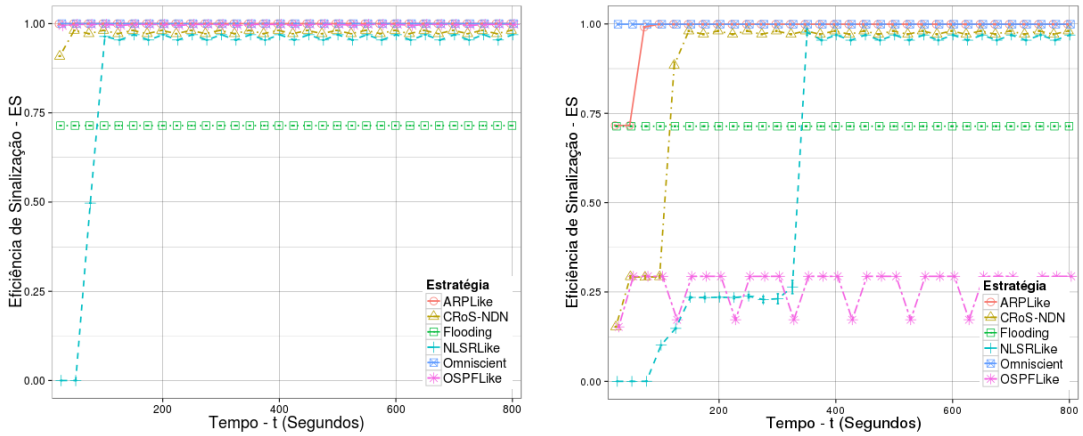



**Figura 3. Atraso de convergência inicial e após falha de enlace em 400s utilizando um único prefixo de conteúdo. A estratégia *NLSRLike* apresenta o maior atraso, a *Flooding* o pior $ES$. As demais convergem para valores próximos de $ES$. Parâmetros: topologia de 11 nós da Figura 1(a), $TC = 20$, $FI = 1$ e $TV = 0, 1$.**

consumidora de conteúdo. O salto adicional explica a diferença entre os valores obtidos para *ES* pela modelagem analítica e os valores obtidos na simulação. O consumidor busca sempre nomes de conteúdos distintos garantindo que o armazenamento local não interfira nos resultados. Os nós não têm limite de registros na FIB, então a fração de prefixos distintos solicitados *FI* só tem impacto até que todos os prefixos *NP* sejam solicitados.

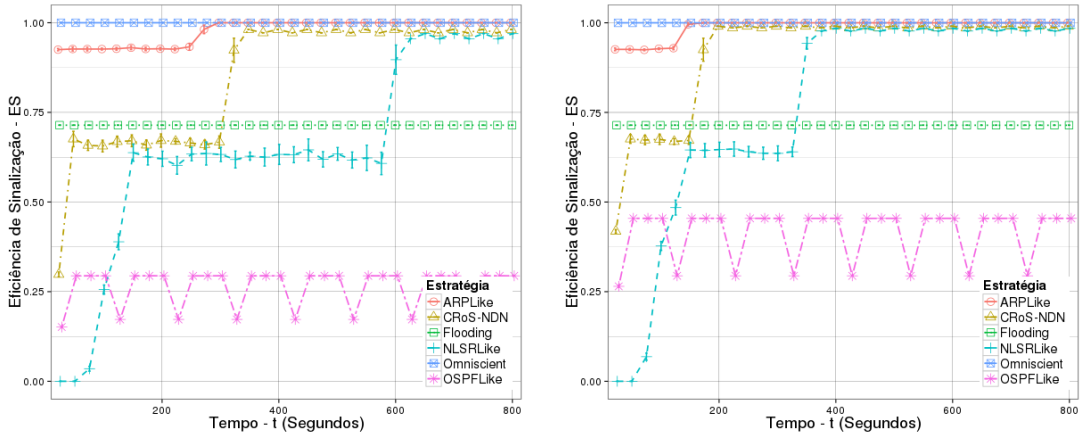
Primeiro avalia-se a consistência da implementação verificando a convergência inicial e em caso de falha de rompimento de enlace. A Figura 3 apresenta o resultado de simulação de falha de rompimento de enlace conforme a topologia da Figura 1(a). A estratégia *Omniscient* não considera caminhos alternativos e a métrica $ES$ é calculada utilizando os valores de desempenho *Omniscient* antes da falha. Após a falha, devido ao maior número de saltos no novo caminho entre consumidor e produtor, $ES$ resulta em desempenho sempre inferior a um (1). A Figura 3 mostra o maior atraso da estratégia




(a) Um prefixo de conteúdo, $NP = 1$.

(b) Mil prefixos de conteúdo, $NP = 1000$.

**Figura 4. Redução da eficiência e aumento do atraso de convergência com aumento do número de prefixos de 1 (a) para 1000 (b). Parâmetros: topologia de 7 nós da Figura 1(b), $TC = 20$, $FI = 1$, $TA = 50$, $TV = 0, 1$ e $TT = 0$.**



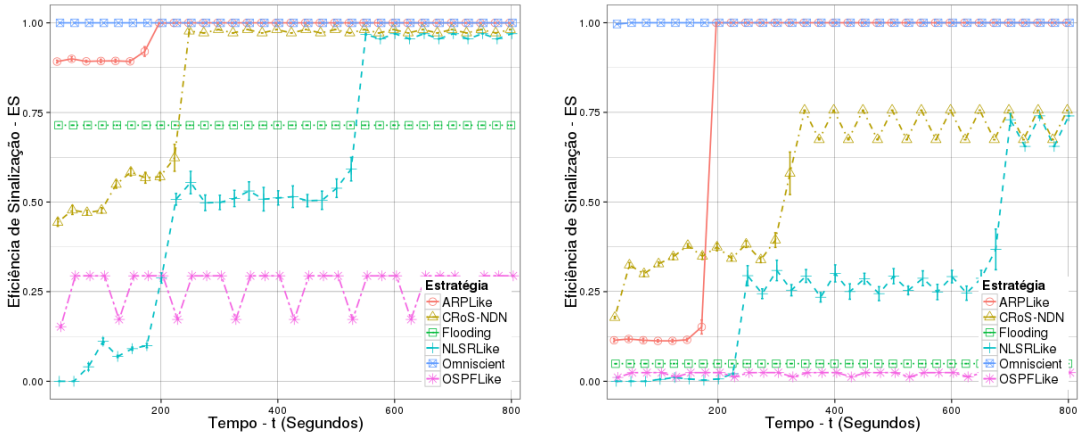

(a) Taxa de 20 interesses por segundo, $TC = 20$. (b) Taxa de 40 interesses por segundo, $TC = 40$.

**Figura 5. Aumento da eficiência de sinalização em função do aumento da taxa de interesses do consumidor de (a) para (b). Parâmetros: topologia de 7 nós da Figura 1(b),** $NP = 1000$**,** $FI = 0,2$**,** $TA = 50$ **e** $TV = 0,1$**.**

*NLSRLike* para início da entrega de dados ao consumidor e a convergência de todas estratégias para valores próximos de $ES$, exceto da *Flooding* que sempre inunda a rede. A estratégia *NLSRLike* tem tempo de convergência ainda maior após a falha. A implementação *NLSRLike* não recalcula os registros da FIB a cada nova LSA recebido, mas sob demanda quando não há registro válido na FIB. Esta abordagem diminui a carga computacional, porém aumenta o atraso de convergência devido ao tempo adicional de expiração de entradas sem resposta na PIT e remoção da entrada FIB utilizada.

A Figura 4 compara o desempenho da $ES$ em função do número total de prefixos anunciados e consumidos. As estratégias *Ominiscient* e *Flooding* não são afetadas pela variação da quantidade de prefixos. O consumidor inicialmente varre todo o conjunto prefixos antes de repetir prefixos já instalados na FIB, $FI = 1$. A estratégia *ARPLike* tem desempenho inicial igual a *Flooding* até que todos os prefixos são instalados na FIB. O desempenho da estratégia *OSPFLike* degrada em função do aumento do número de prefixos. As estratégias *CRoS-NDN* e *NLSRLike* passam por quatro transições com aumento da $ES$: numa primeira fase ocorre a convergência de topologia e não há tráfego útil, na segunda fase os prefixos são registrados na rede e prefixos solicitados pela primeira vez disparam o processo de cálculo e instalação de novos registros na FIB, na terceira fase não há novos registros de prefixos e ainda ocorre instalação de novos registros na FIB, na última fase apenas tráfego útil e de monitoração de vizinhança é encaminhando. O registro de prefixos foi configurado para taxa de 50 prefixos por segundo, totalizando 20 segundos para registrar os 1000 prefixos. A instalação de registros na FIB dura 50 segundos utilizando a taxa de consumo de 20 interesses por segundo com $FI = 1$. A estratégia *NLSRLike* é a mais afetada no tempo de convergência pelo aumento do número de prefixos.

A Figura 5 compara o desempenho da $ES$ em função da taxa de interesses enviada pelo consumidor. As estratégias *Ominiscient* e *Flooding* não são afetadas. Nas demais estratégias, o aumento da taxa de tráfego útil aumenta a eficiência $ES$, pois o tráfego de sinalização não se altera. A simulação utiliza $FI = 0,2$, isto implica que o consumidor consome dois novos prefixos a cada dez interesses enviados. Este valor de $FI$ tem dois




(a) Topologia de 7 nós da Figura 1(b).

(b) Topologia de 122 nós da Figura 2.

**Figura 6. Redução da eficiência e aumento do atraso de convergência em função do aumento do tamanho da topologia de rede de (a) para (b). Parâmetros:** $TC = 20$**,** $NP = 1000$**,** $FI = 0,3$**,** $TA = 10$ **e** $TV = 0,1$**.**

efeitos: primeiro aumenta o tempo da fase instalação de registros na FIB das estratégias *ARPLike*, *NLSRLike* e *CRoS-NDN*; e, em segundo, aumenta a $ES$ nesta fase de instalação de registros na FIB quando comparado com o cenário com $FI = 1$.

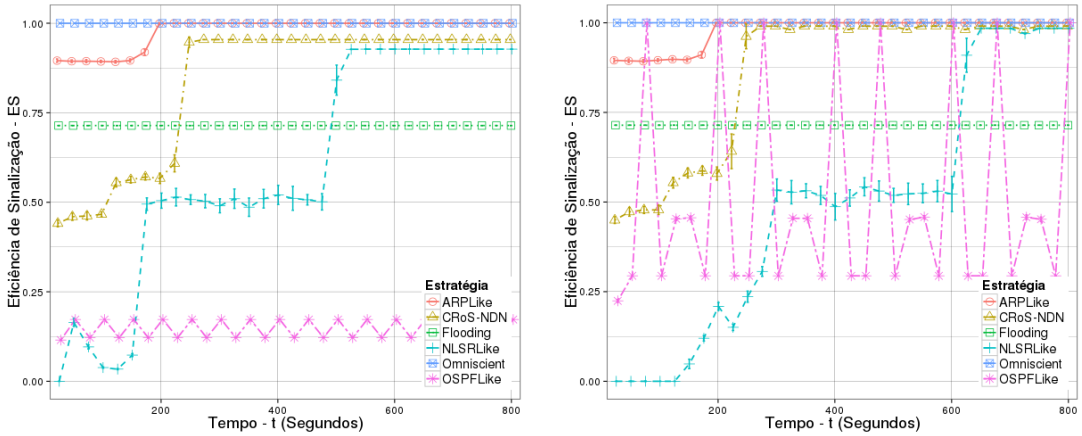
A Figura 6 compara o desempenho da $ES$ em função da topologia de rede. As estratégias *Flooding* e *OSPFLike* são as mais prejudicadas com o aumento do tamanho da rede. O aumento do diâmetro da rede aumenta mais sensivelmente o tempo de convergência da estratégia *NLSRLike*. Na fase de registro de prefixos, o desempenho $ES$ da estratégia *NLSRLike* é quase nulo, pois as novas LSAs são propagadas em um número maior de enlaces gerando maior tráfego de sinalização. Nesta simulação utiliza-se taxa de dez registros de prefixo por segundo, a menor taxa permite melhor visualização desta fase que agora dura cem segundos iniciais.

A Figura 7 compara o desempenho da $ES$ em função da taxa de monitoração de vizinhança $TV$. As estratégias *Ominiscient*, *Flooding* e *ARPLike* não são afetadas pela variação desta taxa. A estratégia *OSPFLike* é mais sensível a variação da $TV$, sendo que a diminuição desta taxa melhora a $ES$, porém aumenta o tempo de detecção de novos caminhos em caso de falha de enlace. A taxa $TV$ afeta ainda o atraso de convergência das estratégias *NLSRLike* e *CRoS-NDN*, sendo este efeito mais pronunciando na primeira.

As simulações demonstram que a proposta *CRoS-NDN* apresenta tempo de convergência inferior a *NLSRLike* e melhor eficiência $ES$ comparada a todas as demais estratégias em função do aumento do tamanho da rede, do número de prefixos e da diversidade de prefixos solicitados. Os resultados estão alinhados com a modelagem analítica.

## 5. Conclusão e Trabalhos Futuros

Este artigo propõe a estratégia de localização e encaminhamento em redes orientadas a conteúdo *CRoS-NDN*. Esta proposta utiliza separação dos planos de controle e de dados para rápida convergência e eficiência de sinalização. A *CRoS-NDN* utiliza inundação pontualmente apenas para localizar o controlador reduzindo a sobrecarga de sinalização. Os resultados analíticos e de simulação demonstram que as demais estra-




(a) Monitoração de vizinhança $TV = 0,2$. (b) Monitoração de vizinhança $TV = 0,05$.

**Figura 7. Aumento da eficiência e do atraso de convergência em função da redução da taxa de interesses de conectividade $TV$ de (a) para (b). Parâmetros: topologia de 7 nós da Figura 1(b), $TC = 20$, $NP = 1000$, $FI = 0,3$ e $TA = 10$.**

tégias baseadas em inundação de interesses apresentam rápida convergência no tempo, porém os procedimentos de inundações as tornam não escaláveis em termos de número de prefixos distintos de conteúdo e do número de nós e enlaces na rede. O efeito negativo da inundação de interesses é observado tanto em estratégias proativas que fazem anúncio de prefixos de conteúdo, quanto em estratégias reativas que inundam de forma recorrente a rede em busca de conteúdos específicos.

Além da quantidade de prefixos, verifica-se que a correlação de interesses distintos com igual prefixo de nome tem impacto significativo nas estratégias de inundação reativa da rede, sendo tanto pior o desempenho quanto menor for esta correlação. Este impacto é minimizado na *CRoS-NDN* que evita inundações fazendo consultas diretas ao controlador.

O trabalho também compara o desempenho com a *NLSRLike*, uma estratégia com anúncios de conteúdo de forma proativa e sem inundação recorrente. A eficiência de sinalização da *NLSRLike* é equivalente a *CRoS-NDN* em cenário sem novos anúncios ou alterações de topologia. Contudo, a *CRoS-NDN* tem convergência mais rápida e não exige que cada nó possua memória e processamento para armazenar e calcular localmente as rotas como ocorre na *NLSRLike*.

No conjunto de cenários analisados, a proposta *CRoS-NDN* apresentou um melhor compromisso entre as métricas de desempenho de atraso e de eficiência. Como trabalhos futuros pretende-se ampliar o conjunto de topologias avaliadas e de possibilidades de localização do elemento controlador.

## Referências

[Afanasyev et al. 2012] Afanasyev, A., Moiseenko, I., and Zhang, L. (2012). ndnSIM: NDN simulator for NS-3. Technical report, Named-Data Networking Project.

[Baid et al. 2012] Baid, A., Vu, T., and Raychaudhuri, D. (2012). Comparing Alternative Approaches for Networking of Named Objects in the Future Internet. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 298 –303.

[Chiocchetti et al. 2013] Chiocchetti, R., Perino, D., Carofiglio, G., Rossi, D., and Rossini, G. (2013). INFORM: A Dynamic Interest Forwarding Mechanism for Information Centric Networking. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pages 9–14, New York, NY, USA. ACM.

[Fernandes et al. 2011] Fernandes, N., Moreira, M., Moraes, I., Ferraz, L., Couto, R., Carvalho, H., Campista, M., Costa, L., and Duarte, O. (2011). Virtual Networks: Isolation, Performance, and Trends. *Annals of Telecommunications*, 66:339–355.

[Hoque et al. 2013] Hoque, A. K. M. M., Amin, S. O., Alyyan, A., Zhang, B., Zhang, L., and Wang, L. (2013). NLSR: Named-data Link State Routing Protocol. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pages 15–20, New York, NY, USA. ACM.

[Jacobson et al. 2009] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking Named Content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12. ACM.

[Mattos et al. 2011] Mattos, D., Fernandes, N., da Costa, V., Cardoso, L., Campista, M., Costa, L., and Duarte, O. (2011). OMNI: OpenFlow MaNagement Infrastructure. In *Network of the Future (NOF), 2011 International Conference on the*, pages 52 –56.

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., S., and Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

[Perino and Varvello 2011] Perino, D. and Varvello, M. (2011). A reality check for content centric networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ICN '11, pages 44–49, New York, NY, USA. ACM.

[Salsano et al. 2013] Salsano, S., Blefari-Melazzi, N., Detti, A., Morabito, G., and Veltri, L. (2013). Information Centric Networking over SDN and OpenFlow: Architectural Aspects and Experiments on the OFELIA Testbed. *Comput. Netw.*, 57(16):3207–3221.

[Torres et al. 2013] Torres, J., Ferraz, L., and Duarte, O. (2013). Redes Orientadas a Conteúdo Baseadas em Controladores Hierárquicos. *XXXI SBRC*, pages 717–730.

[Wang et al. 2012] Wang, L., Hoque, A., Yi, C., Alyyan, A., and Zhang, B. (2012). OSPFN: An OSPF Based Routing Protocol for Named Data Networking. Technical report, University of Memphis and University of Arizona.

[Xie et al. 2013] Xie, H., Wang, Y., and Wang, G. (2013). Scale content centric networks via reactive routing. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3530–3535.

[Yi et al. 2013] Yi, C., Abraham, J., Afanasyev, A., Wang, L., Zhang, B., and Zhang, L. (2013). On the Role of Routing in Named Data Networking. Technical report, Named-Data Networking Project.

[Zhu et al. 2013] Zhu, Z., Afanasyev, A., and Zhang, L. (2013). A New Perspective on Mobility Support. Technical report, Named-Data Networking Project.

# Uma análise do desempenho de redes sem-fio orientadas a conteúdo*

**Gabriel M. de Brito, Pedro B. Velloso e Igor M. Moraes**

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

{gbrito,velloso,igor}@ic.uff.br

***Abstract.*** *Information-Centric Networks (ICNs) are a promising solution to deal with nodes' mobility and low efficiency in wireless networks because they benefit from the native mobility support. This paper analyzes the performance of a wireless content-centric network implementation. The goal is to identify its main weaknesses and characterize an operating range based on these limitations. To this end, a simplified model of this network was implemented in the ndnSIM simulator. The efficiency of content delivery, delay and network load are shown through simulation from a content consumer perspective. The results show the benefits of content-centric wireless networks, which improves content-delivery efficiency and delay up to 63% and 40%, respectively, in certain scenarios, as well as highlight the impact of contention and collisions that define the broadcast storm problem.*

***Resumo.*** *As Redes Orientadas a Conteúdo (ROCs) são uma alternativa promissora para lidar com o problema de mobilidade e de baixa eficiência das redes sem-fio, pois um de seus principais benefícios é o suporte nativo à mobilidade dos nós. Este trabalho analisa o desempenho de uma implementação de rede sem-fio orientada a conteúdo. O objetivo é identificar suas principais deficiências e caracterizar uma faixa de operação a partir dessas limitações. Para tanto, foi implementado um modelo simplificado dessa rede no simulador ndnSIM. Através de simulações, verifica-se, do ponto de vista de consumidores de conteúdos, a eficiência da entrega de conteúdos, o atraso e a carga da rede. Os resultados mostram os benefícios das redes sem-fio orientadas a conteúdo, com melhora de 63% na eficiência e 40% no atraso de entrega dos conteúdos em determinados cenários, assim como evidenciam o impacto de contenções e colisões que definem o problema de* broadcast storm.

## 1. Introdução

O uso de dispositivos móveis tem apresentado significativo crescimento nos últimos anos, devido principalmente à disponibilidade de novas tecnologias, como *tablets* e *smartphones* e a redução do custo destes aparelhos. Uma importante característica deste novo cenário é a alta mobilidade dos nós. Neste contexto, as redes sem-fio, mais especificamente, as redes IEEE 802.11 representam a principal tecnologia de acesso à rede e

*Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ, FINEP e Proppi/UFF.

à Internet. No entanto, como já identificado anteriormente, a arquitetura tradicional da Internet baseada na pilha de protocolos TCP/IP não é adequada para este tipo de rede [Meisel et al. 2010]. O primeiro problema refere-se à mobilidade dos nós, pois as soluções tradicionais dependem de atribuição de endereços IP únicos para cada nó da rede e, assim, aumentam a complexidade do gerenciamento de endereços, sendo pouco eficientes para lidar adequadamente com a alta dinâmica topológica das redes móveis sem-fio. Além disso, o uso do protocolo TCP exige o estabelecimento de uma conexão fim-a-fim baseada nestes endereços, previamente atribuídos. Consequentemente, interrupções de tráfego são percebidas pelos nós como resultado de reposicionamentos em função da movimentação. Outro problema importante está relacionado à necessidade de recuperação de conteúdo apenas dos servidores de origem, baseada no esquema cliente-servidor. Este problema pode causar baixa eficiência e aumento do atraso da comunicação devido a longos caminhos até o servidor de origem, além de aumentar a probabilidade de perda da conexão em consequência da mobilidade dos nós. Assim, diversos trabalhos tentam resolver um destes problemas [Johnson et al. 2004, Moskowitz et al. 2008], mas sem muito êxito no tratamento de ambos ao mesmo tempo.

O paradigma de Redes Orientadas a Conteúdo (ROCs) [de Brito et al. 2012] é uma alternativa para lidar com o problema de mobilidade e de baixa eficiência da rede, pois um de seus principais benefícios é o suporte nativo à mobilidade dos nós [Tyson et al. 2012]. Nas ROCs o foco é a entrega de conteúdo para os usuários, independentemente da localização de tal conteúdo ou destes usuários. Esta abordagem está em consonância com o novo perfil da maioria das aplicações atuais da Internet, as quais são orientadas à entrega do conteúdo, como o Netflix, YouTube, PPStream e BitTorrent. Mais da metade do tráfego da Internet atual pertence a esses aplicativos de distribuição de conteúdo [Sandvine 2011]. Além disso, a infraestrutura de rede contribui ativamente para o armazenamento e distribuição de conteúdo, criando-se assim uma rede de *caches* universal, na qual todos os nós da rede possuem um *cache*. Estas duas características diminuem os problemas relacionados a mobilidade dos nós, visto que os consumidores e publicadores de conteúdo podem mudar suas posições físicas na topologia da rede sem qualquer mudança nos identificadores existentes nem qualquer prejuízo à disponibilidade dos conteúdos [Tyson et al. 2012]. Ademais, é possível reduzir o atraso de entrega do conteúdo em razão da possibilidade de recuperá-lo no *cache* do nó mais próximo. Por último, as redes sem-fio se beneficiam da característica *broadcast* do canal. Esta característica potencializa a eficiência das ROCs, pois provê o suporte nativo à utilização de múltiplos caminhos de comunicação e permite aumentar a disponibilidade de conteúdos através de *cache* de conteúdo oportunista. Alguns trabalhos propõem o uso de ROCs em redes sem-fio [Meisel et al. 2010, Wang et al. 2012], no entanto, nenhum deles realiza uma análise detalhada do desempenho da rede nestas condições, de maneira a avaliar os parâmetros que influenciam na eficiência da rede, bem como identificar os principais problemas desta abordagem.

Este trabalho analisa o desempenho de uma implementação de ROC sem-fio, em particular da arquitetura *Content Centric Networking* (CCN) [Jacobson et al. 2009]. Atualmente, a CCN é um padrão *de facto* no desenvolvimento de pesquisa em ROCs. O objetivo é avaliar o desempenho da CCN sem-fio a fim de identificar seus principais problemas e caracterizar uma faixa de operação a partir de suas limitações. Portanto, pretende-se avaliar, do ponto de vista de consumidores de conteúdos, a eficiência da entrega de conteúdos, o atraso e a carga da rede. Na análise são considerados diversos parâmetros, como mo-

bilidade, carga na rede, densidade da rede, dentre outros. Para tal, foi desenvolvido um modelo simplificado no simulador *ndnSIM* [1], o qual foi utilizado em cenários diversos de simulação. Além disso, também é avaliada o impacto do uso de *cache* oportunístico. Os resultados demonstram os benefícios da CCN sem-fio, com melhora de 63% na eficiência de entrega e 40% no atraso em determinados cenários, assim como evidenciam o impacto de contenções e colisões que definem o problema de *broadcast storm*.

O restante deste artigo está organizado da seguinte forma. Alguns aspectos básicos das CCN e os desafios de mobilidade são discutidos na Seção 2. A Seção 3 apresenta as modificações necessárias à camada de estratégia da CCN, estendendo-a para suportar comunicações sem-fio. O modelo de simulação e os parâmetros utilizados para executar as simulações são apresentados na Seção 4, cujos resultados são apresentados na Seção 5. A Seção 6 apresenta trabalhos relacionados às ROCs e mobilidade. A Seção 7 conclui o trabalho e mostra possíveis direções futuras para sua continuidade.

## 2. A arquitetura CCN (*Content-Centric Networking*)

A arquitetura CCN emprega basicamente dois tipos de pacotes: interesses e dados. Pacotes de interesse são enviados por nós consumidores que desejam receber um conteúdo. Interesses transportam o nome do conteúdo solicitado, um identificador hierárquico estruturado como um *Uniform Resource Identifier* (URI) [Mealling e Denenberg 2002], como `/uff.br/video/intro.avi`. Esses pacotes de interesse são enviados através de todas as interfaces disponíveis para todos os nós vizinhos, sendo reencaminhados pelos últimos, caso estes não possam responder diretamente ao pedido. Pacotes de dados, por sua vez, são enviados em resposta aos interesses por nós produtores ou nós intermediários que podem, eventualmente, ter uma cópia do conteúdo desejado armazenado em seu *cache*. A CCN divide os conteúdos em pedaços, chamados *chunks*, estruturas que são identificadas por nomes hierárquicos únicos. Os *chunks* CCN são bastante semelhantes aos *chunks* de redes *peer-to-peer* (P2P), como os empregados pela aplicação BitTorrent.

Os roteadores CCN usam três estruturas de armazenamento para habilitar o roteamento e encaminhamento de *chunks* para os nós consumidores: a *Content Store* (CS), a *Pending Interest Table* (PIT) e a *Forwarding Information Base* (FIB). A CS é a estrutura de armazenamento de *chunks* dos roteadores, os quais são indexados por nome e atualizados de acordo com políticas de atualização de *cache* como *Least Frequently Used* (LFU) ou *Least Recently Used* (LRU) [Podlipnig e Böszörmenyi 2003]. A PIT armazena informações relativas aos pacotes de interesse já recebidos e retransmitidos por um nó, bem como as interfaces de entrada e saída desses pacotes. A PIT também armazena informações a respeito da validade de suas entradas, as quais são descartadas após expirarem. A FIB é a estrutura usada para armazenar informações de encaminhamento de interesses, realizando o mapeamento de prefixos de nomes de conteúdo em uma ou mais interfaces de saída. As estruturas de roteamento CCN são baseadas em interfaces. No caso de interesses, as interfaces de entrada são mapeadas na PIT e as interfaces de saída, na FIB. No caso de dados, interfaces de entrada podem atualizar entradas FIB, enquanto a PIT é verificada para mapeamento de interfaces de saída. Este modelo básico de CCN permite a utilização de múltiplos caminhos de forma nativa, uma vez que a remoção da

[1] http://www.ndnsim.net

entrada PIT quando os dados são transmitidos impede a formação de *loops* de encaminhamento.

### 2.1. Mobilidade

A CCN provê suporte nativo à mobilidade já que não existe uma associação entre a identificação e a localização dos conteúdos. Assim, a CCN permite que os consumidores e provedores de conteúdo mudem suas posições físicas na topologia da rede sem mudança nos identificadores existentes nem prejuízo à disponibilidade dos conteúdos [Tyson et al. 2012]. Existem, assim, vantagens relacionadas à utilização da CCN na implementação de ROCs sem-fio, tanto em relação à mobilidade física dos nós quanto em relação à mobilidade dos conteúdos, espalhando cópias pelos *caches* da rede. Os nós não precisam atualizar seus endereços quando há reposicionamento já que nenhuma informação topológica é necessária para o envio de pacotes. Além disso, os nomes dos conteúdos podem ser preservados, pois a movimentação de *caches* não implica alteração do conteúdo e, portanto, garante a preservação dos identificadores.

Os benefícios da CCN sem-fio, no entanto, introduzem alguns desafios. Como a comunicação acontece sem qualquer referência à localização dos nós, a CCN explora a natureza *broadcast* do canal sem-fio. Como os pacotes são transmitidos sem qualquer referência de destinatários, qualquer nó pode armazenar dados não solicitados que por ventura receber, aumentando a disponibilidade dos conteúdos. De forma análoga, qualquer nó pode responder um pedido, desde que possua uma cópia válida do conteúdo solicitado. Todavia, existe a possibilidade de que certas áreas da rede estejam no raio de cobertura de múltiplos nós, o que pode levar a retransmissões redundantes naquela região. Ainda, pode haver aumento da contenção do meio caso tais nós estejam próximos. Como não endereçam os pacotes transmitidos a nós específicos, as ROCs não podem valer-se dos mecanismos de *Ready-to-Send/Clear-to-Send* (RTS/CTS), sendo vulneráveis ao problema do terminal escondido. A consolidação destes efeitos é o que define o problema de *broadcast storm* [Ni et al. 1999]. Além disso, conteúdos dinâmicos e alta mobilidade dos nós podem invalidar as entradas FIB e tornar obsoletas as cópias de conteúdo armazenadas localmente. Uma grande quantidade de tráfego de controle seria necessária para a manutenção da rede e sincronização das informações de estado, o que se prova impraticável em grandes topologias [Varvello et al. 2011].

## 3. Camada de Estratégia sem-fio

Os nós sem-fio têm, geralmente, uma única interface de rede já que há apenas um transmissor disponível para explorar o canal de rádio predeterminado. Havendo um único transmissor sem-fio por nó, não há como os nós transmitirem pacotes por múltiplas interfaces e, portanto, a arquitetura básica de CCN deve ser estendida. Conforme apresentado na Seção 2, sempre que um nó recebe um pacote é realizada uma série de verificações na FIB, no caso de interesses, ou na PIT, no caso de dados, para determinar se um pacote precisa ser retransmitido. Uma vez que os nós têm apenas uma interface disponível e são incapazes de determinar rotas diretas para nós além de um salto de distância na topologia, são obrigados a confiar no encaminhamento de interesses e dados baseado em inundação da rede, sujeito aos problemas de *broadcast storm*. A característica *broadcast* do canal sem-fio também pode ser explorada pelos nós através da detecção promíscua de tráfego no canal e, assim, possibilitar a atualização de seus *caches* com conteúdos recebidos de

forma proativa. Este mecanismo de distribuição de conteúdo pode melhorar a percepção de qualidade, diminuir os atrasos de entrega de conteúdo e reduzir a carga da rede.

Tanto as modificações necessárias para permitir o recebimento e retransmissão de interesses por um única interface de rede quanto o armazenamento proativo de *chunks* foram implementados em uma nova camada de estratégia para *ns-3*[2], mais especificamente, para o módulo *ndnSIM*. O *ndnSIM* implementa toda a arquitetura básica da CCN através de modelos funcionais de interfaces de aplicação e de rede, estruturas de encaminhamento como FIB, PIT, CS e a própria Camada de Estratégia, bem como todo o núcleo básico do protocolo CCN baseado em encaminhamento de pacotes de interesse e de dados. Para suportar a comunicação sem-fio, eliminou-se a restrição que impedia o encaminhamento de interesses e dados pela mesma interface de recebimento. Para habilitar o armazenamento proativo de *chunks*, eliminou-se da verificação de entradas PIT válidas para permitir a criação de entradas na CS. Assim, qualquer conteúdo recebido por um nó pode ser armazenado na CS.

## 4. Simulação

As seguintes métricas de desempenho são utilizadas para avaliar os resultados obtidos com a implementação da CCN sem-fio:

- **Eficiência de entrega média:** razão entre a quantidade total de *chunks* efetivamente entregues aos consumidores e a quantidade total de interesses enviados pelos mesmos.
- **Número médio de tentativas:** número médio de interesses necessários para a obtenção de um *chunk* com sucesso, incluindo todas as retransmissões de um determinado interesse, para cada consumidor.
- **Atraso médio:** atraso fim-a-fim médio, em segundos, medido a partir do primeiro interesse enviado até o sucesso da entrega do *chunk* requisitado.
- **Carga média da rede:** média do total de pacotes transmitidos na rede.

Os cenários propostos para avaliação do modelo possuem diferentes características de mobilidade e conectividade. Para cada um dos cenários foram executadas 10 rodadas de simulação, cada uma com 300 s de duração. Calcula-se os intervalos de confiança, representados por barras verticais, para um nível de confiabilidade de 95%. Os padrões de mobilidade foram gerados e analisados com a ferramenta *BonnMotion*[3]. Todos os cenários são compostos por 60 nós, aleatoriamente distribuídos em uma área quadrada de 40.000 $m^2$. Os nós se movem segundo o modelo *Random Walk*, no qual nós escolhem direção e velocidade aleatórios [Camp et al. 2002]. Nas simulações realizadas, são escolhidas direções e velocidades dentro dos limites de [0, $2\pi$] rad e [5, 15] m/s, respectivamente, e os nós se deslocam por 100 m na direção e velocidade escolhidos. Caso atinjam as bordas da área de simulação, os nós ricocheteiam com um ângulo determinado pela direção de chegada. Ao atingir os novos destinos, os nós fazem nova escolha e seguem assim, sucessivamente, por todo o tempo de simulação. Tais parâmetros de mobilidade foram escolhidos visando criar um modelo para simular a movimentação de usuários humanos, caminhando por uma área urbana de convivência coletiva, como parques ou *shopping centers*.

[2]http://www.nsnam.org

[3]http://sys.cs.uos.de/bonnmotion

**Tabela 1. Parâmetros de mobilidade e conectividade.**

| Potência Tx | Alcance Médio | Grau Médio | Disponibilidade |
|---|---|---|---|
| 20 dBm | 200 m | 57,49 | 1,0 |
| 1,65 dBm | 50 m | 9,20 | 0,99 |
| -3 dBm | 35 m | 4,84 | 0,76 |
| -5 dBm | 30 m | 3,64 | 0,44 |
| -20 dBm | 10 m | 0,44 | 0,01 |

Cada nó está equipado com uma interface IEEE 802.11b para permitir a comunicação sem-fio. Utiliza-se o modelo de propagação *log-distance*, adequado para predições de perda de propagação em ambientes urbanos densos, cujos parâmetros foram ajustados para a faixa de 2,4 GHz [Rappaport 1996]. A conectividade média desta rede é alterada através da variação da potência de transmissão dos nós, com valor máximo de 20 dBm, o máximo permitido pelo padrão 802.11b, e valor mínimo de -20 dBm. Um dos efeitos mais interessantes da variação da potência de transmissão reside nas alterações percebidas na disponibilidade de caminhos entre dois nós arbitrários *i* e *j*. A disponibilidade de caminhos indica a fração de tempo na qual existem caminhos válidos entre dois nós [Bai et al. 2003]. Os valores médios de alcance, grau, ou seja, o número de vizinhos de um nó, e disponibilidade de caminhos, para cada um dos níveis de potência utilizados, encontram-se apontados na Tabela 1.

Também são utilizadas diferentes quantidades de nós consumidores e produtores de conteúdo, notadamente 1, 2, 4 e 8 nós (1,67%, 3,33%, 6,67% e 13,33% dos nós da rede, respectivamente). Em todos os cenários, nós consumidores requisitam conteúdos à rede através do envio de pacotes de interesse, carregando identificadores de *chunks* específicos. A identificação dos *chunks* se dá por um número sequencial, inserido após o prefixo agregável, do tipo `/prefixo/<chunk seqnum>`, cujo valor é escolhido dentre um catálogo de 300 *chunks* possíveis. Os consumidores escolhem os *chunks* a serem requisitados obedecendo a uma distribuição de Mandelbrot-Zipf (MZipf) [Silagadze 1999] com parâmetro $\alpha$ = 0,7. Busca-se replicar o parâmetro $\alpha$ encontrado em *traces* reais [Breslau et al. 1999] para definir a característica de popularidade do catálogo de *chunks*. O parâmetro $q$ da distribuição também é configurado com 0,7 já que se deseja utilizar uma distribuição do tipo *Zipf-like*, na qual $q$ deve ser maior do que 0. Nos cenários simulados, todos os *chunks* possuem 1024 Kbytes. Todos os nós possuem uma CS funcional, com capacidade para até 100 dos 300 *chunks* possíveis e aplicando a política de atualização de *caches* LRU. Os nós consumidores são os únicos que não possuem CS ativa, ou seja, não armazenam conteúdos. Tal configuração é necessária a fim de impedir que interesses sejam atendidos com cópias armazenadas na CS dos próprios consumidores.

Além da frequência de solicitação de cada um dos *chunks* individuais, varia-se também a taxa de envio de interesses. Os valores de 1, 5, 15 e 50 pacotes/s são utilizados para verificar o impacto sobre a banda total disponibilizada pela rede. Assim como as taxas de requisição de conteúdos, que variaram desde a menor taxa possível, de 1 pacote/s, até as taxas mais altas, como 50 pacotes/s, busca-se variar os parâmetros de simulação em uma larga faixa a fim de possibilitar a caracterização da faixa de operação da rede.

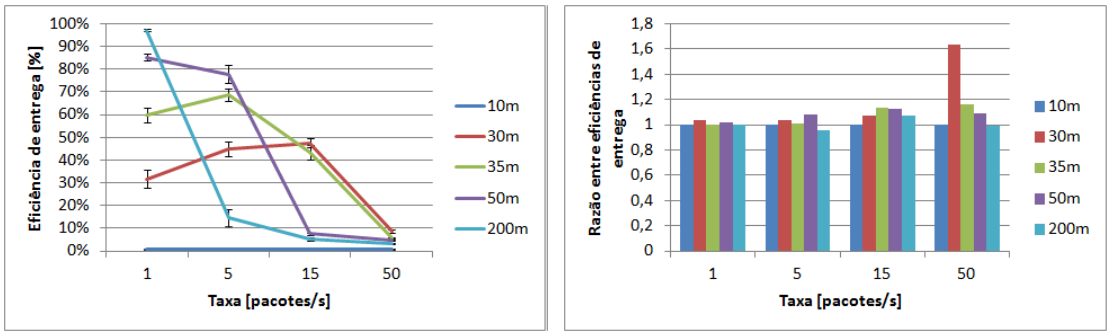
## 5. Resultados

O objetivo da análise é verificar o desempenho da CCN sem-fio em relação ao uso de *cache* oportunista, ao nível de mobilidade e densidade da rede e às diferentes proporções de nós consumidores e produtores. Com estas métricas pretende-se caracterizar a faixa de operação da CCN sem-fio.

### 5.1. Eficiência do *cache* oportunista

Primeiramente, são simulados cenários para comparação do modelo original de CS, com um mecanismo de *cache* que armazena apenas conteúdos solicitados, chamado de regular, e o modelo modificado de CS, o qual propicia o armazenamento de conteúdos recebidos e não solicitados, dito oportunista. Nestes cenários, apenas um nó consumidor e um nó produtor são utilizados a fim de caracterizar os efeitos da difusão de interesses e de conteúdos a partir destes dois nós. Todos os nós movem-se segundo o modelo apresentado na Seção 4. A Figura 1(a) mostra os valores de eficiência de entrega para o mecanismo de *cache* regular. Somente com conectividade plena, com alcance médio igual a 200 m, e com a taxa de emissão de interesses mais baixa é que a rede foi capaz de atingir eficiência de entrega maior que 90%. Nesta situação há apenas um consumidor requisitando interesses em baixa taxa, beneficiando-se da baixa probabilidade de contenção do meio e colisão de pacotes. Para taxas mais altas, como a taxa de 50 pacotes/s, a degradação do desempenho gerada pela retransmissão de pacotes é tão severa que nem mesmo para topologias menos densas a rede se mostra eficiente. Nessa condição, a eficiência é menor do que 10% em todos os casos. Nestes cenários, os pacotes retransmitidos por nós intermediários, na tentativa de entregar o *chunk* solicitado ao consumidor, acabam por inundar a rede por completo e a competir entre si pelos recursos de rede. A contenção do meio fica evidente quando se verifica a Figura 2(a). Pode-se perceber o aumento do número médio de tentativas para a obtenção do *chunk* com sucesso relativo ao aumento da taxa de envio de interesses em todas as densidades de rede. É interessante observar a transição das séries de 30 m e 35 m para as taxas de 5 pacotes/s e 15 pacotes/s, na qual há um abrupto aumento no número médio de tentativas. Esse aumento de densidade também evidencia o aumento da contenção e, como consequência, aumento da colisão de pacotes, representado pela redução da eficiência de entrega de conteúdos e aumento do número médio de tentativas em cenários densos e com taxas maiores de envio de interesses. Esse é um dos sintomas mais evidentes do *broadcast storm* [Ni et al. 1999].

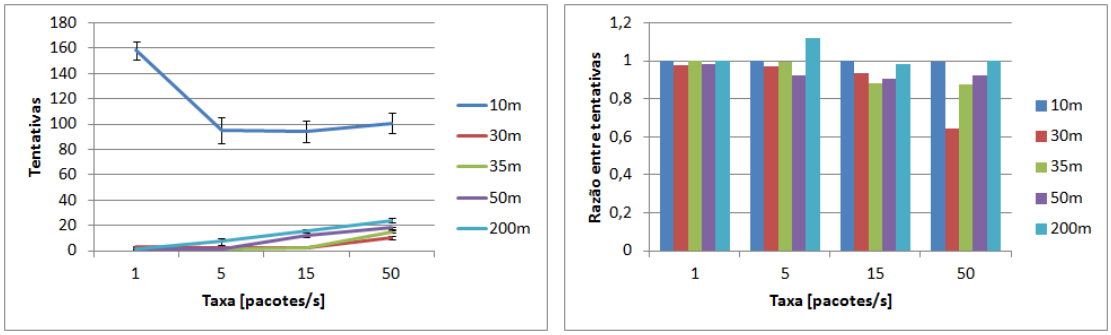
As variações da métrica eficiência e do número de tentativas, com o uso de *cache* oportunista, são representadas nas Figuras 1(b) e 2(b). Pode-se perceber variações positivas, nas quais mais *chunks* foram entregues e variações negativas, nas quais quantidades menores de *chunks* foram entregues. Redes pouco densas, como as obtidas nos cenários de baixa potência de transmissão, não apresentam qualquer variação de desempenho com a utilização do *cache* oportunista devido à baixa disponibilidade de caminhos. Em topologias mais densas o efeito do mecanismo oportunista se torna mais evidente, com desempenho até 63% superior ao mecanismo regular. A eficiência na entrega dos *chunks* aumenta principalmente nos cenários de conectividade intermediária, com alcances de 30 m, 35 m e 50 m. Todavia, quanto mais densa a topologia, mais sensível esta se torna ao aumento da taxa de requisição de conteúdos. Taxas mais altas aumentam a contenção pelo meio e a probabilidade de colisão dos pacotes difundidos. Ao contrário dos demais cenários, a eficiência de entrega com conectividade plena (200 m) é degradada em quase




(a) *Cache* regular.

(b) Razão entre eficiências de entrega média do *cache* oportunista e regular.

**Figura 1. Eficiência de entrega por taxa de requisição de *chunks*.**



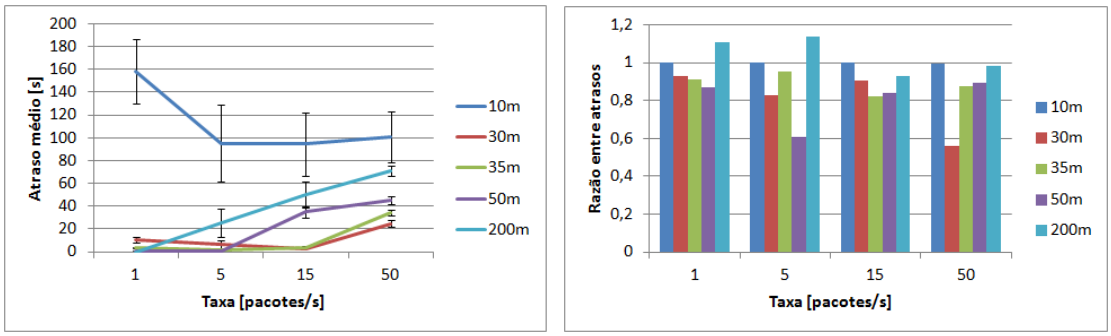

(a) *Cache* regular.

(b) Razão entre número médio de tentativas do *cache* oportunista e regular.

**Figura 2. Número médio de tentativas por taxa de requisição de *chunks*.**

todos os cenários com o uso do *cache* oportunista já que, ao longo do tempo, todos os nós possuem cópias válidas dos *chunks* mais populares, tornando-se elegíveis a responder interesses. Aumenta-se, assim, a probabilidade de contenção e colisão de pacotes. Para taxas mais baixas, a proporção de interesses que são atendidos com o envio de *chunks* provenientes de *caches* intermediários é consideravelmente menor e, por isso, o efeito do *cache* oportunista é menos evidente.

A Figura 3(a) mostra os valores da métrica atraso para o mecanismo de *cache* regular. Em cenários de baixa densidade de nós há muitas áreas desconectadas da rede e tempo curto de existência de enlaces entre os nós. Estes cenários apresentam baixa probabilidade de recepção dos pacotes. Já nos cenários com maior conectividade e disponibilidade de caminhos, os resultados mostram que o atraso também é afetado pelo aumento das taxas de requisição de conteúdos. O intervalo entre pacotes é bastante reduzido, substancialmente menor que o atraso fim a fim nos cenários de 15 e 50 pacotes/s. Nestes cenários novos interesses são enviados antes mesmo de o primeiro interesse ter sido respondido. Esse fato aumenta ainda mais a contenção do meio, pois as diversas requisições de um mesmo consumidor competem pelos recursos de rede, degradando ainda mais o desempenho da rede. Ocorre algo semelhante a uma contenção intrafluxo, porém aplicada a todos os interesses gerados pelo consumidor.



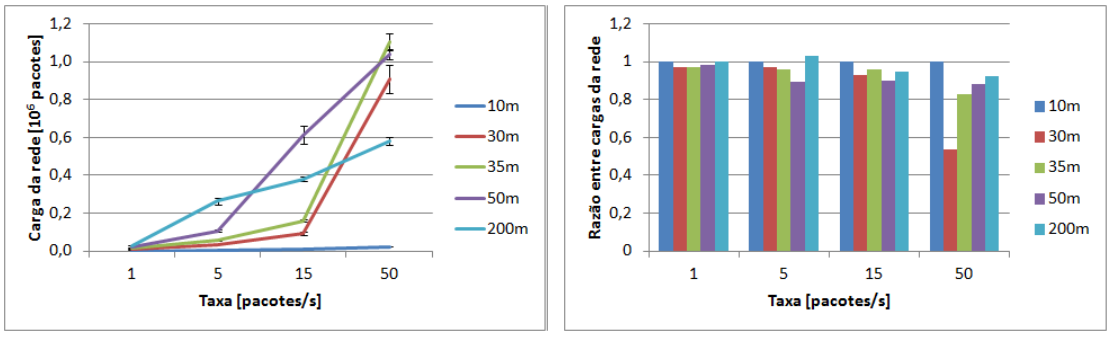

(a) *Cache* regular.

(b) Razão entre atrasos médios do *cache* oportunista e regular.

**Figura 3. Atraso médio por taxa de requisição de *chunks*.**

A Figura 3(b) apresenta as variações da métrica com o uso do *cache* oportunista. De forma análoga à eficiência de entrega de *chunks*, o atraso médio para responder um interesse também é afetado pelo uso do *cache* oportunista. Reduções da ordem de 40% foram obtidas com o mecanismo oportunista. O atraso médio é, também, severamente afetado pelo aumento da probabilidade de colisões. Em cenários de alta conectividade, a probabilidade de colisão de pacotes é maior já que há sobreposição das áreas de coberturas de um número maior de nós. Assim, o uso de taxas mais baixas de envio de interesses frente a maior probabilidade de colisão resulta em baixa entrega efetiva de *chunks*, aumentando o tempo médio para a obtenção com sucesso. O uso de taxas maiores faz com que ao menos parte dos *chunks* solicitados sejam entregues, melhorando tanto a métrica de eficiência quanto a métrica de atraso, uma vez que somente *chunks* efetivamente recebidos são considerados na métrica de atraso. De uma forma geral o uso do *cache* oportunista reduziu o atraso para obtenção de *chunks*.

A carga média da rede, avaliada pela quantidade total de pacotes transmitidos na rede, encontra-se representada na Figura 4(a). A observação dos resultados demonstra como a rede é saturada com o aumento da taxa de emissão de interesses. A carga total da rede cresce com o aumento da taxa de emissão de interesses, caracterizando a saturação da rede e aumento considerável da contenção e colisões.




(a) *Cache* regular.

(b) Razão entre cargas médias da rede do *cache* oportunista e regular.

**Figura 4. Carga média da rede por taxa de requisição de *chunks*.**

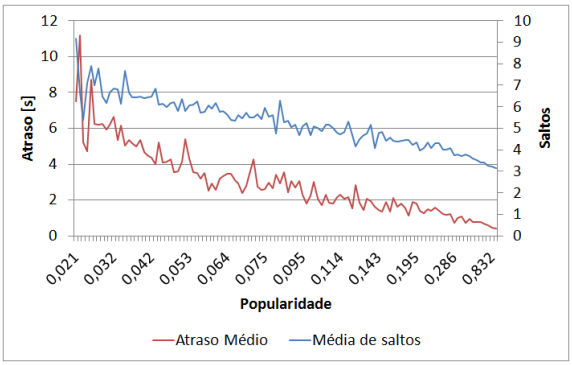



**Figura 5. Atraso e número de saltos médios por popularidade dos *chunks*. Cenário de alcance 30 m com taxa de emissão de interesses de 15 pacotes/s.**

As variações da métrica carga de rede com o uso do *cache* oportunista encontram-se representadas na Figura 4(b). De uma maneira geral, o uso do *cache* oportunista implica redução na carga da rede, sendo seu efeito mais pronunciado nas topologias de média conectividade e alta taxa de emissão de interesses. Como nessas topologias a probabilidade de colisões é menor e a taxa de requisição de *chunks* é maior, há uma dispersão mais efetiva de tais *chunks* pela rede. A dispersão mais efetiva associada ao *cache* oportunista resulta no armazenamento de tais *chunks* em mais nós da rede. Assim, os interesses emitidos para a obtenção de conteúdos mais populares têm grande probabilidade de serem atendidos por cópias do *chunk* armazenadas em nós mais próximos dos consumidores do que o nó produtor original, interrompendo a retransmissão de interesses em ramos da árvore de *broadcast*. Essa constatação é evidenciada na Figura 5. Pode-se perceber que para os *chunks* mais populares, isto é, com popularidade mais próxima de 1, o número médio de saltos entre a cópia armazenada e o nó consumidor é menor, o que se desdobra em um atraso também menor.

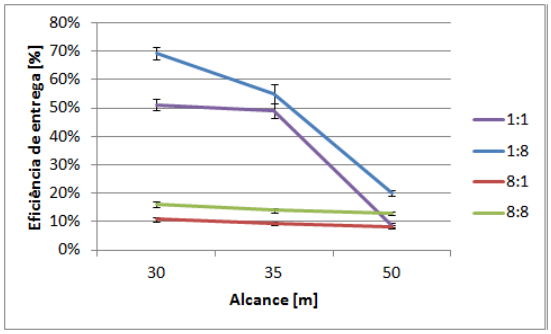
A redução da eficiência de entrega de conteúdos e o aumento do número médio de tentativas em cenários densos e com taxas maiores de envio de interesses são os sintomas mais evidentes do *broadcast storm* [Ni et al. 1999]. Em todos os cenários avaliados, como apontado no início desta seção, há apenas 1 nó consumidor e 1 nó produtor. Muito embora os *chunks* possam ser solicitados múltiplas vezes, tanto por retransmissão de interesses devido à colisão da requisição original quanto por nova requisição devido à popularidade do mesmo, um único nó consumidor não se beneficia plenamente do *cache* oportunista. Dessa forma, a utilização de um número maior de nós consumidores e produtores também deve ser avaliada, pois impacta diretamente o desempenho destes *caches*.

## 5.2. Impacto da mobilidade e da proporção de nós consumidores e produtores

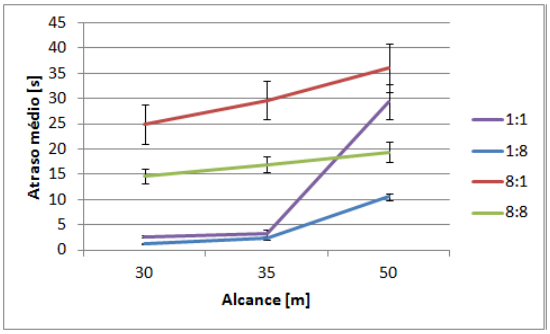
Para atestar o impacto da mobilidade dos nós nas métricas de desempenho, utilizou-se cenários com diferentes densidades de rede. A quantidade de nós consumidores e nós produtores também é variada. São utilizados cenários com 1, 2, 4 e 8 nós consumidores, bem como 1, 2, 4 e 8 nós produtores. Foram também simulados cenários com 8 nós consumidores e 8 nós produtores, simultaneamente. Por simplicidade, todos os produtores possuem os mesmos conteúdos. São quatro os cenários mais representativos do impacto da variação da proporção dos nós: 1 consumidor e 1 produtor (*1:1*), 1 consumidor e 8 produtores (*1:8*), 8 consumidores e apenas 1 produtor (*8:1*) e o cenário com 8

consumidores e 8 produtores (*8:8*).

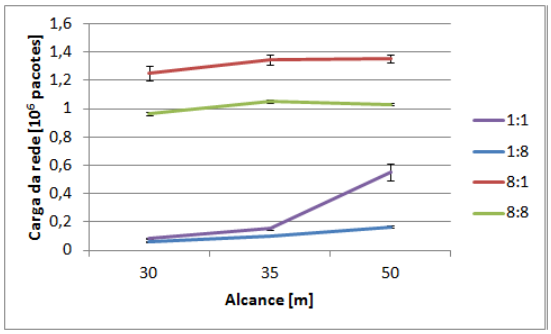
São mostrados, na Figura 6, as métricas relativas à taxa de emissão de interesses de 15 pacotes/s, expurgados os resultados com alcance de 200 m, rede plenamente conectada, e com alcance de 10m, com conectividade muito baixa. Percebe-se uma variação abrupta em todas as métricas com alcances maiores que 35 m nos cenários com apenas 1 consumidor. A Tabela 1 mostra que, para este nível de alcance ou valores maiores, há crescimento exponencial do grau médio dos nós devido o aumento das áreas com coberturas redudantes de múltiplos nós. A redução na carga da rede pode levar a um falso diagnóstico de melhoria do desempenho.




(a) Eficiência de entrega para taxa de emissão de interesses de 15 pacotes/s.




(b) Atraso médio para taxa de emissão de interesses de 15 pacotes/s.




(c) Carga da rede para taxa de emissão de interesses de 15 pacotes/s.

**Figura 6. Resultados para diferentes densidades de rede.**

A partir dos resultados pode-se inferir que a redundância de transmissões, fruto da cobertura de áreas comuns por múltiplos nós, é bastante prejudicial ao desempenho das redes simuladas. Há aumento da contenção e da probabilidade de colisões, evidenciada pelo crescimento do atraso e da redução da eficiência de entrega da rede. Já com alcance de 30 m, com disponbilidade de caminhos inferior, há um desempenho consideravelmente superior nos cenários com apenas 1 consumidor. Este e mais um indicador que a inundação da rede pode ser bastante prejudicial e que mecanismos de controle da difusão dos interesses são necessários para melhorar o desempenho das ROCs sem-fio.

Ao compararmos os cenários *1:1* e *1:8* percebe-se aumento da eficiência de entrega, além da redução do atraso e da carga da rede, com o aumento do número de produtores. Uma vez que nas simulações os produtores possuem cópias de todos os conteúdos de um determinado prefixo, diferentemente dos *caches* intermediários que somente ar-

mazenam os conteúdos recebidos, há o aumento da probabilidade de atendimento às requisições por nós mais próximos dos consumidores. Com isso pode-se perceber a melhoria geral das métricas de desempenho. A adoção de grande número de consumidores leva a uma grande degradação do desempenho, sendo esta mais drástica nos cenários com densidades de nós maiores. O aumento da quantidade de fontes de interesses aumenta, de forma geral, a probabilidade de colisão de pacotes da rede. Com múltiplos consumidores requisitando conteúdos, há aumento da concorrência de recursos de rede para a difusão dos interesses. Nem mesmo o aumento da quantidade de produtores é capaz de melhorar o desempenho frente a inundação da rede com interesses diversos gerados pelo re-encaminhamento por nós intermediários. Assim, percebe-se que o aumento do número de consumidores é o principal ofensor em termos de contenção do meio, contribuindo diretamente para a piora do desempenho da rede devido ao *broadcast storm*.

## 6. Trabalhos relacionados

Diversos trabalhos apresentam propostas para a implementação de ROCs sem-fio, mas as avaliações apresentadas não são extensas. Meisel *et al.*, por exemplo, propõem uma implementaçao de rede *ad hoc* móvel sem-fio baseada na arquitetura CCN [Meisel et al. 2010]. O trabalho apresenta o protocolo *Listen First, Broadcast Later* (LFBL), no qual nós armazenam métricas de atraso para os prefixos conhecidos, priorizando as retransmissões por nós cujas métricas de atraso indiquem maior proximidade ao prefixo. Para obtenção das métricas os nós monitoram o canal de forma promíscua e utilizam informações dos pacotes trafegados, os quais transportam os valores das métricas. A cada salto as métricas transportadas são comparadas com as existentes no nó para verificar se o pacote está se distanciando ou se aproximando do destino. LFBL também explora o canal sem-fio através do *cache* oportunista de conteúdos. Muito embora não haja sincronização de informações topológicas, LFBL depende do armazenamento e gerenciamento de métricas para todos os prefixos, o que pode tornar-se inviável na existência de muitas entradas não-agregáveis.

Wang *et al.* adaptam a arquitetura básica da CCN para a adoção em redes veiculares [Wang et al. 2012]. Muito embora utilize primitivas orientadas a conteúdo, o modelo proposto pelos autores é fortemente dependente da aplicação já que utiliza a semântica dos nomes para limitar o alcance da inundação da rede, não sendo aplicável a aplicações genéricas. Nessa proposta, os nomes dos conteúdos possuem informações de geolocalização, o que possibilita a inundação geográfica [Ni et al. 1999]. São utilizados temporizadores aleatórios para a redução de colisões, priorização baseada em geolocalização e contadores de pacotes para o cancelamento de retransmissões agendadas, medidas que reduzem consideravelmente as colisões de pacotes. Adicionalmente, o modelo não é testado em um cenário verdadeiramente móvel. O trabalho avalia o desempenho do modelo proposto em um cenário no qual não há movimento relativo entre os nós.

Angius *et al.* assumem que nós da CCN sem-fio possuem identificadores binários e armazenam as informações de vizinhança em filtros de *Bloom* [Angius et al. 2012]. Tais filtros são difundidos entre vizinhos, de forma que todos são capazes de atestar as coberturas mútuas. Toda vez que recebe um pacote para retransmissão, os nós checam se há algum vizinho capaz de atingir mais nós da rede. Caso isso seja verdade, o nó cancela a retransmissão e aguarda que o vizinho com cobertura mais adequada faça o

encaminhamento. Este mecanismo, apesar de armazenar o estado mínimo em estruturas de dados eficientes, necessita a constante troca de informações entre nós, o que pode aumentar o *overhead* da rede em cenários com grande dinamismo topológico.

Conceitos básicos de *design* de redes móveis *ad hoc*, com base no paradigma ICN, são investigados por Varvello *et al.* [Varvello et al. 2011]. O trabalho propõe modelos analíticos para três estratégias básicas de encaminhamento: inundação reativa, indundação proativa e as tabelas *hash* geográficas (GHT). Uma de suas principais conclusões é o desempenho competitivo das inundações em relação ao mecanismo mais complexo das GHTs, especialmente em redes de baixa densidade, de até 300 nós. Mecanismos baseados em inundação provém maior disponibilidade e menor atraso, mas levam a uma maior utilização da rede. O custo dos mecanismos de encaminhamento baseados em inundação está estritamente correlacionado com o número de nós, aumentando em topologias mais densas.

## 7. Conclusões

As ROCs possuem suporte nativo à mobilidade característica das redes sem-fio. Todavia, a entrega de conteúdo é consideravelmente impactada pelas características da mobilidade dos nós. Neste trabalho avaliou-se o desempenho de ROCs sem-fio através de simulação de um modelo simplificado da arquitetura CCN sem-fio. Os resultados corroboram a afirmação de suporte nativo à mobilidade, porém evidenciam os problemas relativos à inundação da rede com interesses, especialmente em redes densas e com grande número de consumidores. Maiores taxas de envio de interesses também degradaram as métricas de desempenho, devido ao que chamamos de concorrência intraconsumidor, em que uma série de interesses de um mesmo consumidor competem pelo uso do canal. Ainda, levantou-se que o uso de *cache* oportunista só é vantajoso aos conteúdos mais populares.

Soluções tradicionais para retransmissão otimizada deveriam ser aplicáveis para sanar os problemas de *broadcast storm*, porém requerem o armazenamento de estado e troca de informações entre nós. Varvello *et al* mostram em [Varvello et al. 2011] que não há vantagem em utilizar mecanismos de sincronismo em redes de pequeno e médio porte, com um número máximo de 300 nós. Todavia, os resultados de nossas simulações mostram que o uso de difusão pode tornar inviável a implementação de ROCs utilizando canais *broadcast*, mesmo para um topologia reduzida de 60 nós. A queda de desempenho é fortemente correlacionada com o aumento da contenção e da probabilidade de colisões de pacotes, como indicado pelas métricas obtidas como resultado. Os efeitos da *broadcast storm* podem ser minimizados adotando-se políticas de inundação inteligentes. Como as ROCs não dispõem de informações topológicas e não endereçam pacotes a destinatários, a solução de inundação inteligente deve armazenar o mínimo de estado necessário e ser independente da sincronização de informações entre nós. Sensoriamento do meio e contadores de pacotes podem ser utilizados para a descoberta de vizinhos, determinação de métricas de atraso e para a priorização de retransmissões de pacotes por certos nós, reduzindo a contenção e a carga da rede. Dessa forma pretende-se, como objeto de trabalho futuro, implementar um mecanismo de otimização de retransmissões *broadcast* baseado apenas em informações locais para minimizar a probabilidade de colisão de pacotes, maximizar o total de conteúdos entregues aos consumidores e reduzir o atraso médio percebido pelas aplicações.

## Referências

Angius, F., Gerla, M. e Pau, G. (2012). Bloogo: Bloom filter based gossip algorithm for wireless NDN. Em *ACM Mobihoc NoM*, páginas 25–30.

Bai, F., Sadagopan, N. e Helmy, A. (2003). Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. Em *IEEE INFOCOM*, páginas 825–835.

Breslau, L., Cao, P., Fan, L., Phillips, G. e Shenker, S. (1999). Web caching and zipf-like distributions: Evidence and implications. Em *IEEE INFOCOM*, páginas 126–134.

Camp, T., Boleng, J. e Davies, V. (2002). A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502.

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 2012*, páginas 211–264.

Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N. e Braynard, R. (2009). Networking named content. Em *ACM CONEXT*, páginas 1–12.

Johnson, D., Perkins, C. e Arkko, J. (2004). Mobility support in IPv6. IETF Network Working Group RFC 3775.

Mealling, M. e Denenberg, R. (2002). Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations. IETF Network Working Group RFC 3305.

Meisel, M., Pappas, V. e Zhang, L. (2010). Ad hoc networking via named data. Em *ACM MobiArch*, páginas 3–8.

Moskowitz, R., Nikander, P., Jokela, P. e Henderson, T. (2008). Host identity protocol. IETF Network Working Group RFC 5201.

Ni, S.-Y., Tseng, Y.-C., Chen, Y.-S. e Sheu, J.-P. (1999). The broadcast storm problem in a mobile ad hoc network. Em *ACM MobiCom*, páginas 153–167.

Podlipnig, S. e Böszörmenyi, L. (2003). A survey of web cache replacement strategies. *ACM Computing Surveys*, 35(4):374–398.

Rappaport, T. S. (1996). *Wireless communications: principles and practice*, volume 2. Prentice Hall PTR New Jersey.

Sandvine (2011). Global Internet phenomena report. Relatório técnico, Sandvine.

Silagadze, Z. (1999). Citations and the Zipf-Mandelbrot's law. *Complex Systems 11*, páginas 487–499.

Tyson, G., Sastry, N., Rimac, I., Cuevas, R. e Mauthe, A. (2012). A survey of mobility in information-centric networks: Challenges and research directions. Em *ACM MobiHoc NoM*, páginas 1–6.

Varvello, M., Rimac, I., Lee, U., Greenwald, L. e Hilt, V. (2011). On the design of content-centric MANETs. Em *IEEE WONS*, páginas 1–8.

Wang, L., Afanasyev, A., Kuntz, R., Vuyyuru, R., Wakikawa, R. e Zhang, L. (2012). Rapid traffic information dissemination using named data. Em *ACM Mobihoc NoM*, páginas 7–12.

# GeoZone: Um *Framework* Eficiente de Difusão de Interesses em Redes Veiculares Orientadas a Conteúdo*

**Adriano Antunes Prates**[1,2], **Igor Monteiro Moraes**[1]

[1] Laboratório Mídiacom, PGC-TCC/Instituto de Computação
Universidade Federal Fluminense (UFF) – Niterói – RJ – Brasil

[2]Instituto Federal de Educação, Ciência e Tecnologia do Norte de Minas Gerais
IFNMG – Januária – MG – Brasil

{aprates,igor}@ic.uff.br

***Abstract.*** *This paper proposes the GeoZone framework in order to reduce the broadcast storm of interest packets in content-centric vehicular networks. The proposed framework is composed of (i) a geo-referenced naming scheme, which is also transparent for vehicular applications, and (ii) an interest-packet forwarding mechanism that defines a dissemination zone to reduce network flooding. Simulations are performed in different scenarios to compare the efficiency of our framework with other proposals. Compared to these proposals, the GeoZone framework reduces up to 66 % the number of interest packets transmitted without sacrifice to the content-delivery rate and delay.*

***Resumo.*** *Este artigo propõe o GeoZone, um* framework *para reduzir o problema de inundação de pacotes de interesse em redes veiculares orientadas a conteúdo. O* framework *proposto é composto por (i) um esquema para nomeação geo-referenciada de conteúdos e transparente para aplicações veiculares e por (ii) um mecanismo de encaminhamento de pacotes de interesse, que define uma zona de disseminação para reduzir a inundação da rede. Simulações em diferentes cenários são realizadas para comparar a eficiência do* framework *com outras propostas similares. Comparado a estas propostas, o GeoZone reduz em até 66% o número de pacotes de interesse transmitidos, sem prejuízo ao atraso e à taxa de entrega de conteúdos.*

## 1. Introdução

As redes veiculares (*vehicular ad-hoc networks* - VANETs) são definidas como um sistema de comunicação sem-fio entre veículos automotores e unidades de acostamento [Hartenstein e Laberteaux 2008, Alves et al. 2009]. Tais redes possuem singularidades, como o alto grau de mobilidade dos veículos e a massa crítica da rede, causando grande variação da qualidade dos enlaces e conectividade intermitente [Amadeo et al. 2012a]. Nesse contexto, um dos principais desafios é calcular e/ou manter rotas fim-a-fim entre um dado par de nós da rede [Chen et al. 2011].

A maioria das propostas de arquiteturas para redes veiculares herdam da pilha TCP/IP a necessidade de que o nó de origem conheça o endereço do nó de destino, além

*Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ, FINEP, TBE/ANEEL e Proppi/UFF.

de estabelecer uma rota até ele para obter um conteúdo [Alves et al. 2009]. Tal fato pode inviabilizar o desenvolvimento de aplicações para redes veiculares, uma vez que o estabelecimento de rotas é uma tarefa complexa [Chen et al. 2011]. Por outro lado, muitas aplicações para redes veiculares têm a característica de distribuir um mesmo conteúdo para um grande número de nós interessados como, por exemplo, aplicações que informam sobre condições de vias ou sobre vagas de estacionamento. Uma solução, portanto, é adotar uma arquitetura de rede mais eficiente para essas aplicações e que não exija a manutenção de rotas, como faz a arquitetura TCP/IP. Por isso, são propostas as redes veiculares orientadas a conteúdos (RVOCs) [Amadeo et al. 2013, Wang et al. 2012b, Arnould et al. 2011], baseadas nas redes orientadas a conteúdo (*information-centric networks* - ICNs) [de Brito et al. 2012], que adotam um paradigma de busca e entrega de conteúdos de forma independente da sua localização.

Uma das arquiteturas de orientação a conteúdos propostas na literatura é a *Content-Centric Network* (CCN) [Jacobson et al. 2009]. Na CCN, os conteúdos são identificados através do seu próprio nome e requisitados mediante o envio de pacotes de interesse. Esses pacotes são difundidos pela rede até alcançarem um nó capaz de responder a solicitação. Vale ressaltar que na CCN, cada nó da rede é dotado de um *cache* para armazenar conteúdos, permitindo que qualquer nó possa responder aos interesses recebidos a partir de conteúdos armazenados. Assim, as redes veiculares baseadas na arquitetura CCN podem se beneficiar da característica natural de difusão do meio sem-fio, tanto para disseminar interesses, quanto para espalhar os conteúdos entre os nós que escutam os pacotes de dados. Graças a essa característica, a CCN provê nativamente o suporte à mobilidade de nós consumidores [Amadeo et al. 2012a].

Um problema para a adoção das redes veiculares baseadas na CCN, porém, é a inundação de pacotes de interesse, que pode acarretar no problema de *broadcast storm*, característico de redes sem-fio [Tonguz et al. 2006]. Isto ocorre devido à política de encaminhamento dos pacotes de interesse, que a cada novo salto, reencaminha os pacotes para todos os nós vizinhos. Este trabalho, portanto, propõe um *framework*, chamado GeoZone, para reduzir o problema da inundação de interesses em redes veiculares orientadas a conteúdo. O *framework* proposto é composto por um esquema de nomeação de conteúdos, chamado de *Geographically-Based Naming Scheme* (GBNS) e por um mecanismo de encaminhamento de pacotes de interesse, chamado de *Zone Forwarding Mechanism* (ZFM). Diferentemente de outras abordagens da literatura, o GeoZone limita a área de disseminação dos pacotes de interesse com base em coordenadas geográficas contidas nos nomes dos conteúdos. Para tanto, é definido um esquema de nomeação geo-referenciado e transparente às aplicações veiculares, o que mantém as propriedades e características originais da arquitetura CCN. Simulações em diferentes cenários são realizadas para comparar a eficiência do GeoZone com outras propostas similares da literatura. O GeoZone, se comparado a essas propostas, reduz em até 66% o número total de pacotes de interesses transmitidos sem prejuízo às métricas de atraso e taxa média de entrega de conteúdos.

As próximas seções estão assim organizadas. A Seção 2 apresenta uma visão geral sobre as redes veiculares orientadas a conteúdo. A Seção 3 discute trabalhos recentes sobre o encaminhamento de pacotes em redes veiculares orientadas a conteúdo. A Seção 4 introduz o *framework* GeoZone proposto. As avaliações e os resultados são discutidos na Seção 5. Finalmente, a Seção 6 apresenta as considerações finais e trabalhos futuros.

## 2. As Redes Veiculares Orientadas a Conteúdo

O ponto chave das redes veiculares orientadas a conteúdo é a adoção de um paradigma de comunicação, cujo foco é a entrega de conteúdos de maneira independente da localização física desses conteúdos. Tanto as requisições quanto os conteúdos em si são encaminhados pelos nós da rede com base nos seus próprios nomes. Assim, não é preciso definir endereços para cada nó da rede e nem fazer requisições de conteúdo a um nó específico, de endereço conhecido. Da mesma maneira, não são necessários mecanismos para se estabelecer e/ou gerenciar rotas fim-a-fim em topologias altamente dinâmicas [Amadeo et al. 2012a]. Essas características fazem das redes veiculares orientadas a conteúdo uma solução promissora, em particular, as baseadas na arquitetura *Content-Centric Network* (CCN) [Jacobson et al. 2009], por também se aproveitarem da característica de difusão do meio sem-fio, como será discutido adiante. Wang *et al.* por exemplo, demonstram que, para um dado cenário de comunicação veicular, o IP móvel consegue uma taxa de recuperação de dados de 9,6%, enquanto que uma solução baseada em CCN consegue recuperar cerca de 97% dos dados requisitados [Wang et al. 2010].

O funcionamento da CCN é baseado em dois tipos de pacote: o pacote de interesse e de conteúdo. O pacote de interesse é enviado pelo nó consumidor para solicitar à rede algum conteúdo. O pacote de conteúdo contém o objeto de dados solicitado por um pacote de interesse e são enviados por produtores, ou seja, qualquer nó da rede que possua o conteúdo com o nome indicado no interesse em seu *cache*. Essa é uma das principais características da CCN: cada nó possui um *cache*, chamado *content store* (CS), que armazena conteúdos produzidos pelo nó ou conteúdos já encaminhados através dele. Forma-se assim na CCN, uma rede de *caches*.

Cada nó CCN possui também duas estruturas de dados para realizar o controle de encaminhamento de pacotes. A *pending interest table* (PIT) é responsável por armazenar o estado sobre interesses transmitidos pelo nó e ainda não atendidos pela rede, além das interfaces associadas a cada interesse recebido. A *forwarding interest table* (FIB) atua como uma tabela de encaminhamento para os pacotes de interesse, pois registra as interfaces pelas quais um determinado conteúdo foi recebido anteriormente. Ao receber um pacote de interesse, o nó primeiramente consulta o seu CS para verificar a existência do objeto. Se encontrar, responde o interesse através do envio do pacote de conteúdo. Caso contrário, o nó consulta a PIT para verificar a existência de uma solicitação pendente para esse mesmo conteúdo. Caso já exista esta solicitação, o nó atualiza a lista de interfaces de entrada pelas quais recebeu interesse para o conteúdo e descarta o pacote. Se não existir uma entrada para o interesse na PIT, uma nova entrada é criada e o nó consulta a FIB em busca de interfaces que levem a algum produtor do conteúdo desejado. Não encontrando entradas de encaminhamento na FIB, o nó encaminha o interesse por todas as interfaces disponíveis, a fim de alcançar qualquer produtor do conteúdo desejado. Todo esse processo é realizado a cada recebimento de um pacote de interesse por cada nó da rede. Como a PIT mantém uma lista de interfaces de entrada para cada interesse recebido, o pacote de conteúdo é entregue pelo caminho inverso ao percorrido pelos pacotes de interesse.

O suporte nativo à mobilidade também é uma característica da CCN que justifica sua adoção em redes veiculares. Considerando por exemplo, uma aplicação cliente-servidor na arquitetura TCP/IP, novas rotas devem ser calculadas para alcançar o produtor original do conteúdo à medida que a topologia física da rede se altera. Além disso, os

endereços associados à localização dos nós devem ser alterados à medida que eles se movimentam entre os pontos de acesso. Entretanto na CCN, graças à presença de *cache* nos nós da rede, o nó consumidor pode ser atendido por qualquer outro nó em sua nova área de cobertura, desde que possua o conteúdo de interesse em *cache*. Também em virtude disso, o suporte à comunicação sobre enlaces intermitentes também é nativo. É possível, por exemplo, que um veículo transporte conteúdos a regiões desconectadas do produtor original. Como a comunicação na CCN é orientada à busca de conteúdos, também não é necessário atribuir endereços físicos e/ou lógicos aos nós da rede.

Um dos problemas potencias do emprego da CCN em redes veiculares é a inundação de pacotes de interesse em redes sem-fio altamente dinâmicas, que pode acarretar no problema de *broadcast storm* [Tonguz et al. 2006]. Nessas redes, normalmente, os nós possuem apenas uma interface de rede. Assim, todo interesse recebido será encaminhado pela mesma interface de recepção e todos os vizinhos desse nó o receberão. Nesse caso, a FIB é pouco eficiente para controlar o encaminhamento de pacotes de interesse porque possui apenas uma entrada. Ainda que um nó possua múltiplas interfaces, as entradas da FIB podem se tornar obsoletas rapidamente, devido à alta mobilidade dos veículos.

Como consequência da ineficiência da tabela FIB em uma rede sem-fio, toda requisição de conteúdos em uma rede veicular orientada a conteúdo é realizada através da inundação de pacotes de interesses pela rede. Cada nó da rede ao receber um novo pacote de interesse, o reenvia para todos os seus vizinhos. A inundação prossegue até que se encontre algum produtor, e até que todos os pacotes de interesses, ora disseminados, sejam satisfeitos pelo conteúdo solicitado. Desta forma, pode-se prever que uma simples requisição de conteúdo pouco popular e distante da fonte original, comprometa significativamente os recursos disponíveis, como banda passante e memória do *cache* dos nós, além de aumentar a competição pelo meio, atraso de entrega e probabilidade de colisão.

Outro agravante para as redes veiculares é o fato de que as informações distribuídas pelas aplicações são sensíveis ao tempo em que foram geradas. Como consequência, o *cache* de conteúdos pode se tornar obsoleto com o passar do tempo, forçando que pacotes de interesses sejam encaminhados até suas fontes produtoras originais recorrentemente.

## 3. Trabalhos Relacionados

Existem vários trabalhos que propõem arquiteturas para as redes veiculares orientadas a conteúdo e abordam os desafios discutidos na Seção 2. Arnould *et al.*, por exemplo, propõem uma arquitetura baseada na CCN para disseminar informações críticas em uma rede veicular híbrida. Para isso, acrescentam um novo tipo de pacote denominado *Event Packet*. Esse pacote é propagado pela rede, mesmo quando não há interesses explícitos, e serve para avisar os nós sobre eventos emergenciais, como acidentes de trânsito e zonas de perigo [Arnould et al. 2011]. Entretanto, essa proposta modifica a arquitetura original da CCN e aumenta sua complexidade, pois os autores definem que o controle de disseminação dos *Event Packets* é realizado através do controle da banda utilizada pelos pacotes, o que pode causar falhas no funcionamento em ambientes de alta demanda.

Amadeo *et al.* propõem a arquitetura CRoWN [Amadeo et al. 2012b] e posteriormente a CCVN (*content-centric vehicular networking*), que estendem as características originais da CCN para lidar com seus desafios em redes veiculares [Amadeo et al. 2012a, Amadeo et al. 2013]. Os autores propõem a divisão dos pacotes de interesse em dois

sub-tipos: interesses básicos (*B-Int*) e avançados (*A-Int*). Um *B-Int* é enviado para descobrir produtores de um conteúdo e requisitar o primeiro segmento, enquanto que o *A-Int* é utilizado para requisitar segmentos subsequentes do conteúdo através dos produtores previamente descobertos. Além disso, uma nova estrutura de dados, denominada *Content Provider Table* (CPT) é introduzida no mecanismo em substituição à FIB. A CPT armazena informações sobre produtores já descobertos associando o endereço MAC destes nós aos conteúdos. Dessa forma, a premissa de independência dos conteúdos quanto a sua localização física é quebrada, representando uma ruptura conceitual em relação à proposta original da CCN, e podendo inclusive comprometer o suporte à mobilidade dos nós.

Wang *et al.* propõem um mecanismo de disseminação de pacotes cujo objetivo é reduzir a latência da entrega de conteúdos [Wang et al. 2012a]. Este mecanismo emprega um conjunto de temporizadores para promover a coordenação do envio de pacotes entre os diversos nós da rede, reduzindo o problema de disputa pelo meio e a probabilidade de perdas de pacotes. São definidos quatro temporizadores: (i) *Collision-avoidance timer*, que é um *backoff* aleatório para reduzir a probabilidade de colisão no envio de pacotes entre nós vizinhos; (ii) *Pushing timer*, uma janela de contenção com valor inversamente proporcional à distância do último retransmissor do pacote. Para isso, o mecanismo baseia-se num esquema de rotulação de pacotes, através do qual, a cada salto são agregadas etiquetas contendo as coordenadas geográficas da posição em que os pacotes são retransmitidos. Assim, o valor calculado para o este temporizador privilegia os nós mais distantes do último transmissor, pois possuem um tempo de espera para retransmissão menor do que os nós intermediários; (iii) *Layer retransmission timer*, que funciona como uma camada de proteção para a entrega de pacotes em nível de enlace. Após a transmissão de um pacote, o nó mantém este pacote em uma fila de retransmissão até que perceba que o mesmo fora retransmitido por um nó adiante. Caso o temporizador estoure para o pacote em questão, uma nova tentativa de transmissão é realizada pelo nó; e (iv) *Application retransmission timer*, que é um temporizador que dispara o reenvio de interesses em caso de não recebimento do conteúdo, principalmente devido ao problema de intermitência de conectividade e o particionamento da rede veicular. Cabe entretanto, algumas ressalvas quanto ao trabalho. O mecanismo proposto não estabelece limites para a disseminação de interesses e, portanto, o problema da inundação ainda persiste. Além disso, a avaliação é incipiente, pois não considera o impacto da mobilidade dos nós.

## 4. *Framework* GeoZone

A proposta deste trabalho é um *framework* para disseminação eficiente de interesses em redes veiculares orientadas a conteúdo, que reduz o problema da inundação desses pacotes e que não exige modificações conceituais da arquitetura CCN. Esse *framework*, denominado GeoZone, é composto por dois mecanismos: o *Geographically-Based Naming Scheme* e o *Zone Forwarding Mechanism*, detalhados a seguir.

### 4.1. O Esquema de Nomeação GBNS (*Geographically-Based Naming Scheme*)

A CCN define um esquema hierárquico para nomeação de conteúdos, no qual elementos iniciais do nome possuem nível hierárquico maior do que os posteriores. A partir da definição dos nomes dos objetos da rede, nós consumidores podem enviar pacotes de interesse para a obtenção de dados. Entretanto, um desafio do projeto de nomeação de conteúdos é a transparência destes nomes para os nós consumidores, ou seja, permitir que

quem necessite da informação saiba, a *priori*, o nome do objeto desejado. Uma das vantagens da nomeação hierárquica, é permitir a associação de diferentes níveis de semântica ao nome do conteúdo, como a identificação de aplicações, uso de escopos temporais, geográficos e espaciais [Wang et al. 2012b]. Essa característica serve de base para o esquema de nomeação proposto neste trabalho, o *Geographically-Based Naming Scheme* (GBNS).

O GBNS define que os componentes mais significativos do nome do conteúdo sejam obrigatoriamente associados às coordenadas geográficas (longitude e latitude) do produtor original do conteúdo. Por isso, assume-se que os nós da rede possuem dispositivos GPS (*Global Positioning System*). Níveis hierárquicos menos significativos devem ser usados para identificar a aplicação alvo (*application*), além do tempo (*timestamp*) relativo ao conteúdo que se deseja, conforme a estrutura a seguir:

```
geozone://longitude/latitude/application/timestamp
```

Com essa estrutura, por exemplo, é possível obter a programação do dia em um cinema, assumindo que há uma unidade de acostamento nesse local. Para isso, usuários devem enviar pacotes de interesse com as coordenadas `longitude` e `latitude` do cinema, o identificador do cinema (`application`) e o dia de interesse (`timestamp`). Em outro exemplo de aplicação, usuários podem estar interessados na situação atual do trânsito em uma dada área. Assim devem enviar pacotes de interesse com a `longitude` e `latitude` da área desejada, a informação que desejam obter (p.e. velocidade média) informada pelo elemento `application` e a hora atual (`timestamp`). Veículos que estão, ou que passaram recentemente pela área apontada nas coordenadas, podem responder aos interesses, desde que armazenem periodicamente sua velocidade média e coordenadas. O funcionamento detalhado de aplicações está fora do escopo deste trabalho.

O GBNS não rompe com a premissa de independência dos conteúdos em relação à sua localização física, característica da CCN. A nomeação de conteúdos baseada na identificação geográfica dos produtores originais não indica que somente o nó que está nessa coordenada pode responder um pacote de interesse. Ele indica o interesse em informações daquelas coordenadas. Além disso, não impede que outros nós utilizem-se de *caches* para armazenar estes conteúdos e atuarem também como produtores.

Outra característica do GBNS é que ele é transparente para os consumidores, ou seja, o esquema de nomeação permite que quem necessite da informação saiba, a *priori*, o nome do objeto desejado. Através do dispositivo GPS é possível determinar as coordenadas dos pontos de interesses com base na rota previamente estabelecida, portanto, o próprio consumidor pode criar e emitir interesses destinados a aplicações CCN veiculares. O GBNS difere do esquema de nomeação proposto por Wang *et al.*, pois, ao invés de representar a aplicação como o componente de maior nível hierárquico, o GBNS apresenta as informações de geo-localização como os componentes de mais alto nível. Isto permite que nós consumidores tenham maior controle e autonomia para requisição dos conteúdos desejados, como a possibilidade de solicitar a um ponto de interesse a lista de aplicações disponíveis. Além disso, evita que um interesse seja tratado por aplicações similares em produtores de diferentes regiões.

### 4.2. O Mecanismo de Encaminhamento ZFM (*Zone Forwarding Mechanism*)

A requisição de conteúdo nas redes veiculares orientadas a conteúdo é baseada na inundação de pacotes de interesse por toda a rede. Esta abordagem, no entanto, pode

comprometer a escalabilidade da rede em função do problema de *broadcast storm*, como visto na Seção 2. Para reduzir esse problema, este trabalho propõe-se o *Zone Forwarding Mechanism* (ZFM). Este mecanismo define uma zona de disseminação de pacotes de interesse com base nas coordenadas geográficas do nó consumidor e do próprio nome do conteúdo desejado. Isso só é possível graças ao esquema de nomeação GBNS, cujos nomes carregam as referências geográficas do ponto alvo de interesse. Uma vez definida essa zona de disseminação, somente nós que estejam dentro dessa zona ao receber o pacote de interesse devem reencaminhá-lo. Assim, somente parte da rede é inundada.

Para definir a zona de disseminação, assume-se que conteúdos de interesse têm maior probabilidade de estar em uma zona a frente da posição atual e na mesma direção de deslocamento do nó consumidor. Por exemplo, ao receber um caminho traçado por seu dispositivo GPS, um usuário deseja saber se as vias pelas quais irá passar estão congestionadas ou se existem bons restaurantes no local de destino. As informações dos locais por onde ele já passou não são de interesse. Portanto, considera-se que a probabilidade de um conteúdo chegar ao consumidor a partir de um dos nós situados entre ele a posição indicada no nome do conteúdo é maior do que a probabilidade desse mesmo conteúdo chegar através dos nós situados na direção oposta de deslocamento do nó consumidor.

A Figura 1 ilustra e exemplifica a definição da zona de disseminação. Ela representa a região central da cidade do Rio de Janeiro e mostra três veículos, $V_1, V_2$ e $V_3$, que desejam obter informações de trânsito dos pontos $P_1, P_2$ e $P_3$, no qual existem unidades de acostamento. Considere a zona de disseminação entre $V_3$ e $P_3$, definida pela área som-

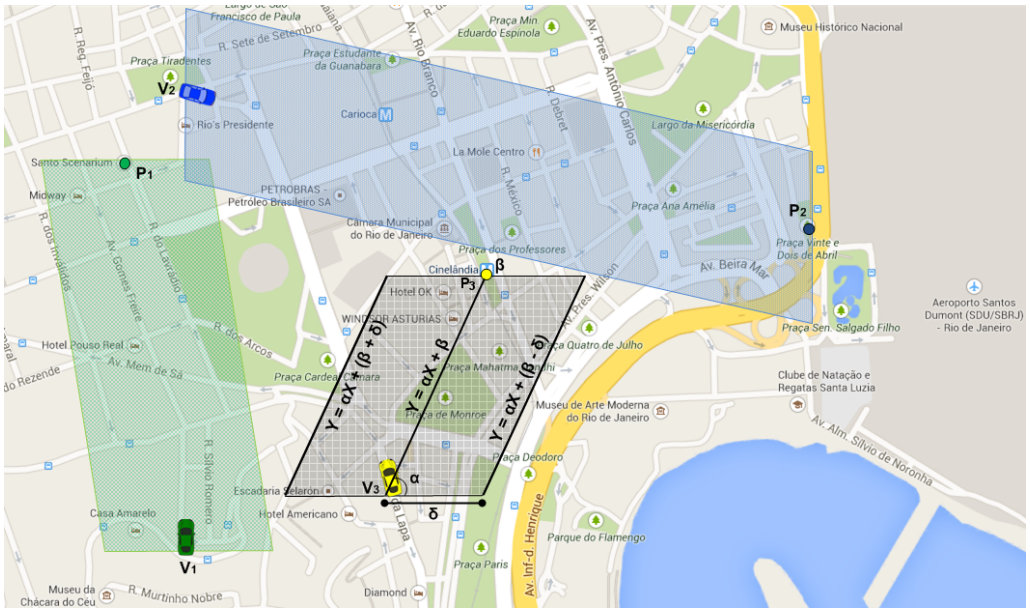



**Figura 1. Exemplos de zona de disseminação definidas pelo GeoZone.**

breada. Esta zona é calculada, como todas as outras, da seguinte forma. Sejam $P_c(x_c, y_c)$ as coordenadas geográficas do nó consumidor e $P_p(x_p, y_p)$ as coordenadas do ponto-alvo de interesse, a Equação 1 define uma reta entre esses dois pontos, na qual $\alpha$ e $\beta$ representam, respectivamente, os coeficientes angular e linear da reta. Os valores $\alpha$ e $\beta$ são calculados pelas Equações 2 e 3.

$$Y = \alpha X + \beta \tag{1}$$

$$\alpha = \Delta y/\Delta x = (y_c - y_p)/(x_c - x_p) \quad (2)$$

$$\beta = y_c - \alpha x_c \quad (3)$$

A zona de disseminação de interesses é, então, definida por duas retas paralelas à reta definida pelos pontos $P_c$ e $P_p$, dadas pela Equação 4.

$$Y' = \alpha X' + (\beta \pm \delta) \quad (4)$$

Tem-se, portanto, a constante $\delta$, que define as margens laterais da área em torno da reta que une $P_c$ e $P_p$, e que delimita a zona de disseminação de interesses. Essa constante é definida de acordo com o grau de inundação da rede que se deseja.

**Algorithm 1** Process_Interest_ZFM ( $interest$ )

```
consumer_position ⇐ extract_source_position_tag ( interest )
producer_position ⇐ extract_producer_position ( interest )
actual_position ⇐ get_position ( )
ZFM ⇐ set_ZFM ( consumer_position , producer_position )
if inside_ZFM ( actual_position , ZFM ) then
   forward ( interest )
else
   discard ( interest )
end if
```

Ao receber um pacote de interesse, cada nó deve executar um teste de pertinência, definido pelo Algoritmo 1, para saber se está ou não dentro da zona de disseminação desse pacote. Para tanto, o consumidor, ou seja, aquele que gera o pacote de interesse inicial, insere uma etiqueta contendo suas coordenadas geográficas no pacote. A esta etiqueta dá-se o nome de *Source Position Tag*. Por simplificação, nas simulações realizadas foi agregado um campo ao pacote de interesse para armazenar essa etiqueta. Dessa forma, o nó possui a posição do consumidor e a posição do produtor, informada no nome do conteúdo, e pode calcular a zona de disseminação segundo as Equações 1 a 4. Como sabe sua posição atual, pois possui um dispositivo GPS, o nó consegue determinar facilmente se deve ou não reencaminhar o pacote de interesse.

Para reduzir a probabilidade de colisões na zona de disseminação definida pelo ZFM, o GeoZone implementa parte do mecanismo proposto em [Wang et al. 2012a]. São usados os temporizadores: *collision-avoidance timer*, *pushing timer* e *layer retransmission timer*, todos já vistos na Seção 3.

### 4.3. GeoZone Não-Proativo e Proativo

O GeoZone reduz a inundação de pacotes de interesse. Consequentemente, menos nós recebem esses pacotes e podem se tornar possíveis encaminhadores de conteúdo. Para avaliar o impacto do *framework* proposto na entrega de conteúdos, são introduzidas duas versões, que diferem em relação à forma de armazenar e encaminhar os conteúdos: o GeoZone não-proativo e o GeoZone proativo.

O GeoZone não-proativo opera conforme a proposta da arquitetura CCN original. Os pacotes de conteúdos são armazenados e retransmitidos somente por nós que possuem a entrada de interesse correspondente na PIT. Caso algum nó receba um pacote de

conteúdo, mas não apresente uma entrada na PIT para esse conteúdo, ele irá descartar o pacote. Esse conceito aplicado ao GeoZone faz com que apenas os nós que, durante a fase de propagação de interesses estavam dentro da zona de disseminação, possam armazenar e retransmitir o conteúdo de interesse. Em redes veiculares esta solução pode ser pouco eficiente, pois, não há garantias de que os nós fiquem sob uma mesma zona por um dado período de tempo, uma vez que estão em movimento.

O GeoZone proativo, por outro lado, emprega mecanismos de *cache* oportunístico e retransmissão proativa [Rao et al. 2013]. Com o *cache* oportunístico, um nó pode armazenar um conteúdo em seu CS mesmo sem ter enviado um interesse por ele. Além disso, os nós também realizam a retransmissão de conteúdos proativamente, contribuindo para que um dado objeto seja popularizado na rede. Em redes altamente dinâmicas, a disseminação de conteúdos populares é benéfica, pois permite que os nós levem conteúdos aos vizinhos. Dessa forma, aumenta-se o número de oportunidades de entrega de conteúdo com menor número de saltos, além de facilitar a comunicação entre regiões desconexas [Vasilakos et al. 2012, Rao et al. 2013].

## 5. Avaliação

Para avaliar o desempenho do *framework* GeoZone, utiliza-se o simulador ndnSIM[1]. Também foi implementado um modelo de mobilidade para simular o padrão de tráfego de veículos em um centro urbano, denominado *Urban-V2V-Mobility Model*. Neste modelo, os veículos são uniformemente distribuídos em uma grade de área delimitada. Cada nó sorteia uma direção e velocidade a cada 100 metros percorridos. O objetivo é simular uma tomada de decisão a cada quarteirão/cruzamento. As direções variam apenas em ângulos de 0°, 90°, 180° ou 270°, excluindo a direção oposta à direção inicial e atual do veículo. Desta forma, dada a direção inicial do veículo, o modelo exclui a possibilidade deste nó retornar à sua posição original, e a possibilidade de um nó selecionar a direção oposta à última sorteada. As velocidades variam de 20 a 60 km/h.

A tecnologia de rede utilizada é o padrão 802.11a em modo *ad-hoc*, como alternativa ao padrão DSRC (*Dedicated Short-Range Communication*) não disponível no simulador, com modulação OFDM (*Orthogonal Frequency-Division Multiplexing*) e taxa de transmissão de 24 Mbps. O modelo de propagação utilizado é o definido pela sub-classe `ns3::YansWifiPhy`, com potência de transmissão de 5 dbm. Para o temporizadores adotou-se os mesmos valores definidos em [Wang et al. 2012a]. Os nós possuem *cache* com tamanho de 10000 bytes e política de descarte LRU (*Least Recently Used*).

O objetivo com os parâmetros definidos é simular cenários urbanos com diferentes densidades, nos quais vários veículos enviam interesses, simultaneamente, para um ponto de interesse que possui uma unidade de acostamento e coordenadas bem conhecidas, como pontes, centros comerciais, avenidas principais etc. A Tabela 1 resume os parâmetros dos quatro cenários avaliados. Cada linha representa um cenário. Em todos, há 50 nós consumidores para um dado conteúdo armazenado inicialmente apenas pela unidade de acostamento do ponto de interesse, por isso, também é chamado de produtor incial. No instante 0, cada nó consumidor envia um pacote de interesse destinado ao produtor inicial. A quantidade de nós na rede, a distância entre os veículos, a área de simulação e a distância que separa os consumidores do ponto de interesse variam em cada

[1]http://ndnsim.net

| Quantidade Consumidores | Qtde. Nós no Núcleo da Rede | Distância entre Veículos (Hor. e Vert.) | Área Total | Distância Consumidor x Produtor inicial |
|---|---|---|---|---|
| 50 | 500 | 100 m. | $\approx 6.2$ km$^2$ | $\approx 1$ km. |
| 50 | 1050 | 75 m. | $\approx 7.2$ km$^2$ | $\approx 1$ km. |
| 50 | 625 | 200 m. | $\approx 30$ km$^2$ | $\approx 5$ km. |
| 50 | 2500 | 100 m. | $\approx 28$ km$^2$ | $\approx 5$ km. |

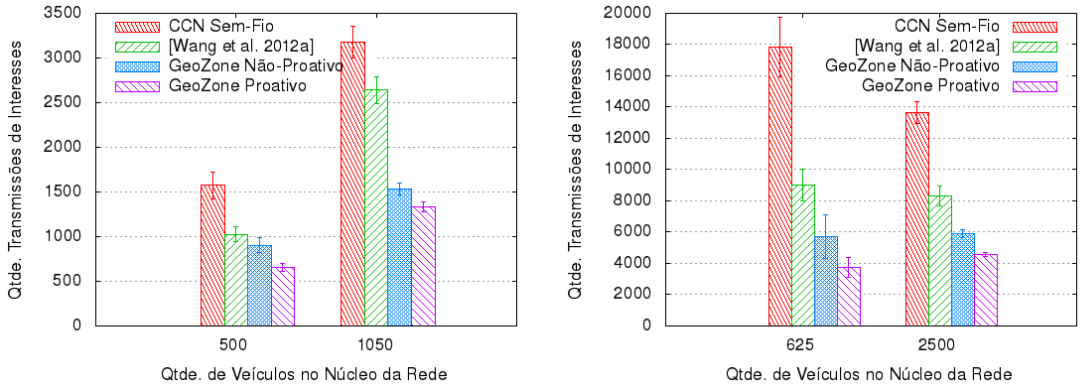
**Tabela 1. Parâmetros dos cenários de avaliação.**

cenário. Os nós consumidores e o produtor inicial são os mesmos para todos os cenários. O pacote de conteúdo contém carga útil fixa de 300 bytes e a constante $\delta$, que delimita a zona de disseminação de interesses, é definida com valor de 200 m. Nas simulações, por simplificação, todos usam o esquema de nomeação GBNS.

Para cada cenário avaliam-se quatro variações da arquitetura CCN:

- Adaptação da arquitetura CCN convencional para redes sem-fio, sem mecanismo de *cache* proativo e retransmissão proativa de conteúdos;
- [Wang et al. 2012a], com *cache* proativo e retransmissão proativa de conteúdos;
- GeoZone não-proativo, sem *cache* proativo nem retransmissão proativa;
- GeoZone proativo, com *cache* proativo e retransmissão proativa de conteúdos;

Apresentam-se a seguir os resultados obtidos a partir da análise de três importantes métricas: quantidade de transmissões de pacotes de interesse, quantidade de transmissões de pacotes de conteúdo e o atraso médio de entrega dos conteúdos aos consumidores. Para cada cenário, foram realizadas 10 rodadas de simulação com diferentes sementes de geração de números aleatórios. Para os gráficos apresentados, calcula-se o intervalo de confiança, representados por barras verticais, para um nível de confiabilidade de 95%.

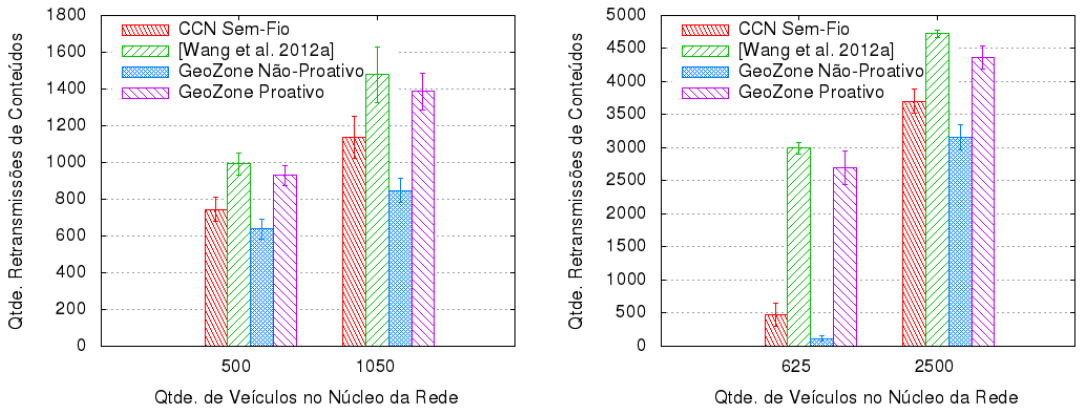



(a) Distância Consumidor-Produtor: $\approx 1$ km. (b) Distância Consumidor-Produtor: $\approx 5$ km.

**Figura 2. Número de transmissões de interesses.**

A Figura 2 apresenta a quantidade de pacotes de interesses transmitidos em função do número de nós no núcleo da rede. Considera-se que o núcleo da rede é formado pelos veículos que não estão entre os 50 consumidores e produtor inicial. Observa-se que o *framework* GeoZone reduz significativamente a quantidade de interesses transmitidos e

necessários para que todos os consumidores obtenham o conteúdo. Entre os mecanismos proativos, a redução proporcionada pelo GeoZone é de aproximadamente 50%, quando o núcleo da rede possui 1050 nós e a distância consumidor-produtor é de 1 km. Na comparação entre as propostas não-proativas, a diferença é de até aproximadamente 57% no cenário com 2500 nós. Já considerando todas as abordagens, no cenário com 2500 nós, o GeoZone proativo reduz em até 66% o número de pacotes de interesse transmitidos em relação a proposta da CCN sem-fio. Este resultado é fruto do mecanismo ZFM que limita a área de disseminação de interesses na rede e, portanto, mostra que o GeoZone é eficiente no controle da inundação desses pacotes em redes veiculares orientadas a conteúdo. Percebe-se também que o GeoZone proativo entrega todos os conteúdos com o menor número de transmissões de interesses nos quatro cenários. Isto ocorre porque todos os nós contribuem para a entrega de conteúdos, inclusive os que durante o envio do interesse estão fora da zona de disseminação. Isto certamente facilita a entrega de pacotes de dados em redes com maior grau de mobilidade. Outra conclusão é que quantidade de interesses transmitidos é diretamente proporcional à distância entre os consumidores e a região-alvo.




(a) Distância Consumidor-Produtor: ≈ 1 km. (b) Distância Consumidor-Produtor: ≈ 5 km.

**Figura 3. Número de transmissões de conteúdos.**

A Figura 3 ilustra a quantidade de pacotes de conteúdos transmitidos em relação ao número de veículos no núcleo da rede. Nota-se que os dois mecanismos não-proativos apresentam o menor número de replicações de conteúdos para atender todos os 50 consumidores. Esse é um resultado esperado, pois somente os nós com a respectiva entrada na tabela PIT podem encaminhar pacotes de conteúdo. Entretanto, como será visto adiante, o impacto no atraso e na taxa de entrega destes conteúdos é alto. Considerando-se uma classificação de soluções proativas ou não-proativas, o GeoZone se destaca com resultados positivos em relação aos concorrentes.

A Figura 4 mostra o atraso médio de entrega do conteúdo aos consumidores que estão distantes cerca de 1 km do produtor inicial. As propostas proativas obtêm uma significativa melhoria de desempenho nesta métrica. Percebe-se, portanto, que a liberdade concedida para que qualquer nó possa contribuir com o armazenamento e a entrega de conteúdos favorece o rápido atendimento das solicitações em cenários móveis. Em cenários mais densos, porém, essa liberdade tem o efeito colateral de aumentar a disputa pelo

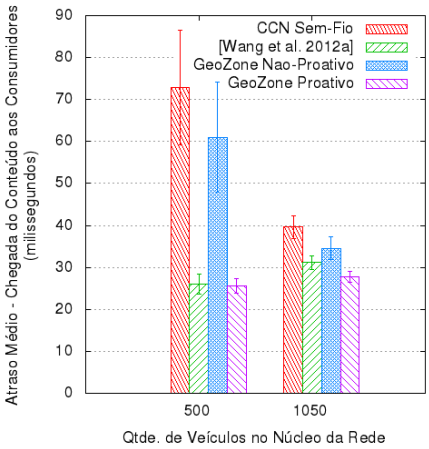
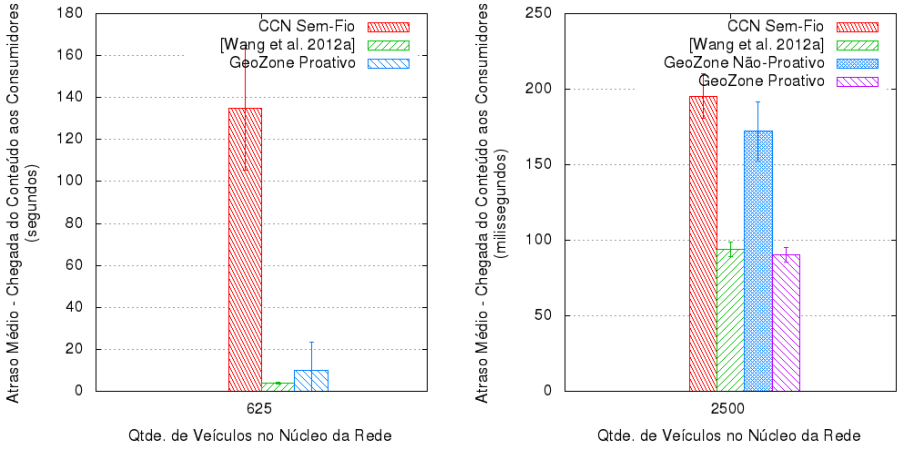



**Figura 4. Atraso médio - distância consumidor x produtor aprox. 1km.**

meio de transmissão. Além disso, as duas versões do GeoZone não aumentam significativamente o atraso de entrega em relação à proposta de Wang*et al.*, cujo objetivo principal é reduzir o tempo de entrega dos conteúdos. Portanto, o *framework* proposto reduz significativamente o número de pacotes de interesse transmitidos pelos nós da rede, sem afetar o desempenho em termos do atraso para entrega do conteúdo.

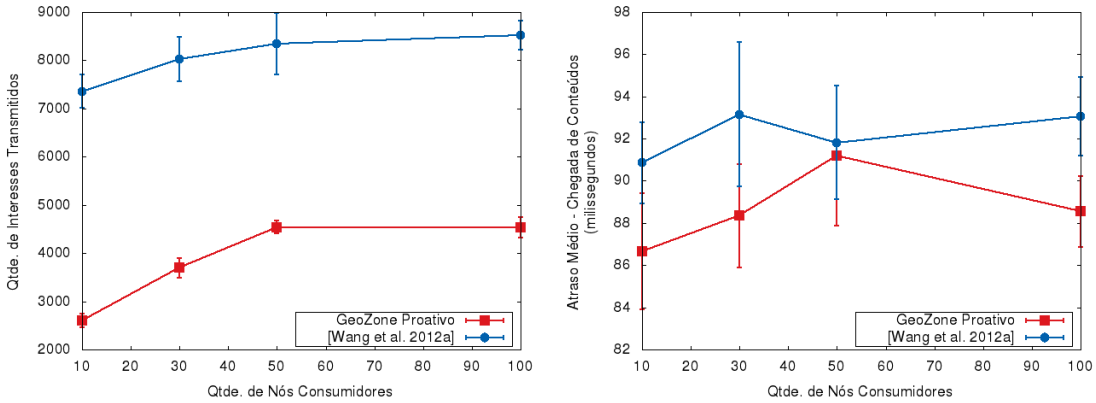



(a) 625 nós (atraso em segundos). (b) 2500 nós (atraso em milissegundos)

**Figura 5. Atraso médio - distância produtor x consumidor aprox. 5km.**

A Figura 5 apresenta o atraso médio de entrega do conteúdo para cenários nos quais o produtor inicial está aproximadamente 5 km distante dos consumidores. Na Figura 5(a), que ilustra o cenário menos denso, a CCN sem-fio obteve sucesso na recuperação de conteúdos em apenas 30% das simulações executadas. Desta forma, o gráfico apresentado considera apenas esses resultados. O atraso médio obtido foi de mais de 2 minutos. O GeoZone não-proativo não teve sucesso na recuperação de conteúdos, por isso, não está representado no gráfico. Esses resultados mostram o quanto as abordagens não-proativas sofrem em ambientes de menor densidade. Em contrapartida, as abordagens proativas recuperaram todos os conteúdos solicitados com atraso até 22 vezes menor. Mais uma vez, conclui-se que a maior participação de nós nos processos de armazenamento e enca-

minhamento de conteúdos é benéfica para o ambiente veicular. Na Figura 5(b), o cenário de maior densidade, observa-se que os mecanismos proativos obtêm resultados melhores, reduzindo em aproximadamente 50% o tempo necessário para recuperação dos conteúdos. Além disso, o GeoZone experimenta uma valor de atraso similar ao de Wang *et al.*.




(a) Interesses transmitidos (b) Atraso de entrega

**Figura 6. Resultados em função da quantidade de consumidores.**

A Figura 6 apresenta os resultados da avaliação do cenário composto por 2500 nós em uma área de aproximadamente 28 km$^2$ em função da quantidade de nós consumidores, que varia de 10 a 100 nós. Somente foram avaliados os mecanismos proativos, por serem os mais eficientes até então. A Figura 6(a) mostra a evolução da quantidade de transmissões de interesses necessárias para entrega de conteúdos. Destaca-se significativa redução de interesses enviados com GeoZone independentemente do número de consumidores. Nota-se também que a curva dos dois mecanismos é crescente até o momento em que se tem mais de 50 consumidores. Neste caso, todos os veículos da rede já possuem entradas para o mesmo interesse na PIT, ignorando novas solicitações. Na Figura 6(b) observa-se que a variação do atraso médio para entrega dos conteúdos é da ordem de poucos milissegundos entre os mecanismos avaliados. Considerando os intervalos de confiança, pode-se concluir que as propostas apresentam resultados equivalentes.

## 6. Conclusões

Este artigo propôs o GeoZone, um *framework* para reduzir a inundação de pacotes de interesse em redes veiculares baseadas na CCN, composto por dois mecanismos: um esquema de nomeação geo-referenciada de conteúdos e um mecanismo de encaminhamento de interesses baseado em zonas de disseminação. Através de simulações para cenários urbanos com diferentes densidades, mostra-se que o GeoZone reduz em até 66% o número total de pacotes de interesse transmitidos, sem impactos negativos ao atraso e à taxa de entrega de dados, em relação a outras propostas, e também, sem prejuízo às vantagens oferecidas pela arquitetura CCN às redes móveis.

Como trabalhos futuros, pretende-se avaliar o desempenho do GeoZone em cenários reais de mobilidade veicular, além de compará-lo com protocolos de roteamento tradicionais para redes *ad-hoc*. Outra proposta para aumentar a eficiência do GeoZone

é tornar o mecanismo ZFM adaptativo. Percebe-se que este mecanismo deve obter melhores resultados em cenários de baixa densidade. Desta forma, propõe-se a aplicação de métodos de percepção do nível de disputa pelo meio para determinar o nível de densidade da rede e, desta maneira, ajustar a área de encaminhamento de interesses dinamicamente.

## Referências


Alves, R. d. S., Campbell, I. d. V., Couto, R. d. S., Campista, M. E. M., Moraes, I. M., Rubinstein, M. G., Costa, L. H. M., Duarte, O. C. M. e Abdalla, M. (2009). Redes veiculares: Princípios, aplicaçoes e desafios. *Minicursos do SBRC*, páginas 199–254.

Amadeo, M., Campolo, C. e Molinaro, A. (2012a). Content-centric networking: is that a solution for upcoming vehicular networks? Em *ACM VANET*, páginas 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.

Arnould, G., Khadraoui, D. e Habbas, Z. (2011). A self-organizing content centric network model for hybrid vehicular ad-hoc networks. Em *ACM DIVANet*, páginas 15–22.

Chen, W., Guha, R. K., Kwon, T. J., Lee, J. e Hsu, Y.-Y. (2011). A survey and challenges in routing and data dissemination in vehicular ad hoc networks. *Wireless Communications and Mobile Computing*, 11(7):787–795.

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*, 2012:211–264.

Hartenstein, H. e Laberteaux, K. (2008). A tutorial survey on vehicular ad hoc networks. *IEEE Communications Magazine*, 46(6):164–171.

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*, páginas 1–12.

Rao, Y., Zhou, H., Gao, D., Luo, H. e Liu, Y. (2013). Proactive caching for enhancing user-side mobility support in named data networking. Em *IMIS*, páginas 37–42.

Tonguz, O., Wisitpongphan, N., Parikh, J., Bai, F., Mudalige, P. e Sadekar, V. (2006). On the broadcast storm problem in ad hoc wireless networks. Em *BROADNETS Conference*, páginas 1–11.

Vasilakos, X., Siris, V. A., Polyzos, G. C. e Pomonis, M. (2012). Proactive selective neighbor caching for enhancing mobility support in information-centric networks. Em *ACM ICN SIGCOMM Wkshp.*, páginas 61–66.

Wang, J., Wakikawa, R. e Zhang, L. (2010). DMND: Collecting data from mobiles using named data. Em *IEEE VNC*, páginas 49–56.

Wang, L., Afanasyev, A., Kuntz, R., Vuyyuru, R., Wakikawa, R. e Zhang, L. (2012a). Rapid traffic information dissemination using named data. Em *ACM NoM Wkshp.*, páginas 7–12.

Wang, L., Wakikawa, R., Kuntz, R., Vuyyuru, R. e Zhang, L. (2012b). Data naming in vehicle-to-vehicle communications. Em *NOMEN Wkshp.*, páginas 328–333.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 5*

## *Redes Veiculares I*

# Análise da Capacidade de uma Rede Tolerante a Atrasos e Desconexões na Bacia Hidrográfica Amazonense

**Alyson de Jesus dos Santos[1], Marcus de Lima Braga[1], Pedro Braconnot Velloso[2], José Geraldo Ribeiro Júnior[1,3] e Luís Henrique M. K. Costa[1]**

[1]Grupo de Teleinformática e Automação - PEE/COPPE
Universidade Federal do Rio de Janeiro - (UFRJ)

[2]Instituto de Computação - PGC
Universidade Federal Fluminense (UFF)

[3]Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG
Leopoldina - MG

{alyson,marcus,jgrjunior,luish}@gta.ufrj.br, velloso@ic.uff.br

***Resumo.*** *Na Amazônia, em função das características regionais, é predominante o transporte hidroviário. Portanto, a implantação de redes veiculares ad hoc (VANETs) é uma importante opção para interconectar toda a região. Neste sentido, é primordial aproveitar o contato entre as embarcações a fim de obter uma maior capilaridade da rede e maior eficiência. Este trabalho apresenta uma avaliação de desempenho experimental da comunicação de dados sem-fio, padrão IEEE 802.11 b/g, entre embarcações no Rio Negro. O principal objetivo é caracterizar a transmissão e os contatos das embarcações, em termos da vazão útil da rede e do tempo de contato. Os resultados obtidos na análise demonstram a viabilidade do tráfego de dados entre as embarcações, com as velocidades praticadas na região. Além disso, combinando os resultados experimentais com os itinerários das embarcações de linhas regulares das hidrovias da Amazônia Ocidental obtemos uma estimativa da capacidade desta rede tolerante a atrasos e desconexões.*

***Abstract.*** *In the Amazon, given that regional characteristics, is prevalent waterborne transportion. Therefore, the deployment of Vehicular ad hoc networks (VANETs) is important choice to interconect the region. This context, is fundametal to utilize the contatc between vessels to obtain an increased capillarity network and increased efficiency. This paper presents an experimental performance evaluation of data communication wireless, IEEE 802.11 b/g, between vessels in Rio Negro. The main goal is to characterize the transmission and the contacts of vessels, according the network goodput and the contact time. The results of the experiments showed the feasibility the data traffic between vessels with the speeds commom in region. Furthermore, combining the experimetal results with itineraries of vessels of regular lines of waterways of west Amazon obtain a estimate capacity this delay tolerant networks.*

## 1. Introdução

A comunicação entre dispositivos móveis desempenha um papel importante no suporte aos Sistemas Inteligentes de Transporte *(Intelligent Transportation Systems - ITS)*,

permitindo a utilização de serviços e aplicações ao longo das vias. No cenário das redes veiculares, entre as aplicações mais utilizadas estão a segurança no trânsito, o entretenimento e a assistência ao motorista [Alves et al., 2009]. Serviços de localização, notícias locais, informações turísticas e mensagens de alerta sobre a via são exemplos de serviços. No entanto, ITS não se restringe a veículos automotores, engloba o transporte ferroviário e o hidroviário, foco deste trabalho.

Na região Amazônica, devido à predominância de rios e florestas, o principal meio de transporte é o hidroviário. Este atinge 65% dos municípios, enquanto o transporte aéreo cerca de 21% e o rodoviário, apenas 13% [Souza, 2009, Neto e Pimentel, 2010]. Nesse cenário, as hidrovias fazem o papel das ruas ou estradas e as embarcações o dos veículos [Moraes, 2013]. O transporte de cargas pela região está diretamente relacionado com a dinamicidade econômica dos municípios, variando durante os períodos de cheia e vazante dos rios no quesito de rotas disponíveis entre os municípios.

Vários municípios da região Amazônica estão excluídos tecnologicamente em virtude da inviabilidade econômica, geográfica ou técnica. Montar uma infraestrutura de comunicação nesses municípios, comunidades ou vilarejos, que estão distante das metrópoles, possui elevado custo. A motivação deste trabalho é prover alternativas para conectar essas áreas remotas aos grandes centros urbanos, por meio da malha hidrográfica. Em função das grandes áreas esparsas, garantir conectividade em qualquer lugar e a todo instante é difícil. Diante deste cenário, as Redes Tolerantes a Atrasos e Desconexões *(Delay and Disruption Tolerant Networks - DTNs)* surgem como alternativa viável e de menor custo. Desta forma, as embarcações transportam os dados no período que não há conexão entre as embarcações e transmitem os dados quando fazem o contato com outra embarcação, em modo ad hoc ou com uma infraestrutura fixa no leito do rio e nos portos das cidades.

Este artigo apresenta através de uma análise experimental um estudo da viabilidade e capacidade de redes DTNs na bacia Amazônica. Primeiramente, coletaram-se informações provenientes da Administração das Hidrovias da Amazônia Ocidental (AHIMOC), órgão tutelar das embarcações que fazem linhas regulares. Em seguida, foram executadas medições às margens do Rio Negro da quantidade de dados transferidos e do tempo de contato entre as embarcações, usando uma rede sem-fio 802.11 b/g. As principais contribuições deste artigo são: (i) o estudo de capacidade das redes DTNs nos rios do Amazonas; (ii) o modelo de regressão linear que expressa a relação entre o tempo de contato e a quantidade de dados transferidos entre as embarcações; (iii) o potencial semanal da capacidade de rede da malha hidroviária na bacia Amazônica, considerando-se as principais calhas dos rios da Amazônia Ocidental. Os resultados obtidos mostram que o potencial de dados transferidos entre as embarcações é da ordem de gigabytes durante o intervalo de tempo de uma semana.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os conceitos que envolvem as redes veiculares e as redes DTNs. A Seção 3 discute os trabalhos relacionados. A Seção 4 descreve o ambiente em que foram realizados os experimentos, as características e os resultados obtidos. Por fim, a Seção 5 apresenta as considerações finais e os trabalhos futuros.

## 2. Redes Veiculares e Redes DTNs

As redes veiculares são formadas por veículos automotores, por equipamentos fixos localizados no entorno das vias e por um sistema de comunicação sem-fio [Alves et al., 2009]. Os principais desafios técnicos que essas redes enfrentam são a alta mobilidade dos nós, o dinamismo dos cenários e a escalabilidade em termos do número de nós. Tais redes possuem um baixo tempo de contato entre os nós, que estão sujeitos a perda de conectividade durante a transmissão dos dados devido a interferência de outras fontes, redução da força do sinal e da propagação multivias. São exemplos de veículos automotores: os automóveis, os caminhões, os ônibus, as motocicletas, os barcos entre outros.

A arquitetura das redes veiculares define a forma como os nós se organizam e se comunicam, que podem ser: ad hoc puro, infraestruturada ou híbrida [Alves et al., 2009, Daher e Vinet, 2012]. Nas redes ad hoc, os veículos se comunicam sem a presença de um elemento centralizador. Nessa arquitetura, a influência na conectividade da rede depende da densidade e do padrão de mobilidade dos veículos. Nas redes infraestruturadas, os nós estáticos estão dispostos ao longo das vias, centralizando tráfego e possibilitando a comunicação com outras redes. Nas redes híbridas, os veículos comunicam entre si e/ou com a infraestrutura fixa, aumentando a conectividade dessas redes. As comunicações entre nós móveis são denominadas V2V *(Vehicle-to-Vehicle)* e entre nós móveis e a infraestrutura, V2I *(Vehicle-to-Infrastructure)*.

Em virtude das características do ambiente veicular, é difícil garantir conectividade plena entre veículos e unidades de acostamento nos cenários hidroviários. Por isso, novas alternativas de comunicação precisam ser investigadas, como o uso de redes DTNs.

As redes DTNs são caracterizadas por operarem com conectividade intermitente, o que pode causar longos atrasos e baixas taxas de entrega de mensagens [Fall, 2003]. Nessas redes, quando uma mensagem precisa ser enviada, ela é armazenada e encaminhada nó a nó desde a origem até o destino [Demmer e Fall, 2007]. Desta forma, os nós intermediários armazenam-carregam-e-encaminham *(store-carry-and-forward)* as mensagens ao estabelecer um contato. Um contato corresponde a uma ocasião favorável para os nós trocarem dados.

A arquitetura das redes DTNs consistem na utilização da técnica de comutação de mensagens e na inserção de uma sobrecamada chamada de camada de agregação ou de empacotamento *(Bundle Layer)* [Demmer e Fall, 2007] entre a camada de aplicação e de transporte, permitindo o desenvolvimento de aplicações sem se preocupar com as características da rede e garantir a interoperabilidade entre diversos tipos de redes. Essa sobrecamada é responsável pela agregação das mensagens e provê o armazenamento persistente dos dados.

## 3. Trabalhos Relacionados

Diversos trabalhos analisam empiricamente a capacidade de transferência de dados nas redes veiculares, em diferentes cenários. Em [Ott e Kutscher, 2004] é estudado o acesso à Internet em vias expressas com o padrão IEEE 802.11b, avaliando a capacidade de transferência de dados. Os veículos movimentam-se em sentidos opostos, com velocidades que variam de 80 a 180 km/h, cruzando os dois pontos de acesso que foram

fixados ao longo de uma estrada. Os protocolos da camada de transporte UDP e TCP são usados, com tamanho do pacote de 1250 bytes. Os resultados experimentais mostram que a quantidade média de dados transferidos varia de 8.8 a 3.7 Mbytes, para as velocidades no intervalo de 80 a 180 km/h, utilizando o protocolo UDP. Enquanto para o TCP a quantidade média de dados transferidos variam de 6 a 1.5 Mbytes.

Em [Rubinstein et al., 2009] são realizados experimentos práticos numa rua reta, de 400 metros de extensão, no campus da UFRJ (Universidade Federal do Rio de Janeiro). Neste cenário, os dois carros moviam-se em sentidos opostos, com velocidades que variavam de 20 a 60 km/h. A plataforma de teste era composta por computadores portáteis, de forma que um carro executava como servidor e o outro executava como cliente. Os padrões empregados para a comunicação de dados foram IEEE 802.11 a/g e a rede funcionava em modo ad hoc. Nos experimentos, analisou-se o desempenho dos protocolos UDP e TCP e a viabilidade dos dados transferidos para executar aplicações V2V. O tamanho dos pacotes UDP e TCP usados são: 150, 500 e 1460 bytes. Os resultados experimentais mostram que para as velocidades variando de 20 a 60 km/h, protocolo UDP e IEEE 802.11g, o tempo médio de contato entre os veículos variou de 45.17 a 10.83 segundos e a quantidade média de dados transferidos de 13 a 1.6 Mbytes.

Em [Neto, 2011] é apresentado um modelo matemático para prever a quantidade de dados que pode ser transferida durante um encontro em uma DTN entre um barco e uma infraestrutura fixada em terra. As embarcações são utilizadas como mula de dados, transferindo dados entre as comunidades ribeirinhas e os municípios com infraestrutura de comunicação. Neste cenário, uma embarcação de médio porte cruza um ponto de acesso fixo numa torre de 40 metros de altura, localizada na Reserva Sustentável do Tupé - Colônia do Julião, às margens do Rio Negro. [Neto, 2011] analisa a vazão útil de agregados e a potência recebida *(Received Signal Strength Indication - RSSI)* para as aplicações DTNs, usando o protocolo de transporte TCP para transferir dados durante um contato. A embarcação navegava a uma velocidade entre 10 e 30 km/h, utilizando o padrão IEEE 802.11n. A distância entre a embarcação e a torre é de aproximadamente 196 metros. Os resultados experimentais mostram que para as velocidades no intervalo de 10 a 30 km/h o volume médio de contato entre o barco e a infraestrutura fixa em terra variam de 184 a 55 Mbytes, respectivamente.

A Tabela 1 apresenta o comparativo dos trabalhos de acordo com os seguintes aspectos: veículo, tipo de via, velocidade[1], tipo de rede, tecnologia, tamanho do pacote, protocolo de transporte e a distância. Os campos preenchidos com "-" indicam que o critério não foi identificado na abordagem.

Os trabalhos de [Ott e Kutscher, 2004] e [Rubinstein et al., 2009] tratam do mesmo componente veículo/via para transferência de dados nas redes sem-fio e foram submetidos à ocorrência dos mesmos fenômenos naturais. As diferenças estão limitadas às velocidades, às redes e ao tamanho dos pacotes praticados. No trabalho de [Neto, 2011], o componente veículo/via é diferente dos trabalhos de [Ott e Kutscher, 2004] e [Rubinstein et al., 2009], estando sujeito a fenômenos físicos diferentes, como ondas, umidade e reflexão do sinal, que ocorre no rio, além da absorção do sinal causada pelas árvores na floresta. Neste artigo, é utilizado o mesmo componente

[1] 1 mph = 1.6 km/h.

**Tabela 1. Trabalhos Relacionados.**

| Parâmetros | Ott et al. | Rubinstein et al. | Neto et al. | Trabalho Proposto |
|---|---|---|---|---|
| **Veículo** | Carro | Carro | Barco | Barco |
| **Tipo de Via** | Estrada | Estrada | Rio | Rio |
| **Velocidade** | 80 a 180 km/h | 20, 40, 60 km/h | 10, 20, 30 km/h | 20, 30, 40, 50, 58 mph |
| **Tipo de Rede** | V2I | V2V | V2I | V2V |
| **Tecnologia** | IEEE 802.11b | IEEE 802.11b/a | IEEE 802.11n | IEEE 802.11b/g |
| **Tamanho do Pacote (Bytes)** | 1250 | 150, 500, 1460 | - | 150, 500, 1460, 2340 |
| **Protocolo de Transporte** | TCP, UDP | TCP, UDP | TCP | UDP |
| **Distância (m)** | - | Menor que 5 | 196 | 100 |

veículo/via usado por [Neto, 2011], mas usando velocidades, redes, protocolos de transporte, tamanho de pacote e distâncias diferentes, a fim de estabelecer transferência de dados entre as embarcações utilizando redes sem-fio padrão IEEE 802.11 b/g. O objetivo é criar um cenário mais realístico que permita avaliar a capacidade de contato e de transferência de dados entre embarcações na bacia amazônica. Além disso, é apresentado um modelo de regressão linear, obtido dos resultados experimentais, que estima a quantidade de dados transferidos a partir do tempo de contato entre as embarcações, compondo desta forma o estudo de capacidade de transferência de redes DTNs na bacia hidrográfica amazonense.

## 4. Medidas de Capacidade entre as Embarcações

Nesta seção é abordado o ambiente em que foram realizadas as medições, bem como as características do experimento e os resultados obtidos.

### 4.1. Ambiente de Medidas

Os experimentos foram realizados às margens do Rio Negro, entre as comunidades de Livramento e Nossa Senhora de Fátima, denominada Tarumãzinho, cidade de Manaus - AM. O período corresponde ao fim da cheia dos rios na região. A extensão total do rio é de 1700 km, com largura que varia de 2 a 24 km. Os barcos utilizados[2] nos experimentos são de pequeno porte (lanchas).

Ao iniciar os experimentos, as embarcações localizavam-se fora da área de cobertura uma da outra e ambas partiam no mesmo instante (t = 0) de tempo, após um contato via rádio. Os barcos deslocavam-se em sentidos opostos, paralelamente à margem do rio, cruzando-se, com velocidades contantes de 20, 30, 40, 50 e 58 mph. Desta forma, um barco movimentava-se a favor e o outro contra a correnteza do rio. Os equipamentos utilizados no experimento são: (a) *Smartphones* Nokia modelo N900, que executa a distribuição Linux - Maemo 5; (b) Roteadores D-Link modelo DIR-320; (c) Antenas externas omnidirecionais (2.4 GHz e 12 dBi); (d) Pendrives USB 2.0 de 32 GB acoplados ao roteador; (e) Circuitos reguladores de tensão (12 Volts para 5 Volts); e (f) Baterias (12 Volts). Para medir a qualidade do enlace e enviar dados aos roteadores, utilizou-se a ferramenta de geração de tráfego Iperf, na versão 2.0.4. A Tabela 2 resume os principais parâmetros utilizados no experimento.

[2]Barcos gentilmente cedidos pela Secretaria Municipal de Educação do Amazonas (SEMED-AM) e pelo Corpo de Bombeiros Militar do Amazonas (CBMAM).

**Tabela 2. Parâmetros de medição configurados.**

| Parâmetro | Valor |
|---|---|
| **Endereço IP** | Fixado |
| **ARP** | Manual |
| **ESSID** | Fixo |
| **Canal** | Fixo (1 e 6) |
| **Padrão sem-fio** | IEEE 802.11 b/g |
| **Protocolo de Transporte** | UDP |
| **Velocidade** | 20, 30, 40, 50, 58 |
| **Tamanho do Pacote (bytes)** | 150, 500, 1460, 2340 |

## 4.2. Características do Experimento

Os testes foram realizados no Rio Negro, em uma extensão de 2.5 km, local onde havia pouco tráfego de embarcações, conforme ilustrado na Figura 1. Nenhuma outra rede do padrão IEEE 802.11 foi identificada ao longo do trecho. Os fatores que podem influenciar nos resultados dos experimentos são descritos a seguir.

**As condições climáticas**: neste dia, a temperatura ambiente era de 26 ºC, umidade relativa do ar de 78%, velocidade do vento variando de 6 a 8 km/h e a velocidade da correnteza do rio de 2 km/h. Estas condições correspondem à média das condições climáticas da região.

**A altura do ponto de acesso em cada uma das embarcações**: os pontos de acesso foram colocados no ponto mais alto do barco, conhecido como *tolder*, a uma altura de 2.67 metros e na parte frontal do barco. O nó móvel *(smartphone)* estava com o usuário localizado dentro da embarcação. Vale lembrar que as embarcações utilizadas são de pequeno porte. Assim, em embarcações maiores, o ponto de acesso pode ser instalado em locais mais altos, melhorando o alcance da rede.

**A distância que as embarcações realizam o cruzamento**: a distância influencia na quantidade de dados que podem ser transferidos nas redes sem-fio, mas também na própria questão de segurança do cruzamento de embarcações. Quanto maior a distância, menor é o tempo de contato e maior é a atenuação do sinal. Com relação à segurança, estabeleceu-se que as embarcações estariam se movendo em sentidos opostos, em trajetórias paralelas separadas por uma distância de 100 metros. Caso as embarcações se movessem com alta velocidade e a distância fosse inferior a 100 metros, as embarcações poderiam tombar, devido à oscilação causada pelas ondas na superfície da água. Esse fenômeno acontece quando um barco se movimenta dentro de certa frequência e a chegada de uma onda com a frequência diferente tira a estabilidade da embarcação, fazendo com que a embarcação tombe. Portanto, esta é a distância mínima de segurança para duas embarcações se cruzarem.

Cada embarcação tinha um ponto de acesso como servidor e um *smartphone* N900 como cliente. A comunicação era realizada entre o cliente e o servidor de barcos distintos. Os clientes não se comunicavam entre si e nem mudavam de ponto de acesso. Para não causar interferência entre os canais, o que reduz a vazão da rede, utilizou-se os canais 1 e 6. Every 1 second was recorded in text file the amount of data transferred and bandwidth between client and server. O tempo de duração das transmissões do cliente é de 180 segundos. A velocidade das embarcações varia de 20 a 58 mph. O protocolo da camada de transporte UDP foi escolhido, em função de ser o mais apropriado para enlaces sem-

fio, pois o controle de congestionamento do TCP é prejudicado em enlaces com altas taxas de erro, por não diferenciar erros de transmissão de perdas por congestionamento. Três repetições foram realizadas para cada uma das velocidades das embarcações, com os respectivos tamanhos de pacote.

(a) Rio Negro. (b) Embarcações utilizadas. (c) Roteador acoplado ao barco.

(d) Coleta de Dados. (e) Visão Panorâmica. (f) Cruzamento de Embarcações.

**Figura 1. Fotos do Ambiente do Experimento.**

### 4.3. Resultados Experimentais

Nesta análise, define-se o tempo de contato como o tempo entre o primeiro e o último pacote de dados recebidos de modo correto entre as embarcações; a vazão útil, como a vazão efetiva dos dados, sem retransmissões, em nível de aplicação; a taxa de pico como a taxa máxima de dados transferidos entre as embarcações.

A Tabela 3 resume os principais resultados do experimento. Assim, com base nesta tabela é possível calcular o tempo médio de contato de cada um dos cenários. Para a velocidade de 20 mph, o tempo médio de contato é de aproximadamente 92 s. Para os demais cenários, verificou-se que o tempo de contato permaneceu inversamente proporcional à velocidade relativa das embarcações, como era esperado. Por exemplo, ao dobrar a velocidade, de acordo com a Tabela 3, o tempo médio de contato para velocidade de 40 mph é de 43 segundos, enquanto que ao praticamente triplicar a velocidade para 58 mph, obteve-se um valor de quase um terço do tempo de contato, com 29 segundos. Vale ressaltar que o trecho do percurso no qual as embarcações estão dentro do alcance de rádio é aproximadamente o mesmo independente da velocidade, aproximadamente 760 metros.

As medições mostram a relação entre a velocidade da embarcação, o tamanho do pacote enviado e a massa de dados transferida. Na análise estatística, usamos o desvio padrão (denotado por $\sigma$) e observamos que as variações de dados transferidos e de vazões úteis não são representativas nas amostras. A causa da variabilidade dos dados é decorrente da redução do tempo de contato, ocasionado pelo aumento de velocidade da

**Tabela 3. Média: Tempo de Contato, Dados Transferidos e Vazão Útil entre duas embarcações com UDP e padrão IEEE 802.11 b/g.**

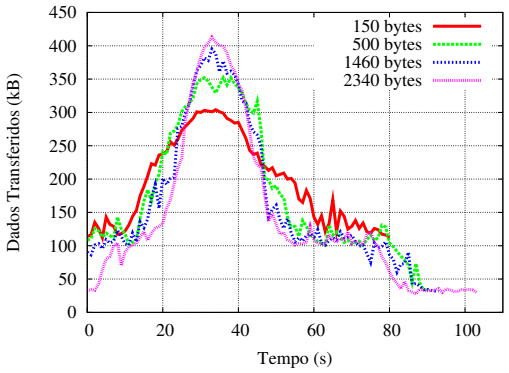
| Velocidades | Tamanho do Pacote (Bytes) | Tempo de Contato (Segundos) | Dados Transferidos (Mbytes) | Vazão Útil (Mbps) |
|---|---|---|---|---|
| **20 mph** | **150** | 80 ($\sigma = 4.58$) | 10.15 ($\sigma = 0.82$) | 1.96 ($\sigma = 0.03$) |
| | **500** | 90 ($\sigma = 3.46$) | 11.55 ($\sigma = 1.32$) | 1.68 ($\sigma = 0.03$) |
| | **1460** | 93 ($\sigma = 7.81$) | 13.77 ($\sigma = 1.72$) | 1.69 ($\sigma = 0.05$) |
| | **2340** | 103 ($\sigma = 8.54$) | 15.49 ($\sigma = 0.82$) | 1.91 ($\sigma = 0.04$) |
| **30 mph** | **150** | 55 ($\sigma = 3.61$) | 6.69 ($\sigma = 0.21$) | 0.77 ($\sigma = 0.04$) |
| | **500** | 61 ($\sigma = 2.65$) | 7.34 ($\sigma = 0.12$) | 1.12 ($\sigma = 0.05$) |
| | **1460** | 56 ($\sigma = 5.29$) | 7.44 ($\sigma = 0.04$) | 1.16 ($\sigma = 0.02$) |
| | **2340** | 55 ($\sigma = 7.81$) | 7.52 ($\sigma = 0.25$) | 1.13 ($\sigma = 0.05$) |
| **40 mph** | **150** | 43 ($\sigma = 2.65$) | 5.80 ($\sigma = 0.14$) | 1.66 ($\sigma = 0.09$) |
| | **500** | 40 ($\sigma = 3.61$) | 6.12 ($\sigma = 0.21$) | 1.91 ($\sigma = 0.03$) |
| | **1460** | 39 ($\sigma = 5.29$) | 6.36 ($\sigma = 0.14$) | 1.70 ($\sigma = 0.06$) |
| | **2340** | 50 ($\sigma = 3.46$) | 7.81 ($\sigma = 0.14$) | 1.93 ($\sigma = 0.02$) |
| **50 mph** | **150** | 33 ($\sigma = 1.73$) | 3.78 ($\sigma = 0.15$) | 1.19 ($\sigma = 0.02$) |
| | **500** | 28 ($\sigma = 3.61$) | 4.20 ($\sigma = 0.05$) | 1.25 ($\sigma = 0.06$) |
| | **1460** | 30 ($\sigma = 4.36$) | 4.22 ($\sigma = 0.12$) | 1.41 ($\sigma = 0.03$) |
| | **2340** | 39 ($\sigma = 7.55$) | 4.37 ($\sigma = 0.17$) | 1.40 ($\sigma = 0.04$) |
| **58 mph** | **150** | 29 ($\sigma = 8.00$) | 3.13 ($\sigma = 0.05$) | 1.48 ($\sigma = 0.09$) |
| | **500** | 22 ($\sigma = 4.36$) | 3.23 ($\sigma = 0.08$) | 1.84 ($\sigma = 0.06$) |
| | **1460** | 30 ($\sigma = 8.54$) | 3.47 ($\sigma = 0.06$) | 1.54 ($\sigma = 0.06$) |
| | **2340** | 32 ($\sigma = 11.14$) | 3.73 ($\sigma = 0.06$) | 1.72 ($\sigma = 0.04$) |

embarcação. Tratando-se especificamente das altas velocidades (50 e 58 mph), o desvio padrão do tempo de contato entre as embarcações é um pouco mais acentuado, devido ao balanço de proa a popa produzido pela agitação do rio, o balanço da embarcação, a marola (pequenas ondas) e os ventos fortes que as embarcações e os equipamentos utilizados no experimento estão sujeitos quando sobem e descem o leito do Rio Negro.

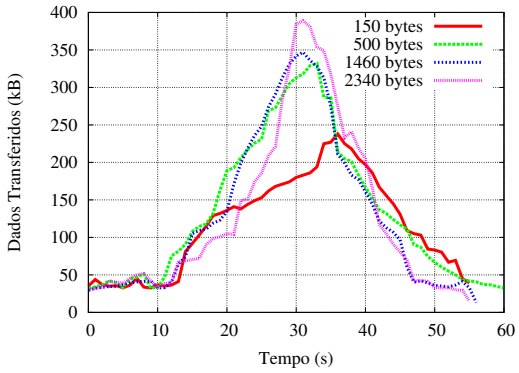
A Figura 2 revela a quantidade média dos dados transferidos entre as embarcações em função do tempo, movimentando-se com velocidades constantes. A partir dos resultados, percebe-se que a velocidade, além de reduzir o tempo de contato, tem influência na taxa de pico, visto que ao dobrar a velocidade (Figuras 2(a) e 2(c)), houve uma queda de aproximadamente 20% na taxa de pico. Esta característica contribui ainda mais para a redução na capacidade de transferência de dados entre duas embarcações em velocidades mais altas.

É interessante observar que a maior parte dos dados são transferidos antes da metade do tempo de contato, isto é, o pico está levemente deslocado para a esquerda. Por exemplo, para velocidades de 50 mph e 58 mph, Figuras 2(a) e 2(c) respectivamente, 64% dos dados são transferidos até a metade do tempo de contato. Tal fato pode ser explicado pelo modo de operação com múltiplas taxas do 802.11. A taxa de transmissão, que começa no máximo no início do contato, é rapidamente reduzida à taxa mínima e aumentada progressivamente conforme as duas embarcações se aproximam. Neste trecho, enquanto a taxa está aumentando, significa que todos os quadros enviados estão chegando corretamente. Após o cruzamento, as embarcações começam a se afastar e a taxa do 802.11 começa a ser reduzida de acordo com as perdas dos quadros transmitidos. Neste trecho, uma parte dos quadros é perdida na tentativa de adaptar para taxa de transmissão mais adequada. Este comportamento já foi verificado em experimentos com carros [Rubinstein et al., 2009].

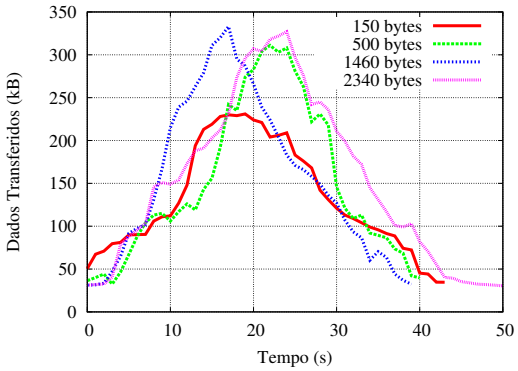
Outro resultado interessante pode ser observado a partir da Figura 2. Clara-



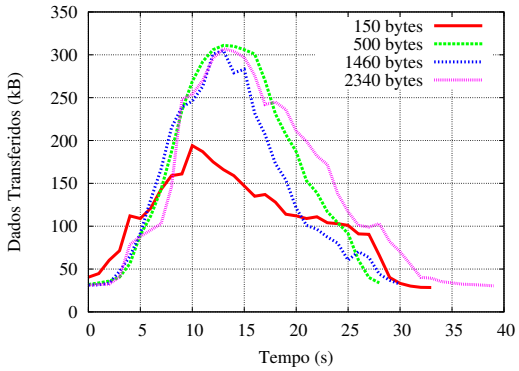

(a) Média de dados transferidos a 20 mph.



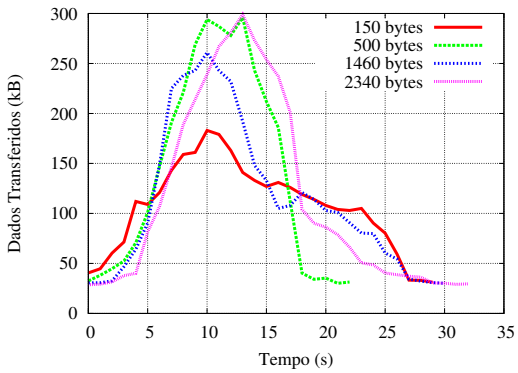

(b) Média de dados transferidos a 30 mph.




(c) Média de dados transferidos a 40 mph.




(d) Média de dados transferidos a 50 mph.




(e) Média de dados transferidos a 58 mph.

**Figura 2. Transferência de dados sobre o UDP entre duas embarcações se cruzando a diferentes velocidades.**

mente, existe uma região de maior capacidade de transferência de dados entre as duas embarcações que se cruzam, independente da velocidade. Esta região é definida basicamente pela distância entre as embarcações e ela é localizada em torno da região de pico. Isto significa que a precisão no tempo de contato é menos importante, isto é, se o tempo de contato é de 92 ou 85 segundos, faz pouca diferença, pois o mais importante é saber que o barco teve 65 segundos dentro da região de maior capacidade de transferência. Pelos gráficos é possível identificar tal região para cada uma das velocidades. Na velocidade de 20 mph, a região de maior capacidade que representa aproximadamente 76% do tempo de contato é responsável por 90% da transferência de dados. Para velocidade de 30, 40, 50

e 58 mph, a região reponsável por 90% da capacidade de transmissão representa aproximadamente 70%, 70%, 76% e 75% do tempo de contato, respectivamente.

Este resultado é importante pois permite uma margem maior de erro nos tempos de contato. No caso das redes hidroviárias, existem muitos parâmetros que podem influenciar a capacidade de transmissão, como descrito na Seção 4.2. Portanto, esta margem maior de erro propicia contemplar pequenas variações nestes parâmetros sem perder muita precisão no cálculo da capacidade. Em consequência, este resultado permite também generalizar os resultados obtidos para obter uma estimativa da capacidade da rede como um todo, como será apresentado na próxima seção.

## 5. Estudo de Capacidade das Redes DTNs

O estudo da capacidade nas redes DTNs na bacia hidrográfica da Amazônia Ocidental foi realizado a partir de dados reais de embarcações obtidos na AHIMOC, combinados com as medições experimentais realizadas no tráfego de dados entre as embarcações no Rio Negro. Nesse estudo, foram consideradas as embarcações que realizam o transporte de carga, transporte misto (carga e passageiros) e as lanchas (Ajato). Para cada uma das embarcações obteve-se as seguintes informações: nome e tipo de embarcação, a origem e o destino, a data e o horário de ida e retorno da embarcação, o tempo estimado de viagem, as distâncias percorridas, a frequência de viagem e o sentido que as embarcações trafegam ao longo das calhas dos rios. As calhas (Figura 3) são importantes vias de transporte social e econômico para o estado do Amazonas, pois são por meio delas que são feitas as ligações entre os municípios e por onde são escoados os produtos produzidos no Pólo Industrial de Manaus (PIM). As calhas analisadas no estudo de capacidade estão descritas na Tabela 4.

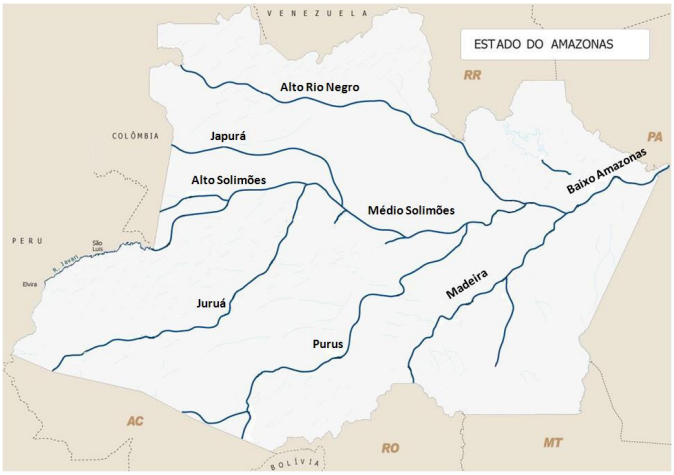



**Figura 3. Calhas Fluviais da Amazônia Ocidental.**

A cidade de Manaus (metrópole regional) foi escolhida como marco zero. Calculou-se o quantitativo de embarcações por calha, as velocidades das embarcações, o tempo de contato e a distância percorrida. Além disso, foi analisada a relação entre o tempo de contato e a quantidade de dados transferidos, resultando no modelo de regressão linear simples. Com base no modelo é calculada a quantidade de dados transferidos e apresentados os resultados do estudo de capacidade de redes DTNs na bacia hidrográfica amazonense.

**Tabela 4. Principais Calhas Fluviais da Amazônia Ocidental.**

| Calhas | Rotas |
|---|---|
| **Alto Rio Negro** | Manaus - São Gabriel da Cachoeira |
| **Alto Solimões** | Manaus - Tabatinga |
| **Baixo Amazonas** | Manaus - Santarém - Belém |
| **Juruá** | Manaus - Eirunepé |
| **Madeira** | Manaus - Porto Velho |
| **Médio Solimões** | Manaus - Coari - Tefé |
| **Purus** | Manaus - Boca do Acre |

### 5.1. Modelo de Tempos de Contato

Foi criado um modelo para o cálculo dos tempos de contatos entre as embarcações da rede analisada. Os parâmetros do modelo são descritos a seguir:

a) Quantitativo de embarcações ($Q$): refere-se ao somatório de embarcações que trafegam nas calhas dos rios. Portanto, $Q = \sum_{i=1}^{N} Q_i$, onde $Qi$ é o número de embarcações na calha $i$ e $N$ o número de calhas.

b) Velocidade escalar média das embarcações ($Vm$): relação entre a distância percorrida pela embarcação e o tempo. Portanto, $Vm = \frac{\Delta s}{\Delta t}$, onde $\Delta s$ é a distância fluvial da cidade de Manaus ao município e $\Delta t$ o tempo estimado de viagem entre as cidades.

c) Tempo de Contato ($Tc$) entre duas embarcações: definido na Seção 4.3, pode ser calculado a partir da equação do Movimento Uniforme (MU), desde que as embarcações desloquem-se com velocidade constante. Portanto, $S = So + Vt, (V \neq 0)$, onde $So$ é a posição da embarcação no instante $t = 0$ e $V$ é a velocidade escalar do movimento. A inequação 1 fornece a condição que os barcos estão dentro da área de alcance. Dadas as equações $S_a$, $S_b$ e o alcance, obtêm-se os tempos $t_1$ e $t_2$ nos quais as embarcações entram e saem do alcance, e então, $Tc$ = $t_2$ - $t_1$.

$$|S_a - S_b| \leq 0.757, \tag{1}$$

onde $S_a$ e $S_b$ são as equações da posição das embarcações $A$ e $B$, respectivamente. Baseando-se na média aritmética dos alcances calculados nos experimentos, será considerado que as embarcações podem se comunicar enquanto a distância entre elas for menor que 0.757 km.

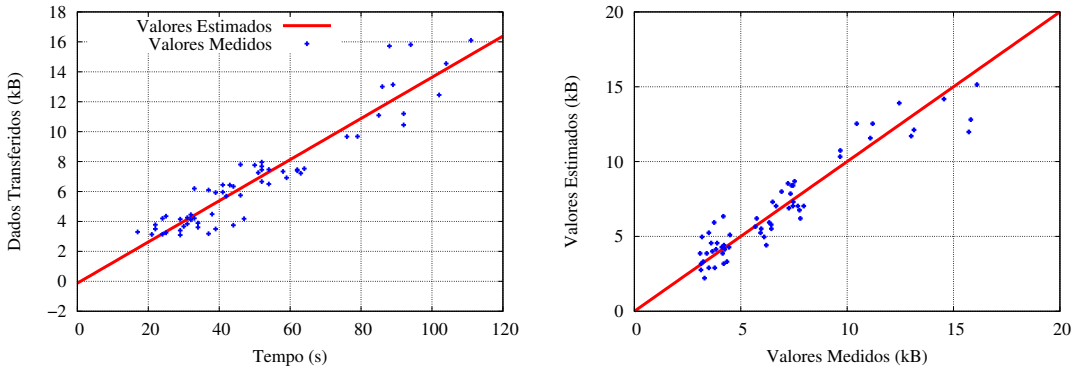
d) Distância total percorrida pelas embarcações ($Dist$): refere-se ao somatório das distâncias percorridas pelas embarcações nas calhas dos rios. Portanto, $Dist = \sum_{i=1}^{N} Dist_i$, onde $Dist_i$ é a distância percorrida pela embarcação em cada calha e $N$ o número de calhas.

### 5.2. Relação entre Tempo de Contato e Quantidade de Dados Transferidos

A partir dos tempos de contato e dados transferidos nas três medições realizadas nos experimentos de cruzamento entre as embarcações no Rio Negro, utilizou-se a técnica estatística de regressão linear [Yan e Su, 2009] para analisar as amostras, e foi construída a reta de ajuste $f(t)$ = a * $t$ + b, obtendo-se os valores a = 0.13 e b = - 0.12 (Equação 2). Essa equação descreve o comportamento dos valores estimados pela reta de regressão com os valores medidos, ilustrados na Figura 4(a). Assim, a quantidade de dados transferidos em função do tempo de contato pode ser estimada pela função:

$$f(t) = 0.13 * t - 0.12, \quad (2)$$

onde $t$ é o tempo de contato em segundos e $f(t)$ a quantidade de dados transferidos entre as embarcações em megabytes.




(a) Dados Transferidos x Tempo de Contato. (b) Gráfico de Resíduos.

**Figura 4. Avaliação do Modelo.**

Neste artigo, para medir a acurácia do modelo foram utilizadas as métricas: *Root Mean Square Error (RMSE)*, *R-Squared* ($R^2$) e o Gráfico de Resíduos [Hellerstein et al., 2004]. A métrica *RMSE* relaciona as diferenças individuais entre o vetor de valores estimados $y(k+1)$ e os valores reais $\hat{y}(k+1)$ de dados transferidos entre embarcações. O valor do RMSE obtido foi de 1.14, considerado como baixa magnitude de erro. A métrica $R^2$ quantifica a variação da saída capturada pelo modelo, no intervalo de 0 (pior) a 1 (melhor). Um $R^2 \geqslant 0.8$ [Yan e Su, 2009] é considerado um bom valor para o estimador. O valor encontrado para o modelo foi $R^2$ = 0.89.

A Figura 4(b) mostra o Gráfico de Resíduos que representa a saída prevista pelo modelo como uma reta e os valores reais obtidos no experimento como pontos. Num modelo perfeito, os pontos experimentais sempre estarão sobre a curva no Gráfico de Resíduos. Analisando a Figura 4(b), observa-se que os pontos experimentais estão próximos da curva.

### 5.3. Estudo de Capacidade das Redes DTNs em Relação às Calhas

As calhas mais importantes economicamente para o estado são a do Baixo Amazonas, Madeira, Alto e Médio Solimões, pois são as responsáveis pelo transporte das mercadorias do PIM. Nas calhas do Baixo Amazonas e do Madeira são escoados os produtos para abastecimento do país e as calhas do Alto e Médio Solimões são responsáveis pela exportação para outros países. Já a calha do Médio Solimões é caracterizada pela grande quantidade de embarcações, isto se deve à predominância de petróleo encontrado na região. As demais calhas são menos povoadas por embarcações por serem menos representativas economicamente.

Na Tabela 5, a quantidade de contatos e os tempos de contatos foram contabilizados a partir da combinação de todos os pares possíveis de embarcações que trafegam nas calhas, de acordo com a inequação 1. De posse dos tempos de contato, substitui-se esse tempo na equação 2, obtendo-se os dados transferidos $f(t)$. Tal procedimento feito por

calha e somado os valores resulta em 93 GB de capacidade máxima da rede DTN numa janela de tempo semanal. Ou seja, Dadas as 357 embarcações foram calculados 4979 contatos entre elas, realizando o cruzamento, com o tempo de contato total de 203.08 horas, quantidade de dados transferidos de 93 GB e distância percorrida entre as cidades de 204332 Km. Neste cenário, as embarcações que realizavam ultrapassagem não foram consideradas no cálculo, o que significa que a presente estimativa é conservadora.

**Tabela 5. Estudo de Capacidade de Redes DTNs na Bacia Hidrográfica Amazonense.**

| Calhas | Quantidade de Embarcações | Quantidade de Contatos | Tempo Total dos Contatos (Horas) | Total de Dados Transferidos (GB) | Distância Total Percorrida pelas Embarcações (km) |
|---|---|---|---|---|---|
| **Alto Rio Negro** | 17 | 70 | 2.34 | 1.07 | 47470 |
| **Baixo Amazonas** | 90 | 1364 | 45.32 | 20.71 | 10906 |
| **Madeira** | 23 | 126 | 5.34 | 2.44 | 16948 |
| **Purus** | 22 | 96 | 5.70 | 2.60 | 17199 |
| **Médio Solimões** | 59 | 831 | 33.79 | 15.44 | 23970 |
| **Alto e Médio Solimões** | 77 | 1431 | 62.32 | 28.48 | 50004 |
| **Juruá e Médio Solimões** | 69 | 1061 | 48.27 | 22.26 | 37835 |

O tempo de contato e a quantidade de dados transferidos medidos neste trabalho são maiores do que no trabalho de [Rubinstein et al., 2009]. A relação entre tempo médio de contato medido neste trabalho ($v = 64$ km/h) com o medido em [Rubinstein et al., 2009] ($v = 60$ km/h) é de 3.4. Da mesma forma, a relação entre a quantidade média de dados transferidos é de 3.8. Entretanto, os experimentos de [Rubinstein et al., 2009] estão sujeitos à interferência de outras redes constatadas no local e não foram utilizadas antenas externas nos veículos, enquanto que nas embarcações do presente trabalho foram utilizadas antenas externas e não existia nenhuma outra rede na região. Estes fatores pesaram mais que o balanço das embarcações e a transmissão sobre o espelho d'água, fatores que prejudicam a comunicação no ambiente hidroviário.

## 6. Conclusão e Trabalhos Futuros

Este trabalho apresentou uma análise experimental do desempenho da capacidade de comunicação entre as embarcações no Rio Negro, utilizando o padrão IEEE 802.11 b/g. Com as velocidades praticadas pelas embarcações na região mostra-se a transferência de dados em função do tempo de contato. Desta forma, é possível o desenvolvimento de aplicações e serviços que se utilizem do contato entre as embarcações que fazem viagens aos municípios amazonenses, produzindo desenvolvimento e modernização ao estado. Além disso, avaliamos a capacidade de transferência de dados entre as embarcações nas principais calhas do estado do Amazonas. Os resultados mostram números razoáveis, com potenciais de transferência da ordem de gigabyte durante uma janela de tempo de uma semana.

Como trabalhos futuros, pretende-se avaliar a capacidade de transmissão de dados do protocolo IEEE 802.11p entre as embarcações na calha do rio e variar a distância entre as embarcações, analisando a quantidade de dados transferidas durante o contato e o tempo de contato, por meio de experimentos práticos e simulações.

## 7. Agradecimentos

Trabalho realizado com os recursos da CAPES, CNPq, Fapeam, Faperj, Fundação Muraki e da GE Centro Brasileiro de Pesquisas. Os autores agradecem pela contribuição neste trabalho a Miguel Elias Mitre Campista e a Eliézer Passos de Moura, e aos órgãos SEMED-AM e CBMAM que cederam as lanchas para a realização dos experimentos.

## Referências

Alves, R. S. A., Campbell, I. V., Couto, R. S., Campista, M. E. M., Moraes, I. M. e Rubinstein, M. G. (2009). *Redes Veiculares: Princípios, Aplicações e Desafios*. Em Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC, p. 199-254.

Daher, R. e Vinet, A. (2012). *Roadside Networks for Vehicular Communications: Architectures, Applications, and Test Fields 1st*. Premier Reference Source. IGI Global, 1ª edição.

Demmer, M. e Fall, K. (2007). Dtlsr: Delay tolerant routing for developing regions. Em *Proceedings of the 2007 Workshop on Networked Systems for Developing Regions*, NSDR '07, p. 5:1–5:6, New York, NY, USA. ACM.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. Em *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, p. 27–34, New York, NY, USA. ACM.

Hellerstein, J. L., Diao, Y., Parekh, S. e Tilbury, D. M. (2004). *Feedback Control of Computing Systems*. John Wiley & Sons, 1ª edição.

Moraes, R. (2013). *Na Planície Amazônica*. Editora Conquista - 1960. Itatiaia, 7ª edição.

Neto, F. A. C. e Pimentel, N. (2010). Perfil Econômico do Município de Manaus. Relatório Técnico CDU: 338.2 (811.), Secretaria de Estado de Planejamento e Desenvolvimento Econômico - SEPLAN.

Neto, J. B. P. (2011). Um Modelo para Previsão do Volume de Contato em Redes Tolerantes a Atrasos e Desconexões: Uma Abordagem Quantitativa. Tese de mestrado, Universidade Federal do Amazonas, Manaus, Amazonas. Departamento de Ciência da Computação.

Ott, J. e Kutscher, D. (2004). Drive-thru internet: IEEE 802.11b for automobile users. Em *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, p. –373.

Rubinstein, M., Ben Abdesslem, F., Dias de Amorim, M., Cavalcanti, S., Dos Santos Alves, R., Costa, L., Duarte, O. e Campista, M. (2009). Measuring the capacity of in-car to in-car vehicular networks. *Communications Magazine, IEEE*, 47(11):128–136.

Souza, M. H. (2009). Contribuição Metodológica para Localizar Terminal de Integração de Passageiros do Transporte Hidro-Rodoviário Urbano. Tese de mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ. Departamento de Engenharia de Transportes.

Yan, X. e Su, X. G. (2009). *Linear Regression Analysis: Theory and Computing*. World Scientific. World Scientific, 1ª edição.

# Applying Machine Learning to Reduce Overhead in DTN Vehicular Networks

**Lourdes P. Portugal-Poma**[1] **, Cesar A. C. Marcondes**[1]**, Hermes Senger**[1]**, Luciana Arantes**[2]

[1]Departamento de Computação – Universidade Federal de São Carlos (UFSCAR)
Caixa Postal 676 13565–905 São Carlos–SP

[2]LIP6 – Université de Paris 6 - CNRS - INRIA
4, Place Jussieu 75005, Paris - France

{lourdes.poma,marcondes,hermes}@dc.ufscar.br, luciana.arantes@lip6.fr

***Abstract.*** *VANETs benefit from Delay Tolerant Networks (DTNs) routing algorithms when connectivity is intermittent because of the fast movement of vehicles. Multi-copy DTN algorithms spread message copies to increase the delivery probability but increasing network overhead. In this work we apply machine learning algorithms to reduce network overhead by discriminating the worst intermediate nodes for the transmission of copies. The scenario is a VANET of public buses that follow specific routes and schedules. This repetitive behavior creates an opportunity for applying trained classifiers to predict the occurrence of performance-related events. As the main contribution, our method decreases overhead without degrading delivery probability.*

## 1. Introduction

Vehicular Ad Hoc Networks (VANETs) are Mobile Ad Hoc networks (MANETs) of vehicles establishing wireless communications. Like other MANETs, VANETs are networks with dynamic topology where nodes can be servers or clients by executing routing algorithms in distributed environments. In addition, VANETs are a special type of MANETs because of the fast movement of its nodes, which creates intermittent end-to-end connectivity paths. Thus, the maintenance and the construction of routing tables in topology based routing algorithms, becomes harder to realize, specially in proactive algorithms that reconstruct periodically the whole routing table [Abolhasan et al. 2004].

To resolve intermittent end-to-end connectivity, VANETs benefit from Store Carry and Forward (SCF) paradigm. This paradigm is the foundation of the Delay Tolerant Networks (DTNs) [Vasilakos et al. 2011], being an approach to deliver messages in computer networks without establishing connectivity paths between source and target nodes. In the SCF, messages are forwarded hop-by-hop in opportunistic contacts between nodes, i.e., when pairs of nodes establish a wireless communication between them. When there is no other neighbor node to receive a message, it remains stored and waiting for a next opportunity to transmit. For this reason, there is a trade-off between: (i) storage resources versus packet loss ratio, and (ii) delay versus delivery probability. However, in challenging scenarios (such as sparse networks), nodes could deliver more messages using the SCF paradigm than using topology based routing algorithms, but

Authors thank to STIC-AMSud and CAPES for their support in the development of this work. Hermes Senger thanks to FAPESP (contract # 2014/00508-7) and CNPQ.

at cost of a considerable increase in delay due to the additional message storage delay [Franck and Gil-Castineira 2007].

In the SCF paradigm, an important issue is the selection of the next intermediate node to whom transmit a message (next hop). This decision is made in every node and based on local information. A multi-copy solution consists in creating and spreading message copies to increase the delivery probability in applications where the target node's location is unknown. However, the flooding of copies could result in network overhead and, depending on the network density and the spatial node distribution, it can induce strong performance degradation.

In this work we apply machine learning algorithms to reduce network overhead by discriminating the worst intermediate nodes for the transmission of message copies in multi-copy DTN routing. The experiment scenario is composed of one VANET of public buses that follow specific routes and schedules. This repetitive behavior allows trained classifiers to predict the occurrence of performance-related events. In summary, our work makes the following contributions:

- We proposed a method that uses a classifier in order to support the decisions of a routing algorithm and to adapt its behavior according to local conditions and the quality of the intermediate nodes. Our method decreases network overhead without degrading the probability of message delivery;
- We evaluated our method with realistic scale traces (254 vehicles in average). The latter revealed zones of highly varied density of vehicles. This behavior is not usually captured by mobility synthetic models;
- We compared the connectivity characteristics between synthetic mobility models and the real traces to highlight the influence of the choice of a model in the overhead evaluation. The result shows that synthetic mobility models can render some bias in the overhead evaluation;
- We conducted performance evaluation experiments of our method with two algorithms whose design has different purposes: Spray and Wait (SaW) [Spyropoulos et al. 2005] is an algorithm with low overhead and good performance in scenarios where nodes move freely, and Epidemic [Vahdat and Becker 2000] is a good algorithm to be used in sparse and low density networks.

The remainder of this paper is organized as follows: section 2 presents different approaches for reducing the overhead in multi-copy routing. The section also presents a discussion on how classifiers are applied in MANETs. Our proposed method is described in section 3 while, in section 4, we highlight the main characteristics observed in scenarios with real traces. In section 5, we discuss some experimental results related to the evaluation of the proposed method with both SaW and Epidemic routing algorithms. Finally, in section 6 we conclude the paper and present suggestions for future work.

## 2. Related Work

Our Machine Learning Classification (+C) approach is related to works of different areas. We start by describing from naive DTN routing algorithms, like Epidemic, up to decision tree based classifiers, which we use in our proposed method.

**Naive multi-copy routing algorithms.** Epidemic is the fundamental multi-copy, high overhead, routing algorithm [Vahdat and Becker 2000]. If applied without any control, nodes transmit in every opportunity to others that do not have a copy, like an epidemic disease. Despite of that, the overhead can be small and delay / delivery performance better in low density scenarios and sparse networks.

**Naive algorithms with flood control.** In order to reduce network overhead, basic flooding control measures can be applied. Some approaches aim at limiting the time to live of messages (TTL), the number of hops, or tuning the message space storage. However, implementing these approaches require intense parameter tuning, not always trivial to implement due to DTNs' dynamic behavior. Some other approaches are inspired in the analogy of vaccines, where once an "anti-packet" is received it induces the node to reject or delete a copy of an specific message [Haas and Small 2006]. Finally, Spray And Wait (SaW) [Spyropoulos et al. 2005] is another solution which sets a maximum number of copies per original message that can exist in the entire network . This value is decreased at every copy transmission. When it equals to one, nodes only transmit directly to the target node. In summary, the goal of all the above solutions is to reduce network overhead without any other consideration about other performance issues. Consequently, the performance of these algorithms is quite dependent on the mobility model and nodes' spatial distribution.

**Statistical and context information algorithms.** Other solutions exploit statistical and context information to select the best intermediate nodes. Nodes use utility functions to qualify intermediate nodes and to decide if they should transmit a copy or not. Commonly, mobility statistics and connectivity information is used since the latter has an influence on performance. The knowledge of network connectivity properties and the qualification of nodes allows the design of more effective routing protocols for specific environments [Vendramin et al. 2012]. For example, the Lobby index ($L_{indx}$) [Korn et al. 2009] is a two-hops connectivity metric suggested by [Pallis et al. 2009] to qualify nodes. It detects dense zones where highly connected nodes are located. To compute the $L_{indx}$ in a network, every neighbor from a node $x$ reports to $x$ its connectivity degree. Then, a node $x$ calculates the maximum integer $l$ such that $x$ has at least $l$ neighbors with a value of degree greater than or equal to $l$. For $L_{indx}$ to be applied, an additional and exclusive channel for beaconing transmission is needed.

**Node encounter probabilities algorithms.** ProPhet [Lindgren et al. 2003] and MaxProp [Burgess et al. 2006] are algorithms that employ utility functions to take advantage of information about node encounter probabilities. However, besides mobility information, nodes can collect other kind of information to find network behavior patterns. For instance, the time interval of the day could be related with the network load. Thus, machine learning algorithms were used with the purpose of detecting convenient opportunities for message transmission according to the learned transmission patterns. Basically, machine learning based approaches consist in collecting data of transmissions, processing the data conveniently and use a trained classifier to support future transmission decisions.

**Classification algorithms.** Classification consists in attributing objects to one of many categories or classes. Formally, according to [Tan et al. 2005], classification is the task of learning an objective function $f$ that maps each set of attributes to one of the prede-

fined classes. Thus, a classifier constructs a classification model from a dataset (training dataset) using a machine learning algorithm. Given a set of $n$ attributes, $\{a, b, \cdots, n\}$, and a set of $m$ classes, $\{C_1, C_2, \cdots, C_m\}$, a training dataset is a set of tuples where each tuple corresponds to:

$$\underbrace{a_{value},\ b_{value},\ \cdots, n_{value}}_{Attribute\ values} \quad \overrightarrow{then} \quad \underbrace{C_i}_{Class\ value} \quad where\ 1 \leqslant i \leqslant m \qquad (1)$$

Using the classification model, the trained classifier assigns a class to an unannotated tuple which has only a set of attribute values, $\{a_{value},\ b_{value},\ \cdots, n_{value}\}$. Also, the classifier must have the capacity of generalization, this mean that the classifier must assign a class even for unannotated tuples that were not seen in the training dataset.

**Routing and broadcasting using classifiers.** There are still few approaches that suggest the use of classifiers to support routing and dissemination decisions. [Colagrosso 2005] use the Naive Bayes classifier to be trained with information of past broadcast retransmissions for message dissemination. Afterward, the classifiers predict if a new broadcast attempt will be successful or not. Classifiers were also used to find mobility patterns from people movement to construct predictive models to forecast: "where", "how long will stay" and "who people will meet" people in different locations [Vu et al. 2011]. [Ahmed and Kanhere 2010] exploit the repetitive behavior of public buses movement to develop an extensible Framework and a single-copy DTN solution. As a matter of fact, this is the first work in DTN routing that use classifiers to improve routing decision, and it is the most related work to our proposal. It consist in keep a resident Naive Bayes classifier that uses information of end-to-end past deliveries (thus, acknowledgment propagation is required) to determine the most likely nodes that will deliver a message to a specific target node. This scheme outperforms a single-copy version of MaxProp. However, when compared with Epidemic, its performance is lower in delivery probability and delay. The reason for that is that in scenarios of low to moderate density Epidemic flooding has usually the best performance in this metrics, but with high network overhead.

**Decision tree classification algorithms.** The C4.5 decision tree [Quinlan 1993] is a machine learning algorithm used for classification. The C4.5 training phase generates a decision tree using training data. The decision tree is composed by: (i) internal nodes that represent conditional operations on the attributes and (ii) terminal nodes associated to class attributes. Initially, all the tuples are attributed to the root node of the decision tree. The next step consists in finding a conditional operation based on one of the tuples' attributes. This operation divides the root node into two or more internal nodes. The new internal nodes contain a subset of the root node's tuples. This division operation is repeated recursively until the generated nodes are pure. The purity of a node is a measure of the homogeneity of its tuples' class distribution. In terminal nodes, the most frequent class determines the class attribute. To choose the attribute and the conditional operation applied to it, all the possible options are evaluated and the one that maximizes the purity of the resulted subsets is used. With all in mind, the merit of the C4.5 comes from [Tan et al. 2005]: (i) it makes a automatic selection of the best attributes to create the structure of the decision tree and (ii) the decision tree is a comprehensible form to visualize the attributes's influence in the classification of tuples. J4.8 is the latest and slightly

improved version, called C4.5 revision 8, implemented in Weka [Witten et al. 2011]. J4.8 was the last public version of this family of algorithms before the commercial C5.0 was released. The full complexity of decision tree induction is $O(\alpha N(lgN))$, where $\alpha$ is the number of attributes and $N$ is the number of instances [Witten et al. 2011].

## 3. Applying Data Classification for Improving Routing Decisions

Our proposal is to apply data classification in order to predict which nodes are the most suitable to behave as intermediate forwarders according to temporal and local connectivity conditions. Our method is composed of three phases, as illustrated in Figure 1. During the first phase, data about message forwarding is collected by all the network nodes. In the second phase, all data collected from the nodes is combined into a single training dataset at some machine with enough capacity (such as a server in a datacenter or cloud provider). This could be done in the end of the workday. Thus, the collected data could be stored and processed incrementally day after day to provide richer information to retrain the classifier. As a result of this, the trained classifier produced by J4.8, i.e., the decision tree built during the classifier training, can be represented using a few dozens of bytes which are then transmitted for every mobile routing node. In the last phase of our method, each mobile node uses the received classifier to predict if neighboring nodes are good forwarders and decide whether to transmit a message copy or not. The classification of each message is very fast, being computed in time $O(h)$, where $h$ is the height of the decision tree.

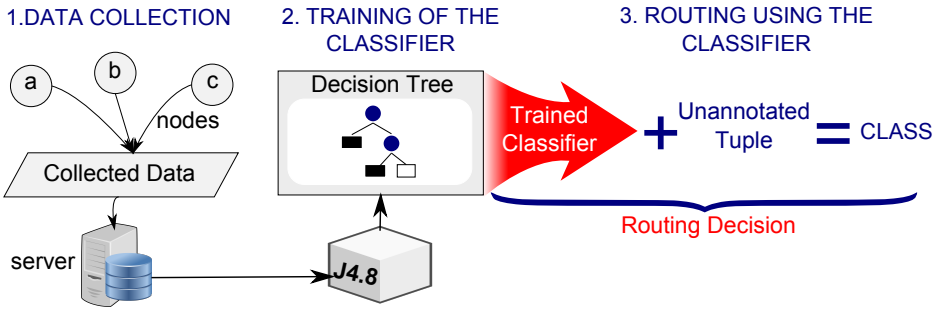



**Figure 1. Phases of the method: (1) Data collection; (2) The classifier is trained with the dataset; (3) Routing is executed based on the classifier's knowledge**

### 3.1. Data collection

The goal of the data collection phase is to store information about message transmissions. This data will be used to train the classifier. Our method organizes the collected data as tuples. Each tuple corresponds to data from a single *message replication*, where message replication is defined as follows:

**Definition 1** *A message replication is an event composed by the reception of a copy of a message $m$ by an intermediate node $n$ and the successful transmission (by $n$) of a new copy of $m$ to another intermediate node $n'$.*

Each tuple $t = \{\eta, \theta, \gamma, l, \tau, \delta\}$ is composed by the following message replication attributes: the **Node ID** ($\eta$) contains the identifier of the node that received the message copy; **Region code** ($\theta$) contains the location of the node at the instant when it received

the message copy; the **Reception time** ($\gamma$) contains a timestamp of the instant when the message copy was received; **Lobby index** ($l$) measures the neighborhood density at two-hops in the moment when the message copy was received; **Time passed** ($\tau$) represents the time interval between the reception time and the successful transmission of the message copy the next node; **Distance** ($\delta$) refers to the distance between the location when the message copy was received and the location when the message copy was transmitted to the next node.

The attributes collected for each message replication are illustrated in Figure 2. To compute the region code attribute we divided the total movement area in squares of $1km \times 1km$. For the reception time attribute, time was discretized into 600s intervals. Our data collection strategy does not require end-to-end acknowledgment. Instead, our method is able to predict suitable intermediate nodes based only on local conditions, without considering the relation between intermediate and target nodes. This is important because end-to-end acknowledgement is a costly procedure in VANETs.

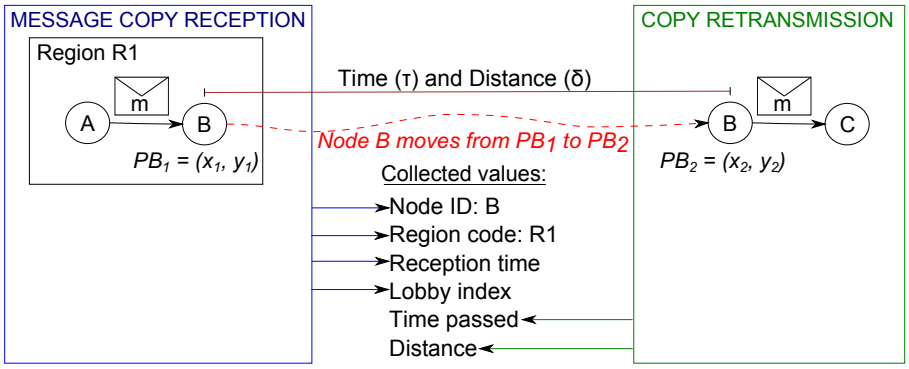



**Figure 2. Data collection. It starts when $B$ receive a copy of message $m$ and it ends when B retransmits a copy of $m$ to another node.**

This set of attributes does not include the class attribute. So, all the set of tuples must be processed to give each tuple its correspond class attribute.

### 3.1.1. Generating the class attribute

In order to train the classifier, our method annotates the tuples generated in the data collection phase with class labels. The class labels $C_i$ are computed from the time passed $\tau$ and distance $\delta$ attributes. First, for each tuple, we computed the fraction:

$$r = \delta/\tau \tag{2}$$

Then, our method uses the value $r$ to group tuples with close values in the same class. For this, we defined threshold values so that each group contains approximately the same number of tuples. Thus, we can obtain $m$ groups that determine a set of $m$ class attributes, $C = \{C_1, \cdots, C_m\}$.

The correspondence between the tuple of collected data and an annotated tuple generated after processing the entire dataset can be described as follows:

$$\underset{\{\eta_1 \quad \theta_4 \quad \gamma_2 \quad l_4 \quad \delta_4 \quad \tau_3\}}{(Collected\ tuple)} \xrightarrow{\frac{\delta_4}{\tau_3}\ is\ related\ to\ C_2} \underset{\{\eta_1 \quad \theta_4 \quad \gamma_2 \quad l_4 \quad C_2\}}{(Annotated\ tuple)} \tag{3}$$

Once the annotation process is complete, the set of resulting tuples constitutes the training dataset. Then, for any given two tuples $t$ and $t'$ with different class labels, the following relation holds:

$$If \;\; t \mapsto C_i \;\; and \;\; t' \mapsto C_{i+k} \;\; then \;\; \frac{\delta}{\tau} < \frac{\delta'}{\tau'} \;\; where \;\; k \in \mathbb{Z}^+ \tag{4}$$

### 3.2. Training the classifier

The training phase consists in generating a decision tree using the J4.8 classifier. The classifier is trained and the resulting decision tree is sent for each node. A segment of the resulting decision tree is illustrated in Figure 3. The J4.8 chose the node identifier to be placed at the root level of the decision tree, and the region code attribute for the second level of the decision tree. Thus, a trained classifier can predict the class of a specific node, according to the region where the node will transmit. The other levels of the decision tree are quite different depending of the information collected by every node.

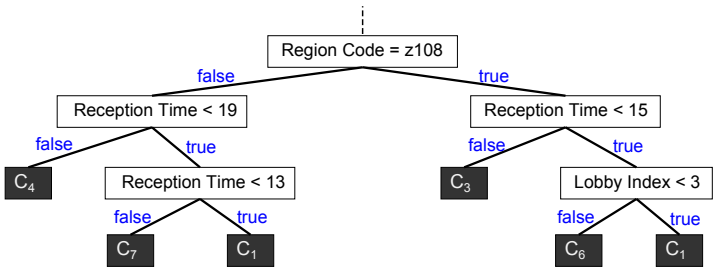



**Figure 3. Example of a part of a decision tree obtained after the training phase. The leaf nodes correspond to class attributes $C_i$.**

### 3.3. Routing using the classifier

The trained classifier is used to decide if a message copy should be forwarded to a neighbor node. Before transmitting a message copy, the transmitting node requests to its neighbor nodes their lobby index $l$ and region code $\theta$ values. These values are combined to the node ID $\eta$ and reception time $\gamma$ to generate a tuple. The classifier takes this tuple as input to predict a class $C_i$ for the intermediate node. Once the class is predicted, the node is able to decide if it will transmit a copy or not.

According to Equation 4, the maximum values of $i$ in $C_i$ reflect nodes that are able to retransmit a copy at greater distance or in lower times, i.e., nodes more capable of spreading copies when compared with nodes of classes with lower values of $i$. However, given that we applied this solution to different algorithms, namely Epidemic and Spray and Wait (SaW) in its binary mode, the values of $C_i$ qualify nodes differently depending on the algorithm. Figure 4 shows the relation of quality ($Q_j$), i.e., convenient nodes to be intermediates, and the value of class ($C_i$) for each algorithm.

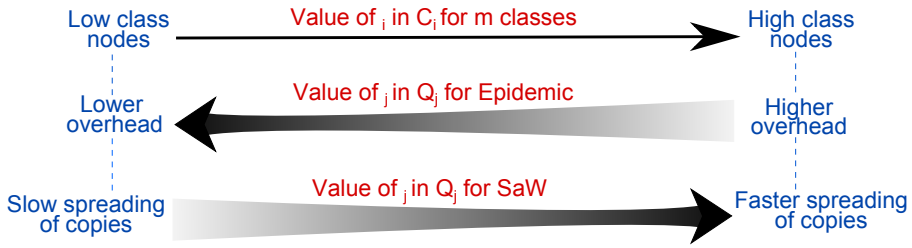



**Figure 4. Relation between the class value and the transmission decision in Epidemic and SaW**

Our approach to reduce overhead for the two routing algorithms can be described as follows.

**Epidemic.** We set a retransmission probability vector $V$, where $V_i$ is the probability to transmit copies to a node of class $C_i$. Thus, the overhead is reduced by transmitting copies with low probability to nodes with a high spreading capacity (high classes values), and with great probability to low class nodes, i.e., those nodes in conditions of difficult spreading of copies, such as nodes in region of low density, that require Epidemic to deliver a message.

**SaW.** Since a maximum value of copies ($L$) for a message is defined, the only way to decrease network overhead is to deliver messages using fewer copies than $L$. To this end, those high class nodes are chosen as intermediate nodes while low class nodes should not take part in the spreading process. The rationale for this choice is that high class nodes have higher capacity to retransmit copies than the low class nodes. The reason for such a difference is both the vehicle's route characteristics and the regions' density where vehicles transit. Hence, if high class nodes can find intermediate nodes easily, they will probably also find target nodes easily due to vehicles concentration in determinate zones. If this assumption holds, it is possible to use fewer copies than $L$ and, therefore, deliver a message before applying direct transmission.

## 4. Scenario and Simulation Environment

Evaluation experiments were conducted using the version 1.5.1 RC1 of The Opportunistic Networking Environment (ONE) Simulator [Keränen et al. 2010]. The ONE is a movement and networking simulator for evaluation of DTN routing and applications. In order to include the classification functionality, we used the API of the machine learning software WEKA [Frank et al. 2009]. For evaluating the proposed method, we used the ONE simulator to reproduce real traces of buses. The latter include location information in short time intervals of buses in the King County, Seattle, Washington [Jetcheva et al. 2003], and were obtained from CRAWDAD[1]. However, we considered only a subset of these traces which correspond to 144 km$^2$ of area and movements from 7a.m to 12p.m. (five hours of simulation). Additionally, for the comparison of real traces versus synthetic mobility models, we applied two mobility models included in the ONE: (i) Random Waypoint (RWP) and (ii) Shortest path map based movement (SPMB). We, thus, performed two group of simulations: (i) simulations for the comparison and characterization of scenarios with real traces and synthetic mobility models, and (ii) simulations for the evaluation of

[1]http://crawdad.cs.dartmouth.edu/

the proposed method, using only scenarios with real traces. The simulation parameters are summarized in Table 1 of the Section 4.2.

### 4.1. Comparison of scenarios with real traces and mobility models

Aiming at highlighting the movement characteristics of the real traces scenario, we have compared its connectivity properties and performance with those scenarios with synthetic mobility models. To this end, we have chosen the (i) Random Waypoint (RWP), where nodes move randomly in the area, and (ii) Shortest path map based movement (SPMB), where nodes move randomly from a selected point to another in the area using the shortest path and according to a map topology. The results of such comparison allow us to conclude that:

**Density and Connectivity.** Differently from synthetic models, the real traces scenario presents varied number of nodes which depends on the time of the day (see Figure 5). Thereby, the scenario of real traces has two types of nodes: (i) vehicles that always stay inside the area, and (ii) vehicles that enter and leave the area. Therefore, the former corresponds to vehicles that can be selected as source and target nodes when messages are created while the latter includes vehicles that should only participate in copy spreading. In average, in the traces scenario, the area has 254 vehicles and this value was used to set the total number of nodes in synthetic model scenarios. Due to the movement restrictions, the traces scenario presents particular connectivity properties. It presents groups of nodes with higher connectivity than the other scenarios. Thus, the lengths of the diameter and the average shortest path are greater than the obtained values from synthetic model scenarios (see Figure 8). Also, these measures reveal that there are regions in the city where groups of nodes are highly connected even in hours of less transit activity. Figure 7 shows that nodes are highly concentrated in specific zones.

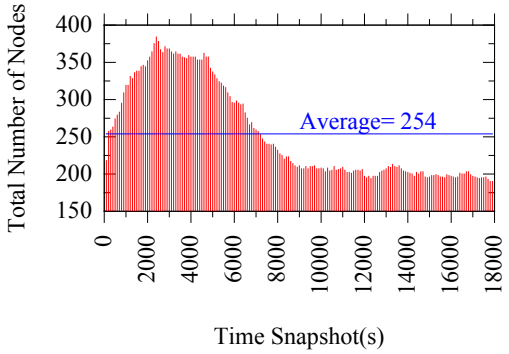



**Figure 5. Number of nodes in real traces scenario.**

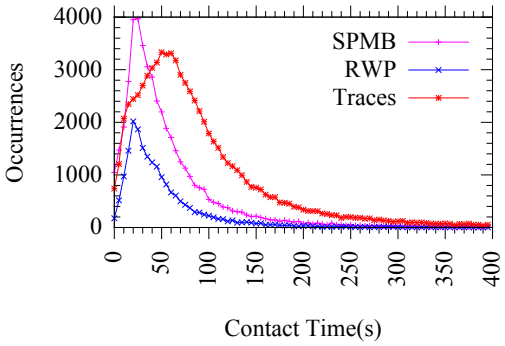



**Figure 6. Comparison of contact times.**

**Contact Time.** Contact time seems to characterize how much each model restricts node movements. Higher restricted movement models present higher values of contact time (see Figure 6). Real traces scenario has higher values because vehicular movement is ruled by traffic lights, roads and points of interest that lead to vehicular congestion. On the one hand, high contact time may help to the spreading of copies. On the other hand, in zones of high concentration of nodes, it might increase network overhead.

**Performance.** We evaluated the performance by using three measures: (i) the delivery probability, (ii) the average delay to deliver a message and (iii) the network

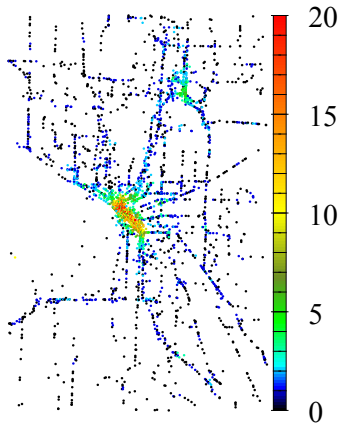



**Figure 7. Neighborhood density in traces scenario.**

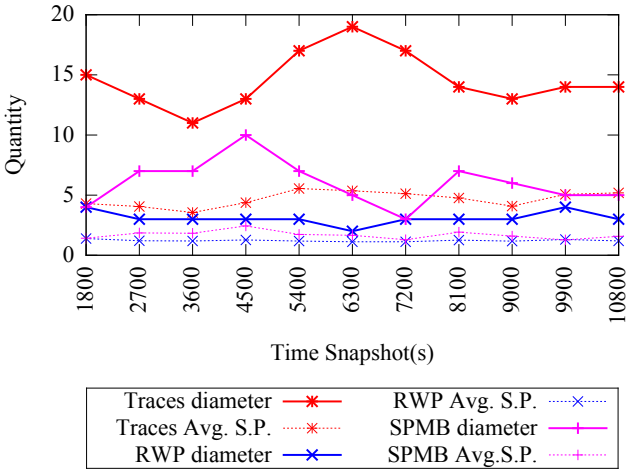



**Figure 8. Diameter and average shortest path (S.P.) lengths.**

overhead, i.e., $\frac{\#\ of\ \ copies\ \ successfully\ \ transmitted - \#\ of\ \ messages\ \ delivered}{\#\ of\ \ messages\ \ delivered}$. Given that vehicles follow specific routes, movements in the real traces scenario are more restricted than in the SPMB scenario. Thus, in the former, the message delivery probability between two randomly chosen nodes is lower than in the latter. On the other hand, since real traces scenario has highly connected groups of nodes, the delivered messages could be delivered with lower delays (see Figure 9). However, such a fast propagation of copies not only reduces delivery time but also increases network overhead (see Figure 10). Consequently, the scenario with real traces is more challenging for message delivery and prone to cause much more network overhead.

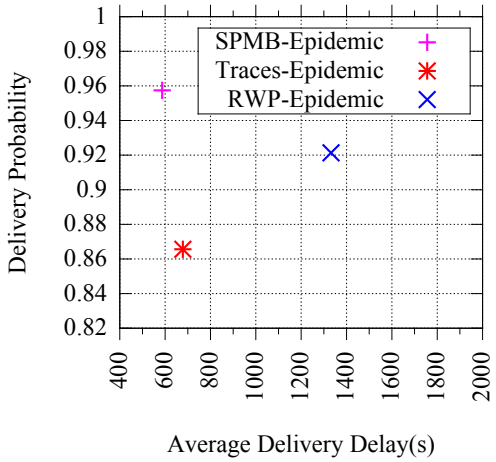



**Figure 9. Comparison of delivery Prob. vs delivery delay.**

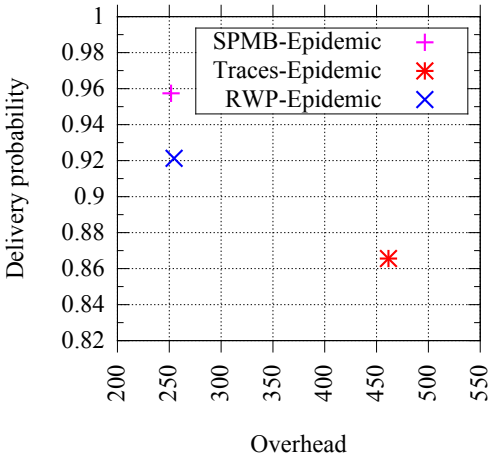



**Figure 10. Comparison of delivery probability vs overhead.**

## 4.2. Simulation parameters

The basic parameters used in our experiments are shown in Table 1. The time for message creation is the interval of time after one message is created in the whole network. In synthetic model scenarios, nodes were distributed in groups with different speed ranges

to approximate the speed distribution to the one that we find in real traces. The proposed method was evaluated using the parameters of the real traces scenario (see Table 1). Furthermore, additional parameters were setting in order to perform the first and the third phases of our method, i.e, data collection and routing using the classifier phases. It is described as follows:

**Table 1. Simulation parameters.**

| | **Real Traces** | **Synthetic models(RWP / SPMB)** |
|---|---|---|
| **Transmission rate** | 5Mbps | 5Mbps |
| **Transmission range** | 250m | 250m |
| **Storage capacity** | unlimited | unlimited |
| **Message size** | 128KB | 128KB |
| **Time for message creation** | $\approx$ 60s | $\approx$ 60s |
| **Number of nodes** | variable | **Group 1:** 71 nodes<br>**Group 2:** 99 nodes<br>**Group 3:** 61 nodes<br>**Group 4:** 23 nodes<br>**Total:** 254 nodes |
| **Node speed (m/s)** | 0 to 22 | **Group 1:** 0 to 2<br>**Group 2:** 2 to 5<br>**Group 3:** 5 to 10<br>**Group 4:** 10 to 22 |

**Data collection phase.** We used SaW and Epidemic algorithms with their default behavior. For SaW routing, the maximum number of copies per message ($L$) was set to 300. Based on the results of the simulations, we obtained seven classes and three classes for Epidemic and SaW training classifiers respectively. We have chosen this two algorithms because is interesting to reveal how our method performs in scenarios with strict overhead control measures (SaW) and in scenarios of free spreading of copies (Epidemic). Also, in our experiments, SaW had a better perform that more complex algorithms like Prophet, with a network overhead of more than 260 units lower, a average delay of more than 100 seconds below and with similar delivery probability of $\approx$84 %.

**Routing phase.** Since the classifiers has been previously trained, the routing algorithms were modified to make routing decisions according to the decision trees. By notation, we distinguish the algorithms with classifier using the standard nomination of the algorithms plus the string "+C". For "Epidemic+C" we use a retransmission probability vector $V$ for the seven classes obtained, $V = \{1, 1, 0.80, 0.50, 0.20, 0.10, 0\}$ (values were set accordingly to the rationale of Section 3.3), where the element $i$ of $V$ corresponds to the probability of retransmission to nodes of class $C_i$. In the case of "SaW+C", we exclude the worst intermediate nodes from the copy spreading process, i.e., the nodes of the most inferior class. Hence, "SaW+C" transmits a copy to any node predicted with a different class than $C_1$.

Finally, for the evaluation of the method, we experiment with "Epidemic+C" without any additional overhead control measure and, for "SaW+C" evaluation, we defined six scenarios with different values of $L = \{5, 10, 30, 50, 100, 150\}$. All of them exploit the same trained classifier.

## 5. Evaluation on Real Traces

We evaluate our proposed method (+C) using movement traces of buses of Seattle (Washington) reproduced on The ONE simulator. For this evaluation, we performed two groups of simulations as specified in the Section 4.2: (i) one group of two simulations to collect data using the algorithms SaW and Epidemic in their normal behavior, and (ii) another group of simulations using a different subset of messages and the classifier support to make routing decisions. The results in Figure 11 show a performance comparison of original SaW and SaW with classifier support ("SaW+C").

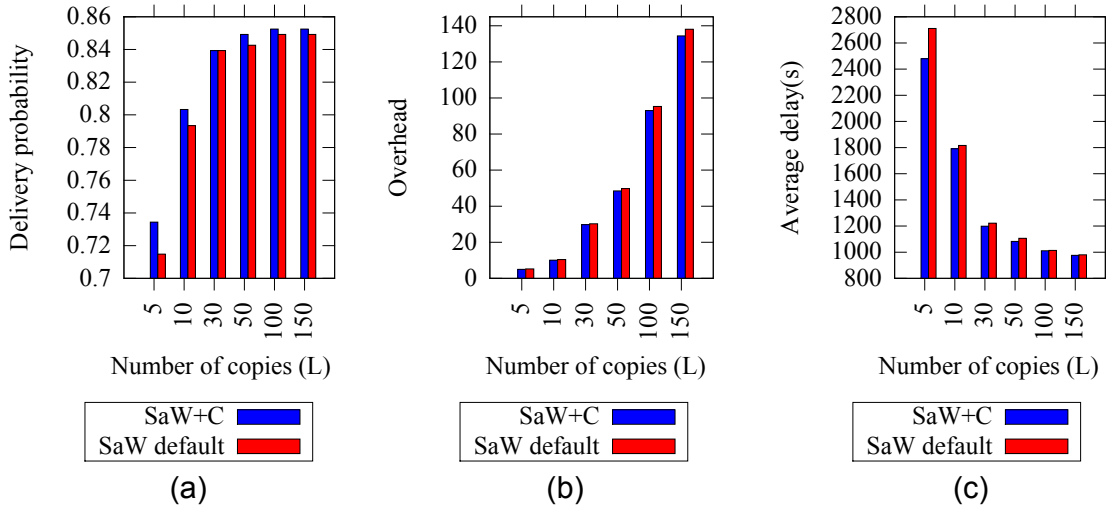



**Figure 11. SaW and "SaW+C" performance comparison. (a)Delivery probability. (b) Network overhead. (c) Avg. delivery delay.**

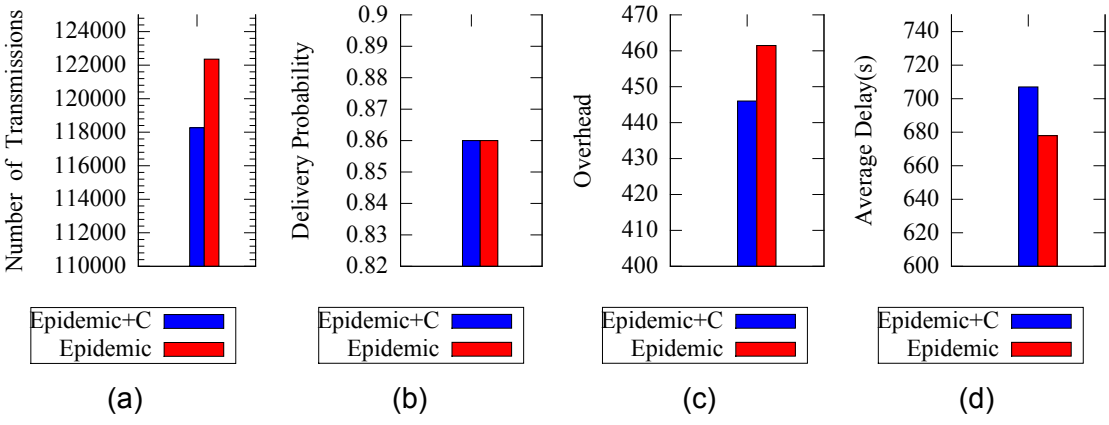



**Figure 12. Epidemic and "Epidemic+C" performance comparison. (a)Number of transmissions. (b)Delivery probability. (c) Network overhead. (d) Avg. delivery delay.**

As desired, our method improve the performance of the algorithm SaW in delivery probability as well as network overhead. Although the quantitative values are still modest, they are consistent in all scenarios when the number of copies of SaW varies. The delivery delay has also improve since, for each number of copies configured (see Figure 11(c)), the average delay decreases by 231.3, 25.2, 23.5, 24.2, 3.0 and 4.1 seconds respectively.

We also compared Epidemic and "Epideimc+C" as shown in Figure 12. The results were encouraging, "Epidemic+C" reduced significantly the amount of transmissions, and consequently the network overhead, without decreasing the delivery probability. Despite of that, there was a small increment of 29.04s in message delivery delay. This can explained by the way Epidemic floods the networks and thus delay is minimal. But, as we compare horizontally SaW and Epidemic delay ( Figure 11(c) vs Figure 12(d)) it is easy to see that replicating less copies has implications in delivery delay.

## 6. Conclusion and Future Work

To reduce network overhead in multi-copy DTN routing, we applied a machine learning algorithm to exclude nodes predicted as bad intermediates from the copy spreading process. We have applied our classification mechanism (+C) on top of two regular DTN routing algorithms in order to verify the performance improvement that our approach provides under real trace scenarios (Seattle buses). The results confirmed that performance of SaW can be improved with the classifier support (SaW+C). Such gain is due to a limited number of copies distributed more conveniently by transmitting them to more suitable spreading nodes, on regions of moderate concentration of vehicles. We also applied our method on the Epidemic routing algorithm (Epidemic+C) and we have obtained a reduction in network overhead without degrading message delivery probability. Despite of such a promising result, we have observed a slight increase in delivery delay due to the use of fewer copies.

We have also conducted evaluation experiments about the impact of mobility on multi-copy routing. We observed that the network overhead is greater in real traces scenarios than in synthetic movement models. This can be attributed to the fact that real scenarios present non-uniform distribution of nodes, movement restrictions, and high variability density in different regions of the city. The experiments also helped us to model the decision tree parameters.

For future work, we intend to focus on mobility information and use machine learning algorithms to predict where and for how long the buses will stay in certain part of the map. Also, given that the spreading capacity of nodes can be quantified, we can explore machine learning algorithm for regression, such as CART, ANNs or SVMs.

## References

Abolhasan, M., Wysocki, T., and Dutkiewicz, E. (2004). A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2(1):1–22.

Ahmed, S. and Kanhere, S. S. (2010). A Bayesian Routing Framework for Delay Tolerant Networks. *IEEE Wireless Communication and Networking Conference*, pages 1–6.

Burgess, J., Gallagher, B., Jensen, D., and Levine, B. N. (2006). MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. *IEEE INFOCOM 25TH*, pages 1–11.

Colagrosso, M. (2005). A classification approach to broadcasting in mobile ad hoc network. In *Intl. Conference on Communications (ICC)*, volume 2, pages 1112–1117.

Franck, L. and Gil-Castineira, F. (2007). Using delay tolerant networks for car2car communications. *IEEE Intl. Symp. on Industrial Electronics (ISIE)*, pages 2573–2578.

Frank, M. H. E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11:10–18.

Haas, Z. J. and Small, T. (2006). A new networking model for biological applications of ad hoc sensor networks. *IEEE/ACM Transactions on Networking*, 14:27–40.

Jetcheva, J., Hu, Y., Palchaudhuri, S., Kumar, A., and Johnson, S. D. B. (2003). Design and evaluation of a metropolitan area multitier wireless ad hoc network architecture. In *5th IEEE Workshop on Mobile Computing Systems and Applications*, pages 32–43.

Keränen, A., Kärkkäinen, T., and Ott, J. (2010). Simulating mobility and dtns with the one. *Journal of Communications*, 5:92–105.

Korn, A., Schubert, A., and Telcs, A. (2009). Lobby index in networks. *Physica A: Statistical Mechanics and its Applications*, 388:2221–2226.

Lindgren, A., Doria, A., and Schelén, O. (2003). Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20.

Pallis, G., Katsaros, D., Dikaiakos, M. D., Loulloudes, N., and Tassiulas, L. (2009). On the structure and evolution of vehicular networks. In *IEEE MASCOTS*, pages 1–10.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Spyropoulos, T., Psounis, K., and Raghavendra, C. S. (2005). Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of ACM SIGCOMM workshop on DTN*, WDTN, pages 252–259. ACM.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Vahdat, A. and Becker, D. (2000). Epidemic routing for partially-connected ad hoc networks. Technical report.

Vasilakos, A. V., Zhang, Y., and Spyropoulos, T. (2011). *Delay Tolerant Networks: Protocols and Applications*. CRC Press, Inc., 1st edition.

Vendramin, A. C. K., Munaretto, A., Delgado, M. R., and Viana, A. (2012). Grant: Inferring best forwarders from complex networks' dynamics through a greedy ant colony optimization. *Computer Networks*, 56(3):997–1015.

Vu, L., Do, Q., and Nahrstedt, K. (2011). Jyotish: A novel framework for constructing predictive model of people movement from joint wifi/bluetooth trace. *IEEE Intl. Conf. on Pervasive Computing and Communications (PerCom)*, pages 54–62.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier.

# Um Protocolo de Identificação e Minimização de Congestionamentos de Tráfego para Redes Veiculares

**Guilherme Braga Araújo**[1], **Anna Izabel J. Tostes** [2],
**Fátima de L. P. Duarte-Figueiredo**[1], **Antônio A. F. Loureiro**[2]

[1]Departamento de Informática - Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
[2]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)
guilhermebaraujo@gmail.com, annatostes@gmail.com,
fatimafig@pucminas.br, loureiro@dcc.ufmg.br

***Resumo.*** *O congestionamento de trânsito é tratado pelos órgãos governamentais como problema de mobilidade urbana, o qual gera stress aos motoristas e perdas econômicas. Para auxiliar na identificação e redução de congestionamentos no tráfego veicular, são propostos sistemas inteligentes de transporte. Este trabalho propõe o CARTIM, um protocolo de identificação colaborativa e minimização de congestionamentos veiculares. O protocolo se baseia na comunicação V2V (Vehicle-to-Vehicle) para mensurar cooperativamente o nível local de congestionamento de tráfego. Se houver infraestrutura presente, pode ocorrer disseminação de informações consolidadas a veículos em outras regiões através da comunicação V2I (Vehicle-to-Infrastructure). Para identificar localmente um congestionamento de forma eficaz, o CARTIM possui um sistema baseado em lógica fuzzy (nebulosa), que faz o tratamento de informações qualitativas (densidade de veículos etc.). O protocolo também utiliza eficientemente o canal colaborativo de comunicação, evitando sobrecarga. Os resultados obtidos nas simulações mostraram que o CARTIM consegue detectar congestionamentos (melhor que trabalhos relacionados) e minimizá-los.*

***Abstract.*** *Traffic congestion is handled by government agencies as an urban mobility problem, which generates stress to drivers and economic loss. Intelligent transport systems can assist in the identification and reduction of vehicular traffic congestion. In this context, this paper proposes CARTIM, a protocol for collaborative identification and minimization of vehicular congestion. The protocol uses V2V (Vehicle-to-Vehicle) communication to cooperatively measure the local level of vehicular traffic congestion. Additionally, if any infrastructure is present, consolidated information dissemination may occur to vehicles in other regions through V2I (Vehicle-to-Infrastructure) communication. To effectively identify a traffic congestion locally, CARTIM employs a fuzzy logic-based system, which is used in the treatment of qualitative information (vehicle density etc.). The protocol also efficiently uses the collaborative communication channel, preventing overload. The simulation results showed that CARTIM can detect congestion (better than related work) and minimize it.*

## 1. Introdução

O transporte sempre desempenhou um papel de destaque na história da humanidade. Durante as últimas décadas, a densidade de veículos no transporte rodoviário no Brasil tem aumentado consideravelmente [DENATRAN]. Esse processo deu origem ao fato de que,

atualmente, os congestionamentos de tráfego nas estradas são um dos fenômenos mais comuns que os motoristas têm que enfrentar. Além de ser uma experiência estressante e desagradável para motoristas, os congestionamentos de tráfego também causam um impacto negativo na economia e no meio ambiente.

Grandes avanços tecnológicos vêm sendo aprimorados e embarcados em sistemas, nos veículos automotores, para melhorar a experiência dos motoristas como, por exemplo, sistemas de frenagem, alarmes de velocidade acima do permitido [Alves et al. 2009]. No entanto, o crescimento populacional e, consequentemente, da frota de veículos, torna necessário redes que conectem esse tipo de ambiente. Veículos com conectividade com outros veículos compartilhando o mesmo meio (as vias) auxiliarão motoristas, a fim de proporcionar viagens mais rápidas até seus destinos.

Este trabalho pretende responder à seguinte pergunta: como identificar e minimizar congestionamentos de tráfego utilizando redes veiculares? Esse problema pode ser dividido nas etapas: (i) identificar um congestionamento, (ii) conseguir uma decisão colaborativa do congestionamento estimado, (iii) repassar informações da situação atual do tráfego e (iv) utilizar políticas de alteração de rotas para reduzir o congestionamento.

O objetivo principal, portanto, é propor um protocolo, denominado CARTIM (do inglês **C**ooperative vehicul**AR** **T**raffic congestion **I**dentification and **M**inimization), para detecção e redução de congestionamento em redes veiculares, utilizando o padrão IEEE 802.11p [Alves et al. 2009] para troca de informações, em tempo real, da situação atual do tráfego. A proposta utiliza para isso um sistema baseado em lógica *fuzzy*, e uma heurística que visa reduzir o tempo de viagem dos motoristas, permitir alteração das rotas e melhorar o fluxo do tráfego de veículos, em caso de congestionamento.

Os resultados obtidos pelo CARTIM mostram que o sistema *fuzzy* obteve boa eficácia para quantificar o nível local de congestionamento dos veículos, em cenários heterogêneos (autoestrada e vias urbanas). Também foram analisados os níveis gerais de congestionamento nos cenários avaliados. Essa visão global dos congestionamentos é utilizada por órgãos gestores como principal métrica para quantificar, caracterizar e representar a intensidade de momento da situação do tráfego, nos grandes centros urbanos. A heurística para minimização dos congestionamentos e alteração das rotas também obteve resultados satisfatórios, reduzindo o tempo médio de viagem dos veículos.

Este artigo está organizado em 6 seções. A seção 2 contextualiza redes veiculares e simulação de redes veiculares. A seção 3 apresenta uma revisão bibliográfica dos principais trabalhos relacionados. Suas características são analisadas, classificadas e comparadas com o CARTIM. Na seção 4, detalhes do protocolo proposto são discutidos. A definição dos cenários, a avaliação e os resultados obtidos nas simulações são abordados na seção 5. Por fim, a seção 6 conclui este trabalho e lista potenciais trabalhos futuros.

## 2. Redes Veiculares

Redes veiculares, também chamadas de VANETs (*Vehicular Ad hoc NETworks*), são redes formadas por veículos e por equipamentos fixos (*Road Side Units* - RSUs) localizados às margens das estradas. Essas redes se caracterizam pela alta mobilidade dos nós (veículos), enlaces intermitentes e requisitos estritos de latência. São esses atributos que impedem que alguns protocolos utilizados em redes *ad hoc* sejam utilizados em redes veiculares, pois eles não apresentam desempenho satisfatório [Alves et al. 2009].

A arquitetura das redes veiculares define a forma como os nós da rede se organizam e se comunicam. As arquiteturas definidas para redes veiculares são: (i) *ad hoc* puro, (ii) infraestruturada ou (iii) híbrida [Alves et al. 2009]. Na arquitetura (i), os veículos se comunicam sem nenhum suporte externo ou elemento centralizador.

Nas redes veiculares, devido à natureza especial do ambiente, a simulação deve considerar dois aspectos importantes que caracterizam esse tipo de rede: a alta mobilidade dos nós e a comunicação via rádio entre os nós. Como a mobilidade e a comunicação via rádio possuem características bem distintas, tipos diferentes de simuladores para redes veiculares são utilizados. Em [Martinez et al. 2011] os simuladores são divididos em três tipos: (i) simuladores de rede, (ii) geradores de mobilidade e tráfego, (iii) simuladores de VANET. Contudo, alguns simuladores dos tipos (i) e (ii) têm trabalhado de forma isolada, o que não permite simular cenários mais próximos do ambiente real. Com a necessidade de comunicação *on-line* entre esses simuladores, alguns *frameworks* foram implementados. Tal ambiente integrado, conhecido como simulador de VANET, proporciona resultados mais satisfatórios e próximos da realidade de redes veiculares.

## 3. Trabalhos Relacionados

Mecanismos de controle de congestionamento adequados são essenciais para otimizar o fluxo de veículos em redes veiculares. As aplicações de controle de congestionamento em redes veiculares podem ser discriminadas em identificação, minimização e prevenção de congestionamento. A identificação de congestionamento de tráfego representa a eficácia em se caracterizar um congestionamento no trânsito. A minimização corresponde à tentativa de reduzir os congestionamentos detectados. A prevenção de congestionamento é uma abordagem complexa, uma vez que precisa gerenciar fluxos de veículos nas vias, para prevenir o surgimento dos congestionamentos. Esse problema de prevenção de congestionamento de tráfego pode ser reduzido em um problema de minimização de fluxos em grafos, o que o remete à classe de problemas NP-Completos.

Os trabalhos de [Fukumoto et al. 2007] e [Fahmy and Ranasinghe 2008] apresentam propostas de monitoramento e detecção de condições de tráfego, utilizando comunicação V2V (*Vehicle-to-Vehicle*). Para isso, são transmitidas mensagens do tipo *beacons*. Tais mensagens são bem simples, possuem um baixo *overhead* e são usadas para anunciar a presença de um veículo aos vizinhos. Outras abordagens, como o ConProVa [Silva et al. 2013] utilizam apenas a comunicação V2I (*Vehicle-to-Infrastructure*), o que limita a proposta. No ConProVa, estruturas para lidar com questões relativas ao provimento de contextos em redes veiculares são criadas na infraestrutura. Para isso, o ConProVa implementa um *middleware*, a fim de ajudar na tomada de decisão e resolução de conflitos de interesse sobre a situação real do tráfego relatada pelos veículos, uma vez que os veículos não conseguem validar colaborativamente as condições detectadas na via.

Existem também abordagens como o ECODE [Younes and Boukerche 2013], que utilizam ambas as comunicações V2V e V2I. No entanto, o ECODE por depender da infraestrutura (RSU) nas áreas de monitoramento, limita a proposta a cenários específicos.

Como as abordagens na literatura possuem características distintas, uma classificação das propostas de monitoramento de tráfego veicular foi adaptada para este trabalho, a fim de permitir a comparação das técnicas analisadas. A tabela 1 mostra informações sumarizadas e comparativas em relação às referências deste trabalho.

**Tabela 1. Propostas de monitoramento de tráfego veicular avaliadas**

| Proposta | *Validação Cooperativa V2V* | *Identificação de Congestionamento* | *Tamanho do Engarrafamento* | *Baixo Overhead* | *Disseminação* | *Arquitetura(s)* | *Mudança de Rota* |
|---|---|---|---|---|---|---|---|
| COC [Fukumoto et al. 2007] | sim | sim(1) | sim | não | não | V2V | não |
| [Fahmy and Ranasinghe 2008] | não | sim(1) | não | sim | não | V2V | não |
| ConProVa [Silva et al. 2013] | não | sim(1) | não | sim | sim | V2I | sim |
| ECODE [Younes and Boukerche 2013] | não | sim(1) | não | sim | sim | V2V e V2I | sim |
| [Lu and Cao 2003] | não | sim | não | sim | não | V2I | não |
| [Binglei et al. 2008] | não | sim | não | sim | não | V2I | não |
| COTEC [Bauza et al. 2010] | sim | sim | sim | sim | sim | V2V | não |
| CARTIM | sim | sim | sim | sim | sim | V2V e V2I | sim |

1 - Apenas detecta um congestionamento, não classifica e nem estima níveis de congestionamento.

A classificação adotada distingue os trabalhos de acordo com: validação cooperativa V2V se refere à capacidade da proposta em utilizar a comunicação cooperativa entre veículos para chegar a uma decisão (consensual) sobre a situação do tráfego em uma região. A identificação de congestionamento retrata a eficácia em mensurar as condições do tráfego na via. O tamanho do engarrafamento indica a eficácia em estimar a extensão de um congestionamento. O *overhead* é a capacidade de se evitar a sobrecarga do canal de comunicação. A disseminação de informação retrata a característica da proposta em propagar dados da situação do tráfego na rede. A arquitetura mostra como a rede veicular se organiza. A mudança de rota indica se os veículos podem alterar suas rotas iniciais.

O trabalho de [Lu and Cao 2003] propõe um algoritmo para classificação automática do nível de congestionamento veicular, utilizando para isso um mecanismo de inteligência computacional baseado em lógica *fuzzy*. Em [Binglei et al. 2008] e [Bauza et al. 2010], a ideia proposta em [Lu and Cao 2003] foi utilizada para aumentar a eficiência na detecção de congestionamentos. Como retratado em [Binglei et al. 2008], o uso de limites fixos entre categorias de tráfego veicular leva a altas taxas de erros, principalmente quando uma categoria está próxima da fronteira de classificação de outra. Isso ocorre devido à característica do fluxo de tráfego veicular, que não possui limites claros entre tipos de tráfegos distintos. Nesse caso, para problemas de decisão não determinísticos, lógica *fuzzy* obtém bons resultados, por permitir que variáveis de entrada do sistema *fuzzy* contenham elementos com grau parcial de adesão, em regiões de fronteira.

O COTEC, proposta apresentada em [Bauza et al. 2010], é a principal referência deste trabalho. Conforme mencionado, ele utiliza lógica *fuzzy* para detectar o nível de congestionamento. O protocolo também dispõe de uma abordagem para validação cooperativa das condições do tráfego detectadas pelo sistema. Os conceitos abordados pelos autores em [Bauza et al. 2010] foram incorporados a este trabalho. No entanto, o COTEC possui deficiências que limitam aspectos positivos da proposta, tais como: (i) a ineficiência em evitar que um veículo alcance uma área congestionada e (ii) a falta de uma política de permissão de alteração de rotas. (iii) Novas configurações no sistema *fuzzy* também são necessárias para que tal técnica não se limite apenas ao cenário ideal (autoestrada) proposto pelos autores.

Neste contexto, propõe-se um protocolo de identificação e redução de congestionamento veicular, que possua todas as características apresentadas na tabela 1. O CARTIM, aqui proposto, supera as limitações apresentadas nas referências avaliadas neste trabalho. Com todas as características listadas na tabela 1, o CARTIM é uma proposta viável para cenários heterogêneos, que fazem parte do escopo de redes veiculares.

# 4. CARTIM

Esta seção apresenta o protocolo proposto neste trabalho. A explicação do CARTIM (***C**ooperative vehicul**AR** **T**raffic congestion **I**dentification and **M**inimization*) está separada em três partes: (i) estimativa local de tráfego veicular, (ii) validação cooperativa de congestionamento de tráfego, e (iii) heurística para minimização de congestionamento de tráfego.

## 4.1. Estimativa Local de Tráfego Veicular

O CARTIM utiliza a arquitetura V2V para estimar o nível de congestionamento do tráfego. As informações são obtidas através da troca periódica de mensagens *beacons* entre os veículos, em que cada veículo informa aos seus vizinhos sua velocidade e sua localização atual. Para não sobrecarregar o canal de comunicação com mensagens *beacons*, a frequência de transmissão dessas mensagens no protocolo é ajustada inicialmente para 0.5 Hz, ou seja, a cada 2 segundos o veículo envia um *beacon* por *broadcast*.

O processo de identificação de congestionamento não é uma tarefa simples. Várias informações qualitativas são tratadas como, por exemplo, densidade baixa de veículos etc. Os trabalhos de [Lu and Cao 2003], [Binglei et al. 2008] e [Bauza et al. 2010] utilizam lógica *fuzzy* para detecção do nível de congestionamento de tráfego. Fundamentado nesses trabalhos, o CARTIM também utiliza lógica *fuzzy*.

Como em qualquer sistema de decisão baseado em lógica *fuzzy*, as variáveis possuem valores linguísticos e são separadas em conjuntos *fuzzy*. A classificação desses conjuntos no CARTIM se baseia nas referências do HCM (*Highway Capacity Manual*) [Board 2000] . O manual de capacidade das estradas contém conceitos, diretrizes e procedimentos computacionais para medir a capacidade, o desempenho e a qualidade do tráfego, baseado na velocidade e densidade de veículos nas vias.

O sistema *fuzzy* proposto para o CARTIM também utiliza como parâmetros de entrada a velocidade do veículo e a densidade veicular da via. A velocidade é um atributo simples de se obter e pode ser calculada através da distância percorrida em um intervalo curto de tempo. No entanto, a obtenção da densidade de veículos na via é uma tarefa complexa. Com base no número de mensagens *beacons* recebidas, cada veículo estima a quantidade de veículos vizinhos em um intervalo de tempo. Assim, é possível calcular a densidade de veículos, que corresponde ao número de veículos por quilômetro, por faixa.

As classificações das variáveis linguísticas (conjuntos *fuzzy*) de entrada do sistema para a velocidade são: rápida, moderada, lenta, muito lenta; e para densidade são: baixa, média, alta, muito alta. O conjunto *fuzzy* de saída varia entre [0,1], de acordo com a seguinte classificação: 0 - equivale a fluxo livre, 1/3 - equivale a fluxo fraco, 2/3 - equivale a fluxo moderado, 1 - equivale a condições severas.

**Tabela 2. Regras *fuzzy* para o CARTIM**

| Nível de Congestionamento | Densidade do Tráfego | | | |
|---|---|---|---|---|
| Velocidade | **Baixa** | **Média** | **Alta** | **Muito Alta** |
| **Muito Lenta** | fraco | moderado | moderado | severo |
| **Lenta** | livre | fraco | moderado | moderado |
| **Moderada** | livre | fraco | fraco | moderado |
| **Rápida** | livre | livre | livre | fraco |

Fonte: [Bauza et al. 2010]

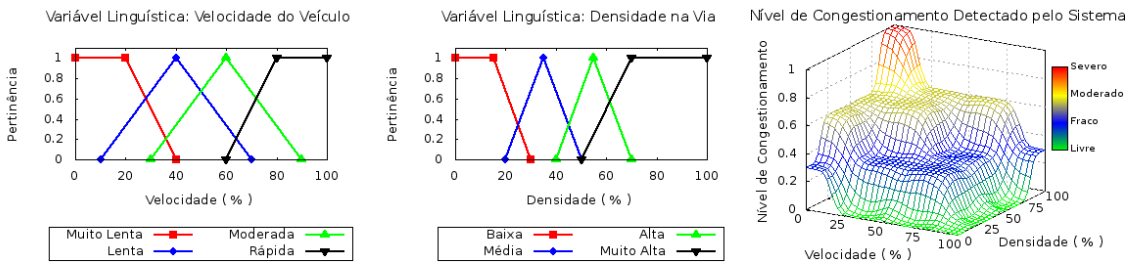
A tabela 2 apresenta as regras *fuzzy* adotadas para obtenção do nível de congestionamento do sistema, de acordo com o respectivo conjunto *fuzzy* de entrada. As figuras 1 (a) e 1 (b) mostram as funções de pertinência para os conjuntos *fuzzy* de entrada do sistema, em que é possível verificar o grau parcial de adesão dos conjuntos, nas regiões de fronteira. O conjunto *fuzzy* de saída, representado pela figura 1 (c), permite visualizar a estrutura para detecção do nível de congestionamento. Por exemplo, condições severas de congestionamento ocorrem apenas quando a velocidade do veículo está muito lenta e a densidade de veículos vizinhos na via está muito alta.

Um grande benefício do sistema *fuzzy* proposto para o CARTIM, e ilustrado na figura 1, está relacionado à conversão dos valores das variáveis linguísticas de entrada do sistema para porcentagem. Desse modo, cada veículo estima o valor da velocidade baseado na razão entre sua velocidade momentânea e a velocidade máxima permitida na via. A densidade de veículos vizinhos no CARTIM se baseia na ocupação do segmento de estrada que o veículo percorre no momento. Utilizando o raio de transmissão (alcançado na comunicação V2V para disseminação de mensagens), em metros, estima-se a quantidade máxima de veículos por faixa no segmento. Assim, cada veículo, de acordo com a razão entre o número de vizinhos momentaneamente no segmento e a quantidade máxima de veículos suportada no local, consegue estimar a ocupação do segmento em porcentagem. A utilização dessa abordagem pelo CARTIM faz com que o protocolo consiga se adaptar e ajustar o sistema *fuzzy* dinamicamente de acordo com o ambiente (contexto).

### 4.2. Validação Cooperativa de Congestionamento de Tráfego

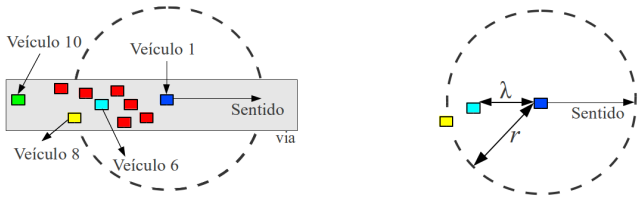
O CARTIM utiliza um processo colaborativo (comunicação V2V) para validar o nível local de congestionamento detectado pelo sistema *fuzzy*. Para isso, são utilizadas mensagens CTE (*Cooperative Traffic Estimation*), um conceito apresentado em [Bauza et al. 2010], em que um limiar de congestionamento ($C_{th}$ - *congestion threshold*) é estipulado previamente, e, assim que atingido, um processo colaborativo de validação das condições detectadas se inicia. A transmissão das mensagens cooperativas ocorre exclusivamente da extremidade dianteira do congestionamento para a extremidade traseira, através de um encaminhamento *multi-hop*, *unicast*.

O intervalo de frequência e número de vizinhos são usados pelos veículos no CARTIM para calcular cooperativamente o nível de congestionamento. Proposta no COTEC, essa abordagem usa apenas veículos que retransmitem a mensagem CTE, em que cada veículo usa sua quantidade de vizinhos para calcular a média ponderada do nível de congestionamento. Dessa forma, tal proposta procura representar as condições detectadas pelos demais veículos na via, sem aumentar o *overhead* de comunicação.




(a) (b) (c)

**Figura 1. CARTIM, sistema *fuzzy* para detecção do nível de congestionamento**




(a) Comunicação V2V (b) Distância ótima - $\lambda$

**Figura 2. CARTIM, processo de retransmissão de mensagens *unicast*.**

As mensagens CTE também são utilizadas para estimar o tamanho de um congestionamento, de acordo com a posição inicial do primeiro veículo que a transmitiu. Assim, a descoberta do veículo na dianteira do congestionamento é uma tarefa importante para o funcionamento do CARTIM. Tal veículo, além de ser o responsável pela frequência (Hz) de geração de mensagens CTE na via congestionada, serve como referencial aos demais veículos na pista, para cálculo do tamanho do engarrafamento.

A alta mobilidades dos nós, as mudanças na topologia da rede e as desconexões frequentes são características já mencionadas de VANET, que podem levar a altas taxas de perdas no envio de mensagens *unicast*. Além disso, para não sobrecarregar a rede, o CARTIM não utiliza mensagens ACK (*acknowledgement*) para confirmação do recebimento das mensagens CTE. Portanto, elevadas taxas de perda de mensagens comprometem o funcionamento do protocolo, uma vez que a comunicação fica interrompida.

O CARTIM realiza uma avaliação para identificar qual é o veículo mais adequado em cada *hop*, para encaminhar a mensagem *unicast*. A figura 2 (a) ilustra um cenário em que o veículo 8 é o mais distante em relação ao veículo 1, considerando-se o raio ($r$) de comunicação. Tal veículo, no entanto, não é apontado como o veículo ideal para retransmissão da mensagem CTE. O veículo 6, considerado ideal, é obtido pelo protocolo através da equação (1), que calcula dinamicamente a distância ótima ($\lambda$) do veículo, para retransmissão da mensagem. Essa distância, ilustrada pela figura 2 (b), obtida em metros (unidade na simulação), é calculada subtraindo-se do tamanho do raio de transmissão ($r$) dos veículos a razão entre a velocidade máxima permitida na via (m/s) e a frequência (Hz) de transmissão das mensagens *beacons*.

$$\lambda(m) \quad = \quad r(m) - \frac{v(m/s)}{f(Hz)} \tag{1}$$

Por exemplo, veículos com a frequência de transmissão de *broadcast beacons* ajustada em 0.5 Hz e com raio de transmissão de 150 metros, trafegando em uma via qualquer, cuja velocidade máxima permitida seja de 16,67 m/s (60 km/h), obtêm a distância ótima de aproximadamente 116,67 metros. No CARTIM, esse cálculo é necessário para encontrar o veículo vizinho mais próximo dessa distância de referência, a fim de retransmitir a mensagem *unicast*. Apesar das mensagens cooperativas de tráfego serem *unicast*, veículos no raio de transmissão escutam tais mensagens e obtêm seu conteúdo.

Utilizando essa abordagem, o CARTIM minimiza problemas causados pelas contantes mudanças na topologia da rede. O protocolo considera que, mesmo que o veículo selecionado para retransmissão da mensagem se desloque na via com a velocidade máxima permitida, ele ainda tem alta probabilidade de estar dentro do raio de transmissão do veículo atual, durante o intervalo de tempo de um ciclo de transmissão de mensagem *beacon*. Esse intervalo de tempo é o mesmo utilizado para descoberta dos vizinhos.

As mensagens CTE possuem um alto *overhead* quando comparadas às mensagens *beacons*. Para evitar sobrecarga do canal de comunicação, tais mensagens só são enviadas quando o nível de congestionamento local no veículo é superior ao $C_{th}$ e em uma frequência de 0.1 Hz. Além disso, apenas veículos com nível de congestionamento maior que $C_{th}$ participam do processo de retransmissão e, caso já exista algum veículo à frente transmitindo, o veículo de trás cancela sua transmissão. O processo de retransmissão na rede termina quando as mensagens chegam em uma região onde o nível de congestionamento local detectado pelos veículos é inferior ao $C_{th}$.

### 4.3. Heurística para Minimização de Congestionamento de Tráfego

Gerenciar o fluxo de veículos e prever congestionamento de tráfego em redes veiculares é uma tarefa complexa computacionalmente. Nesse contexto, uma heurística para redução de congestionamento de tráfego é adotada pelo CARTIM. O objetivo da heurística é tentar reduzir o tempo gasto no trânsito pelos motoristas, sugerindo alterações nas rotas quando começar a ocorrer condições severas de congestionamento em uma determinada via.

A proposta de minimização de congestionamento também utiliza o processo colaborativo para validação das condições detectadas. A mensagem CTE, ao atingir nível de congestionamento cooperativo maior que o limiar $CTA_{th}$, se transforma em mensagem CTA (*Congestion Traffic Avoid*). Veículos no raio de transmissão, ao capturarem tais mensagens na rede, podem alterar suas rotas iniciais, evitando passar na via sob condições severas. A minimização de congestionamento do CARTIM pode utilizar as arquiteturas V2V e V2I. Contudo, a infraestrutura, composta pelas RSUs, é transparente para o protocolo. Caso esteja presente, a função da RSU é escutar mensagens CTA na rede e enviar *broadcast* dessas mensagens em uma região maior que as alcançadas na comunicação V2V. Caso a infraestrutura não esteja presente, apenas a comunicação V2V é utilizada.

No CARTIM, quando o nível de congestionamento em uma via não estiver mais sob condições severas, as mensagens CTA param de ser disseminadas. Assim, o protocolo regula dinamicamente os alertas para alteração das rotas, procurando desse modo evitar o surgimento de novos pontos de congestionamento no tráfego.

## 5. Simulações e Resultados

Neste trabalho, simulações foram conduzidas para comparar o sistema *fuzzy* do CARTIM, discutido na seção 4.1, com o do COTEC. A finalidade é verificar o comportamento dos sistemas e atestar a eficácia de cada um em detectar congestionamentos na rede. Em seguida, simulações foram realizadas para mostrar como a utilização do CARTIM pode auxiliar na minimização de congestionamentos de tráfego veicular. Para isso, um cenário maior e mais próximo de ambientes urbanos foi criado.

A ferramenta de simulação de rede utilizada foi o OMNET++ [Varga and Hornig 2008]. O simulador de tráfego e de mobilidade utilizado foi o SUMO [Martinez et al. 2011]. O *framework* VEINS [Sommer et al. 2011] também foi usado neste trabalho para simulação de VANET. A tabela 3 apresenta um quadro sumarizado das configurações realizadas nas simulações, bem como os principais parâmetros de comunicação ajustados no *framework* VEINS. A potência de transmissão na comunicação V2V de 0.61 mW (tabela 3) resulta em um raio de transmissão ($r$) aproximado de 150 metros. Na comunicação V2I o raio ($r$) estimado é de 300 metros.

**Tabela 3. Principais parâmetros de simulação**

| **W**ireless Communication | |
|---|---|
| *Frequency* | 5.8 GHz |
| *Transmission power - V2V Commun.* | 0.61 mW |
| *Transmission power - V2I Commun.* | 2.20 mW |
| *Bit rate* | 6 Mbit/s |
| *Signal Attenuation* | -94 dBm |

| **C**ARTIM | |
|---|---|
| *Beacon frequency* | 0.5 Hz |
| *CTE frequency* | 0.1 Hz |
| *Congestion threshold* ($C_{th}$) | 0.5 |
| *CTA threshold* ($CTA_{th}$) | 0.8 |
| *Vehicle average space* | 10 m |

### 5.1. Comparação do Sistema *Fuzzy*

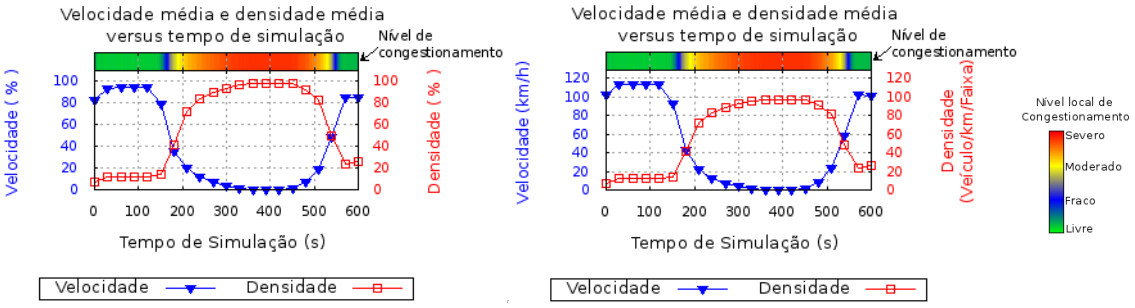
Para comparar o sistema *fuzzy* dos protocolos CARTIM e COTEC, dois cenários ($A$ e $B$) foram avaliados. Em ambos os cenários, uma autoestrada (*freeway*) de dois quilômetros com duas faixas foi utilizada. Os mapas dos cenários foram criados no SUMO, cuja taxa de chegada de veículos corresponde a 1 veículo por segundo. O cenário $A$ é similar ao cenário utilizado pelos autores do COTEC. Tal cenário possui velocidade máxima na via de 120 km/h. O cenário $B$ apresenta velocidade máxima permitida de 40 km/h, que corresponde à velocidade máxima em algumas vias urbanas. Na simulação, com a intenção de gerar congestionamento, foi forçado um acidente entre alguns veículos, começando a partir do tempo de simulação $t = 150$ s, obstruindo a via durante 5 minutos (300 s) em ambos os cenários. Os dados obtidos nas simulações avaliaram veículos em 800 metros da autoestrada, na região do acidente.

Os gráficos das figuras 3 e 4 têm como objetivo mostrar o comportamento dos protocolos em estimar localmente (nos veículos) o tráfego veicular. Para isso, os conjuntos *fuzzy* de entrada do sistema também foram avaliados. Esses gráficos representam as velocidades e densidades médias, obtidas pelos veículos durante as simulações. Os níveis de congestionamento local inferidos pelos sistemas *fuzzy* (nos veículos), durante as simulações, também são ilustrados nesses gráficos. Essas informações agrupadas em um gráfico permitem visualizar o nível de congestionamento estimado pelo sistema, à medida que a velocidade e densidade de veículos vizinhos variam.

Nos gráficos do cenário $A$ (figura 3) se observa que o CARTIM obtém resultados similares ao COTEC, pois ambos os sistemas *fuzzy* estimaram níveis de congestionamento na mesma intensidade durante as simulações. O comportamento similar dos protocolos neste cenário $A$ valida a proposta *fuzzy* do CARTIM. De acordo com a figura 3, ainda se verifica que à medida que os conjuntos *fuzzy* de entrada do sistema variam, os níveis de congestionamento estimados também variam (de acordo com as regras *fuzzy* definidas).

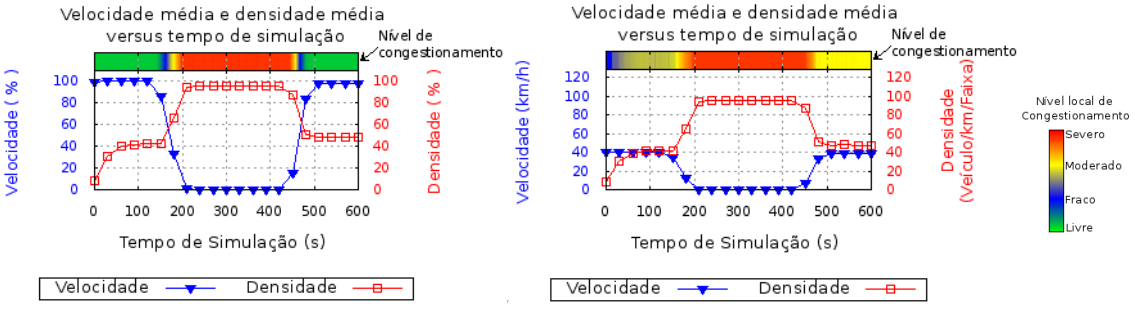
A figura 4 (a) mostra os níveis de congestionamento estimados pelo CARTIM no cenário $B$, bem como as velocidades e densidades médias dos veículos durante as simulações. Contudo, o COTEC neste cenário $B$ apresentou uma diferença considerável nos níveis de congestionamento, figura 4 (b), comparado com o CARTIM. Isso é atribuído ao *fuzzy* estático do protocolo, que apresenta uma desregulagem em cenários distintos do proposto para o protocolo. Assim, cenários com mais faixas (nas vias) afetariam ainda mais a estimativa de tráfego do COTEC.

As simulações mostraram que a proposta *fuzzy* dinâmica do CARTIM se adapta melhor ao cenário (contexto), obtendo informações mais precisas da situação do tráfego. É essa abordagem que ajuda no funcionamento do protocolo em cenários heterogêneos, comuns em redes veiculares. Além disso, ao estimar o tráfego veicular inferindo níveis de congestionamento mais elevados, o COTEC inicia antecipadamente o processo colaborativo (por se atingir o $C_{th}$) para validar as condições detectadas na via. Isso, no entanto, provoca um *overhead* desnecessário na rede.




(a) sistema *fuzzy* CARTIM (b) sistema *fuzzy* COTEC

**Figura 3. Nível de congestionamento no cenário *A*, aferido pela velocidade média e densidade média de veículos na autoestrada, versus tempo de simulação**




(a) sistema *fuzzy* CARTIM (b) sistema *fuzzy* COTEC

**Figura 4. Nível de congestionamento no cenário *B*, aferido pela velocidade média e densidade média de veículos na autoestrada, versus tempo de simulação**

## 5.2. Avaliação do CARTIM

**Cenário de Simulação:** Para avaliar o CARTIM, simulações foram realizadas em um cenário Manhattan Grid (10 x 10) sob uma região de 20,25 km$^2$. Esse tipo de mapa é comumente usado em redes veiculares, tendo como objetivo avaliar protocolos em ambientes urbanos. A interseção das ruas no mapa representa um cruzamento e cada rua possui extensão de 500 m, faixa única, mão, contramão e velocidade máxima permitida variando entre 50 e 60 km/h em ruas perpendiculares. No entanto, é importante salientar que não é objetivo deste trabalho propor uma solução para esse cenário específico, mas sim mostrar como o CARTIM pode auxiliar usuários das redes veiculares.

A tabela 4 apresenta os cinco cenários utilizados para avaliar o CARTIM. O número de veículos caracteriza a quantidade de nós no instante inicial da simulação. Com destinos aleatórios, cada veículo realizou uma viagem com distância mínima de 4 km. Os obstáculos representam estruturas (prédios etc.) nos cenários, que podem dificultar a comunicação via rádio entre os nós. A mudança de rota corresponde à proposta de minimização de congestionamento do CARTIM. A arquitetura da rede define como a rede veicular se organiza. O tempo de simulação em ambos os cenários foi de 10 minutos.

**Tabela 4. Cenários de simulação**

| Cenário | Nº de Veículos | Obstáculos | Mudança de rota | Arquitetura(s) | Tempo de Simulação |
|---|---|---|---|---|---|
| 1 | 1000 | Não | Não | V2V | 600 s |
| 2 | 1000 | Sim | Não | V2V | 600 s |
| 3 | 2000 | Sim | Não | V2V | 600 s |
| 4 | 2000 | Sim | Sim | V2V | 600 s |
| 5 | 2000 | Sim | Sim | V2V e V2I | 600 s |

Cada cenário foi simulado 33 vezes e o tempo total para simular um cenário variou em função da quantidade de veículos. Os cenário com 1000 veículos gastaram aproximadamente 50 minutos e os cenário com 2000 veículos cerca de 1 hora e 40 minutos.

Os gráficos desta seção que representam valores médios possuem intervalos de confiança de 95%. Contudo, alguns intervalos de confiança estão bastante próximos do valor médio, o que dificulta a visualização nas figuras.

**Mensagens de Congestionamento de Tráfego:** O processo de retransmissão de mensagens de congestionamento de tráfego, ilustrado na figura 2, foi avaliado nos cenários simulados. A tabela 5 discrimina a quantidade de mensagens de congestionamento de tráfego transmitidas, retransmitidas e recebidas durante as simulações. Os cenários 1 e 2, com menor densidade de veículos, tiveram maiores perdas de mensagens.

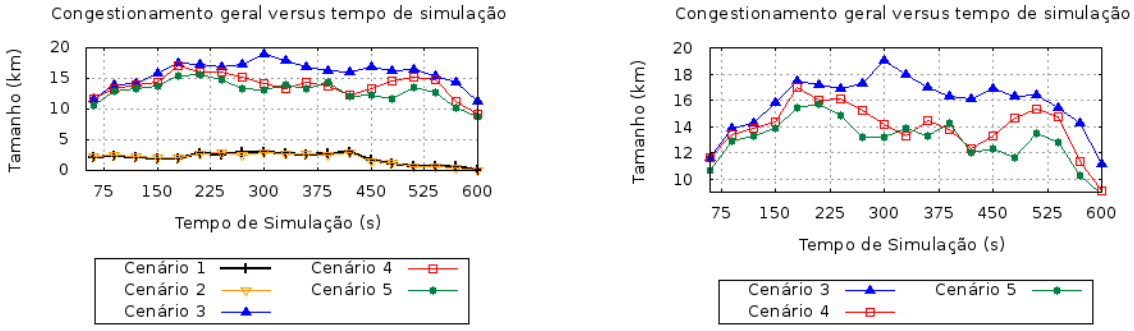
Contudo, esses cenários foram criados apenas para comparar e avaliar o processo de retransmissão de mensagens do CARTIM com presença de obstáculos. A tabela 5 mostra que o cenário 2 teve um aumento de aproximadamente 2.05% na taxa de mensagens perdidas, comparado com o cenário 1. Isso ocorreu em razão dos obstáculos. No entanto, devido à especificidade das redes veiculares, as taxas alcançadas não comprometeram o funcionamento do CARTIM. Além disso, à medida que os congestionamentos aumentam, a densidade de veículos na região também aumenta, o que diminui a mobilidade dos nós na rede, reduzindo assim a perda de mensagens. Essa afirmação é verificada na tabela 5 comparando as taxas de mensagens perdidas nos cenários 1 e 2 com os cenários 3, 4 e 5.

A heurística de minimização de congestionamento do CARTIM, adotada nos cenários 4 e 5, provocou uma ligeira elevação na taxa de perda de mensagens, comparado com o cenário 3. Isso ocorreu porque tal heurística, ao permitir mudanças de rotas, gera mais alterações na topologia da rede. A tabela 5 ainda mostra que no cenário 5 os veículos retransmitiram um número maior de mensagens, em relação aos veículos nos cenários 3 e 4. Isso ocorreu pois, devido à infraestrutura presente no cenário 5, as RSUs passaram a retransmitir mensagens CTA ao escutá-las.

**Sobrecarga de Comunicação:** É necessário avaliar o impacto da adoção do CARTIM em relação às mensagens de congestionamento de tráfego. A tabela 5 é utilizada para contabilizar o total de mensagens enviadas e retransmitidas pelos veículos. Essas mensagens possuem uma carga útil de aproximadamente 256 bytes (2048 bits) no protocolo. Nos cenários 3, 4 e 5 um total de 5047, 4990 e 5286 mensagens foram disseminadas respectivamente durante as simulações, o que corresponde às taxas médias de 8.4, 8.3 e 8.8 mensagens por segundo. Porém, essas taxas de mensagens disseminadas durante as simulações não foram suficientes e significativas para sobrecarregar a rede, uma vez que a taxa de transferência era de 6 Mbit/s ($6 \times 10^6$ bits/segundo). Nessas circunstâncias, a rede suportaria a transmissão de quase 3000 mensagens de congestionamento por segundo.

**Tamanho do Congestionamento:** A figura 5 mostra o tamanho total dos congestionamentos nos cenários por unidade de tempo. Essa métrica é avaliada por órgãos gestores do tráfego para quantificar a situação de momento no trânsito. Neste trabalho, tal métrica é utilizada para medir e avaliar a influência do CARTIM nos cenários simulados.

A figura 5 (b) mostra que o CARTIM ajudou a reduzir o tamanho do congestionamento geral nos cenários 4 e 5 durante as simulações. O cenário 3, que só utiliza a abordagem de identificação de congestionamento, obteve engarrafamentos maiores. Tal




(a) Todos os cenários (b) Cenários com 2000 veículos

**Figura 5. Congestionamento geral por cenário versus tempo de simulação**

fato ocorre pois veículos nos cenários 4 e 5, assim que escutam mensagens CTA na rede, obtêm aprovação para mudar de rota, evitando passar em vias sob condições severas de tráfego já validadas. O cenário 5, de acordo com o gráfico ilustrado na figura 5 (b), ainda obteve congestionamento geral durante boa parte da simulação menor que os verificados no cenário 4. Esse fato é atribuído à infraestrutura (RSUs), que aumenta o raio de comunicação ao retransmitir mensagens CTA a veículos mais distantes.

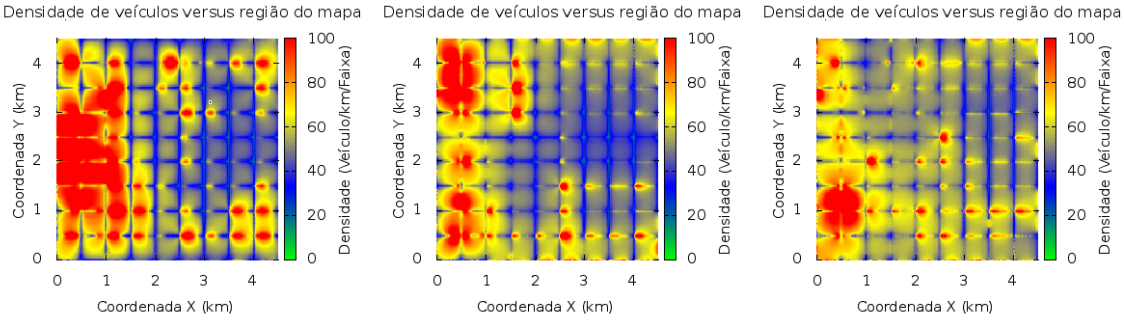
**Tabela 5. Mensagens de congestionamento por cenário**

| **Cenário** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| Transmitidas | 811 | 799 | 3413 | 3386 | 3378 |
| Retransmitidas | 54 | 53 | 1634 | 1604 | 1908 |
| Recebidas | 770 | 741 | 4599 | 4502 | 4766 |
| Perdidas (%) | 10.98 | 13.03 | 8.87 | 9.78 | 9.84 |

A figura 6 ilustra a densidade de veículos por região (inferida pelo sistema *fuzzy*) durante as simulações, nos cenários com 2000 veículos. O objetivo é observar o fluxo de veículos no mapa. A figura 6 (a) mostra uma grande região (em vermelho) com alta densidade de veículos no cenário 3, em que, em comparação com os cenários 4 e 5 (figuras 6 (b) e (c)), se verifica um espalhamento menor dos veículos sobre o mapa. Tal observação é feita em razão da minimização de congestionamento do CARTIM, que permite mudanças de rotas perante condições severas e, assim, tende a melhorar o fluxo de veículos nas vias.

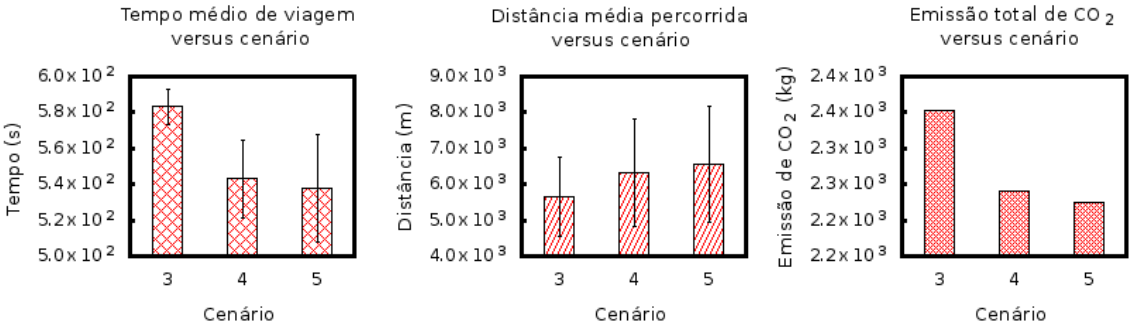
**Tempo de Viagem, Distância Percorrida e Emissão de $CO_2$:** O tempo de viagem é a métrica de interesse dos usuários das redes veiculares. A figura 7 (a) ilustra o tempo médio de viagem dos veículos por cenário. Em outras palavras, a figura mostra quanto tempo na média um veículo gastou no seu trajeto completo. Os dados apresentados na figura 7 (a) apontam que o CARTIM reduziu o tempo médio de viagem nos cenários 4 e 5 (em relação ao 3), pois nesses cenários a minimização de congestionamento foi utilizada pelos veículos. No cenário 5, como a infraestrutura estava presente, é verificado no gráfico da figura 7 (a) que os veículos realizaram os trajetos no menor tempo médio.

A distância percorrida e a emissão de $CO_2$ [Silva et al. 2013], [Sommer et al. 2011] são métricas utilizadas neste trabalho para avaliar a adoção do CARTIM. A figura 7 (b) mostra que a distância média percorrida pelos veículos nos cenários 4 e 5 aumentou (em razão das mudanças de rotas). Mas como o principal objetivo é reduzir o tempo de viagem, o aumento do trajeto não é significativo, ao ponto de comprometer os benefícios apresentados do CARTIM. A figura 7 (c) mostra que o




(a) Cenário 3 (b) Cenário 4 (c) Cenário 5

**Figura 6. Mapa de calor com a densidade de veículos por região do mapa. Estimado pelos veículos durante as simulações usando o CARTIM.**




(a) (b) (c)

**Figura 7. Tempo de viagem, distância percorrida e emissão de $CO_2$ por cenário**

CARTIM ainda reduziu a emissão de $CO_2$ nos cenários 4 e 5 comparado com cenário 3. Isso ocorreu devido à redução do tempo médio de viagem dos veículos nesses cenários.

## 6. Conclusões

Este trabalho apresentou o CARTIM, um protocolo de identificação e minimização de congestionamento de tráfego. O CARTIM foi planejado considerando as características e realidades das redes veiculares, buscando ser adaptável ao contexto e assim viável em cenários heterogêneos (autoestrada e vias urbanas). Essas características mencionadas distinguem o CARTIM das propostas encontradas na literatura.

Dois cenários foram utilizados para avaliar o protocolo proposto: (i) autoestrada de 2 km e (ii) Manhattan Grid (10 x 10). No cenário (i), os resultados mostraram a eficácia do sistema *fuzzy* dinâmico e comprovaram a capacidade do CARTIM em se adaptar ao contexto. Os resultados obtidos no cenário (ii) mostraram a capacidade do protocolo de minimizar congestionamentos, melhorando o fluxo de veículos. Dessa forma, o CARTIM atingiu a principal finalidade deste trabalho, reduzir o tempo médio de viagem dos veículos no tráfego, procurando aumentar a satisfação dos usuários da rede veicular. O protocolo ainda reduziu a emissão de $CO_2$ na atmosfera durante as simulações.

Como trabalho futuro, pode-se citar a simulação de cenários em mapas geográficos. Outras propostas de minimização e roteamento também devem ser avaliadas e comparadas. Protocolos de disseminação de mensagens *geocast* também podem ser implementados nas RSUs. Além disso, a utilização de microcontroladores e sensores pode viabilizar testes do CARTIM em cenários reais, com baixo custo financeiro por veículo.

## Referências

Alves, R. d. S., Campbell, I. d. V., Couto, R. d. S., Campista, M. E. M., Moraes, I. M., Rubinstein, M. G., Costa, L. H. M. K., Duarte, O. C. M. B., and Abdalla, M. (2009). Redes Veiculares: Princípios, Aplicações e Desafios. In *Minicursos do XXVII Simpósito Brasileiro de Redes de Computadores., chapter 5*, pages 199–254.

Bauza, R., Gozalvez, J., and Sanchez-Soriano, J. (2010). Road traffic congestion detection through cooperative vehicle-to-vehicle communications. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 606–612.

Binglei, X., Zheng, H., and Hongwei, M. (2008). Fuzzy-logic-based traffic incident detection algorithm for freeway. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 3, pages 1254–1259.

Board, N. R. C. U. T. R. (2000). *Highway Capacity Manual – HCM 2000*. Transportation Research Board, National Research Council.

DENATRAN. Departamento nacional de trânsito - frota de veículos. Disponível em `http://www.denatran.gov.br/frota.htm`. Acessado em: Julho 2013.

Fahmy, M. and Ranasinghe, D. N. (2008). Discovering automobile congestion and volume using vanet's. In *ITS Telecommunications, 2008. ITST 2008. 8th International Conference on*, pages 367–372.

Fukumoto, J., Sirokane, N., Ishikawa, Y., Wada, T., Ohtsuki, K., and Okada, H. (2007). Analytic method for real-time traffic problems by using Contents Oriented Communications in VANET. *2007 7th International Conference on ITS Telecommunications*, pages 1–6.

Lu, J. and Cao, L. (2003). Congestion evaluation from traffic flow information based on fuzzy logic. In *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, volume 1, pages 50–53 vol.1.

Martinez, F. J., Toh, C. K., Cano, J.-C., Calafate, C. T., and Manzoni, P. (2011). A survey and comparative study of simulators for vehicular ad hoc networks (vanets). *Wireless Communications and Mobile Computing*, 11(7):813–828.

Silva, F. A., Silva, T. R., Ruiz, L. B., and Loureiro, A. F. A. (2013). Um Middleware para Provisionamento de Contextos para Redes Veiculares. *XXXI Simpósito Brasileiro de Redes de Comutadores e Sistemas Distribuidos - SBRC*, pages 615–628.

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 *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10.

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.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 6*

## *Computação e Redes Móveis*

# Uma Análise do Impacto da Qualidade da Internet Móvel na Utilização de Cloudlets

**Philipp B. Costa**[1,2], **Paulo A. L. Rego**[1,2], **Emanuel F. Coutinho**[1,2], **Fernando A. M. Trinta**[1,2], **José N. de Souza**[1,2]

[1]Mestrado e Doutorado em Ciência da Computação (MDCC)
[2]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

{philippcosta,emanuel}@great.ufc.br, {pauloalr,trinta,neuman}@ufc.br

***Abstract.*** *An increasingly concern in the use of mobile devices is the battery life due to the increase of its embedded sensors and its features. The Mobile Cloud Computing (MCC) paradigm studies how to extend computational resources and the energy of mobile devices through the use of offloading techniques. In this context, cloudlets and the public clouds emerge as alternatives to the execution of cpu-intensive tasks of mobile devices. This paper presents an evaluation of the mobile Internet quality in three cities of the Brazilian northeast and a comparative study between the use of cloudlets and public cloud to perform offloading. Furthermore, the paper discusses the factors that impact the decision making of when and where to offloading.*

***Resumo.*** *Uma preocupação cada vez mais evidente no uso de dispositivos móveis é a autonomia da bateria em função do aumento de seus sensores embutidos e de suas funcionalidades. O paradigma de Mobile Cloud Computing (MCC) estuda formas de estender os recursos computacionais e energéticos dos dispositivos móveis, através da utilização das técnicas de offloading. Neste contexto, cloudlets e nuvens públicas surgem como alternativas para a execução de tarefas computacionalmente pesadas do dispositivo móvel. Este trabalho apresenta uma avaliação da qualidade da Internet móvel em três capitais do Nordeste do Brasil e um estudo comparativo entre o uso de cloudlets e a nuvem pública para realizar offloading. Além disso, o trabalho discute os fatores que impactam na tomada de decisão de quando e onde fazer offloading.*

## 1. Introdução

Com o surgimento dos *smartphones*, iniciou-se uma nova era para a computação móvel, na qual a inovação tecnológica passou a ocorrer em uma taxa acelerada no que diz respeito à capacidade de processamento, memória e armazenamento dos dados, além das constantes evoluções das redes de celulares [Bahl et al. 2012]. Com os avanços dessas redes, os *smartphones* agregaram ao longo do tempo diversas tecnologias, como WiMAX, WiFi, UMTS, HSDPA e LTE, que permitiram extrapolar o conceito de transferência entre canais de uma mesma tecnologia, para transferência entre tecnologias diferentes, encorajando assim novas pesquisas na área de mobilidade [Quental and Gonçalves 2013].

Apesar dos avanços tecnológicos, os *smartphones* continuam limitados computacionalmente, se comparados com um computador de mesa ou um *notebook*, por causa

de diversos fatores relacionados principalmente com suas dimensões físicas e restrições térmicas. Além disso, nos últimos anos, a questão energética tem se destacado como a maior limitação dos dispositivos móveis, dada a evolução mais rápida das tecnologias de *hardware*, em comparação com a evolução tecnológica empregada nas baterias [Satyanarayanan et al. 2011, Fernando et al. 2013].

Uma das estratégias utilizadas para aumentar o desempenho do dispositivo móvel e reduzir o consumo de energia é a execução de tarefas computacionais em servidores remotos, técnica conhecida na literatura como *offloading* [Cuervo et al. 2010] ou *cyber foraging* [Satyanarayanan 2001]. Esta técnica consiste na migração do processamento ou dados dos dispositivos móveis para outra infraestrutura, com maior capacidade computacional e de armazenamento. Segundo [Verbelen et al. 2012], o *offloading* pode ser executado em máquinas virtuais de uma nuvem pública ou em qualquer máquina que esteja na mesma rede local em que estão os dispositivos móveis, sendo denominado, nesse caso, de *cloudlet*.

O conceito de *cloudlet* foi introduzido por [Satyanarayanan et al. 2011] e tem como principal objetivo trazer os serviços de *offloading* para serem executados em máquinas locais. A ideia é utilizar as redes WiFi, que possuem em geral velocidades maiores e latências menores, por serem menos congestionadas do que as redes de Internet móvel. Como consequência, um *cloudlet* consegue entregar um serviço de melhor qualidade, principalmente se empregado numa determinada categoria de aplicações que são sensíveis à percepção da latência, como aplicativos de processamento de imagem, visão computacional, realidade aumentada, reconhecimento de voz e jogos [Bahl et al. 2012].

O paradigma *Mobile Cloud Computing* (MCC) surgiu nesse contexto e envolve uma combinação de tecnologias em diversas áreas, sendo as principais: computação em nuvem, computação móvel e redes sem fio. [Shiraz et al. 2013] definem MCC como o mais recente paradigma computacional prático, que estende a visão da computação utilitária da computação em nuvem, para aumentar os recursos computacionais e energéticos dos dispositivos móveis, através da utilização das técnicas de *offloading*.

O conceito de MCC surgiu com o objetivo de fornecer uma variedade de serviços, equivalentes aos da nuvem, adaptados à capacidade dos dispositivos com recursos limitados, o que passa também pela melhoria das infraestruturas de telecomunicações, a fim de aperfeiçoar o provisionamento de serviços [Dinh et al. 2011].

Diversos trabalhos [Cuervo et al. 2010, Chun et al. 2011, Kosta et al. 2012, Barbera et al. 2013] mostraram como a qualidade das redes sem fio influencia no desempenho da técnica de *offloading*. A literatura apresenta resultados em que o ganho de desempenho é maior ao utilizar *cloudlets*, se comparado ao uso de recursos da nuvem pública, acessados através da Internet. Entretanto, estes estudos não abrangem em seus experimentos a utilização das redes 4G LTE, que foram planejadas para ter qualidade e velocidade superior à geração anterior da rede móvel.

Neste contexto, este trabalho apresenta uma avaliação da qualidade da Internet móvel em algumas capitais do Brasil e um estudo comparativo entre o uso de *cloudlets* e a nuvem pública para realizar *offloading*. Neste trabalho, o impacto do uso das redes móveis de quarta geração (4G LTE) no desempenho da aplicação, ao utilizar *offloading*, foi avaliado.

O trabalho está organizado da seguinte maneira: a Seção 2 apresenta uma discussão sobre os fatores que impactam na decisão de quando fazer *offloading*; na Seção 3, os trabalhos relacionados são apresentados; a Seção 4 apresenta os experimentos que foram realizados para avaliar a qualidade da Internet, comparar o uso de *cloudlet* e nuvem pública e avaliar o impacto do uso do 4G; finalmente, a Seção 5 apresenta a conclusão e trabalhos futuros.

## 2. Quando Fazer *Offloading*?

Enviar o processamento para outra máquina não é uma ideia nova. O tradicional modelo cliente-servidor é bastante usado para esse fim, assim como o *offloading*. O *offloading* de processamento é uma técnica que data da época dos "terminais burros", que utilizavam *mainframes* para o processamento. Com a introdução dos computadores pessoais, a necessidade de descarregar (ou migrar) o processamento diminuiu. Porém, com o surgimento dos dispositivos portáteis, uma nova necessidade de poder de computação remota surgiu [Dinh et al. 2011].

*Offloading* é diferente do modelo cliente-servidor tradicional, em que o cliente sempre solicita um conteúdo ou função do servidor. É também diferente do modelo de migração usado em sistema de multiprocessadores e grades computacionais, onde um processo pode ser migrado para fins de balanceamento de carga [Kumar et al. 2013]. Hoje, as técnicas de *offloading* estão sendo utilizadas para aumentar a capacidade computacional e de armazenamento dos dispositivos móveis, ao migrar processamento e dados para computadores com mais recursos, visando aumentar o desempenho e economizar energia do dispositivo. Porém, utilizar *offloading* nem sempre aumenta o desempenho do dispositivo.

O trabalho [Kumar et al. 2013] apresenta um modelo analítico para responder a pergunta *"Quando a técnica de offloading aumenta o desempenho do dispositivo móvel?"*. O modelo compara o tempo para executar todo o processamento de uma tarefa da aplicação no dispositivo móvel ($\frac{W}{P_m}$) e o tempo para transferir os dados e executar o processamento fora do dispositivo ($\frac{D_u}{V_u} + \frac{W}{P_c} + \frac{D_d}{V_d}$), seja na nuvem ou em um *cloudlet*. O modelo considera os seguintes parâmetros: $W$ é o total de computação a ser realizada; $P_m$ é o poder de processamento do dispositivo móvel; $P_c$ é o poder de processamento da nuvem; $D_u$ é a quantidade de dados enviados do dispositivo para a nuvem, enquanto $D_d$ são os dados recebidos pelo dispositivo; e $V_u$ e $V_d$ são as vazões de *upload* e *download*, respectivamente.

$$\frac{W}{P_m} > \frac{D_u}{V_u} + \frac{W}{P_c} + \frac{D_d}{V_d} \tag{1}$$

A técnica de *offloading* melhora o desempenho quando a Equação (1) é satisfeita. Além de ganhar desempenho, ao utilizar *offloading*, o dispositivo móvel pode economizar energia [Fernando et al. 2013]. Analisando o modelo, pode-se notar que, para ganhar desempenho, a computação deve ser pesada (alto valor para $W$) e a comunicação entre o dispositivo móvel e a nuvem deve ser leve (baixo valor para $\frac{D_u}{V_u} + \frac{D_d}{V_d}$), seja por transferir poucos dados, seja por ter uma alta vazão. Nota-se também que aumentar a diferença entre o poder de processamento do dispositivo móvel e a nuvem não traz grande impacto para a decisão. Este fato pode ser observado em (2), ao considerar que a nuvem é $K$ vezes mais rápida do que o *smartphone* ($P_c = K \times P_m$). Nota-se que, para grandes valores de

$K$, (1) se simplifica para $\frac{W}{P_m} > \frac{D_u}{V_u} + \frac{D_d}{V_d}$. Resumindo, a Figura 1 ilustra o *tradeoff* entre a quantidade de comunicação e computação para decidir se deve-se ou não fazer *offloading*.

$$W \times \left(\frac{1}{P_m} - \frac{1}{P_c}\right) > \frac{D_u}{V_u} + \frac{D_d}{V_d} \Rightarrow W \times \left(\frac{K-1}{K \times P_m}\right) > \frac{D_u}{V_u} + \frac{D_d}{V_d} \Rightarrow \frac{W}{P_m} > \frac{D_u}{V_u} + \frac{D_d}{V_d} \quad (2)$$

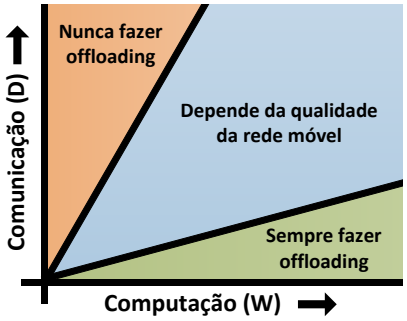



**Figura 1. *Tradeoff*: decisão sobre *offloading*. Fonte: [Kumar and Lu 2010].**

## 3. Trabalhos Relacionados

Esta seção foi dividida em duas subseções. A Subseção 3.1 apresenta os trabalhos relacionados ao tema medição da rede móvel, enquanto a Subseção 3.2 apresenta os trabalhos que utilizaram *offloading* e mensuraram seu desempenho para diferentes redes móveis.

### 3.1. Medição da Rede Móvel

Existem trabalhos na literatura que realizaram medição de diferentes redes móveis (e.g., 2G, 3G, WiFi, WiMaX, 4G LTE). Entretanto, até onde foi pesquisado, nenhum outro trabalho científico realizou experimento que medisse a rede móvel brasileira.

[Sommers and Barford 2012] compararam o desempenho da rede celular e WiFi, usando dados de milhões de usuários, obtidos com a ferramenta Speedtest[1]. O experimento cobriu 15 áreas metropolitanas dos EUA, Europa e Ásia e focou nas métricas taxa de *download*, *upload* e latência. [Shepard et al. 2011] apresentam o LiveLab, uma metodologia para realizar estudos de campo e medir a qualidade das redes sem fio e o uso dos *smartphones*. Para demostrar a viabilidade e capacidade do LiveLab, 25 usuários utilizaram aparelhos configurados com o LiveLab durante 6 meses. O estudo, além de medir a qualidade da rede móvel, caracteriza o uso dos dispositivos (e.g., para jogar, se comunicar, uso de mapas, navegar na Internet, etc).

[Chen et al. 2013] realizaram experimentos para medir e comparar a rede móvel de 3 operadoras dos EUA (duas delas com a tecnologia 4G LTE e uma com 3G CDMA). Apesar de calcular vazão e latência, o foco do trabalho é avaliar o desempenho de utilizar TCP de múltiplos caminhos (MPTCP - *Multipath* TCP) em uma ambiente sem fio, formado por redes 3G, 4G e WiFi. [Huang et al. 2012] também coletaram dados de *smartphones* utilizando aplicações, tais como 3GTest[2] e 4GTest. Os autores realizaram

[1]Speedtest - http://speedtest.net.
[2]MobiPerf (3GTest) - http://mobiperf.com.

dois experimentos principais. No primeiro, com os resultados obtidos através da execução da aplicação 4GTest por mais de 3000 usuários nos EUA, eles mediram taxa de *download* e *upload*, RTT (*Round-Trip Time*) e *jitter* de redes WiFi, 3G e 4G. No segundo (foco do trabalho), os autores geraram um modelo de consumo de energia para redes 4G LTE, ao analisar dados que detalhavam o uso dos *smartphones* de 20 usuários, durante cinco meses.

Apesar de alguns trabalhos disponibilizarem os aplicativos de medição da rede, o desenvolvimento das aplicações utilizadas nesse trabalho foi necessário, pois permitiu que o servidor de destino fosse configurado no Brasil, o que influencia nos resultados de algumas métricas. Além disso, foi possível adicionar outras métricas (e.g., taxa de perda de pacotes, localização) e um coletor que detecta o tipo de rede móvel utilizada (e.g., EDGE, UMTS, HSPA, HSPA+, LTE, etc).

Além desses trabalhos científicos, o aplicativo desenvolvido pela Entidade Aferidora de Qualidade[3] (EAQ), criada em atendimento às resoluções 574 e 575 da Agência Nacional de Telecomunicações do Brasil (ANATEL), merece destaque, pois permite que dispositivos móveis das plataformas Android e iOS afiram indicadores de qualidade das redes de telecomunicações. Este aplicativo afere as mesmas métricas de rede que foram utilizadas nesse trabalho, mas não permite identificar o tipo de rede móvel ativa, nem a localização, por isso não foi utilizado.

### 3.2. *Offloading*: *Cloudlet* x Nuvem Pública e Rede Móvel x WiFi

Muitos trabalhos recentes focam no *offloading* de processamento intensivo dos *smartphones* para servidores remotos na nuvem, com o objetivo de aumentar a autonomia da bateria e reduzir o tempo de execução das aplicações. MAUI [Cuervo et al. 2010] utiliza chamada remota de procedimento para executar métodos complexos na nuvem. Ele tem um sistema de decisão que, baseado na qualidade da rede, decide quais métodos devem ser executados localmente e quais devem ser executados na nuvem. Durante os experimentos, os autores simularam redes WiFi com diferentes taxas de RTT e compararam a execução local com execuções no *cloudlet* e na nuvem pública (via WiFi e 3G).

Alguns trabalhos utilizam virtualização para criar um clone do *smartphone* com sistema operacional Android na nuvem pública e, assim, executar tarefas fora do dispositivo móvel. [Kosta et al. 2012] apresentam o ThinkAir, um *framework* que explora o conceito de virtualização de *smartphones* na nuvem (i.e., clones) para fazer *offloading* de métodos. Os autores desenvolveram quatro aplicações de *benchmark* e compararam a execução no dispositivo móvel com a execução parcial na nuvem (via WiFi dedicado, *hotspot* WiFi compartilhado e 3G). CloneCloud [Chun et al. 2011] e COMET [Gordon et al. 2012] também utilizaram clones para realizar tarefas do dispositivo móvel e avaliaram a utilização de WiFi e 3G para comunicação entre o dispositivo móvel e a nuvem. Ambas as soluções fizeram alterações na máquina virtual do Android (DalvikVM). A solução CloneCloud identifica os métodos da aplicação que fazem processamento intensivo e particiona os aplicativos automaticamente, possibilitando a migração automática de *threads* entre o dispositivo móvel e o clone, a fim de otimizar o tempo de execução e o consumo de energia do dispositivo móvel. Já os autores da solução COMET implantaram um sistema de memória distribuída e compartilhada na DalvikVM, para permitir

[3]Brasil Banda Larga - http://www.brasilbandalarga.com.br.

que *threads* pudessem ser migradas de maneira transparente entre o dispositivo móvel e a nuvem, possibilitando o *offloading* de qualquer aplicação.

O trabalho [Barbera et al. 2013] apresenta duas estratégias para sincronizar o *smartphone* e o clone. Os autores avaliaram o impacto da banda e latência de redes 2G, 3G e WiFi nas duas estratégias e no consumo de energia do dispositivo. À época dos experimentos, a cobertura da rede móvel em Roma (Itália) e Cambridge (Reino Unido) não era a ideal (chegando a estar disponível apenas a rede 2G em alguns momentos). Além disso, os resultados mostram maior desempenho e economia de energia ao utilizar WiFi, se comparado às redes 2G e 3G. Assim como nos outros trabalhos, não foram realizados testes para avaliar o impacto de utilizar a rede 4G para fazer *offloading*.

## 4. Experimentos de Avaliação e Comparação

Nesta seção, são apresentados os experimentos realizados para avaliar a qualidade da Internet móvel em algumas capitais do Brasil e para comparar o desempenho da aplicação ao usar *cloudlet* e a nuvem pública para realizar *offloading*. A Subseção 4.1 descreve a metodologia utilizada para a experimentação; as subseções 4.2, 4.3 e 4.4 apresentam os experimentos e seus resultados; e a Subseção 4.5 faz uma breve discussão sobre os resultados obtidos.

### 4.1. Metodologia

Para a realização desse trabalho, duas aplicações para a plataforma Android foram desenvolvidas e estão disponíveis para a comunidade[4].

**NETester:** Esta aplicação realiza operações que permitem identificar taxa de *upload*, *download*, RTT, total de pacotes perdidos e *jitter* entre o dispositivo móvel e máquinas virtuais da Amazon EC2[5], em São Paulo. A aplicação foi desenvolvida baseada nas ideias e metodologia de [Huang et al. 2012] e nos cálculos de [Demichelis and Chimento 2002]. Além disso, a aplicação tem um modo de execução (SignalHunter) que captura, de tempos em tempos, o tipo de conexão que está habilitada no dispositivo móvel no momento (e.g., EDGE, UMTS, HSDPA, HSPA+, LTE), a fim de avaliar a estabilidade da conexão e a cobertura da rede.

**BenchImage:** Esta aplicação de processamento de imagem aplica efeitos (filtros) em fotos. O aplicativo foi desenvolvido com as opções de executar totalmente no dispositivo móvel e executar parcialmente, ao fazer *offloading* da tarefa para a nuvem pública ou para o *cloudlet*. Ao executar fora do dispositivo móvel, a aplicação transfere a foto para o servidor remoto e a recebe após o efeito ser aplicado.

Os serviços de dados de quatro operadoras (chamadas de Operadora 1, Operadora 2, Operadora 3 e Operadora 4) foram contratados para avaliação. Apenas três operadoras (1, 2 e 3) disponibilizavam serviço de dados 4G LTE à data da aquisição dos planos de dados. Dessa forma, um serviço de dados 3G foi utilizado para a Operadora 4.

Os experimentos foram realizados em três cidades (Fortaleza, Salvador e Recife), com a maioria dos experimentos sendo realizados em Fortaleza. Salvador e Recife foram

[4]As duas aplicações estão disponíveis no Google Play. Os códigos-fonte e mais detalhes podem ser encontrados em http://github.com/ufc-great/apps.

[5]Amazon EC2 - http://aws.amazon.com/ec2.

escolhidas pela proximidade de Fortaleza e por também serem cidades-sede da Copa do Mundo de 2014, fazendo assim parte do grupo das cidades que já dispõem de infraestrutura e ofertas de planos de dados 4G.

### 4.2. Experimento 1 - Medição da Qualidade da Internet Móvel

O primeiro experimento consistiu de 10 execuções da aplicação NETester para cada operadora, em quatro localidades das três cidades. Em cada local, as operadoras foram sorteadas para definir a ordem das execuções, que foram realizadas no mesmo dia e horário, uma após a outra, para estarem sujeitas às mesmas condições de rede nas máquinas virtuais da Amazon EC2. Além disso, o mesmo *smartphone* foi utilizado em todas as execuções, para evitar que configurações de *hardware* e *software* do dispositivo (e.g., tamanho da janela TCP) afetassem os resultados. Os resultados das execuções foram agrupados para calcular média, limite inferior e superior, com confiabilidade de 95%.

A Figura 2 apresenta o resultado do Experimento 1. Nota-se que a variabilidade das taxas de *download* e *upload* é grande, e a qualidade da Internet móvel depende da operadora e da localidade. Além disso, percebe-se que nem todas as operadoras estão com a rede 4G cobrindo todas as localidades, apesar das cidades escolhidas serem cidades-sede da Copa do Mundo de 2014, e os experimentos apresentados terem sido realizados em aeroportos (porta de entrada dos turistas e foco dos investimentos iniciais do setor). Das operadoras com plano de dados 4G, a Operadora 1 foi a única em que a rede 4G estava disponível em todas as localidades. A rede 4G da Operadora 2 só não estava disponível no aeroporto de Recife, que foi o único local em que a rede 4G da Operadora 3 estava disponível. Estes resultados mostram que as redes ainda estão sendo estruturadas e que mais investimentos são necessários para expandir a cobertura.

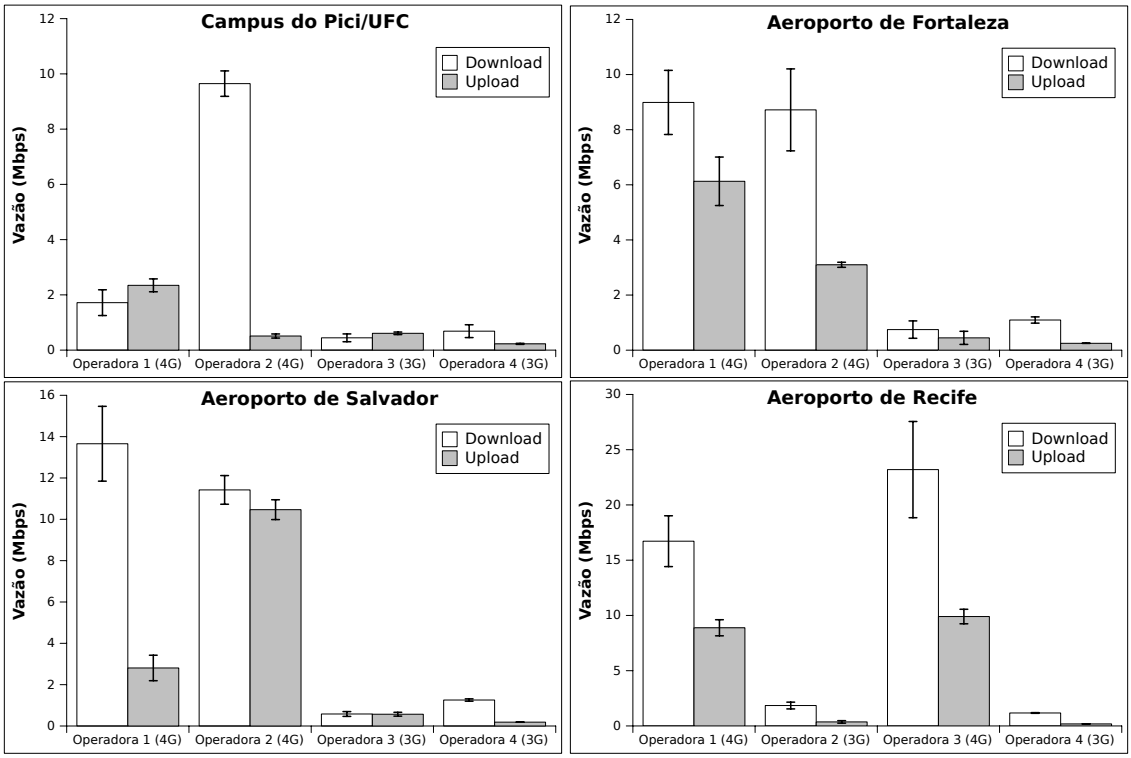



**Figura 2. Taxa de *download* e *upload* da rede móvel das operadoras.**

As operadoras 1, 2 e 3, nas localidades em que a rede 4G estava disponível, conseguem prover taxas de transmissão similares às conexões a cabo de banda larga, com uma variação de 2 a 25 Mbps para *download* e de 1 a 12 Mbps para *upload*. Estas taxas são consideravelmente maiores do que as obtidas em redes 3G (vide resultados da Operadora 4) e estão de acordo com estudos mais amplos, como o realizado pela AT&T nos EUA [Huang et al. 2012], que tem redes mais maduras do que as brasileiras.

Este experimento também mensurou, para cada operadora, o RTT entre o dispositivo móvel e máquinas virtuais localizadas no *datacenter* da Amazon, em São Paulo. A Figura 3 apresenta o gráfico com os resultados. Pode-se observar que o menor RTT foi obtido pela Operadora 2 e foi de 60 ms em média, enquanto o RTT mais alto chegou a aproximadamente 180 ms, no aeroporto de Recife, quando a rede 4G não estava disponível. A Operadora 1, apesar de ter cobertura 4G em todas as localidades, teve uma alta variabilidade no RTT (variando de 90 a 160 ms, aproximadamente). Tamanha variação do RTT pode causar degradação de desempenho, principalmente das aplicações de tempo real [Cuervo et al. 2010]. Este é um fator importante para garantir a qualidade do serviço, e precisa ser considerado, principalmente quando o número de usuários dos planos de dados 4G aumentar, o que tende a elevar os congestionamentos na rede.

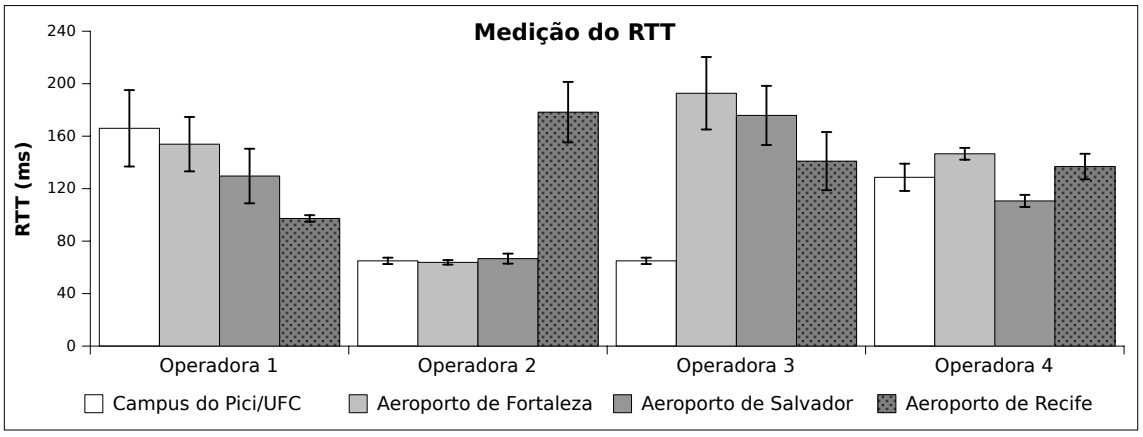



**Figura 3. RTT da rede móvel das operadoras em diferentes localidades.**

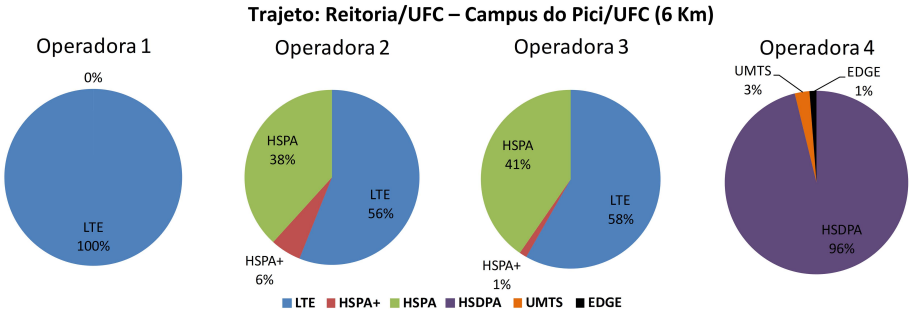
### 4.3. Experimento 2 - Medição da Cobertura da Rede 4G

Analisando os resultados do experimento anterior, que considera diferentes cidades, pode-se afirmar que a cobertura da rede 4G ainda está aquém da ideal. Já no Experimento 2, o objetivo é avaliar a cobertura da rede para pequenos trajetos dentro da mesma cidade (Fortaleza), simulando um trajeto típico do dia a dia de um usuário.

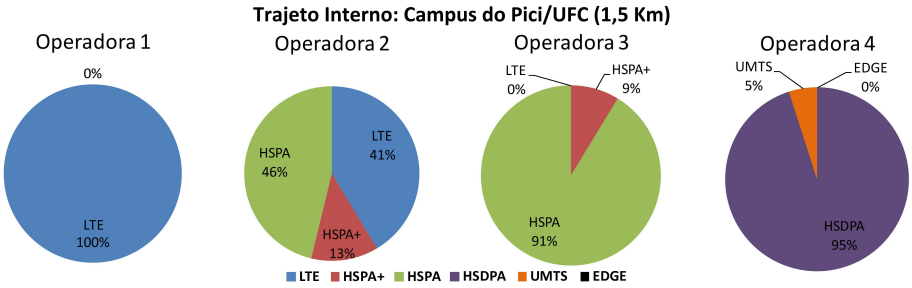
O experimento consistiu de duas execuções da aplicação NETester para cada operadora, no modo de execução SignalHunter com tempo de coleta de um em um segundo. A primeira execução foi feita durante uma caminhada de 1,5 Km no campus da Universidade Federal do Ceará (UFC), enquanto a segunda execução foi realizada durante um trajeto de 6 Km, de carro, entre o campus e a reitora da universidade. O mesmo *smartphone* foi utilizado em todas as execuções, para evitar que o *hardware* do dispositivo afetasse os resultados.

A Figura 4 apresenta o resultado para os dois trajetos. Pode-se observar que a Operadora 1 é a única a prover cobertura de rede 4G para os dois trajetos completos.

As Operadoras 2 e 3 cobrem parcialmente os trajetos, enquanto a Operadora 4 teve o pior desempenho, uma vez que inclusive a cobertura 3G é insuficiente, pois, em certos momentos dos trajetos, as únicas redes disponíveis eram a UMTS e EDGE.




(a) Conexão 4G: Operadoras 1, 2 e 3. Conexão 3G: Operadora 4.




(b) Conexão 4G: Operadoras 1 e 2. Conexão 3G: Operadoras 3 e 4.

**Figura 4. Resultado do Experimento 2: Cobertura da rede durante dois trajetos.**

A cobertura da maioria das operadoras precisa ser melhorada para que as aplicações móveis possam utilizar os serviços através da Internet sem sofrer com degradação do serviço. Inclusive as trocas entre células e o *handover* entre diferentes tipos de redes devem ser feitos de maneira transparente e sem perda de conectividade para o usuário, pois estes fatores impactam na qualidade do serviço.

### 4.4. Experimento 3 - Onde Fazer *Offloading* e o Impacto da Rede 4G

Para mostrar na prática as implicações do *tradeoff* apresentado na Seção 2, o terceiro experimento foi realizado utilizando a aplicação de processamento de imagem. Neste experimento, a aplicação foi executada 10 vezes para cada tamanho de imagem (0,3 MP, 1 MP, 2 MP, 4 MP e 8 MP), aplicando o efeito chamado Cartoonizer[6] em seis diferentes ambientes de execução. A aplicação foi executada totalmente no dispositivo móvel e fazendo *offloading* em *cloudlets* e na nuvem pública. Dois *smartphones* de modelos diferentes foram utilizados para aplicar o efeito localmente. Nos casos em que o *offloading* foi utilizado, todas as execuções foram realizadas no mesmo *smartphone*, a computação do efeito foi realizada em dois tipos diferentes de instância da Amazon EC2 e em dois *cloudlets* com configurações distintas. A Tabela 1 apresenta detalhes sobre os ambientes de execução.

Para realizar *offloading* de processamento, o dispositivo móvel precisa trocar dados com o servidor remoto. Por isso, para executar no *cloudlet*, o *smartphone* utilizou

[6]Cartoonizer aplica quatro filtros na imagem, dentre eles o desenfoque gaussiano e o ColorDodgeBlend.

**Tabela 1. Configuração dos Ambientes de Execução**

| Ambiente de Execução | Configuração |
|---|---|
| LG Local | LG Optimus G E977<br>Processador Qualcomm Krait, 4 cores, 1.5 Ghz e 2 GB de memória |
| S3 Local | Samsung GT-i8190 Galaxy S III Mini<br>Processador ARM Cortex-A9, 2 cores, 1 Ghz e 1 GB de memória |
| Cloudlet Ci5 | Ubuntu Desktop 12.04 64-bit<br>Processador Intel Core i5-4570, 4 cores, 3.2 Ghz e 16 GB de memória |
| Cloudlet C2D | Ubuntu Desktop 12.04 64-bit<br>Processador Intel Core 2 Duo T5500, 2 cores, 1.66 Ghz e 3 GB de memória |
| EC2 Medium | Ubuntu Server 12.04 64-bit. Tipo de instância: c1.medium<br>2 VCPUs, 5 ECUs e 1,7 GB de memória |
| EC2 Micro | Ubuntu Server 12.04 64-bit. Tipo de instância: t1.micro<br>1 VCPU, 1∼2 ECU e 0,615 GB de memória |

uma rede WiFi dedicada (ponto de acesso TP-Link WR740N, 802.11n), enquanto para executar na nuvem, foram utilizadas as redes WiFi (mesmo ponto de acesso conectado à banda larga ADSL2+ com contrato de 10 Mbps para *download* e 0,5 Mbps para *upload*) e 4G LTE (da Operadora 1, que obteve melhor desempenho no Experimento 1).

**Tempo Total.** A Tabela 2 apresenta o tempo total de execução do aplicativo de processamento de imagem (média e intervalo de confiança), para cada tamanho de foto e ambiente de execução. Pode-se observar que a execução local foi a mais demorada, independente do tamanho das fotos. Ou seja, para essa aplicação, fazer *offloading* em todos os casos é a melhor opção, mesmo para a imagem de 0,3 MP.

**Tabela 2. Tempo total de execução (segundos)**

| | 8 MP | 4 MP | 2 MP | 1 MP | 0,3 MP |
|---|---|---|---|---|---|
| LG Local | 176,67 ± 2,17 | 86,94 ± 0,72 | 46,04 ± 0,60 | 24,44 ± 0,21 | 7,10 ± 0,06 |
| S3 Local | 219,61 ± 2,68 | 109,73 ± 0,93 | 57,16 ± 0,68 | 30,28 ± 0,16 | 8,94 ± 0,04 |
| Cloudlet Ci5 | 11,71 ± 1,05 | 5,47 ± 0,12 | 3,20 ± 0,04 | 1,70 ± 0,05 | 0,68 ± 0,04 |
| Cloudlet C2D | 31,01 ± 5,96 | 15,31 ± 2,37 | 8,85 ± 2,13 | 3,78 ± 0,06 | 1,42 ± 0,07 |
| EC2 Micro (WiFi) | 124,85 ± 11,47 | 34,87 ± 5,88 | 19,05 ± 0,94 | 10,89 ± 0,38 | 6,28 ± 0,22 |
| EC2 Micro (4G) | 71,76 ± 8,26 | 14,07 ± 3,38 | 14,20 ± 2,49 | 5,11 ± 0,51 | 2,65 ± 0,25 |
| EC2 Medium (WiFi) | 94,55 ± 0,79 | 33,90 ± 4,71 | 17,86 ± 0,37 | 9,55 ± 0,20 | 5,66 ± 0,08 |
| EC2 Medium (4G) | 27,42 ± 2,95 | 12,26 ± 0,20 | 9,86 ± 2,37 | 5,38 ± 0,50 | 2,56 ± 0,13 |

**Tempo de Processamento.** Para visualizar melhor a diferença entre o tempo total de execução em cada ambiente, a Figura 5 apresenta o tempo de computação do efeito Cartoonizer para fotos de 8, 2 e 0,3 MP. É importante ressaltar que o tempo de transferência da foto entre o dispositivo móvel e o servidor remoto foi desconsiderado, assim como o tempo gasto pelo dispositivo móvel para carregar e salvar a foto, que foi de aproximadamente 1 segundo.

A computação do efeito no Cloudlet Ci5 é aproximadamente 24 vezes mais rápida do que no *smartphone* mais potente, que foi de aproximadamente 175 segundos (tempo de execução total menos tempo para carregar e salvar a foto), e aproximadamente 2,7 vezes mais rápida do que no Cloudlet C2D, ambas para a foto de 8MP. Enquanto a execução na EC2 Medium é aproximadamente 12 vezes mais rápida do que no *smartphone* e aproximadamente 4 vezes mais rápida do que na instância EC2 Micro. Os ambientes de execução

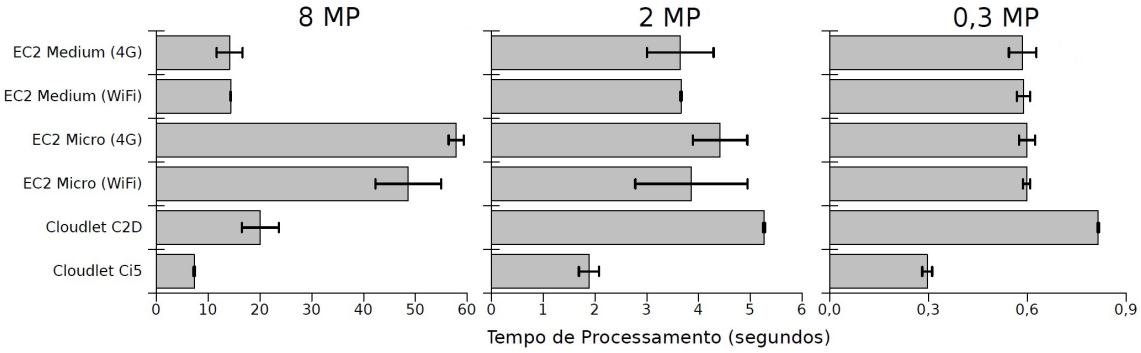



**Figura 5. Tempo de processamento para diferentes tamanhos de fotos.**

com melhor configuração têm um menor tempo de processamento, por isso a diferença entre o tempo de processamento nas diferentes instâncias da nuvem e nos dois tipos de *cloudlet* era esperada, e está de acordo com a Fórmula 2, apresentada na Seção 2. Porém, para fotos pequenas (pouca computação) a diferença entre os tempos de processamento é pouca.

**Tempo de Transferência.** A Figura 6 compara o tempo médio para fazer *download* e *upload* das fotos, entre o dispositivo móvel e o *cloudlet* (usando a rede WiFi dedicada) e entre o dispositivo móvel e a nuvem pública (usando WiFi e 4G). Pode-se perceber que o tempo de transferência é menor na rede WiFi local (802.11n), pois a taxa de transferência não está limitada por provedor, uma vez que não passa pela Internet. Ressalta-se que a instância EC2 Micro aplica o efeito mais rápido do que o Cloudlet C2D para as fotos de 0,3 a 4 MP, por causa da tecnologia de economia de energia Intel SpeedStep, presente no Cloudlet C2D. Apesar disso, o tempo gasto com a transferência de dados (seja com WiFi ou 4G) entre o dispositivo e a nuvem pública faz com que o tempo total de execução usando o Cloudlet C2D seja menor. Este fato mostra que o tempo gasto com transferência de dados é fator que impacta fortemente no tempo total de execução.

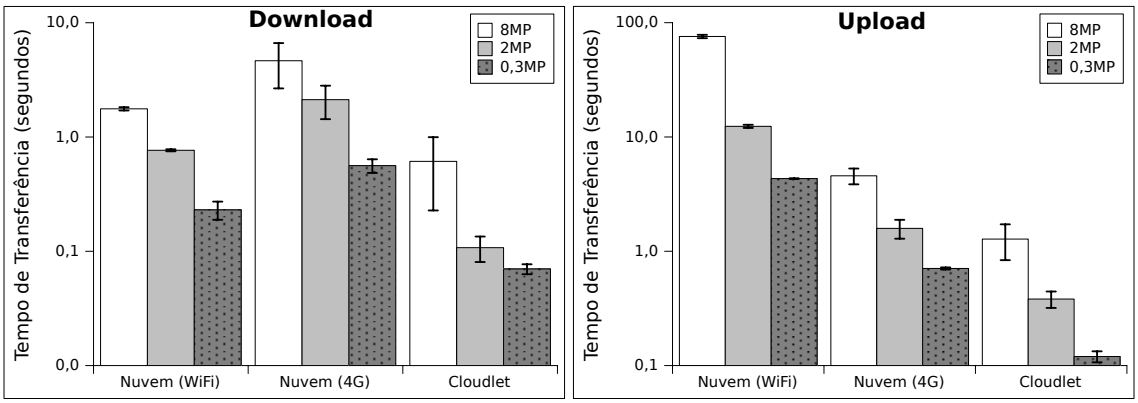



**Figura 6. Tempo gasto com transferência de dados (segundos).**

Os resultados mostram que fazer *offloading* no Cloudlet Ci5 gera o maior ganho de desempenho para a aplicação. Em geral, ao utilizar os *cloudlets*, obtêm-se os menores tempos de execução. Esse fato só é modificado ao comparar o tempo total de execução no Cloudlet C2D e na instância EC2 Medium, para fotos maiores que 2 MP e transferência de dados usando 4G. A explicação para este fato é que a instância EC2 Medium faz a

computação do efeito mais rápido, ganhando alguns segundos de vantagem, e a transferência de dados com 4G não é tão mais lenta do que na rede WiFi dedicada a ponto de perder a vantagem. Este é um cenário em que a qualidade da rede 4G trouxe vantagens, uma vez que os *cloudlets* sempre teriam vantagens se a rede 3G tradicional fosse utilizada. O uso da rede 4G tornou a decisão de quando e onde fazer *offloading* mais difícil.

**4G x Banda Larga Cabeada.** Percebe-se que o uso do 4G e sua alta taxa de *download* e *upload* fez com que o tempo total de execução na instância EC2 Medium diminuísse aproximadamente 3,4 vezes para a foto de 8 MP, se comparado ao uso do WiFi ligado à banda larga ADSL2+. Este fato mostra que o uso do 4G pode fortalecer e impulsionar a adoção das técnicas de *offloading*, pois as aplicações podem se aproveitar da conectividade com mobilidade das redes móveis e, por conseguinte, das altas taxas de *download* e, principalmente, de *upload*, uma vez que, por enquanto, estas taxas ainda não estão sendo limitadas pelas operadoras.

### 4.5. Discussão dos Resultados

Os três experimentos, apresentados nas subseções anteriores, mostraram que a Internet 4G brasileira pode alcançar velocidades equivalentes à Internet a cabo, mas revelaram a existência de uma grande variação no RTT. Além disso, os experimentos mostraram que a cobertura da rede ainda está longe da ideal e que existem diversos fatores, relacionados à qualidade da Internet, que impactam diretamente na decisão de quando e onde fazer *offloading*.

O experimento que utilizou a aplicação BenchImage mostrou que o tempo de execução de uma aplicação de processamento de imagem, executando no *smartphone*, pode ser até 24 vezes mais lento do que executar o aplicativo com *offloading* no *cloudlet*. Além disso, mostrou que as taxas de *download* e *upload* tornam-se gargalos, principalmente quando a quantidade de dados a ser transferida é grande (4,5 MB, no caso da foto de 8MP).

Estes resultados podem ser utilizados para motivar o uso de *offloading*. Porém, é importante destacar que a tecnologia 4G trouxe a Internet banda larga para os dispositivos móveis, e este fato traz novas oportunidades e desafios para MCC. A maioria dos estudos presentes na literatura abordava apenas o uso das redes 3G e WiFi como suporte para realizar *offloading*. Tais estudos enfatizavam que a rede WiFi era a melhor solução para trabalhar com *offloading*, porém o usuário perdia no aspecto mobilidade.

A tecnologia 4G, com sua alta taxa de transferência, surge como uma ótima opção para explorar a mobilidade. Acreditamos que, pelo fato dos usuários estarem quase sempre ao alcance de um ponto de acesso WiFi em suas residências e trabalho, os *cloudlets* podem e devem ser utilizados de forma oportunista para melhorar o desempenho das aplicações móveis. Nesse cenário, a rede 4G surge como opção para manter a conectividade e os serviços em execução quando o dispositivo móvel estiver fora do alcance do WiFi/*cloudlet*, notadamente enquanto o usuário estiver se deslocando.

## 5. Conclusão e Trabalhos Futuros

Este trabalho apresentou uma avaliação da qualidade da Internet móvel em algumas capitais do Brasil e um estudo comparativo entre o uso de *cloudlet* e nuvem pública, conectadas via diferentes redes sem fio, para realizar *offloading*. Além de avaliar a qualidade

da Internet móvel, o trabalho discutiu os fatores que impactam na decisão de fazer ou não *offloading*. O trabalho mostrou que a quantidade de dados a serem transferidos, complexidade das tarefas e vazão de rede são os fatores com mais peso para a decisão. Ou seja, a qualidade da Internet móvel e a cobertura da rede são fatores-chave para a adoção do 4G LTE para realização de *offloading*.

Entre as contribuições do trabalho estão os aplicativos desenvolvidos, que estão disponíveis para a comunidade, e podem ser utilizados em outros estudos sobre MCC e *offloading*. Os aplicativos também estão entre as principais dificuldades enfrentadas, principalmente com relação ao melhor modo de mensurar as métricas de rede, assim como a metodologia, que exigia a realização dos experimentos em diferentes cidades.

Além dos fatores já mencionados, o custo energético e dos planos de dados também devem ser considerados em futuros estudos. Diante deste cenário, diversas oportunidades de pesquisa surgem para otimizar a quantidade de dados transmitidos, orquestração e adaptação de serviços, e sincronização de dados entre o dispositivo móvel, a nuvem e os *cloudlets*.

Como trabalho futuro, pretende-se estender esse trabalho para avaliar a eficiência energética de se utilizar a rede 3G, 4G e WiFi para realizar *offloading* de dados e processamento. Além disso, pretende-se desenvolver um *framework* para tomada de decisão de onde e quando fazer *offloading*, e para realizar *offloading* em múltiplas plataformas. A ideia é criar uma solução híbrida entre *cloudlets* e a nuvem pública, para que o *framework* possa se aproveitar dos *cloudlets* presentes na vizinhança e, quando eles não estiverem disponíveis, utilizar a nuvem pública através do 4G, se disponível na área.

## Agradecimentos

Este trabalho é parcialmente financiado pelo projeto INCT-MACC (processo CNPq 573710/2008-2).

## Referências

Bahl, P., Han, R. Y., Li, L. E., and Satyanarayanan, M. (2012). Advancing the state of mobile cloud computing. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, MCS '12, pages 21–28, New York, NY, USA. ACM.

Barbera, M., Kosta, S., Mei, A., and Stefa, J. (2013). To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. In *INFOCOM, 2013 Proceedings IEEE*, pages 1285–1293.

Chen, Y.-C., Lim, Y.-s., Gibbens, R. J., Nahum, E. M., Khalili, R., and Towsley, D. (2013). A measurement-based study of multipath tcp performance over wireless networks. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 455–468, New York, NY, USA. ACM.

Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., and Patti, A. (2011). Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA. ACM.

Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. (2010). Maui: Making smartphones last longer with code offload. In *MobiSys 2010, Proceedings ACM*, pages 49–62, New York, NY, USA. ACM.

Demichelis, C. and Chimento, P. (2002). RFC 3393: IP Packet Delay Variation Metric for IP Performance Metrics (IPPM).

Dinh, H. T., Lee, C., Niyato, D., and Wang, P. (2011). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing*.

Fernando, N., Loke, S. W., and Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84 – 106.

Gordon, M. S., Jamshidi, D. A., Mahlke, S., Mao, Z. M., and Chen, X. (2012). Comet: Code offload by migrating execution transparently. In *USENIX 2012, Proceedings ACM*, pages 93–106, Berkeley, CA, USA. USENIX Association.

Huang, J., Qian, F., Gerber, A., Mao, Z. M., Sen, S., and Spatscheck, O. (2012). A close examination of performance and power characteristics of 4g lte networks. In *MobiSys 2012, Proceedings ACM*, pages 225–238, New York, NY, USA. ACM.

Kosta, S., Aucinas, A., Hui, P., Mortier, R., and Zhang, X. (2012). Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *INFOCOM, 2012 Proceedings IEEE*, pages 945–953.

Kumar, K., Liu, J., Lu, Y.-H., and Bhargava, B. (2013). A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, 18(1):129–140.

Kumar, K. and Lu, Y.-H. (2010). Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51–56.

Quental, N. C. and Gonçalves, P. A. d. S. (2013). Uma Estratégia de Tentativas de Handover Vertical em Grupo. In *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2013)*.

Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2011). The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE*, PP(99):1–1.

Shepard, C., Rahmati, A., Tossell, C., Zhong, L., and Kortum, P. (2011). Livelab: Measuring wireless networks and smartphone users in the field. *SIGMETRICS Perform. Eval. Rev.*, 38(3):15–20.

Shiraz, M., Gani, A., Khokhar, R., and Buyya, R. (2013). A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *Communications Surveys Tutorials, IEEE*, 15(3):1294–1313.

Sommers, J. and Barford, P. (2012). Cell vs. wifi: On the performance of metro area mobile connections. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, pages 301–314, New York, NY, USA. ACM.

Verbelen, T., Simoens, P., Turck, F. D., and Dhoedt, B. (2012). Aiolos: Middleware for improving mobile application performance through cyber foraging. *Journal of Systems and Software*, 85(11):2629 – 2639.

# Um mecanismo para escalonamento de pacotes no uplink da rede LTE no contexto da comunicação Máquina-a-Máquina

**Adyson M. Maia**[1], **Miguel F. de Castro**[1], **Dario Vieira**[2], **Yacine Ghamri-Doudane**[3]

[1] Universidade Federal do Ceará (UFC)

[2]Ecole d'Ingénieur Généraliste en Informatique et Technologies du Numérique (EFREI)

[3]Université de La Rochelle

adysonmaia@great.ufc.br, miguel@lia.ufc.br

dario.vieira@efrei.fr, yacine.ghamri@univ-lr.fr

***Abstract.*** *This paper proposes a mechanism for uplink packet scheduler in LTE network in the context of Machine-to-Machine (M2M) communications that uses the current and past information of the system to satisfy the Quality of Service (QoS) requirements, ensure fairness in the allocation of resources and control the congestion caused by the M2M devices. The results indicate that the proposed approach can reduce the impact of M2M communication on the Human-to-Human (H2H) communication and avoid the problem of starvation.*

***Resumo.*** *Este artigo propõe um mecanismo para o escalonamento de pacotes no uplink da rede LTE no contexto da comunicação Máquina-a-Máquina (M2M). O mecanismo proposto utiliza as informações atuais e passadas da rede e de cada dispositivo para satisfazer os requisitos de Qualidade de Serviço (QoS) dos dispositivos M2M, garantir justiça na alocação dos recursos e controlar o congestionamento causado por estes dispositivos. Os resultados obtidos indicam que a proposta consegue reduzir o impacto da comunicação M2M sobre a Humano-a-Humano (H2H) e evitar o problema de inanição.*

## 1. Introdução

No paradigma da Internet das Coisas (*Internet of Things* - IoT), onde uma gama de objetos inteligentes (eletrodomésticos, automóveis, celulares etc) estarão conectados à Internet, a comunicação Máquina-a-Máquina (*Machine-to-Machine* - M2M) terá uma importância ainda maior que a atual comunicação Humano-a-Humano (*Human-to-Human* - H2H) [Wu et al. 2011].

Espera-se que as redes celulares desempenhem um papel fundamental na implantação da IoT, como também na comunicação M2M, destacando-se a rede LTE para esta implantação [Miorandi et al. 2012]. Porém, melhorias na rede LTE são necessárias devido às características intrínsecas da comunicação M2M, discutidas com mais detalhes na Seção 2, que diferenciam da comunicação H2H para a qual a rede foi inicialmente projetada. Dentre estas melhorias inclui-se o aperfeiçoamento no escalonamento de pacotes no *uplink*.

Segundo [Lioumpas and Alexiou 2011], as soluções existentes na literatura para o escalonamento da comunicação H2H não são adequadas para a comunicação M2M

devido a suposição da pouca quantidade de serviços e requisitos de QoS existentes da H2H (*e.g.*, voz, vídeo e web) que se diferencia da vasta variedade de aplicações M2M e consequentemente de seus requisitos de QoS. Outro fator importante da não adequação é que essas soluções não tratam do congestionamento causado pela implantação de uma grande quantidade de objetos inteligentes em uma mesma área e, portanto, utilizando os mesmo recursos de rádio compartilhados da rede.

A maioria das soluções encontradas na literatura, discutidas na Seção 3, que tratam da comunicação M2M possuem como objetivo satisfazer os requisitos de QoS dos dispositivos M2M[1], como o atraso [Lioumpas and Alexiou 2011] ou o *jitter* [Lien et al. 2011] máximo tolerável. Contudo, ao satisfazer estes requisitos a justiça na alocação de recursos é comprometida podendo levar ao caso de inanição para os dispositivos M2M que possuem baixa prioridade de alocação. Além disso, estas soluções não tratam do congestionamento na comunicação H2H causado pela introdução da comunicação M2M na rede.

Dentro deste contexto, este trabalho aborda a problemática da alocação de recursos compartilhados de rádio na rede LTE, levando em consideração tanto a comunicação M2M quanto a H2H, propondo um mecanismo de escalonamento de pacotes para o *uplink* cujos detalhes são apresentados na Seção 4. Este escalonamento utiliza informações atuais do tráfego de cada dispositivo assim como informações do histórico de alocações para classificar o tráfego e também para priorizar a alocação dos recursos de rádio. Esta priorização tem como objetivo (i) controlar o impacto da comunicação M2M na H2H, (ii) garantir justiça, consequentemente, evitar o problema de inanição e (iii) satisfazer os requisitos de QoS.

Na Seção 5, os experimentos através de simulações e os seus resultados são apresentados e debatidos. Estes resultados indicam que a abordagem proposta consegue reduzir o impacto da comunicação M2M sobre a H2H e evita o problema de inanição ao garantir justiça na alocação de recursos. Finalizando, a Seção 6 aborda as conclusões deste trabalho.

## 2. Fundamentação Teórica

Nesta seção serão abordados com mais detalhes a comunicação M2M (Subseção 2.1) e o escalonamento de pacotes na rede LTE (Subseção 2.2).

### 2.1. Características da Comunicação M2M

As aplicações que utilizam a comunicação M2M são geralmente caracterizadas pela transmissão e coleta automática de dados de uma fonte remota com pouca ou nenhuma intervenção humana através de uma rede pública (Wi-Fi, WiMax, UMTS, LTE etc) [Boswarthick et al. 2012]. Geralmente, as aplicações M2M utilizam um grande número de dispositivos para monitorar algum aspecto (temperatura, umidade, velocidade, posição, batimento cardíaco, etc.) do ambiente implantado através de sensores [Amokrane et al. 2011]. Os dados capturados pelo monitoramento são enviados para um servidor (programa de software). No servidor, os dados são processados e traduzidos em informações úteis (detecção de ameaças, armazenamento de dados etc). Finalmente, respostas podem ser enviadas de volta para os dispositivos.

[1]Os dispositivos M2M são os objetos inteligentes das aplicações que utilizam a comunicação M2M.

### 2.2. Escalonador de Pacotes para a rede LTE

A transmissão de dados via rádio na rede LTE é feita através de canais (*downlink* e *uplink*) compartilhados de transmissão. Assim sendo, os dispositivos precisam requisitar à rede recursos de rádio para fazer a transmissão. A menor unidade de recurso que o dispositivo pode receber é chamada de bloco de recurso (*Resource Block* - RB).

O escalonamento nos dois canais (*downlink* e *uplink*) é realizado separadamente (escalonador para o *downlink* e o para o *uplink*) na estação base. O escalonamento de ambos os canais é realizado a cada TTI (*Transmission Time Interval* - 1 ms) e decide quais RBs serão alocados para quais dispositivos. Esta decisão tem a finalidade de satisfazer as necessidades de recursos dos dispositivos, assim como maximizar uma performance do sistema (*e.g.*, maximizar a taxa de transferência da rede, satisfazer os requisitos de QoS, garantir justiça na alocação de recursos).

## 3. Trabalhos Relacionados

Várias soluções sobre o escalonamento de pacotes no *uplink* da rede LTE podem ser encontradas na literatura [Kwan and Leung 2010, Salah et al. 2011]. Contudo, poucas soluções tratam da comunicação M2M [Lien et al. 2011, Lioumpas and Alexiou 2011, Zhenqi et al. 2013, Gotsis et al. 2013, Abdalla and Venkatesan 2013, Afrin et al. 2013]. No entanto, estas soluções possuem algumas deficiências. Entre essas deficiências, destaca-se a não garantia de alocação justa dos recursos entre os dispositivos. Outro ponto fraco é relacionado ao não controle do impacto da introdução da comunicação M2M na rede sobre a performance da comunicação H2H. Mesmo quando feito, este controle é fracamente realizado considerando o uso eficiente dos recursos entre as comunicações M2M e H2H. No restante desta seção, nós discutimos algumas das soluções que consideram a comunicação M2M que foram utilizadas como base para o nosso trabalho.

Em [Lien et al. 2011], os autores propõem um escalonador baseado em grupos no qual os dispositivos formam um *cluster* segundo seus requisitos de QoS. As características e requisitos de QoS dos dispositivos em um mesmo *cluster* são definidos por dois parâmetros: (i) taxa de chegada de pacotes e (ii) o máximo *jitter* tolerável. O *cluster* com maior taxa de chegada de pacotes possui maior prioridade sendo que seus dispositivos só podem requisitar recursos em intervalos inversamente proporcionais a essa taxa. Portanto, esta solução utiliza o padrão do tráfego para garantir seus requisitos de QoS. Contudo, em cenários onde há uma massiva quantidade de dispositivos, o escalonamento não é justo para os dispositivos nos *clusters* com baixa prioridade. Ademais, a solução não utiliza as informações da qualidade do canal entre o dispositivo e a estação base que influencia na taxa de transferência.

Duas soluções são apresentadas em [Lioumpas and Alexiou 2011] com o objetivo de garantir os requisitos de QoS. Para isto, dois parâmetros são utilizados: a qualidade do canal entre os dispositivos e a estação base, e o atraso máximo tolerável como métrica de QoS. As duas soluções se diferenciam pelo peso dos dois parâmetros na tomada de decisão. Na primeira, a qualidade do canal tem mais peso na priorização, enquanto na segunda solução o maior peso é nos dispositivos menos toleráveis ao atraso. As duas soluções não são justas para os dispositivos que possuem as piores condições no parâmetro mais relevante. Por exemplo, no cenário onde há uma grande quantidade de dispositivos, a primeira solução não é justa na alocação de recursos para os dispositi-

vos que possuem uma baixa qualidade do canal, e a segunda solução não é justa para os dispositivos mais toleráveis ao atraso.

## 4. Mecanismo Proposto para o Escalonamento

Como discutido nas seções anteriores, melhorias no escalonamento de pacotes no *uplink* na rede LTE são necessárias para tratar a comunicação M2M. Nesta seção, serão abordados os detalhes do mecanismo proposto sobre o problema citado. Na primeira subseção, o problema do escalonamento é formulado, e as novas classes de QoS propostas para a comunicação M2M são descritas na subseção 4.2. Na Subseção 4.3, os recursos de rádio são divididos em dois grupos, um para a comunicação H2H e o outro para a M2M. Por fim, o algoritmo proposto para o escalonamento é apresentado na subseção 4.4.

### 4.1. Formulação do Problema

Seja $\mathcal{U} = \{1, \ldots, U\}$ o conjunto dos dispositivos que requerem recursos no TTI $t$, tal que $\mathcal{U}_{H2H} \cup \mathcal{U}_{M2M} = \mathcal{U}$, onde $\mathcal{U}_{H2H}$, $\mathcal{U}_{M2M}$ são os conjuntos dos dispositivos H2H e M2M respectivamente. Considere $B$ como a largura de banda utilizada e com $R$ RBs indexados pelo conjunto $\mathcal{RB} = \{1, \ldots, R\}$. Assumindo isto, o problema de alocação de recursos pode ser definido como um problema de otimização onde o objetivo é maximizar a seguinte função:

$$M_{sum} = \sum_{\substack{u \in \mathcal{U} \\ r \in \mathcal{RB}}} M_{u,r} A_{u,r} \tag{1}$$

onde $M_{u,r}$ é a métrica que avalia a performance da alocação do RB $r$ para o dispositivo $u$ com o intuito de alcançar o objetivo do escalonamento. $A_{u,r} = \{0, 1\}$ é uma função no qual o valor é 1 se o RB $r$ for alocado para o dispositivo $u$ e 0 caso contrário.

Além disso, duas restrições são necessárias para a equação (1). A primeira restrição (equação (2)) é que cada RB só pode ser alocado para no máximo um dispositivo. A segunda, visualizada na equação (3), é que os recursos alocados para um dispositivo devem ser contínuos como exigido na escalonamento do *uplink*.

$$\sum_{r=1}^{R} A_{u,r} \leq 1; \forall u \in \mathcal{U} \tag{2}$$

$$\begin{aligned} &\forall u \in \mathcal{U}: \\ &\forall r \left(r \in \{r' | A_{u,r'} = 1\} \implies r \in \{i, i+1, \ldots, i+l\}\right) \\ &\qquad\qquad \text{para algum } i, l | 1 \leq i \leq i+l \leq R \end{aligned} \tag{3}$$

### 4.2. Classes de QoS

O padrão do sistema LTE suporta que os tráfegos tenham garantias de QoS. Estes tráfegos podem ser classificados em uma das nove classes de QoS definidas de acordo com os seus requisitos de QoS como especificado em [3GPP 2013]. Os requisitos de QoS são definidos pelas seguintes métricas: (i) o atraso máximo tolerável entre o dispositivo e o *gateway* de saída da rede e (ii) a taxa de perda de pacotes. As classes de QoS são identificáveis através de seus QCIs (*QoS Class Indicator*) e foram definidas para suportar as principais aplicações H2H (*e.g.*, VoIP, *streaming* de vídeo ou áudio e web).

Contudo, as classes de QoS do padrão não são adequadas para a grande variedade do atraso máximo tolerável (de poucos milissegundos até vários minutos) das aplicações baseadas em tempo. Portanto, no presente trabalho é proposto uma nova relação entre as classes de QoS do padrão LTE e as categorias de aplicações M2M apresentadas em [Amokrane et al. 2011]. As aplicações da categoria baseada em consulta podem ser relacionadas com os QCI 8 ou 9, pois estas aplicações utilizam o modo de transmissão requisição-resposta típica das aplicações baseadas no TCP. Além disso, as aplicações M2M baseadas em evento requerem alta prioridade, baixa tolerância a atraso e alta confiabilidade (baixa taxa de perda de pacotes). Assim, a única relação possível destas aplicações é uma classe (QCI 5) que é usada exclusivamente para mensagens de controle.

Este trabalho propõe duas abordagens para estender as classes de QoS definidas no padrão do sistema LTE. As duas propostas possuem como vantagem a identificação dos dispositivos (M2M ou H2H) e de suas aplicações através da classe de QoS no qual o seu tráfego está relacionado. As abordagens propostas são as seguintes:

1. Na primeira abordagem, duas classes são adicionadas ao padrão. Uma classe será utilizada para as aplicações M2M baseadas em evento e a outra será utilizada para as aplicações baseadas em tempo. Contudo, o parâmetro de atraso máximo permitido não será utilizado nesta última classe e cabe a cada dispositivo das aplicações baseadas em tempo enviar para a rede (estação base) o seu próprio valor para este parâmetro.
2. Na segunda abordagem, $1 + N$ classes são adicionadas ao padrão. Como na abordagem anterior, uma classe será utilizada para as aplicações baseadas em evento, entretanto, $N$ classes são utilizadas para as aplicações baseadas em tempo. Estas $N$ classes possuirão como diferencial entre elas o atraso máximo tolerável. Assim, um dispositivo que transmite seus dados a cada $x$ intervalo de tempo escolherá a classe com o maior atraso máximo permitido igual ou inferior à $x$.

Na Tabela 1 é ilustrado um exemplo das abordagens propostas. A primeira abordagem utiliza somente as duas primeiras classes (QCI 10 e 11) e a segunda, todas as classes ilustradas. Os valores dos parâmetros para a classe (QCI 10) das aplicações baseadas em evento foram escolhidas para atender os seus requisitos de QoS utilizando valores das classes já existentes no padrão. Além disso, a classe com QCI 10 possui a maior prioridade após a classe utilizada para as mensagens de controle. O atraso máximo tolerável foi incrementado para cada classe subsequente das aplicações baseadas em tempo. A diferença do atraso máximo entre duas classes consecutivas aumenta conforme as aplicações sejam mais toleráveis ao atraso. Assim, haverá uma menor granularidade na classificação das aplicações menos toleráveis ao atraso que também possuem maior prioridade.

### 4.3. Cálculo da Demanda por Recursos

Um dos objetivos deste trabalho é controlar o impacto da comunicação M2M na H2H. Para isto, o escalonador irá reservar RBs para o tráfego dos dispositivos H2H baseado nas suas demandas por recursos. Para o cálculo da demanda atual de recurso é utilizado o histórico de alocação de recursos e o tamanho atual dos dados no *buffer* de transmissão dos dispositivos como mostrado a seguir:

$$\widehat{RB}_{H2H}(u,t) = \max\left(RB^{min}_{H2H}, \frac{BS_{H2H}(u,t) \times RB^{past}_{H2H}(u,t-1)}{BS^{past}_{H2H}(u,t-1)}\right) \quad (4)$$

**Tabela 1. Classes de QoS para as aplicações M2M**

| QCI | Prioridade | Atraso Máximo | Perda de Pacotes | Aplicações |
|---|---|---|---|---|
| 10 | 2 | 50 ms | $10^{-6}$ | Baseada em Evento |
| 11 | 10 | 50 ms | $10^{-2}$ | Baseada em Tempo |
| 12 | 11 | 100 ms | $10^{-2}$ | Baseada em Tempo |
| 13 | 12 | 150 ms | $10^{-2}$ | Baseada em Tempo |
| 14 | 13 | 300 ms | $10^{-2}$ | Baseada em Tempo |
| 15 | 14 | 400 ms | $10^{-2}$ | Baseada em Tempo |
| 16 | 15 | 500 ms | $10^{-2}$ | Baseada em Tempo |
| 17 | 16 | 1 s | $10^{-2}$ | Baseada em Tempo |
| 18 | 17 | 5 s | $10^{-2}$ | Baseada em Tempo |
| 19 | 18 | 15 s | $10^{-2}$ | Baseada em Tempo |
| 20 | 19 | 30 s | $10^{-2}$ | Baseada em Tempo |
| 21 | 20 | 1 min | $10^{-2}$ | Baseada em Tempo |

onde $\widehat{RB}_{H2H}(u,t)$ é a demanda atual de RBs do dispositivo $u$ no TTI $t$, $BS_{H2H}$ é o tamanho atual dos dados no *buffer*, $RB^{past}_{H2H}$ é a quantidade média de RBs alocados para o dispositivo e $BS^{past}_{H2H}$ é a média do tamanho dos dados no *buffer* de transmissão. As funções $RB^{past}_{H2H}$, $BS^{past}_{H2H}$ são calculadas utilizando uma média móvel exponencial. Ademais, cada dispositivo H2H possui um limite mínimo $RB^{min}_{H2H}$, maior que zero, de recursos requeridos.

A demanda total de recursos de todos os dispositivos H2H é calculada como a soma das demandas de cada dispositivo e com o limite superior do total de recursos disponíveis como mostrado a seguir:

$$\widehat{RB}_{H2H}(t) = \min\left(|\mathcal{RB}|, \sum_{u \in \mathcal{U}_{H2H}} \widehat{RB}_{H2H}(u,t)\right) \tag{5}$$

Contudo, os dispositivos H2H podem requerer todos os recursos disponíveis causando inanição nos dispositivos M2M. Assim sendo, uma porcentagem constante miníma de recursos ($0 \leq p^{min}_{M2M} \leq 1$) para os dispositivos M2M é garantida com objetivo de evitar a inanição. Outra importante característica da solução proposta neste trabalho é que como os dispositivos M2M transmitem pouca quantidade de dados por vez (menos de 1000 bits) [3GPP 2012], então, cada dispositivo M2M irá receber uma quantidade fixa e pequena de recursos $RB^{min}_{M2M}$. Estas restrições são apresentadas nas equações (6b) e (6c) que calculam a quantidade de recursos para a comunicação M2M. Consequentemente, os recursos reservados para a H2H será igual os recursos restantes como ilustrado na equação (6a).

$$RB_{H2H}(t) = |\mathcal{RB}| - RB_{M2M}(t) \tag{6a}$$

$$RB_{M2M}(t) = RB^{min}_{M2M} \times \min\left(|\mathcal{U}_{M2M}|, \widehat{RB}_{M2M}(t)\right) \tag{6b}$$

$$\widehat{RB}_{M2M}(t) = \max\left(\left\lfloor \frac{p^{min}_{M2M} \times |\mathcal{RB}|}{RB^{min}_{M2M}} \right\rfloor, \left\lfloor \frac{|\mathcal{RB}| - \widehat{RB}_{H2H}(t)}{RB^{min}_{M2M}} \right\rfloor\right) \tag{6c}$$

### 4.4. Descrição do Algoritmo

Com as demandas de recursos para os dispositivos calculadas, os RBs são divididos em dois grupos consecutivos em relação a frequência, um para cada tipo de comunicação.

Deste modo, as alocações de recursos para as comunicações H2H e M2M podem ser tratadas separadamente. Portanto, o mecanismo proposto neste trabalho focará somente no escalonamento para os dispositivos M2M, beneficiando-se do uso de qualquer solução existente na literatura para o escalonamento dos outros dispositivos.

Baseando-se no modelo especificado em [Pokhariyal et al. 2007], o algoritmo proposto é divido em duas fases. Na primeira fase, os dispositivos são priorizados e então, todos os dispositivos ou somente aqueles com maior prioridade são escolhidos para a próxima fase. Na segunda fase é realizado o mapeamento entre RBs e dispositivos. Nas próximas subseções, estas duas fases são apresentadas com mais detalhes.

### 4.4.1. Primeira Fase

Considere que $RB_{M2M}$ recursos estão disponíveis e que cada dispositivo receba uma quantidade fixa de recursos $RB^{min}_{M2M}$. Portanto, no máximo $RB_{M2M}/RB^{min}_{M2M}$ dispositivos poderão receber os recursos. Esta fase tem o papel de escolher os dispositivos que receberão estes recursos. Esta decisão é feita de tal forma que os dispositivos que receberam menos recursos ao longo do tempo e que estejam mais próximos de superar o atraso máximo tolerável tenham maior prioridade na alocação de recursos. Portanto, nesta fase é garantida a justiça na alocação de recursos. Outro característica fundamental desta fase é de maximizar a satisfação dos requisitos de QoS medidos pela métrica de atraso máximo tolerável.

A função de priorização $F^{TD}_{M2M}$ é apresentada na equação (7), onde $T^{past}_{M2M}(u,t)$ é uma média móvel exponencial da taxa de transferência do dispositivo $u$. $T^{past}_{M2M}$ indica se um dispositivo recebeu uma quantidade justa de recursos até o TTI $t$. $\Delta D(u)$ é a função que mede se este dispositivo está próximo de não satisfazer os seus requisitos de QoS. Estas duas funções serão descritas com mais detalhes nos próximos parágrafos. A constante $\varpi$ ($0 \leq \varpi \leq 1$) indica o peso de garantir justiça e de satisfazer os requisitos de QoS na priorização dos dispositivos.

$$F^{TD}_{M2M}(u,t) = (1-\varpi)\left(1 - \frac{T^{past}_{M2M}(u,t-1)}{\max_{n\in\mathcal{U}_{M2M}} T^{past}_{M2M}(n,t-1)}\right) + \varpi\left(1 - \frac{\Delta D(u)}{\max_{n\in\mathcal{U}_{M2M}} \Delta D(n)}\right) \tag{7}$$

$\Delta D(u)$ é calculada de duas formas. Se a última requisição por recursos foi atendida ou nenhuma requisição foi feita, então $\Delta D(u)$ é igual ao atraso máximo tolerável. Caso contrário, $\Delta D(u)$ é igual a diferença entre o atraso máximo e o tempo esperado desde a última requisição não atendida ou zero caso esta diferença seja negativa.

Utilizando a função de priorização descrita anteriormente, o algoritmo desta fase escolhe os $RB_{M2M}/RB^{min}_{M2M}$ dispositivos com maiores valores de $F^{TD}_{M2M}$ para a próxima fase. Os dispositivo que não forem escolhidos terão que esperar $x$ TTIs para fazer uma nova requisição, onde o valor $x$ é escolhido aleatoriamente entre 1 à $\Delta D(u)/10$. Deste modo, as requisição são espalhadas no decorrer do tempo com a finalidade de reduzir um possível congestionamento da rede gerado pela grande quantidade de dispositivos requisitando recursos.

### 4.4.2. Segunda Fase

Nesta fase, $RB_{M2M}/RB^{min}_{M2M}$ grupos de recursos consecutivos em relação à frequência são criados, um para cada dispositivo selecionado pela fase anterior. A decisão de qual grupo será alocado para qual dispositivo é feita utilizando os seguintes passos:

1. Para cada grupo de recursos $g$ e dispositivo $u$, calcule a taxa de transferência caso o dispositivo utilize este grupo de recursos. Seja $\mathcal{S}$ o conjunto com estes valores.
2. Se $\mathcal{S} \neq \emptyset$, então remova deste conjunto o maior valor. Caso contrário, o algoritmo é finalizado.
3. Aloque os recursos do grupo $g$ ao dispositivo $u$ que estão associados ao valor removido somente se o dispositivo ainda não recebeu recursos.
4. Repita o passo 2.

A taxa de transferência $T_{M2M}(u,t)$ esperada para o dispositivo $u$ no TTI $t$ é igual à taxa de transferência $T_{M2M}(u,g)$ para o grupo de recursos $g$ selecionado. O cálculo de $T_{M2M}(u,g)$ pode ser realizado utilizando o limite teórico de Shannon para a capacidade máxima do sistema como mostrado em [Lim et al. 2006]. Neste cálculo, a taxa de transferência é influencia pela qualidade do canal entre o dispositivo e o RB e é medida através da relação sinal-ruído (*Signal-to-Noise Ratio* - SNR).

## 5. Experimentos

Esta seção apresenta o ambiente de simulação no NS-3 [NS-3 2013] (Subseção 5.1) e os resultados obtidos (Subseção 5.2).

### 5.1. Ambiente de Simulação

Para a avaliação do mecanismo proposto com as duas abordagens de classes de QoS, a abordagem proposta no presente trabalho foi comparada com duas soluções existentes na literatura. A primeira solução é o escalonador *Proportional Fair* (PF), que é uma das soluções mais utilizadas e pesquisadas na literatura para o escalonamento justo de recursos da comunicação H2H [Lee et al. 2009]. Para fins de comparação, o PF foi utilizado para o escalonamento da comunicação H2H no mecanismo proposto neste trabalho. Como discutido na Seção 3, dois algoritmos são proposto em [Lioumpas and Alexiou 2011] sendo o segundo algoritmo escolhido para os experimentos visto que o mesmo apresenta melhor desempenho que o primeiro algoritmo na satisfação do QoS.

Algumas lacunas do segundo algoritmo proposto em [Lioumpas and Alexiou 2011] precisaram ser preenchidas para que este seja comparado com a abordagem proposta no presente trabalho e com a PF. A comunicação H2H não é abordada em [Lioumpas and Alexiou 2011] apesar de ter maior prioridade em relação à M2M. Assim, a divisão de recursos que é descrita na Seção 4.3 foi também utilizada para estas soluções. Além disso, a quantidade de recursos requeridos por cada dispositivo também não é definida. Portanto, os recursos foram divididos igualmente entre os dispositivos, mas com um limite inferior igual à $RB^{min}_{M2M}$.

As aplicações H2H utilizadas nas simulações foram VoIP, vídeo e FTP. A modelagem dos tráfegos destas aplicações foi baseada nos trabalhos [Salah et al. 2011] (VoIP e FTP) e [Potsch et al. 2013] (Vídeo).

As categorias baseadas em evento e em tempo foram utilizadas para simular a comunicação M2M. A transmissão em rajadas das aplicações M2M baseadas em eventos foram modeladas através do processo de Poisson com a taxa $\gamma = 0,02$ pacotes/TTI [Gotsis et al. 2013]. O intervalo de transmissão de cada dispositivo das aplicações baseadas em tempo foi distribuído uniformemente entre 50 ms e 550 ms. Ambas as categorias de aplicações M2M possuem o tamanho do pacote de 125 bytes [3GPP 2012].

O cenário simulado foi composto por 30 dispositivos H2H, 10 para cada tipo de tráfego. Os dispositivos foram distribuídos uniformemente em torno de uma única estação base. A quantidade de dispositivos por categoria de aplicações M2M foi definida de tal forma que a categoria baseada em tempo tenha uma quantidade maior dos dispositivos uma vez que a maioria das aplicações M2M está nesta categoria.

Para avaliar os desempenhos das soluções, três métricas foram utilizadas: (i) a taxa de transferência, (ii) a porcentagem de pacotes que não satisfizeram a restrição do atraso máximo tolerável e (iii) o índice de justiça de Jain [Jain 1991] com relação à taxa de transferência. Para isto, 30 execuções independentes com duração de 3 s ou 3000 TTIs foram realizadas e utilizado o nível de confiança de 95% para a análise dos resultados. A Tabela 2 resume os principais parâmetros utilizados nas simulações. Outros parâmetros foram omitidos por terem os valores padrão do simulador NS-3 [NS-3 2013].

**Tabela 2. Parâmetros das simulações.**

| **Parâmetros Gerais** | | | |
|---|---|---|---|
| Largura de banda | 5 MHz (25 RBs) | | |
| Número dispositivos M2M | 0, 50, 100, 150, 200, 250; 1/3 Baseados em Evento, 2/3 Baseados em Tempo | | |
| Número dispositivos H2H | 10 VoIP, 10 Vídeo, 10 FTP | | |
| **Tráfego H2H** | **Tamanho do Pacote** | **Tempo entre Transmissões** | **QCI** |
| Vídeo | 1200 bytes | 75 ms | 2 |
| VoIP | 40 bytes | 20 ms | 1 |
| FTP | 256 bytes | 16,625 ms | 8 |
| **Tráfego M2M** | | | |
| Baseados em Evento | 125 bytes | Processo de Poisson, $\gamma$ = 50 ms | |
| Baseado em Tempo | 125 bytes | Distribuído uniformemente [50, 550] ms | |
| **Parâmetros do Mecanismo Proposto** | | | |
| $RB_{H2H}^{min}$ | 3 | $RB_{M2M}^{min}$ | 3 |
| $p_{m2m}^{min}$ | 0,48 | $\varpi$ | 0,72 |

Com relação aos parâmetros do mecanismo proposto, a Tabela 1 foi utilizada para classificar o tráfego M2M segundo as duas abordagens já discutidas na Seção 4.2. Além disso, na fase inicial da implementação dos experimentos, foram realizadas simulações com o objetivo de enriquecer a solução proposta e definir os seus parâmetros. Deste modo, as quantidades mínimas de recursos ($RB_{H2H}^{min}$ e $RB_{M2M}^{min}$) requisitadas por dispositivo foram escolhidas para que seja possível transmitir todos os dados no *buffer* apenas com essa quantidade de recursos caso o dispositivo apresente uma boa qualidade no canal e a quantidade de dados seja pequena como nas aplicações M2M. Ademais, a porcentagem mínima de recursos para os dispositivos M2M foi definida para que haja pelo menos 4 ($25 \times p_{m2m}^{min}/RB_{M2M}^{min}$) dispositivos M2M recebendo recursos por TTI. Na equação (7), o valor da constante $\varpi$ foi definido para que a satisfação dos requisitos de QoS tenha maior peso do que a justiça na priorização das requisições.

## 5.2. Resultados

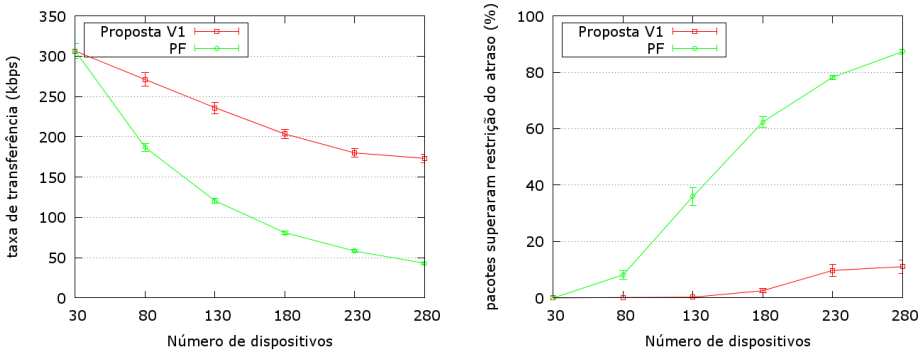
Como descrito na Seção 4.2, duas abordagens foram propostas para classificar o tráfego M2M segundo seus requisitos de QoS. Ao longo desta seção, o uso da primeira abordagem pelo escalonador proposto será referido como "Proposta-V1" e ao utilizar a segunda abordagem, o escalonador proposto será referido como "Proposta-V2". As modificações, descritas na Seção 5.1, para o segundo escalonador proposto em [Lioumpas and Alexiou 2011] será designada de "Lioumpas 2".

A Figura 1(a) apresenta os valores médios da taxa de transferência (vazão) dos dispositivos H2H em relação à variação no número de dispositivos M2M para os escalonadores PF e Proposta-V1. Os outros dois escalonadores (Proposta-V2 e o Lioumpas 2) não são mostrados na figura pois utilizam a mesma abordagem de separação de recursos entre H2H e M2M que o Proposta-V1.

Com 30 dispositivos e nenhum dispositivo M2M os escalonadores PF e Proposta-V1 apresentam a mesma taxa de transferência. Contudo, com o aumento dos dispositivos M2M, há uma grande diferença entre os valores apresentados para estes escalonadores. A taxa de transferência do PF foi reduzida em 86% com 280 dispositivos. O escalonador Proposta-V1 também apresenta redução na taxa de transferência, porém a sua redução é de 43% com 280 dispositivos.

Os valores médios dos percentuais de pacotes do tráfego H2H que não satisfizeram o atraso máximo tolerável é apresentado na Figura 1(b). Observa-se que até a quantidade de 130 dispositivos o escalonador Proposta-V1 apresenta valores próximos de zero. Com o aumento do número de dispositivos de 180 para 230 tem-se o aumento do valor de 2% para 9% e finalmente para 11% com 280 dispositivos. Contudo, a situação é bem diferente para o PF que apresenta um aumento de 87% com 280 dispositivos.

Conclui-se ao analisar os resultados ilustrados na Figura 1 que a abordagem utilizada em Proposta-V1 de separar o escalonamento da comunicação H2H da M2M apresenta um melhor controle do impacto da comunicação M2M na H2H do que o PF.




(a) Média da taxa de transferência do tráfego H2H

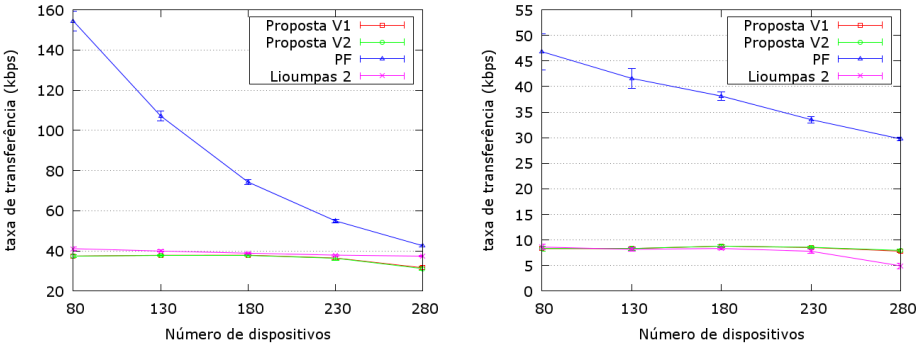
(b) Média do nível de satisfação de QoS do tráfego H2H

**Figura 1. Impacto da comunicação M2M sobre a H2H**

A Figura 2 apresenta os valores médios da taxa de transferência do tráfego das aplicações baseadas em evento (Figura 2(a)) e baseadas em tempo (Figura 2(b)). Observa-se baixa variação nos valores do escalonador Lioumpas 2 para a categoria baseada em

eventos, enquanto na categoria baseada em tempo a baixa variação é existente até a quantidade de 230 dispositivos com uma redução de 37% da taxa de transferência entre 230 e 280 dispositivos. Nos escalonamentos Proposta-V1 e Proposta-V2, observa-se uma baixa variação da taxa de transferência para a categoria baseada em tempo, enquanto na categoria baseada em evento os valores permanecem quase constantes até 230 dispositivos e com redução de 13% entre 230 e 280 dispositivos. Além disso, o Proposta-V1 e Proposta-V2 apresentam valores muito próximos em todas as medidas. O PF apresenta redução de 72% na categoria baseada em evento e 36% para a baseada em tempo com 280 dispositivos.

O escalonador Lioumpas 2 apresentou resultados similares aos escalonadores Proposta-V1 e Proposta-V2 em relação à taxa de transferência pois no cenário simulado com muitas requisições, a quantidade de recursos requisitados por dispositivo M2M foi limitado pelo valor mínimo $RB^{min}_{M2M}$. A diferença de priorização na alocação de recursos destes escalonamentos tiveram influências diferentes na taxa de transferência somente quanto um grande número de dispositivos M2M foi utilizado, ou seja, para evitar o problema de inanição da categoria baseado em tempo o Proposta-V1 e o Proposta-V2 reduzem a vazão da outra categoria. Ademais, a taxa de transferência da categoria baseada em tempo foi afetada pela prioridade mais alta da categoria baseada em eventos no escalonador Lioumpas 2.




(a) Tráfego do baseado em evento (b) Tráfego do baseado em tempo

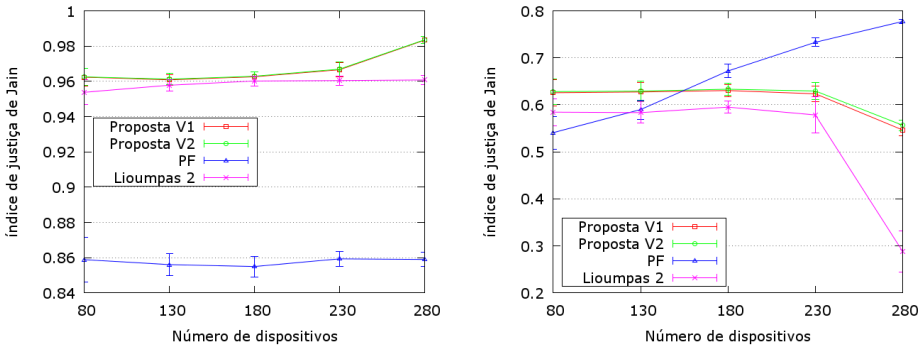
**Figura 2. Média da taxa de transferência to tráfego M2M**

Na Figura 3 os valores médios do índice de justiça de Jain [Jain 1991] são apresentados para o tráfego M2M. Se $n$ dispositivos requisitaram recursos e o índice é igual a $k/n$, então, $k$ dispositivos receberam recursos igualmente e $n - k$ não receberam recursos. Portanto, quanto mais próximo de 1 o índice está, mais justo o escalonador é. Consequentemente, quanto mais próximo de 0, maior a possibilidade do problema de inanição ocorrer.

Observa-se que houve pouca variação nos valores ao aumentar a quantidade de dispositivos para o PF e o Lioumpas 2 na categoria baseada em evento. Os escalonadores Proposta-V1 e Proposta-V2 apresentaram valores com pouca variação e similares ao Lioumpas 2 até 230 dispositivos e com um pequeno aumento de 2% no valor com 280 dispositivos na categoria baseada em evento. Na categoria baseada em tempo o Proposta-V1 e o Proposta-V2 também apresentaram valores com pouca variação, mas um pouco maiores que Lioumpas 2 até 230 dispositivos, enquanto uma redução de 13% foi observada entre 230 e 280 dispositivos para estes escalonadores e uma redução maior de 50% para

o Lioumpas 2. Contudo, o PF apresentou aumento no valor do índice para a categoria baseada em tempo pois esta categoria possui maior prioridade devido à infrequência de transmissão.

Conclui-se ao analisar o nível de justiça que com um grande número de dispositivos M2M, o Lioumpas 2 possui uma alta possibilidade da ocorrência do problema de inanição para a categoria baseada em tempo. Porém, os escalonadores PF, Proposta-V1 e Proposta-V2 apresentam bons índices em ambas as categorias.




(a) Tráfego do baseado em evento (b) Tráfego do baseado em tempo

**Figura 3. Média do índice de justiça do tráfego M2M**

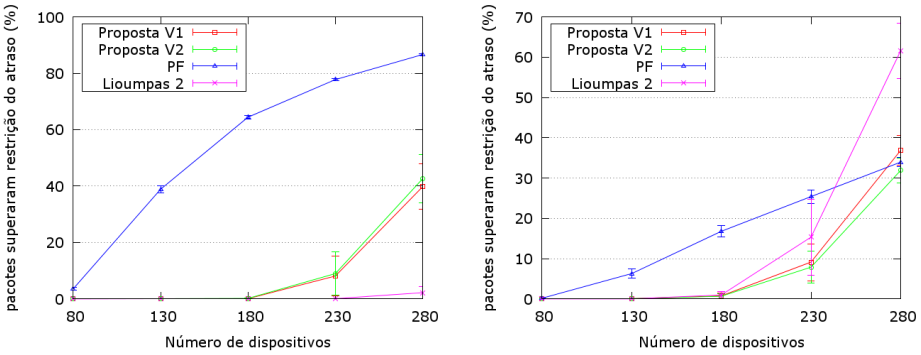
A Figura 4 apresenta os valores médios dos percentuais de pacotes do tráfego M2M que não satisfizeram o atraso máximo tolerável. Observa-se que o Lioumpas 2 possui os valores próximos de 0% até 230 dispositivos e com o valor inferior à 3% com 280 dispositivos para a categoria baseada em evento (Figura 4(a)). Proposta-V1 e Proposta-V2 possuem valores próximos de 0% até 180 dispositivos e chegando a quase 40% com 280 dispositivos para esta categoria. Ainda nesta categoria, PF apresenta um aumento elevado nos valores percentuais chegando a mais de 80% de pacotes não satisfazendo os requisitos de QoS. Considerando a categoria baseado em tempo (Figura 4(b)), Lioumpas 2 apresenta um alto crescimento do valor após 180 dispositivos chegando a um pouco mais de 60% com 280 dispositivos. Proposta-V1 e Proposta-V2 apresentam padrão de crescimento similar à Lioumpas 2, porém seus valores são bem menores, 36% com 280 dispositivos. PF apresenta um crescimento quase constante e chegando à valores similares com 280 dispositivos aos escalonadores Proposta-V1 e PropostaV2.

O escalonador Lioumpas 2 apresenta melhor desempenho na satisfação dos requisitos de QoS da categoria baseada em evento. Porém, a satisfação de QoS da categoria baseado em tempo é comprometida além do problema de inanição que pode ocorrer nela.

## 6. Conclusão e Trabalhos Futuros

Este trabalho apresentou um mecanismo para o escalonador de pacotes no *uplink* da rede LTE para tratar a comunicação M2M utilizando informações do histórico de alocações de recursos, qualidade do canal e os requisitos de QoS dos dispositivos para (i) controlar o impacto da comunicação M2M sobre a H2H, (ii) evitar o problema de inanição com a alocação justa dos recursos e (iii) satisfazer as garantias de QoS.

A partir das análises realizadas dos resultados obtidos pelas simulações, foi observado que a abordagem da separação de recursos entre a comunicação H2H e a M2M,




(a) Tráfego do baseado em evento (b) Tráfego do baseado em tempo

**Figura 4. Média do nível de satisfação de QoS do tráfego M2M**

de tal forma que a quantidade de recursos alocados para a M2M seja controlada, é uma alternativa viável para controlar o impacto da comunicação M2M na H2H. Além disso, o escalonador proposto evita o problema de inanição ao garantir alocação justa dos recursos. Contudo, em um ambiente com uma grande quantidade de tráfego H2H e para cumprir os objetivos (i) e (ii) supracitados, a satisfação dos requisitos de QoS das aplicações baseadas em evento é comprometida.

Como trabalhos futuros, estão planejados implementação e testes em diferentes cenários de tráfego H2H e M2M. Além disso, é pretendido alterar o mecanismo proposto para que as constantes utilizadas sejam variáveis segundo o estado da rede com objetivo de melhorar o desempenho da satisfação dos requisitos de QoS.

## Referências

3GPP (2012). Analysis on traffic model and characteristics for mtc and text proposal. Technical Report R1-120056, 3GPP.

3GPP (2013). Policy and charging control architecture. Technical Specification 23.203, 3GPP.

Abdalla, I. and Venkatesan, S. (2013). A qoe preserving m2m-aware hybrid scheduler for lte uplink. In *Mobile and Wireless Networking (MoWNeT), 2013 International Conference on Selected Topics in*, pages 127–132.

Afrin, N., Brown, J., and Khan, J. (2013). Performance analysis of an enhanced delay sensitive lte uplink scheduler for m2m traffic. In *Telecommunication Networks and Applications Conference (ATNAC), 2013 Australasian*, pages 154–159.

Amokrane, A., Ksentini, A., and Hadjadj-Aoul, Y. (2011). Congestion control in the context of machine type communications in long term evolution networks. Master's thesis, ENS Cachan Bretagne.

Boswarthick, D., Elloumi, O., and Hersent, O. (2012). *M2m communications: a systems approach*. Wiley. com.

Gotsis, A. G., Lioumpas, A. S., and Alexiou, A. (2013). Analytical modelling and performance evaluation of realistic time-controlled m2m scheduling over lte cellular networks. *Transactions on Emerging Telecommunications Technologies*, 24(4):378–388.

Jain, R. (1991). *The art of computer systems performance analysis*, volume 182. John Wiley & Sons Chichester.

Kwan, R. and Leung, C. (2010). A survey of scheduling and interference mitigation in lte. *JECE*, 2010:1:1–1:10.

Lee, S.-B., Pefkianakis, I., Meyerson, A., Xu, S., and Lu, S. (2009). Proportional fair frequency-domain packet scheduling for 3gpp lte uplink. In *INFOCOM 2009, IEEE*, pages 2611–2615.

Lien, S.-Y., Chen, K.-C., and Lin, Y. (2011). Toward ubiquitous massive accesses in 3gpp machine-to-machine communications. *Comm. Mag., IEEE*, 49(4):66–74.

Lim, J., Myung, H., Oh, K., and Goodman, D. (2006). Channel-dependent scheduling of uplink single carrier fdma systems. In *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, pages 1–5.

Lioumpas, A. and Alexiou, A. (2011). Uplink scheduling for machine-to-machine communications in lte-based cellular systems. In *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, pages 353–357.

Miorandi, D., Sicari, S., Pellegrini, F. D., and Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497 – 1516.

NS-3 (2013). The network simulator ns-3. `http://www.nsnam.org`.

Pokhariyal, A., Pedersen, K., Monghal, G., Kovacs, I. Z., Rosa, C., Kolding, T., and Mogensen, P. (2007). Harq aware frequency domain packet scheduler with different degrees of fairness for the utran long term evolution. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 2761–2765.

Potsch, T., Marwat, S., Zaki, Y., and Gorg, C. (2013). Influence of future m2m communication on the lte system. In *Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP*, pages 1–4.

Salah, M., Ali, N., Taha, A.-E., and Hassanein, H. (2011). Evaluating uplink schedulers in lte in mixed traffic environments. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5.

Wu, G., Talwar, S., Johnsson, K., Himayat, N., and Johnson, K. (2011). M2m: From mobile to embedded internet. *Communications Magazine, IEEE*, 49(4):36–43.

Zhenqi, S., Haifeng, Y., Xuefen, C., and Hongxia, L. (2013). Research on uplink scheduling algorithm of massive m2m and h2h services in lte. In *Information and Communications Technologies (IETICT 2013), IET International Conference on*, pages 365–369.

# Proposta de um Mecanismo de Escalonamento em Dois Estágios para o Tráfego de Aplicações em Tempo Real para Redes LTE.

**Johann M. H. Magalhães[1], Paulo R. Guardieiro[2]**

[1]Instituto Federal do Triângulo Mineiro (IFTM)
Campus Uberaba – Uberaba – MG – Brasil

[2]Faculdade de Engenharia Elétrica (FEELT)
Universidade Federal de Uberlândia – Uberlândia – MG - Brasil

johann@iftm.edu.br, guardieiro@ufu.br

***Abstract.*** *In this paper, we propose a new scheduling mechanism for real time traffic applications for LTE networks considering two scheduler steps. In the first step, we use a system based on the theory of digital filters mechanism called Moving-Average Scheduler. In the second step, the scheduler provides the allocation of resource blocks using a Proportional Fair scheduler. The proposed mechanism is compared with other proposed schedulers widely known and exploited in the literature. Simulation results show the proposed mechanism can achieve better performance than other schemes.*

***Resumo****. Neste artigo, propõe-se um novo mecanismo de escalonamento para o tráfego de aplicações em tempo real para redes LTE considerando dois estágios de escalonamento. No primeiro estágio, utiliza-se um mecanismo baseado na teoria de filtros digitais, denominado Moving-Average Scheduler. No segundo estágio, o escalonador provê a alocação dos blocos de recursos utilizando um escalonador Proportional Fair. O mecanismo proposto é comparado com outras propostas de escalonadores amplamente conhecidos e explorados na literatura. Os resultados obtidos por meio de simulações mostraram que o mecanismo proposto apresenta um desempenho superior em relações aos demais escalonadores avaliados.*

## 1. Introdução

No atual cenário das telecomunicações, observa-se o surgimento de várias tecnologias com a finalidade de suprir a demanda dos usuários por serviços móveis com taxas de transmissão de dados cada vez mais elevadas. Nesse contexto, tem-se a tecnologia *Long Term Evolution* (LTE), que vem sendo adotada como o próximo padrão de telefonia móvel pela maioria das operadoras de telefonia celular do mundo. O LTE surge como uma evolução das redes 3G (UMTS) existentes e é padronizado pelo *3rd Generation Partnership Project* (3GPP) [3GPP 2008], que é baseada, principalmente, nas técnicas *Orthogonal Frequency Division Multiplexing* (OFDM) e *Multiple Input Multiple Output* (MIMO), as quais possibilitam melhorias significativas na alocação de recursos de rádio e na adaptação das condições dos enlaces. A tecnologia LTE possibilitará o uso de serviços e aplicações que demandam uma alta taxa de dados, tais como aplicações multimídias, sendo asseguradas as garantias de Qualidade de Serviço (QoS) exigidas por essas aplicações.

Os mecanismos de alocação de recursos têm-se revelado um dos principais desafios para os sistemas LTE [Dahlman, Parkvall and Sköld 2011]. Estes mecanismos são responsáveis por definir como os recursos são distribuídos entre os diferentes usuários. Nos sistemas LTE, o escalonador está localizado na estação rádio base, denominada de *envolved NodeB* (eNodeB), e é responsável pela alocação dinâmica dos recursos dos enlaces de subida e de descida.

A arquitetura de QoS definida pelo padrão LTE inclui mecanismo de escalonamento, entretanto, as políticas que devem ser usadas na implementação desses mecanismos não são especificadas. Nesse caso, tais políticas de escalonamento devem ser implementadas pelos fabricantes e desenvolvedores de equipamentos, permitindo, assim, que eles diferenciem seus produtos. Um escalonador eficiente deve levar em consideração a qualidade do canal do enlace da eNodeB para o terminal de usuário (UE – *User Equipment*) e a capacidade das portadoras, tendo como objetivo mais importante satisfazer as exigências dos usuários, tentando alcançar, ao mesmo tempo, um compromisso ideal entre a utilização e a justiça. Esse objetivo se torna bastante desafiador na presença de aplicações em tempo real, especialmente considerando as restrições em relação aos requisitos para atraso e largura de banda.

O LTE utiliza, em seu enlace de descida, a tecnologia *Orthogonal Frequency Division Multiple Access* (OFDMA), que possibilita explorar a diversidade multiusuário e ter-se o escalonamento dos pacotes, tanto no domínio do tempo (TDS – *Time Domain Scheduler*) quanto no domínio da frequência (FDS – *Frequency Domain Scheduler*). Diante disto, neste artigo, propõe-se um mecanismo que realiza o escalonamento em dois estágios para o tráfego de aplicações em tempo real em redes LTE. No primeiro estágio é proposto um mecanismo de alocação de recursos baseado na teoria de filtros digitais denominado *Moving-Average Scheduler* (MAS). O MAS define, em cada tempo de amostragem, a quantidade de dados que cada fluxo em tempo real deve transmitir para satisfazer às necessidades de qualidade de serviço do fluxo. No segundo estágio, o escalonador provê a alocação dos blocos de recursos utilizando um escalonador *Proportional Fair* (PF) [Jalali, Padovani and Pankaj 2000] considerando as restrições impostas pelo MAS.

O restante deste artigo está organizado da seguinte maneira. Na Seção 2, os trabalhos relacionados são apresentados. Uma breve explanação sobre o mecanismo de alocação de recursos em redes LTE é feita na Seção 3. A seguir, na Seção 4, a proposta do mecanismo de escalonamento é apresentada. Na Seção 5, as avaliações e os respectivos resultados são discutidos e, finalmente, na Seção 6, são apresentas as considerações finais a respeito do trabalho.

## 2. Trabalhos relacionados

Vários mecanismos de escalonamento de pacotes para redes LTE são propostos na literatura. O *Maximum Throughput* (MT) e o *Proportional Fair* (PF) [Jalali, Padovani and Pankaj 2000], para os quais as decisões de escalonamento estão estritamente relacionadas com a qualidade do canal experimentada pelos UEs, não são muito adequados para tráfego em tempo real. Em se tratando de tráfego em tempo real, têm-se as seguintes opções de mecanismos de escalonamento: *Modified Largest Weighted Delay First* (M-LWDF) [Andrews *et al.* 2001], *Exponential-PF* (EXP/PF) [Basukala, Mohd Ramli and Sandrasegaran 2009], *Exponential Rule* (EXP-RULE) [Shakkottai and

Stolyar 2002] e *Log Rule* (LOG-RULE) [Sadiq, Baek and de Veciana 2009]. Esses mecanismos citados acima estão entre os mais conhecidos e explorados na literatura referente a escalonamento em redes LTE.

Nos últimos anos, pesquisadores têm proposto escalonadores que trabalham conjuntamente no domínio do tempo e no domínio da frequência em mais de uma etapa. Diante disso, [Pokhariyal; Kolding and Mogensen 2006] avaliaram o desempenho do escalonamento de pacotes no domínio da frequência considerando-se as condições instantâneas do canal. Comparando com um escalonador que trabalha apenas no domínio do tempo, o FDS apresentou um ganho da ordem de 40% na capacidade média do sistema e na taxa média de dados. Em outro trabalho [Pokhariyal *et al.* 2006], também se avalia o desempenho de um escalonador desenvolvido em duas camadas, um escalonador no domínio do tempo, seguido de um escalonador no domínio da frequência. Por meio dos resultados obtidos, o escalonamento, considerando-se ambos os domínios, obteve uma melhora de 35% na taxa de vazão de dados em comparação com algoritmos que realizam o escalonamento considerando apenas o domínio do tempo.

Também explorando o ganho potencial da diversidade conjunta no domínio do tempo e da frequência, pesquisadores [Beh, Armour and Doufexi 2008] utilizaram diversos escalonadores já conhecidos e presentes na literatura para realizarem o escalonamento no domínio da frequência e propor um novo escalonador no domínio tempo. Os resultados obtidos também mostraram que, com o uso conjunto de escalonadores em ambos os domínios, foi possível obter uma significativa melhoria quando comparado com um escalonador apenas no domínio do tempo.

Considerando-se escalonadores para o tráfego em tempo real, Sandrasegaran, Ramli and Basukala (2010) propõem um algoritmo de escalonamento que, no domínio do tempo, faz-se o cálculo para determinar os usuários com o tempo mais próximo de expirar e, no domínio da frequência, seleciona-se o melhor RB para a transmissão dos dados dos usuários selecionados anteriormente no domínio do tempo. Em Piro *et al.* (2011), também se propõe um escalonador em duas camadas para o tráfego em tempo real, em que, na camada superior, utiliza-se um algoritmo de escalonamento baseado na teoria de controle, denominado *Frame Level Scheduler* (FLS), e, na camada inferior, utilizou-se o algoritmo PF para alocação dos RBs no domínio da frequência. Como uma extensão deste trabalho, Tsang (2013) propõe um algoritmo de escalonamento para o tráfego de vídeo denominado *QoS-aware Two-layer Scheduling*. Nesse algoritmo, introduz-se, na segunda camada, um fator dinâmico que determina a urgência dos pacotes e a justiça entre os fluxos para que o escalonador assegure a transmissão dos pacotes mais importantes na eNodeB.

Assim como em Piro *et al.* (2011), o trabalho proposto no presente artigo, considera um primeiro estágio no domínio do tempo que determina a quantidade de dados a ser retirada das filas a cada tempo de amostragem. Entretanto, diferentemente do proposto em Piro *et al.* (2011), utiliza-se uma abordagem baseada na teoria de filtros digitais do tipo FIR para se determinar essa quantidade de dados. Uma das vantagens de se utilizar um filtro FIR é que não há a necessidade de se preocupar com a estabilidade do sistema, conforme será mostrado na Seção 4.2.

## 3. Alocação de recursos em redes LTE

A alocação de recursos é responsável por definir como estes recursos serão distribuídos entre os diferentes usuários. Nas redes LTE, as transmissões nos enlaces de descida e de subida são organizadas em quadros de 10 ms, sendo que cada quadro é composto de 10 subquadros com duração de 1 ms. Além disso, cada subquadro é dividido em dois *slots* com duração de 0,5 ms cada [Khan 2009]. Esta estrutura de quadros é mostrada na Figura 1.

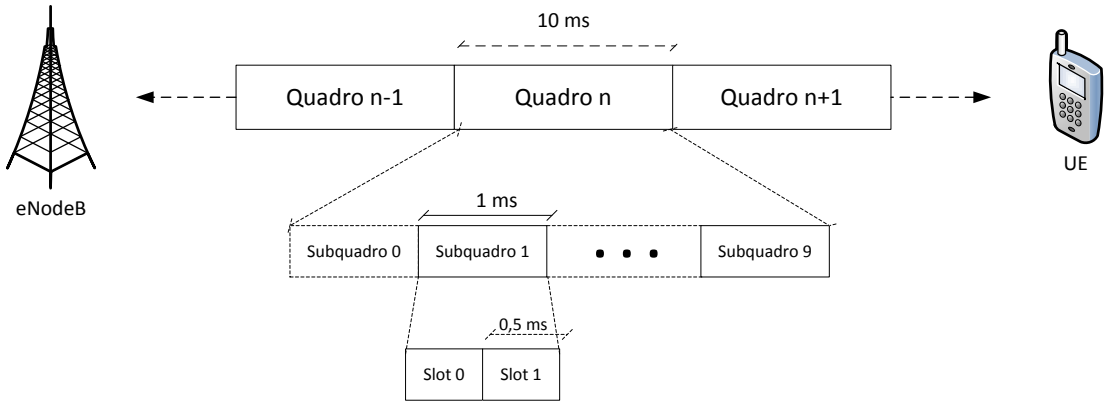



Figura 1. Estrutura de quadros em redes LTE.

Cada *slot* constitui um *Resource Block* (RB), o qual é caracterizado por ser composto por um conjunto de subportadoras consecutivas no domínio da frequência e um número de símbolos OFDM consecutivos no domínio do tempo (12 subportadoras x 7 símbolos OFDM). Um subquadro constitui um *Transmission Time Interval* (TTI), o qual é a menor unidade de transmissão de dados. A cada TTI, o UE calcula o *Channel Quality Indicator* (CQI) de acordo com a relação sinal/ruído (SINR - *Signal to Interference plus Noise Ratio*) e informa para a eNodeB as condições instantâneas do canal do enlace de descida. Para cada usuário, é associado um *buffer* na eNodeB. Os pacotes que chegam nesses *buffers* recebem uma marca de tempo e são enfileirados para transmissão baseada em um sistema FIFO. Para cada pacote na fila, calcula-se o *Head of Line Delay* (HOL). Se o HOL exceder o limite de atraso especificado para o fluxo, então, o pacote é descartado.

O escalonador de pacotes determinará quais usuários serão escalonados de acordo com um algoritmo de escalonamento. A seleção desses usuários pode ser realizada por meio de priorização conforme o cálculo de uma métrica, a qual se pode basear na condição do canal, no atraso dos pacotes, no estado do *buffer*, no tipo de serviço, etc.

## 4. Mecanismo de escalonamento proposto

Considerando-se as vantagens de utilizar o escalonamento de pacotes em mais de um estágio, este artigo propõe um mecanismo que utiliza tanto o escalonamento no domínio do tempo, quanto o escalonamento no domínio da frequência. No domínio do tempo, o escalonador MAS define, em cada tempo de amostragem, a quantidade de dados que cada fluxo em tempo real deve transmitir para satisfazer as necessidades de qualidade

de serviço do fluxo. No domínio da frequência, o escalonador provê a alocação dos RBs utilizando-se o escalonador PF. No caso dos fluxos de melhor esforço, utiliza-se um escalonador PF para compartilhar os RBs não utilizados pelos fluxos em tempo real. Um modelo simplificado do mecanismo de escalonamento proposto é mostrado na Figura 2.

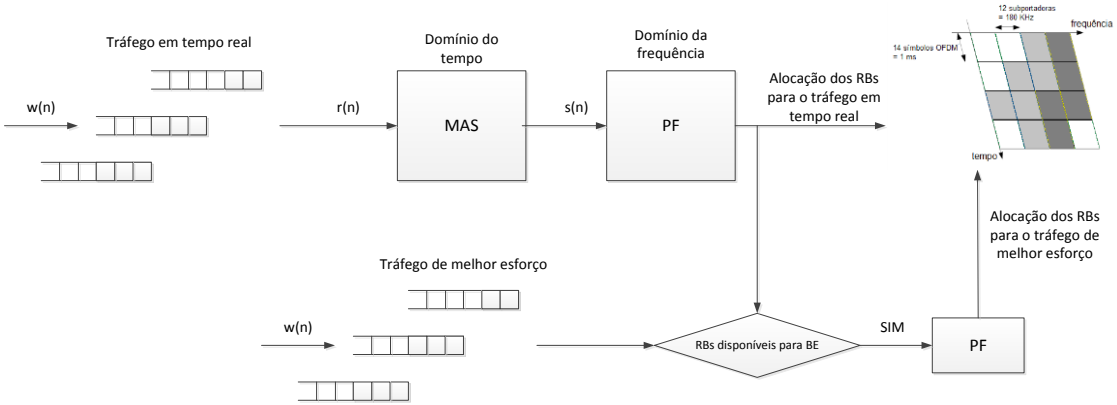



Figura 2. Modelo simplificado do mecanismo de escalonamento em dois estágios proposto.

O escalonador MAS é projetado utilizando-se a teoria de filtros digitais do tipo *Finite Impulse Response* (FIR) [Oppenheim and Schafer 2010]. Para isso, consideram-se *M* fluxos de dados ativos compartilhando a interface aérea de uma eNodeB. Para cada um destes fluxos, é associada uma fila, em que os pacotes são armazenados, esperando para serem transmitidos. Essas filas podem ser modeladas pela seguinte equação de diferenças [Chisci *et al.* 2004]:

$$r_i(n+1) - r_i(n) = w_i(n) - s_i(n) \quad (1)$$

Onde:

$r_i(n)$: comprimento da i-ésima fila no instante de amostragem $n$.

$r_i(n+1)$: comprimento da i-ésima fila no instante de amostragem $n+1$.

$s_i(n)$: quantidade de dados a ser transmitida no instante de amostragem $n$.

$w_i(n)$: quantidade de dados que chega na fila durante o instante de amostragem $n$.

Conforme mostrado na Figura 3, o sinal de entrada para o filtro FIR será o comprimento da fila *i* no instante de amostragem *n*. Dessa maneira, a saída do filtro $s_i(n)$ irá resultar na quantidade de dados que deverá ser retirado da fila a cada instante de amostragem para atender às necessidades de qualidade de serviço dos fluxos em tempo real. Essa quantidade é informada para o escalonador no próximo estágio, o qual irá alocar os RBs para os usuários.

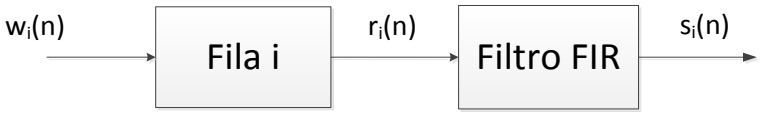



Figura 3. Sinais de entrada e de saída para o filtro.

No sistema proposto, o tempo de amostragem é dado pelo tempo de duração de um quadro LTE, ou seja, 10 ms. Dessa maneira, o intervalo de amostragem será dado por $\Delta t(n) = t_{n-1,i} - t_{n,i}$ , sendo $t_{n,i}$ o tempo de início do *n-ésimo* quadro. O tempo de amostragem será denominado como $T_q$ (tempo de quadro).

### 4.1. Modelagem do escalonador MAS

Os filtros FIR são conhecidos também como filtros *feedforward*, pois utilizam uma cópia atrasada do sinal de entrada (por número *N* de amostras) e combina este sinal com o novo sinal de entrada.

Para a proposta do escalonador MAS, optou-se por um filtro FIR de média móvel, considerando-se que a ordem *N* do filtro será definida de acordo com as necessidades dos fluxos, em relação ao atraso e à vazão dos dados.

Um filtro FIR de média móvel tem a sua forma geral definida segundo a equação (2). Este filtro calcula a *n-ésima* amostra do sinal de saída como sendo a média das *(N1+N2+1)* amostras do sinal de entrada em torno da *n-ésima* amostra.

$$y(n) = \frac{1}{N_1+N_2+1} . \sum_{k=-N_1}^{N_2} x(n-k) \quad (2)$$

$$= \frac{1}{N_1+N_2+1} . [x(n+N_1) + x(n+N_1-1) + \ldots + x(n) + x(n-1) + \ldots + x(n-N_2)]$$

Considerando que o tamanho da fila nunca será negativo, o filtro FIR de média móvel utilizado no escalonador MAS será representado pela equação (3) e pelo diagrama de blocos apresentado na Figura 4.

$$s(n) = \frac{1}{N+1} . \sum_{k=0}^{N} r(n-k) \quad (3)$$

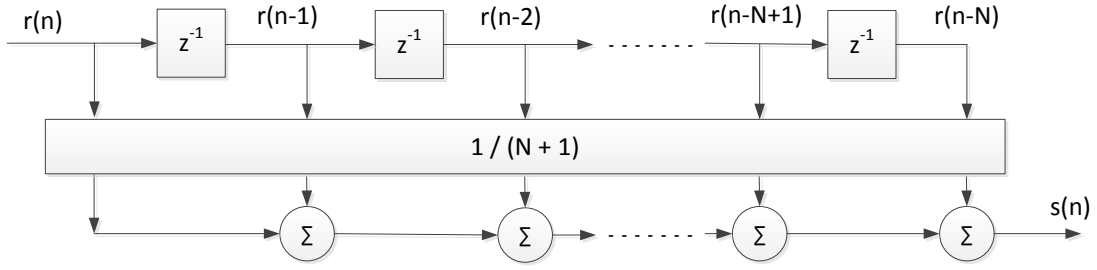



Figura 4. Diagrama de blocos de um filtro FIR de média móvel de ordem *N*.

### 4.2. Estabilidade do filtro

Um filtro FIR de média móvel tem a função de transferência dada pela equação (4). Nessa equação, as raízes do polinômio no numerador são conhecidas como zeros e as raízes do polinômio no denominador são conhecidas como polos. Para a estabilidade de um filtro digital discreto no tempo, todos os polos devem estar dentro do círculo unitário do plano *z* e, conforme a equação (4), observa-se que todos os polos estão situados no plano *z*. Portanto, esse filtro de ordem *N* é estável.

$$H(z) = \frac{1}{N+1}\sum_{k=0}^{N} z^{-k} = \frac{1}{N+1}.\left[\frac{\sum_{k=0}^{N} z^{(N-k)}}{z^N}\right] \quad (4)$$

### 4.3. Resposta ao impulso unitário

A resposta ao impulso unitário é a saída de um filtro quando o sinal de entrada é um impulso unitário,

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \quad (5)$$

A sequência de entrada pode ser considerada como uma combinação de uma sequência de impulsos dado por,

$$r(n) = \sum_{k=-\infty}^{\infty} r_k.\delta(n-k) \quad (6)$$

A saída *s(n)* do filtro será dada por:

$$s(n) = \sum_{k=0}^{N} b_k.\delta(n-k) = \quad (7)$$
$$= b_0\delta(n) + b_1\delta(n-1) + b_2\delta(n-2) + \ldots + b_n\delta(n-n) + \ldots + b_N\delta(n-N)$$
$$= 0 + \ldots + 0 + b_n\delta(0) + 0 + \ldots + 0 = b_n$$

Assim,

$$s(n) = \sum_{k=0}^{N} b_k.\delta(n-k) = b_n \quad (8)$$
$$para\ n = 0,1,2,\ldots,N \text{ e } b_k = \frac{1}{N+1}$$

Nesse caso, a resposta ao impulso será simplesmente a sequência de números que corresponde aos coeficientes do filtro. Como *s(n)* contém todas as informações a respeito do filtro, o comprimento de *s(n)* será igual ao número de coeficientes *(N+1)*, que é um número inteiro e finito.

Diante disso, o tempo máximo de espera de um pacote na fila será de *N* intervalos de amostragem,

$$\tau_i = N_i.T_q \quad (9)$$

Assim, a ordem do filtro irá determinar o atraso máximo que o pacote poderá esperar na fila antes de ser transmitido. Considerando o tempo de quadro ($T_q = 10\ ms$) e $\tau_{max,i}$ o atraso máximo permitido para o fluxo *i*, a ordem do filtro a ser utilizado no escalonador MAS será definida por,

$$N = \left(\frac{\tau_{max,i}}{T_q}\right) - 1 \quad (10)$$

## 5. Avaliação de desempenho da proposta

Esta seção apresenta os resultados obtidos por meio de simulações, comparando-se o desempenho do escalonador proposto com outras disciplinas de escalonamento amplamente exploradas na literatura (M-LWDF, EXP-PF, EXP-RULE e LOG-RULE). Para essas simulações, utilizou-se o simulador LTE-Sim [Piro *et al.* 2011].

### 5.1. Ambiente de simulação

Para a realização das simulações, utilizou-se um cenário composto por 19 células com raio igual a 1 km, considerando-se *clusters* compostos por quatro células e interferência intercelular. Considerou-se um número de usuários escolhidos dentro de uma faixa de 10 a 60, movendo-se a velocidades de 3 e 120 km/h no interior da célula central. Cada UE recebe, ao mesmo tempo, um fluxo de vídeo, um fluxo de VoIP e um fluxo de melhor esforço (BE). Cada simulação tem a duração de 70 segundos. A Tabela 1 resume os parâmetros utilizados nas simulações.

Tabela 1. Parâmetros utilizados nas simulações.

| **Parâmetro** | **Valor** |
|---|---|
| Tempo de simulação | 70 s |
| Enlace de descida | 10 MHz |
| Estrutura do quadro | FDD |
| TTI | 1 ms |
| Número de RBs | 50 |
| Esquema de reutilização de frequências | *Clusters* de 4 células |
| Modelo de propagação | *Macro-cell urban model* |
| Número de células | 19 |
| Raio da célula | 1 km |
| Número de UEs | 10, 20, 30, 40, 50 e 60 |
| Mobilidade do UE | *Random way-point* |
| Velocidade do UE | 3 km/h e 120 km/h |
| Tráfego em tempo-real | H264, VoIP |
| Fluxo de melhor esforço | *Infinite buffer* |
| Atraso máximo | 50 e 150 ms |

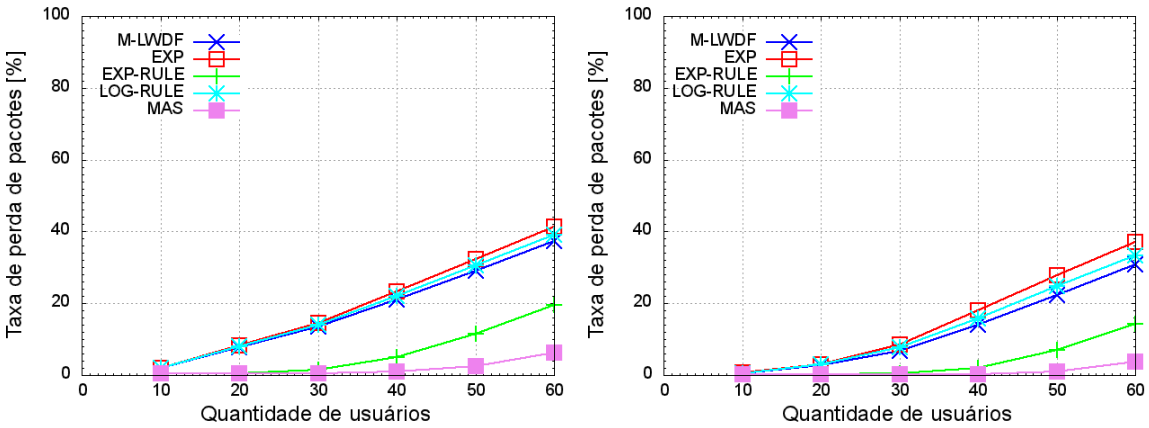
Nas simulações realizadas, foram utilizados como atrasos máximos suportados pelos fluxos em tempo real os valores de 50 e 150 ms. Conforme a equação (10), para estes valores de atraso, a ordem dos filtros utilizados é apresentada na Tabela 2. Os pacotes que ultrapassarem esses limites de atraso serão descartados.

Tabela 2. Ordem do filtro de acordo com os atrasos máximos permitidos.

| *Atraso Máximo Permitido ($\tau_{max,i}$)* | *Ordem do Filtro (N)* | *Comprimento do Filtro (N+1)* |
|---|---|---|
| 50 ms | 4 | 5 |
| 150 ms | 14 | 15 |

### 5.2. Apresentação e análise de resultados

Inicialmente, avaliou-se a taxa de perda de pacotes (PLR - *Packet Loss Ratio*), considerando-se a variação da quantidade de usuários simultaneamente na célula central e considerando os atrasos máximos permitidos, conforme a Tabela 2. A análise da PLR é importante em se tratando de tráfego em tempo real, considerando-se que os escalonadores analisados descartam os pacotes que extrapolam o atraso máximo permitido. Isso é baseado no fato de que não seria vantagem nenhuma transmitir ou receber pacotes de aplicações de tempo real que já tenham extrapolado os requisitos de atraso máximo, caracterizando um desperdício de recursos. As Figuras 5 e 6 apresentam os resultados para a PLR para os escalonadores avaliados em relação ao tráfego de vídeo. Observa-se que a taxa de perda dos pacotes aumenta em relação à quantidade de usuários na célula, devido ao aumento de carga na rede. O escalonador MAS consegue manter uma PLR baixa mesmo com o aumento desta carga na rede. Para condições mais adversas, com velocidade de deslocamento dos usuários a 120 km/h e imposição de atraso máximo de 50 ms (Figura 6a), o MAS obteve valores de PLR abaixo dos demais escalonadores avaliados.




(a) (b)

Figura 5. Taxa de perda de pacotes dos fluxos de vídeo, com deslocamento dos usuários a 3 km/h e atrasos máximos permitidos (a) 50 ms e (b) 150 ms.

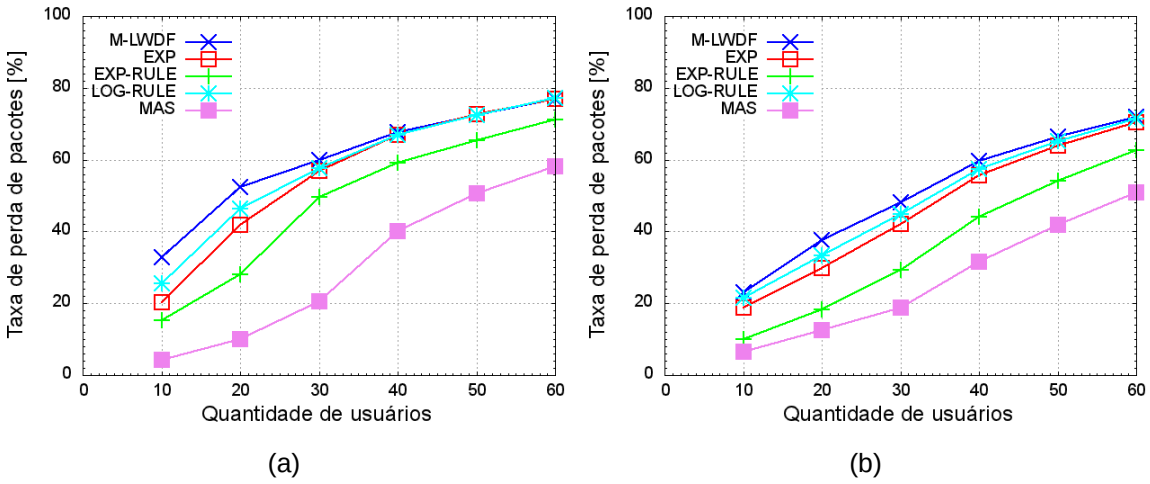



Figura 6. Taxa de perda de pacotes dos fluxos de vídeo, com deslocamento dos usuários a 120 km/h e atrasos máximos permitidos (a) 50 ms e (b) 150 ms.

Para os fluxos VoIP, mostrados na Figura 7, em geral, todos os escalonadores apresentaram uma baixa taxa de perda de pacotes considerando-se uma velocidade de deslocamento do UE de 3 km/h. Para uma velocidade de deslocamento de 120 km/h, os escalonadores EXP-RULE e MAS apresentaram um aumento menor na PLR à medida que o número de usuários se tornou maior, conforme mostrado na Figura 8. Comparando-se com os fluxos de vídeo, os fluxos VoIP experimentam uma taxa de perda bem menor, devido, principalmente, à pequena taxa de bits que esse tipo de fluxo requer. Esses fluxos também obtêm uma maior prioridade nos escalonadores, nos quais se considera a taxa média de dados obtida para esses fluxos no cálculo da métrica.

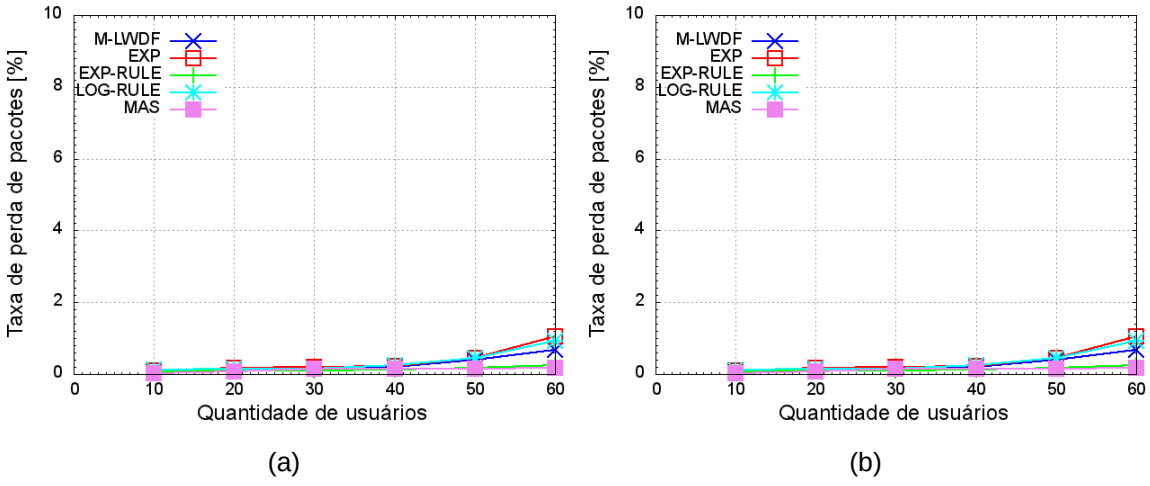



Figura 7. Taxa de perda de pacotes dos fluxos VoIP, com deslocamento dos usuários a 3 km/h e atrasos máximos permitidos (a) 50 ms e (b) 150 ms.

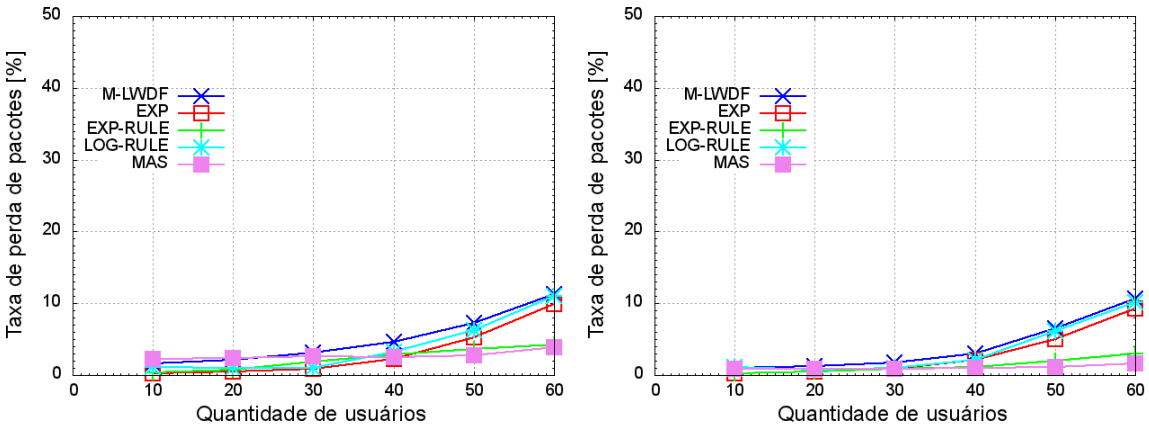



Figura 8. Taxa de perda de pacotes dos fluxos VoIP, com deslocamento dos usuários a 120 km/h e atrasos máximos permitidos (a) 50 ms e (b) 150 ms.

Os valores obtidos para a PLR, considerando os fluxos de vídeo e VoIP analisados, estão relacionados com a capacidade que o escalonador tem de servir a maior quantidade possível de pacotes dentro dos limites de atraso estabelecidos, possibilitando que mais pacotes sejam preservados do descarte por terem atingido esses limites.

Em seguida, avaliou-se o comportamento dos fluxos de melhor esforço. Para tanto, utilizou-se a medida da vazão agregada, a qual mede a vazão total obtida por estes fluxos durante todo o período de simulação. Os resultados apresentados nas Figuras 9 e 10 mostram que, à medida que o número de usuários aumenta, os demais escalonadores mantêm uma alta vazão quando comparados com o MAS. Dessa maneira, o MAS consegue prover uma melhor vazão para o tráfego em tempo real em detrimento dos fluxos de melhor esforço. Quando a quantidade de usuários é pequena, o MAS consegue prover uma vazão elevada para os fluxos BE. Entretanto, quando a quantidade de usuários aumenta e os recursos ficam mais escassos, o MAS sacrifica a vazão deste tipo fluxo para manter uma vazão elevada dos fluxos de vídeo e VoIP.

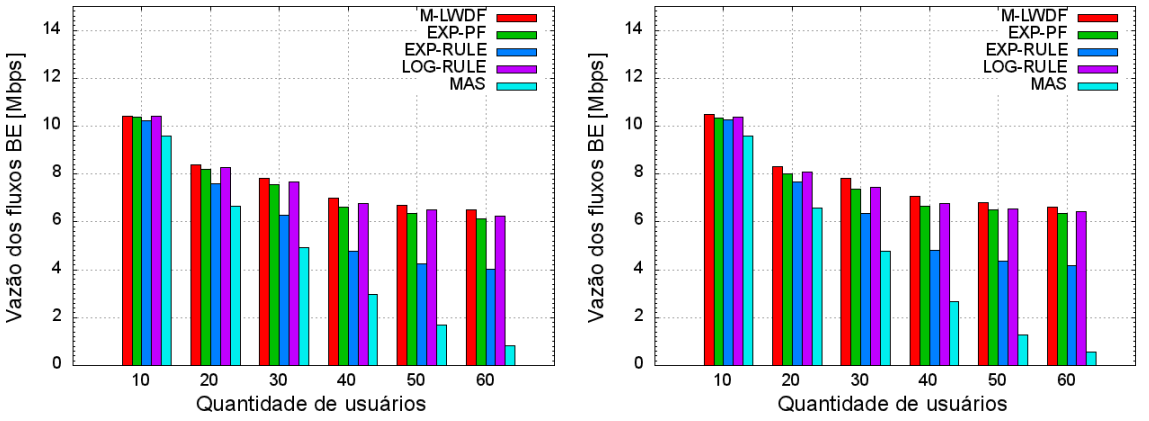



Figura 9. Vazão agregada dos fluxos de melhor esforço, com deslocamento dos usuários a 3 km/h e atrasos máximos permitidos (a) 50 ms e (b) 150 ms.

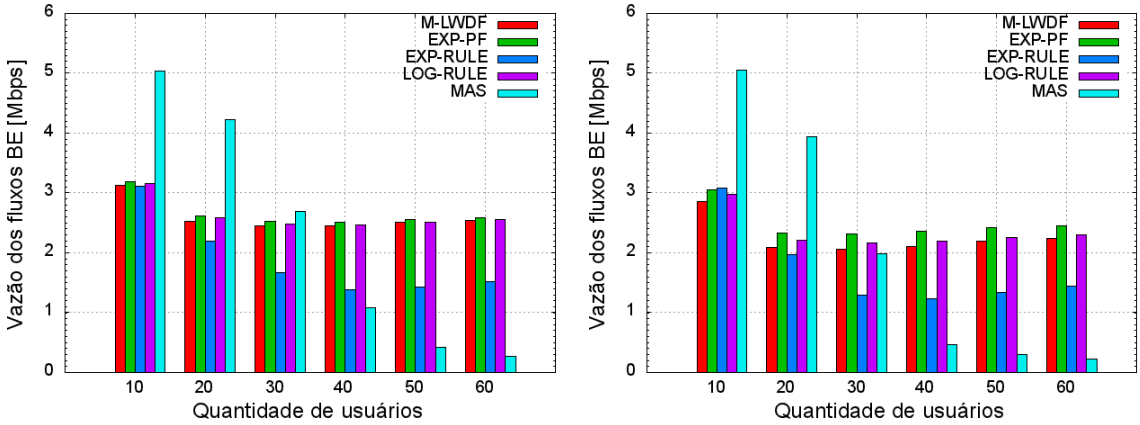



Figura 10. Vazão agregada dos fluxos de melhor esforço, com deslocamento dos usuários a 120 km/h e atrasos máximos permitidos (a) 50 ms e (b) 150 ms.

Outra métrica importante a ser avaliada é o índice de justiça (*fairness*). Essa é uma medida que deve ser levada em conta para garantir um desempenho mínimo também para os usuários que experimentam condições de canal não tão favoráveis. As Tabelas 3 e 4 mostram que o escalonador proposto consegue manter um índice de justiça elevado. As piores situações são observadas no caso dos usuários deslocando-se com velocidade de 120 km/h, para os quais as condições de canal podem mudar muito rapidamente.

Tabela 3 – Índice de justiça para os fluxos de vídeo do escalonador MAS.

| *Quantidade de Usuários* | *3 km/h* | | *120 km/h* | |
|---|---|---|---|---|
| | *50 ms* | *150 ms* | *50 ms* | *150 ms* |
| 10 | 0,99938 | 0,99959 | 0,99830 | 0,99899 |
| 20 | 0,99927 | 0,99957 | 0,99293 | 0,99524 |
| 30 | 0,99909 | 0,99951 | 0,97789 | 0,97938 |
| 40 | 0,99787 | 0,99942 | 0,93386 | 0,82195 |
| 50 | 0,99415 | 0,99811 | 0,82268 | 0,67380 |
| 60 | 0,98167 | 0,98110 | 0,70795 | 0,56946 |

Tabela 4 – Índice de justiça para os fluxos VoIP do escalonador MAS.

| *Quantidade de Usuários* | *3 km/h* | | *120 km/h* | |
|---|---|---|---|---|
| | *50 ms* | *150 ms* | *50 ms* | *150 ms* |
| 10 | 0,97646 | 0,98718 | 0,97651 | 0,98193 |
| 20 | 0,97614 | 0,97757 | 0,97568 | 0,96954 |
| 30 | 0,97360 | 0,97697 | 0,96963 | 0,97056 |
| 40 | 0,97178 | 0,97589 | 0,97237 | 0,97356 |
| 50 | 0,97088 | 0,9748 | 0,97494 | 0,97759 |
| 60 | 0,96932 | 0,97323 | 0,97607 | 0,97522 |

## 6. Conclusões

Neste artigo, abordou-se o problema relativo ao escalonamento de pacotes provenientes do tráfego de aplicações em tempo real para o enlace de descida em redes LTE. Com o intuito de prover um melhor desempenho para o escalonamento desses pacotes foi proposto um escalonador que atua tanto no domínio do tempo quanto no domínio da frequência. No domínio do tempo, o escalonador faz uso da teoria de filtros digitais, mais especificamente um filtro FIR de média móvel, o qual é responsável por determinar a quantidade de dados das filas para o tráfego em tempo real que deve ser alocada em cada quadro. Considerando o cenário e as métricas utilizadas, os resultados apresentados por meio de simulações mostraram que o escalonador proposto obteve um melhor desempenho quando comparado com outras estratégias de escalonamento bem conhecidas na literatura.

## 7. Referências

3GPP (2008), Tech. Specif. Group Radio Access Network - Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN), 3GPP TS 25.913.

Andrews, M., Kumaran, K., Ramanan, K., Stolyar, A., Whiting, P. and Vijayakumar R (2001) “Providing quality of service over a shared wireless link,” IEEE Communications Magazine, vol.39, no.2, pp.150-154.

Basukala, R., Mohd Ramli, H. A. and Sandrasegaran, K. (2009) “Performance analysis of EXP/PF and M-LWDF in downlink 3GPP LTE system”, First Asian Himalayas International Conference on Internet, pp.1-5.

Beh, K. C., Armour, S. and Doufexi, A. (2008) “Joint Time-Frequency Domain Proportional Fair Scheduler with HARQ for 3GPP LTE Systems,” 68th IEEE Vehicular Technology Conference, pp.1,5.

Chisci, L.; Fantacci, R.; Francioli, F. and Pecorella, T. (2004) “Multi-terminal dynamic bandwidth allocation in GEO satellite networks,” 59th IEEE Vehicular Technology Conference, vol.5, pp.2797,2801.

Dahlman, E., Parkvall, S and Sköld, J (2011) "4G LTE/LTE-Advanced for Mobile Broadband" Elsevier, 1st. Edition, UK. ISBN: 978-0-12-385489-6.

Jalali, A., Padovani, R. and Pankaj, R. (2000) "Data throughput of CDMA-HDR a high efficiency-high data rate personal communication wireless system," 51st IEEE Vehicular Technology Conference Proceedings, vol.3, pp.1854,1858.

Khan, F. (2009) "LTE for 4G Mobile Broadband - Air Interface Technologies and Performance", 1st Edition, Cambridge University Press. ISBN: 978-0-511-51666-5.

Oppenheim, A. V. and Schafer, R. W. (2010) "Discrete-time signal processing," 3th Edition, Prentice-Hall. ISBN 0-13-198842-5.

Piro, G., Grieco, L., Boggia, G., Capozzi, F. and Camarda, P. (2011) "Simulating LTE cellular systems: an open-source framework", IEEE Transaction on Vehicular Technology, vol. 60, no. 2, pp. 498 –513.

Piro, G.; Grieco, L.A.; Boggia, G.; Fortuna, R.; Camarda, P. (2011) "Two-Level Downlink Scheduling for Real-Time Multimedia Services in LTE Networks," IEEE Transactions on Multimedia, vol.13, no.5, pp.1052,1065.

Pokhariyal, A.; Kolding, T.E. and Mogensen, P.E. (2006) "Performance of Downlink Frequency Domain Packet Scheduling for the UTRAN Long Term Evolution," 17th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp.1,5.

Pokhariyal, A.; Pedersen, K.I.; Monghal, G.; Kovacs, I. Z.; Rosa, C.; Kolding, T.E. and Mogensen, P.E. (2007) "HARQ Aware Frequency Domain Packet Scheduler with Different Degrees of Fairness for the UTRAN Long Term Evolution," 65th IEEE Vehicular Technology Conference, pp.2761,2765.

Sadiq, B., Baek, S.J. and de Veciana, G. (2009) "Delay-optimal opportunistic scheduling and approximations: the Log rule," 27th Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '09), pp. 1–9.

Sandrasegaran, K.; Ramli, H.A.M.; Basukala, R. (2010) "Delay-Prioritized Scheduling (DPS) for Real Time Traffic in 3GPP LTE System," IEEE Wireless Communications and Networking Conference (WCNC), pp.1,6.

Shakkottai S. and Stolyar, A. (2002) "Scheduling for multiple flows sharing a time-varying channel: the exponential rule," in Analytic Methods in Applied Probability, vol. 207 of American Mathematical Society Translations, Series 2, A Volume in Memory of F. Karpelevich, pp. 185–202.

Tsang, T.(2013) "Performance Analysis for QoS-Aware Two-Layer Scheduling in LTE Networks," International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 2.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 7*

## *Tolerância a Falhas e Resiliência*

# Árvores Geradoras Mínimas Distribuídas e Autonômicas

**Luiz A. Rodrigues[1,2], Elias P. Duarte Jr.[2] e Luciana Arantes[3]**

[1] Colegiado de Ciência da Computação – Universidade Estadual do Oeste do Paraná
Caixa Postal 801 – 85819-110 – Cascavel – PR – Brasil

[2]Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – 81531-980 – Curitiba – PR – Brasil

[3]Laboratoire d'Informatique - Université Pierre et Marie Currie
CNRS/INRIA/REGAL – Place Jussieu, 4 – 75005 – Paris, France

luiz.rodrigues@unioeste.br, elias@inf.ufpr.br, luciana.arantes@lip6.fr

***Abstract.*** *This paper presents an autonomic algorithm for constructing minimum spanning trees in a distributed system. Processes are virtually organized on a hypercube-like topology, called VCube. The spanning trees are dynamically created from any source process and regenerated transparently when processes become faulty. Any tree reconstruction is autonomously done based on the virtual topology and on fault information, supporting up to n-1 faulty processes. Two applications using the autonomic spanning tree algorithm are proposed, one for the best-effort broadcast and another for reliable broadcast. Besides the formal specification, simulation results are presented, including comparison results with other alternative solutions.*

***Resumo.*** *Este artigo apresenta um algoritmo autonômico para a construção de árvores geradoras mínimas (spanning trees) em um sistema distribuído. Os processos do sistema são virtualmente organizados em uma topologia baseada em hipercubo, denominada* VCube*. As árvores são criadas dinamicamente a partir de qualquer processo como raiz e regeneradas de forma transparente quando processos falhos são detectados. A reconstrução é feita autonomicamente com base na topologia virtual e nas informações de diagnóstico, suportando até* $n - 1$ *processos falhos. Com base no algoritmo de árvore, duas aplicações são propostas: uma para o broadcast de melhor-esforço e outra para broadcast confiável. Além da especificação formal, resultados de simulação são apresentados, inclusive comparando os algoritmos com outras soluções alternativas.*

## 1. Introdução

Árvores geradoras (*spanning trees*) são utilizadas na solução de diversos problemas em sistemas distribuídos, tais como exclusão mútua, agrupamento, fluxo em redes, sincronização e difusão de mensagens (*broadcast*) [Elkin 2006, England et al. 2007, Dahan et al. 2009]. A principal motivação é encontrar uma rede de baixo custo que conecta todos os processos do sistema. Para a difusão de mensagens, por exemplo, uma solução alternativa simples é utilizar inundação (*flooding*) para enviar uma mensagem a todos os processos do sistema. O inconveniente desta solução é o custo da comunicação devido ao grande número de vezes que a mensagem é retransmitida a um mesmo processo. Por outro lado, as árvores geradoras minimizam este custo, visto que a mensagem precisa ser transmitida somente pelas $n - 1$ arestas da árvore [Gärtner 2003].

A possibilidade de ocorrência de falhas é intrínseca aos sistemas distribuídos. Uma aplicação distribuída tolerante a falhas precisa continuar sua execução corretamente na presença de falhas, sem um comprometimento do desempenho. Idealmente a adaptação do serviço deve ocorrer de forma transparente, como acontece nos sistemas denominados autonômicos [Kephart e Chess 2003]. Para os algoritmos de árvores geradoras distribuídas, além do problema de construção, existe ainda o custo de manutenção das árvores quando ocorrem falhas dos seus nodos, que implica na sua reconstrução ou reconfiguração após uma falha.

O algoritmo proposto neste trabalho permite a propagação de mensagens de uma aplicação distribuída por meio de uma árvore geradora mínima construída de forma distribuída a partir de uma topologia baseada em hipercubo virtual, denominada VCube. No VCube, os processos são organizados em clusters hierárquicos progressivamente maiores. A partir da raiz principal da árvore, cada cluster possui um processo lider que é a raiz da sub-árvore naquele cluster, responsável pela propagação das mensagens nos clusters menores. Em função da topologia em hipercubo, a solução apresenta importantes propriedades logarítmicas, mesmo quando processos falham. Além disso, as árvores são reconstruídas de forma autonômica após a ocorrência de falhas.

Duas aplicações distribuídas para *broadcast* foram implementadas sobre o algoritmo autonômico proposto: uma para *broadcast* de melhor-esforço e outra para *broadcast* confiável. Resultados de simulação mostram que o uso da topologia virtual e do algoritmo de árvore aumentam a escalabilidade do sistema e reduzem a latência de propagação das mensagens nas duas aplicações.

O restante deste artigo está organizado nas seguintes seções. A Seção 2 descreve os trabalhos relacionados. A Seção 3 apresenta as definições e o modelo do sistema. A topologia virtual VCube é descrita na Seção 4. A Seção 5 apresenta o algoritmo de árvore geradora proposto. As aplicações baseadas em árvore são descritas na Seção 6 e resultados de simulação são apresentados na Seção 7. A Seção 8 apresenta a conclusão e os trabalhos futuros.

## 2. Trabalhos Relacionados

Os dois algoritmos clássicos para a obtenção de árvores geradoras mínimas a partir de um grafo são o algoritmo de Kruskal e Joseph (1956) e o proposto por Prim (1957). O algoritmo de Kruskal inicialmente cria uma floresta na qual cada vértice é uma árvore. A cada passo, as árvores são conectadas entre si através das arestas de menor peso. As arestas que não interligam duas árvores são descartas, evitando ciclos. Ao final, uma única componente conexa é gerada e esta constitui a árvore geradora mínima do grafo. O algoritmo de Prim utiliza uma abordagem diferente, que emprega cortes mínimos para escolher as arestas de menor peso para incluí-las na árvore.

Muitos algoritmos distribuídos para construção de árvores geradoras são baseados nos algoritmos centralizados de Kruskal-Joseph e Prim. O primeiro deles foi definido por Gallager et al. (1983). O processo é semelhante ao utilizado por Kruskal. Inicialmente cada nodo é uma árvore. A cada nível, um nodo é eleito líder e uma aresta de peso mínimo que o interliga a um nodo em outra árvore é adicionada. O processo é repetido até formar uma única componente conexa. O algoritmo proposto por Dalal (1987) utiliza o modelo de Prim para conectar segmentos da árvore escolhendo a aresta de menor peso que conecta dois segmentos.

Avresky (1999) apresenta três algoritmos para construção e manutenção de árvores em sistemas baseados em hipercubos sujeitos a falhas. Na fase inicial a árvore é construída usando busca em largura. Em caso de falha, o processo falho é desconectado da árvore e uma nova árvore é reconstruída pela conexão dos filhos do processo falho. Os algoritmos toleram falhas simples de processo e enlace, mas podem bloquear em certas combinações com falhas múltiplas.

Leitão, Pereira e Rodrigues (2007) propõem um protocolo híbrido combinando árvore com uma estratégia de *gossip*. Nesta solução, chamada HyParView, uma árvore de *broadcast* é criada sobre uma rede de sobreposição baseada em *gossip*. Outras soluções utilizando inundação são empregas para criar protocolos probabilísticos. Eugster et al. propõem um mecanismo no qual cada processo tem conhecimento de uma quantidade fixa de vizinhos escolhidas aleatoriamente. Pereira et al. utilizam um protocolo hierárquico que se adapta de acordo com a capacidade dos nodos em disseminar mensagens.

O trabalho de Flocchini et al. (2012) propõe uma solução tolerante a falhas para árvores geradoras que reconstrói a árvore após a uma falha simples utilizando árvores alternativas pré-computadas. Isto é feito pelo cálculo e armazenamento distribuído das $n$ árvores que podem ser geradas com um único processo falho. Em caso de recuperação do processo, a árvore anterior é restaurada para incluí-lo novamente na topologia.

## 3. Definições e Modelo do Sistema

Um sistema distribuído consiste de um conjunto finito $\Pi$ de $n > 1$ processos independentes $\{p_0, ..., p_{n-1}\}$ que colaboram para a realização de alguma tarefa. Considera-se que cada processo é executado em um nodo distinto. Sendo assim, os termos *nodo* e *processo* são usados indistintamente. O sistema considerado é síncrono. Portanto, os atributos temporais relativos à velocidade de execução dos processos e ao atraso de mensagens nos canais de comunicação possuem limites conhecidos.

Seja $G = (V, E)$ o grafo conexo e não-direcionado que representa $\Pi$, no qual $n = |V|$ são os vértices que representam os processos e $m = |E|$ são as arestas, isto é, os enlaces de comunicação. Uma aresta $(i, j)$ indica que o processo $i$ pode se comunicar diretamente com o processo $j$ e vice-versa. Uma árvore geradora (*spanning tree*) de $G$ é um sub-grafo $T = (V, E')$ conexo e acíclico no qual $E' \subseteq E$, isto é, $T$ contém todos os vértices de $G$ e $|E'| = |V| - 1$. Se as arestas possuem pesos, uma *árvore geradora mínima* é aquela cujo a soma dos pesos das arestas é mínima. Se cada aresta possui um peso diferente, existe uma única árvore mínima. Se todas as arestas possuem o mesmo peso, todas as árvores do grafo são mínimas [Gallager et al. 1983]. Neste trabalho todos os enlaces possuem o mesmo peso.

Os processos comunicam-se enviando e recebendo mensagens. A topologia é totalmente conectada e cada par de processos está conectado por um enlace ponto-a-ponto bidirecional. Entretanto, os processos são organizados em um hipercubo virtual. Um hipercubo de $d$ dimensões possui $n = 2^d$ processos. Cada processo $i$ tem identificador entre $0$ e $n - 1$, composto por um endereço binário de $d$ *bits* $i_{d-1}, i_{d-2}, ..., i_0$. Dois processos estão conectados se os seus endereços diferem em apenas um *bit*. O envio e recebimento de mensagens são operações atômicas, mas se requer suporte extra para *multicast* e *broadcast*. Os enlaces são confiáveis e nunca falham. Portanto, nenhuma mensagem é perdida, corrompida ou duplicada durante a transmissão. Não existe particionamento da rede.

Os processos podem falhar por colapso (*crash*) e as falhas são permanentes. Um

processo é dito *correto* ou *sem-falha* se não falha durante toda a execução do sistema. Caso contrário, o processo é considerado *falho*. Durante o colapso, o processo não executa qualquer ação e não responde aos estímulos externos, isto é, não executa processamento, nem envia ou recebe mensagens. Os processos falhos são detectados por um serviço de detecção de falhas perfeito, isto é, nenhum processo falho $q$ é suspeito a menos que esteja realmente falho e, se $q$ está falho, todo processo correto $p$ será notificado pelo módulo detector sobre a falha de $p$ em um tempo finito [Freiling et al. 2011].

## 4. VCube: Uma Topologia Virtual Escalável

O uso de uma topologia virtual para conectar processos de um sistema distribuído facilita a construção das aplicações, pois abstrai a estrutura física e favorece a reconfiguração do sistema quando necessário. A topologia empregada neste trabalho, denominada VCube, organiza os processos em um hipercubo virtual quando não existem falhas. Porém, se uma falha é detectada, os enlaces virtuais são reconfigurados para se adaptar ao novo estado do sistema. Mesmo na presença de falhas, VCube mantém as seguintes propriedades logarítmicas: em um hipercubo de $d$ dimensões com $n = 2^d$ vértices, cada vértice está conectado a até $\log_2 n$ vizinhos e a distância máxima entre dois vértices é $\log_2 n$.

Para construir o VCube é utilizada a organização hierárquica proposta por Duarte Jr. e Nanya (1998). Os processos são organizados em clusters progressivamente maiores, conforme ilustrado na Figura 1. Cada cluster $s = 1, .., \log_2 n$ possui $2^{s-1}$ processos. Nos clusters de nível $s = 1$ cada cluster possui um elemento. No segundo nível, dois clusters de tamanho um são agrupados, formando um cluster $s = 2$ de tamanho dois. No terceiro nível, dois clusters de tamanho dois se unem para formar um cluster $s = 3$ de tamanho quatro, e assim sucessivamente.

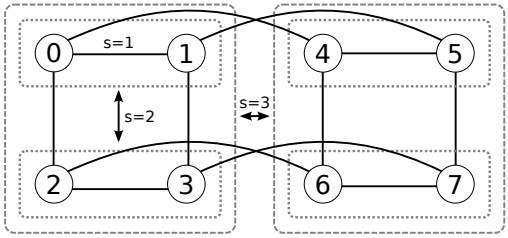



**Figura 1. Organização em clusters no VCube de três dimensões.**

Os processos integrantes de cada cluster $s$ em relação a um processo $i$ são listados em ordem pela função $c_{i,s} = \{i \oplus 2^{s-1}, c_{i\oplus 2^{s-1},1}, ..., c_{i\oplus 2^{s-1},s-1}\}$, na qual $\oplus$ representa a operação binária de *ou* exclusivo (*xor*).

Utilizando a topologia criada pelo VCube, foram definidas algumas funções. Sejam $i$ e $j$ dois processos do sistema. O conjunto $correct_i$ contém a relação dos processos considerados corretos por $i$. A função $cluster_i(j) = s$ determina a qual cluster $s$ do processo $i$ o processo $j$ pertence. Como exemplo, no hipercubo de três dimensões da Figura 1, $cluster_0(1) = 1$, $cluster_0(2) = cluster_0(3) = 2$, $cluster_0(5) = 3$. Note que para todo par $i, j$, $cluster_i(j) = cluster_j(i)$.

A segunda função é $FF\_neighbor_i(s) = j$. Esta função calcula o primeiro processo $j$ correto do cluster $c_{i,s}$. Se todos os processos no cluster estão falhos, a função retorna $\perp$. No hipercubo da Figura 1, por exemplo, em um cenário sem fa-

lhas $FF\_neighbor_4(1) = 5$, $FF\_neighbor_4(2) = 6$, $FF\_neighbor_4(3) = 0$. Por outro lado, se $p_6$ está falho e $p_4$ já foi informado sobre esta falha pelo módulo detector (isto é, $p_6$ não pertence a $correct_4$), $FF\_neighbor_4(2) = 7$. Se $p_6$ e $p_7$ estão falhos, $FF\_neighbor_4(2) = \perp$. É importante ressaltar que esta função é dependente do conhecimento atual que um processo tem a respeito das falhas no sistema. Devido à latência para a detecção, em um mesmo espaço de tempo, é possível que dois processos possuam visões diferentes sobre quais processos estão corretos ou falhos.

Por fim, a função $neighborhood_i(h) = \{j \mid j = FF\_neighbor_i(s), j \neq \perp, 1 \leq s \leq h\}$ é definida. Esta função gera um conjunto que contém todos os processos sem-falha virtualmente conectados ao processo $i$ de acordo com $FF\_neighbor_i(s)$, para $s = 1, .., h$. O parâmetro $h$ pode variar de 1 a $\log_2 n$. Se $\log_2 n$ é utilizado, o conjunto resultante contém todos os vizinhos corretos do processo $i$ no VCube. Para qualquer outro valor de $h < \log_2 n$, a função retorna apenas um subconjunto dos vizinhos contidos nos clusters $s = 1, .., h$. Assim como $FF\_neighbor_i$ esta função depende do conhecimento local que o processo $i$ tem sobre o estado dos outros processos. Como exemplo, para o VCube da Figura 1 e em um cenário sem falhas, $neighborhood_0(1) = \{1\}$, $neighborhood_0(2) = \{1, 2\}$ e $neighborhood_0(3) = \{1, 2, 4\}$. Se o processo $p_4$ é detectado como falho pelo processo $p_0$ ($4 \notin correct_0$) então $neighborhood_0(3) = \{1, 2, 5\}$. Se $p_1$ está falho, $neighborhood_0(1) = \emptyset$.

Assim, a topologia do sistema no VCube é formada pela conexão de cada processo $i$ com todos os seus vizinhos determinados por $neighborhood_i(\log_2 n)$. De forma análoga, os vizinhos de um processo $i$ restritos ao cluster $s$ ao qual $i$ pertence em relação a outro processo $j$ são determinados por $neighborhood_i(cluster_i(j) - 1)$.

## 5. O Algoritmo Autonômico para Árvores Geradoras

Esta seção apresenta o algoritmo proposto para construir de forma autonômica uma árvore geradora mínima em um sistema distribuído com base na topologia VCube. O Algoritmo 5.1 propõe um mecanismo de disseminação de informação para um sistema distribuído que propaga as mensagens do sistema a partir de um processo qualquer, denominado fonte. O algoritmo é considerado autonômico porque reconstrói a árvore dinamicamente à medida que processos falhos são detectados. Uma das vantagens da solução é que, ao invés de iniciar a reconstrução a partir do processo fonte, a regeneração da árvore é realizada apenas localmente, de acordo com o ramo afetado.

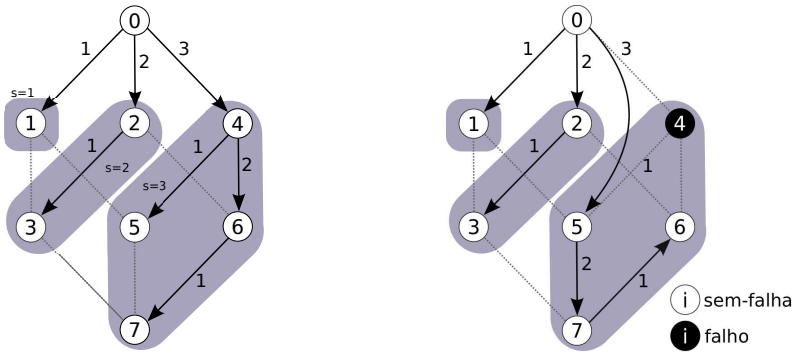
**Algoritmo 5.1** Algoritmo Distribuído de Árvore Geradora Mínima no processo $i$

```
1: correct_i ← {0, .., n − 1} //lista dos processos corretos
2: procedure STARTTREE( )
3:     //envia a todos os vizinhos
4:     for all k ∈ neighborhood_i(log_2 n) do
5:         SEND(⟨TREE⟩) para p_k
6: procedure RECEIVE(⟨TREE⟩) from p_j
7:     if j ∈ correct_i then
8:         //retransmite aos vizinhos dos clusters internos
9:         for k ∈ neighborhood_i(cluster_i(s) − 1) do
10:            SEND(⟨TREE⟩) para p_k
11: procedure CRASH(processo j) //j é detectado como falho
12:     correct_i ← correct_i ∖ {j}
13:     if k = FF_neighbor_i(cluster_i(j)), k ≠ ⊥ then
14:         SEND(⟨TREE⟩) to p_k
```

Considere inicialmente uma execução sem falhas. No primeiro passo, a propagação é iniciada no procedimento STARTTREE. Uma mensagem TREE é enviada

para os $\log_2 n$ vizinhos sem falha, um em cada cluster $s = 1, .., \log_2 n$ (linhas 4-5). Ao receber uma mensagem de um processo $j$, o procedimento RECEIVE é executado e o processo $i$ encaminha a mensagem para os *clusters* internos ao seu próprio cluster, isto é, aos clusters $s' = 1, .., cluster_i(j) - 1$ (linha 9).

Como exemplo, considere o VCube sem processos falhos da Figura 2(a). O processo $p_0$ é a raiz e envia TREE para os vizinhos $FF\_neighbor_0(1) = 1$, $FF\_neighbor_0(2) = 2$ e $FF\_neighbor_0(3) = 4$. O processo $p_1$ recebe a mensagem, mas não retransmite, visto $cluster_1(0) = 1$ e $neighborhood_1(0) = \perp$. O processo $p_2$ recebe a mensagem e retransmite para seu vizinho $p_3$ no cluster $s = 1$. Quando $p_3$ recebe a mensagem ele calcula $cluster_3(2) = 1$ e para a retransmissão. No caso de $p_4$ a mensagem e recebida e retransmitida para os vizinhos $5 \in c_{4,1}$ e $6 \in c_{4,2}$. Finalmente, sendo $FF\_neighbor_6(1) = 7$, o processo $p_6$ envia a mensagem para o processo $p_7$.




(a) sem-falhas (b) processo $p_4$ falho

**Figura 2. Árvore geradora no VCube de 3 dimensões.**

Os casos com falhas podem ser divididos em dois cenários. Primeiramente, considere que um processo $j \in c_{i,s}$ está falho e o processo $i$ já foi informado sobre esta falha pelo detector, isto é, $j \notin correct_i$ (linha 11). Neste caso, o processo $i$ envia a mensagem para o vizinho sem falha $k = FF\_neighbor_i(s)$ e a mensagem é propagada corretamente por $k$ a todos os clusters internos do cluster $s$. Em um segundo cenário, este mesmo processo $j$ está falho, mas o processo $i$ ainda não foi informado pelo detector, isto é, $j \in correct_i$. Neste caso, se $FF\_neighbor_i(s) = j$, a mensagem é enviada ao processo falho $j$ e é descartada. A propagação termina prematuramente e a sub-árvore interna ao cluster $s$ é desconectada. Entretanto, assim que o módulo detector informa $i$ sobre a falha, um novo $FF\_neighbor_i(s) = k$ é eleito e a mensagem é retransmitida para $k$, reconstruindo-se assim a sub-árvore do cluster $s$ (linha 13).

A Figura 2(b) ilustra um cenário com falhas. Seja o processo $p_0$ novamente a raiz e $p_4$ falho. Estando $p_0$ ciente da falha do processo $p_4$, ao invés de enviar TREE para $p_4$, $p_0$ transmite a mensagem a $p_5$ que é o primeiro processo sem falha de $c_{0,3}$. O processo $p_5$ por sua vez, repassa a mensagem para $p_7 \in c_{5,2}$. Por fim, $p_7$ retransmite a mensagem para $p_6 \in c_{7,1}$, completando a árvore. Se quando $p_0$ inicia o *broadcast* e $p_4 \in correct_i$, $p_0$ envia a mensagem para $p_4$ e, quando o detector informá-lo sobre a falha de $p_4$, a mensagem será retransmitida para $p_5$. Deste ponto em diante a propagação é análoga ao caso anterior.

O Teorema 1 formaliza a propagação em árvore proposta pelo Algoritmo 5.1.

**Teorema 1.** *Seja $m$ uma mensagem propagada por um processo fonte $src$ correto. Todo processo correto no sistema $\Pi$ recebe $m$.*
**Prova.** A prova deste lema é por indução. Considere como base da indução um sistema com $n = 2$ processos: $p_0$ é o processo $src$ que inicia o envio da mensagem $m$ e $p_1 \in c_{0,1}$. Se $p_1$ é correto, $FF\_neighbor_0(1) = 1$ e $p_0$ envia $m$ para $p_1$ (linha 4). Portando, $p_1$ recebe $m$ e o lema é válido.

Como hipótese da indução, considere que o lema é válido para um sistema com $n = 2^k$ processos.

No passo da indução é demonstrado que o lema é válido para um sistema com $n = 2^{k+1}$ processos. Pela organização hierárquica do VCube, este sistema é constituído de dois subsistemas com $n = 2^k$ processos, como ilustrado pela Figura 3. A figura mostra que $src$ e $j$ são as raízes destes subsistemas. O processo $src$ executa o algoritmo e envia $m$ para cada processo retornado por $FF\_neighbor_{src}(s)$, $s = 1, .., k$ (linha 4). Sendo $j = FF\_neighbor_{src}(k)$ um processo correto, $j$ corretamente recebe $m$. Se $j$ é detectado como falho, uma cópia da mensagem é retransmitida para o próximo processo correto no mesmo cluster de $j$ (linha 13). Assim, a mensagem $m$ é transmitida nos dois subsistemas e, pela hipótese, todo processo correto recebe $m$ em cada subsistema. Como todo processo em $\Pi$ pertence a um destes sistemas, todo processo correto em $\Pi$ recebe $m$.

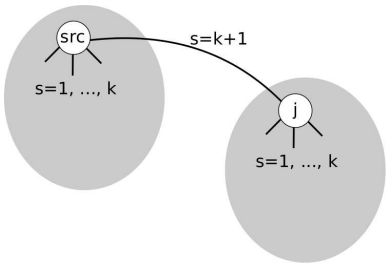



**Figura 3. Propagação das mensagens de *broadcast*.**

## 6. Broadcasts com VCube

*Broadcast* é um componente básico para implementar muitos algoritmos e serviços distribuídos como notificação, entrega de conteúdo, *publish/subscribe*, replicação e comunicação em grupo. Um processo utiliza *broadcast* para enviar uma mensagem a todos os outros processos do sistema. No entanto, se um processo falha durante o procedimento de difusão, alguns processos podem receber a mensagem enquanto outros não. O *broadcast* de melhor-esforço garante que, se o emissor é correto, todos os processos corretos recebem a mensagem difundida por ele. Por outro lado, se o emissor pode falhar, estratégias de *broadcast* confiável precisam ser implementadas [Hadzilacos e Toueg 1993].

Algoritmos de *broadcast* tolerante a falhas são normalmente implementados utilizando enlaces ponto-a-ponto confiáveis e primitivas SEND e RECEIVE. Os processos invocam BROADCAST($m$) e DELIVER($m$) para difundir e receber mensagens para/de outros processos da aplicação, respectivamente. Para incluir tolerância a falhas, um detector de falhas perfeito pode ser utilizado para notificar o algoritmo de *broadcast*, que deve reagir apropriadamente quando uma falha é detectada.

Neste artigo, dois algoritmos de *broadcast* tolerante a falhas são descritos. O primeiro implementa *broadcast* de melhor-esforço e o segundo propõe uma solução confiável. As duas soluções utilizam o modelo de árvores geradoras proposto na Seção 5.

### 6.1. Broadcast de Melhor-esforço

*Broadcast* de melhor-esforço garante que todos os processos corretos entregarão o mesmo conjunto de mensagens se o emissor (fonte) é correto. Três propriedades caracterizam este modelo: entrega confiável (*validity*), não-duplicação e não-criação de mensagens [Guerraoui e Rodrigues 2006]. A entrega confiável garante que, se um processo $p_i$ envia uma mensagem $m$ para um processo $p_j$ e nenhum deles falha, $p_j$ recebe $m$ em um tempo finito. A não-duplicação garante que nenhuma mensagem é entregue mais de uma vez e a não-criação garante que nenhuma mensagem é entregue a menos que tenha sido previamente enviada.

O Algoritmo 6.1 apresenta uma solução para *broadcast* de melhor-esforço que utiliza o mecanismo de árvore proposto neste trabalho. Dois tipos de mensagens são utilizados: $\langle TREE, m\rangle$ para identificar as mensagens de aplicação e $\langle ACK, m\rangle$ para confirmar o recebimento das mensagens de aplicação. Os processos são informados sobre o estado dos demais processos pelo algoritmo que mantém o VCube. O algoritmo executa corretamente mesmo que $n-1$ falhas ocorram. Um versão preliminar deste algoritmo foi publicada em Rodrigues (2013) aplicado a uma solução tolerante a falhas de exclusão mútua distribuída.

As variáveis locais mantidas pelos processos são:

- $correct_i$: conjunto dos processos considerados corretos pelo processo $i$;
- $last_i[n]$: a última mensagem recebida de cada processo fonte;
- $ack\_set_i$: o conjunto com todos os ACKs pendentes no processo $i$. Para cada mensagem $\langle TREE, m\rangle$ recebida pelo processo $i$ de um processo $j$ e retransmitida para o processo $k$, um elemento $\langle j, k, m\rangle$ é adicionado a este conjunto.

O símbolo $\perp$ representa um elemento nulo. O asterisco é usado como curinga para selecionar ACKs no conjunto $ack\_set$. Um elemento $\langle j, *, m\rangle$, por exemplo, representa todos os ACKs pendentes para uma mensagem $m$ recebida pelo processo $j$ e retransmitida para qualquer outro processo.

Um processo $i$ que deseja difundir uma mensagem $m$ por *broadcast* invoca o método BROADCAST. A linha 5 garante que um novo *broadcast* só é iniciado após o término do anterior, isto é, quando não há mais ACKs pendentes para a mensagem $last_i[i]$. Nas linhas 9-11 a nova mensagem é enviada a todos os vizinhos considerados corretos. Para cada mensagem enviada, um ACK é incluído na lista de ACKs pendentes.

Quando um processo recebe uma mensagem TREE de um processo $j$ (linha 16), primeiramente ele verifica se tanto o processo fonte da mensagem quanto o processo $j$ são considerados corretos. Se um deles esta falho, o recebimento é abortado, pois se $j$ está falho, o processo que transmitiu $m$ para $j$ fará uma nova transmissão quando detectar a falha e $i$ irá receber a mensagem através da nova árvore que será reconstruída. Além disso, se a fonte está falha, não é mais necessário continuar a retransmissão. Se a fonte e $j$ estão corretos, o processo $i$ verifica se a mensagem é nova comparando os *timestamps* da última mensagem armazenada em $last_i[j]$ e da mensagem recebida $m$ (linha 21). Se $m$ é nova, $last_i[j]$ é atualizado e a mensagem é entregue à aplicação. Em seguida, $m$ é retransmitida para os vizinhos em cada cluster interno ao cluster de $i$. Se não existe vizinho correto ou se $i$ é uma folha na árvore ($cluster_i(j) = 1$), nenhum ACK pendente é adicionado ao conjunto $ack\_set_i$ e CHECKACKS envia um ACK para $j$.

Se uma mensagem $\langle ACK, m\rangle$ é recebida, o conjunto $ack\_set_i$ é atualizado e,

**Algoritmo 6.1** Broadcast de melhor-esforço hierárquico no processo $i$

```
1: last_i[n] ← {⊥, .., ⊥}
2: ack_set_i = ∅
3: correct_i = {0, .., n − 1}

4: procedure BROADCAST(message m)
5:     wait until ack_set_i ∩ {⟨⊥, ∗, last_i[i]⟩} = ∅
6:     last_i[i] = m
7:     DELIVER(m)
8:     //envia a todos os vizinhos
9:     for all j ∈ neighborhood_i(log_2 n) do
10:        ack_set_i ← ack_set_i ∪ {⟨⊥, j, m⟩}
11:        SEND(⟨TREE, m⟩) to p_j

12: procedure CHECKACKS(processo j, mensagem m)
13:    if ack_set_i ∩ {⟨j, ∗, m⟩} = ∅ then
14:        if {source(m), j} ⊆ correct_i then
15:            SEND(⟨ACK, m⟩) to p_j

16: procedure RECEIVE(⟨TREE, m⟩) from p_j
17:    if {source(m), j} ⊈ correct_i then
18:        return
19:    //verifica se m é nova
20:    if last_i[source(m)] = ⊥ or
21:       ts(m) = ts(last_i[source(m)]) + 1 then
22:        last_i[source(m)] ← m
23:        DELIVER(m)
24:    //retransmite aos vizinhos dos clustes internos
25:    for all k ∈ neighborhood_i(cluster_i(j) − 1) do
26:        if ⟨j, k, m⟩ ∉ ack_set_i then
27:            ack_set_i ← ack_set_i ∪ {⟨j, k, m⟩}
28:            SEND(⟨TREE, m⟩) to p_k
29:    CHECKACKS(j, m)

30: procedure RECEIVE(⟨ACK, m⟩) from p_j
31:    k ← x : ⟨x, j, m⟩ ∈ ack_set_i
32:    ack_set_i ← ack_set_i ∖ {⟨k, j, m⟩}
33:    if k ≠ ⊥ then
34:        CHECKACKS(k, m)

35: procedure CRASH(processo j) //j é detectado como falho
36:    correct_i ← correct_i ∖ {j}
37:    k ← FF_neighbor_i(cluster_i(j))
38:    for all p = x, q = y, m = z : ⟨x, y, z⟩ ∈ ack_set_i do
39:        if {source(m), p} ⊈ correct_i then
40:            //remove ACKs pendentes para ⟨j, ∗, ∗⟩ e
41:            //⟨∗, ∗, m⟩ : source(m) = j
42:            ack_set_i ← ack_set_i ∖ {⟨p, q, m⟩}
43:        else if q = j then //retransmite para novo vizinho k
44:            if k ≠ ⊥ and ⟨p, k, m⟩ ∉ ack_set_i then
45:                ack_set_i ← ack_set_i ∪ {⟨p, k, m⟩}
46:                SEND(⟨TREE, m⟩) to p_k
47:            ack_set_i ← ack_set_i ∖ {⟨p, j, m⟩}
48:            CHECKACKS(p, m)
```

se não existem mais ACKs pendentes para a mensagem $m$, CHECKACKS envia um $\langle ACK, m\rangle$ para o processo $k$ do qual $i$ recebeu a mensagem TREE anteriormente. No entanto, se $k = \perp$, o ACK alcançou o processo fonte e não precisa mais ser propagado.

A detecção de um processo falho $j$ é tratada no procedimento CRASH. Três ações são realizadas: (1) atualização da lista de processos corretos; (2) remoção dos ACKs pendentes que contém o processo $j$ como destino ou aqueles em que a mensagem $m$ foi originada em $j$; (3) reenvio das mensagens anteriormente transmitidas ao $j$ para o novo vizinho $k$ no mesmo cluster de $j$, se existir um. Esta retransmissão desencadeia uma propagação na nova estrutura da árvore.

No Teorema 2 é provada a correção da solução de *broadcast* de melhor-esforço implementada pelo Algoritmo 6.1.

**Teorema 2.** *O Algoritmo 6.1 é uma solução para broadcast de melhor-esforço: se o emissor é correto, todo processo correto receberá o mesmo conjunto de mensagens.*

**Prova.** As propriedades de não-duplicação e não-criação são derivadas das propriedades dos enlaces, os quais são considerados confiáveis. Além disso, toda mensagem possui um *timestamp* único e, mesmo que seja retransmitida após uma falha, o receptor fará a entrega uma única vez (linha 21).

A entrega confiável é garantida pelo processo emissor (fonte). Quando um emissor envia uma mensagem $m$, ele aguarda pelas mensagens de confirmação (ACKs) de todos os vizinhos corretos. O Teorema 1 provou que se o emissor é correto, todo processo correto recebe $m$, mesmo que um processo intermediário $j$ falhe durante a retransmissão.

Neste caso, a mensagem é retransmitida para o próximo vizinho sem falha $k$ no mesmo cluster do processo $j$ (linha 37).

**Teorema 3.** *O total de mensagens enviadas para cada mensagem de aplicação em uma execução sem falhas do Algoritmo 6.1 é* $2 * (n - 1)$. *Se um processo* $j$ *que recebeu a mensagem de um processo* $j$ *falha antes de confirmar o recebimento, o total de mensagens extra depende da quantidade de processos no cluster de* $j$.

**Prova.** Em uma execução sem falhas, para cada mensagem TREE enviada, uma mensagem ACK é retornada. Seja $n - 1$ o número de arestas na árvore com $n$ processos, o total de mensagens é o dobro do total de arestas.

Se um processo $j$ é detectado como falho por um processo $i$ depois que a mensagem TREE foi enviada para $j$, uma nova mensagem será enviada para o próximo vizinho $k = FF\_neighbor_i(cluster_i(j))$. Seja $s = cluster_i(j)$. No melhor caso, $k = \perp$ e nenhuma mensagem extra é enviada ($j \in c_{i,1}$ ou não existem mais processos corretos no cluster $s$). No entanto, se $k \neq \perp$, a quantidade de mensagens extras depende do número de processos detectados corretos no cluster $s$. Seja $n' = |c_{i,s}|$ o total de processos no cluster $c_{i,s}$. No pior caso, todos os processos do cluster estão corretos, exceto $j$, e $j$ enviou a mensagem a todos os vizinhos antes de falhar. Assim, o total de mensagens extras será $1 + 2 * (n' - 2)$, uma TREE extra para $k$ e $(n' - 2)$ TREEs + $(n' - 2)$ ACKs na subárvore. De forma geral, se existem $f$ processos falhos em $c_{i,s}$ incluindo $j$, a quantidade de mensagens extras retransmitidas é $1 + 2 * (n' - 1 - f)$.

### 6.2. Broadcast Confiável

Um algoritmo de *broadcast* confiável garante que o mesmo conjunto de mensagens é entregue a todos os processos corretos, mesmo se o emissor (fonte) falhar durante o procedimento de difusão. Para tanto, o *broadcast* confiável herda as propriedades de entrega confiável, não-criação e não-duplicação do melhor-esforço e acrescenta a propriedade de acordo (*agreement*). Assim, a solução para *broadcast* confiável proposta neste trabalho é uma modificação do *broadcast* de melhor-esforço descrito na Seção 6.1 que inclui o tratamento para a falha do processo fonte. O Algoritmo 6.2 apresenta as modificações realizadas. As variáveis $last_i$, $ack\_set$ e $correct_i$ são as mesmas, bem como os métodos RECEIVE($ACK, m$) e CHECKACKS.

Um processo $p_i$ que deseja efetuar o *broadcast* invoca o procedimento BROADCAST($m$). Se a mensagem $m$ tem origem no mesmo processo, $source(m) = i$ e o tratamento é o mesmo realizado pelo algoritmo de melhor-esforço. A mensagem é então propagada para todos os vizinhos de $p_i$. Quando um processo $p_i$ recebe a mensagem $\langle TREE, m\rangle$ de um processo $p_j$ ele primeiramente verifica se $j \notin correct_i$. Se $p_j$ está falho, a mensagem é descartada. Note que esta verificação difere do algoritmo de melhor-esforço pois não descarta mensagens recebidas de um $source(m)$ falho. A segunda modificação está nas linhas 15-17. Se $p_i$ recebe uma mensagem nova de um processo fonte considerado falho, ele inicia um novo *broadcast* com a mensagem recebida para garantir que os demais processos recebam $m$ corretamente. Neste caso, quando a linha 2 é executada, $source(m)$ não será igual a $i$ e a mensagem é retransmitida aos demais processos através da árvore geradora de $p_j$.

A recuperação em caso de falhas também é muito semelhante à solução de melhor-esforço, exceto pelo *broadcast* da última mensagem recebida do processo detectado como

falho (linha 33). Embora a prova de correção da solução tenha sido omitida, esta retransmissão, em conjunto com a retransmissão da linha 16, garante que todos os demais processos corretos receberão a última mensagem transmitida pelo processo fonte $p_j$ falho, mesmo que um único processo tenha recebido a mensagem antes da falha de $p_j$.

**Algoritmo 6.2** Broadcast confiável hierárquico no processo $i$

```
1: procedure BROADCAST(message m)
2:     if source(m) = i then
3:         wait until ack_set_i ∩ {⟨⊥, ∗, last_i[i]⟩} = ∅
4:         last_i[i] = m
5:         DELIVER(m)
6:     ...
7: procedure RECEIVE(⟨TREE, m⟩) from p_j
8:     if j ∉ correct_i then
9:         return
10:    //verifica se m é nova
11:    if last_i[source(m)] = ⊥ or
12:       ts(m) = ts(last_i[source(m)]) + 1 then
13:        last_i[source(m)] ← m
14:        DELIVER(m)
15:        if source(m) ∉ correct_i then
16:            BROADCAST(m)
17:            return
18:    ...
19: procedure CRASH(processo j) //j é detectado como falho
20:    correct_i ← correct_i ∖ {j}
21:    k ← FF_neighbor_i(cluster_i(j))
22:    for all p = x, q = y, m = z : ⟨x, y, z⟩ ∈ ack_set_i do
23:        if p ∉ correct_i then
24:            //remove ACKs pendentes para ⟨j, ∗, ∗⟩
25:            ack_set_i ← ack_set_i ∖ {⟨p, q, m⟩}
26:        else if q = j then //retransmite para novo vizinho k
27:            if k ≠ ⊥ and ⟨p, k, m⟩ ∉ ack_set_i then
28:                ack_set_i ← ack_set_i ∪ {⟨p, k, m⟩}
29:                SEND(⟨TREE, m⟩) to p_k
30:            ack_set_i ← ack_set_i ∖ {⟨p, j, m⟩}
31:            CHECKACKS(p, m)
32:    if last_i[j] ≠ ⊥ then
33:        BROADCAST(last_i[j])
```

## 7. Avaliação Experimental

Os algoritmos de *broadcast* propostos foram implementados utilizando o Neko [Urbán et al. 2002], um *framework* Java para simulação de algoritmos distribuídos, e comparados com duas outras soluções. Para facilitar a discussão dos resultados, o *broadcast* de melhor-esforço foi chamado de ATREE-B e a abordagem confiável de ATREE-R. A primeira solução comparada utiliza uma abordagem todos-para-todos (nomeadas ALL-B e ALL-RB) e a segunda implementa uma estratégia baseada em árvore, porém, não autonômica (nomeadas NATREE-B e NATREE-R). ATREE, ALL e NATREE são utilizadas para referenciá-las de forma genérica. Para implementar a estratégia ALL, a função $neighborhood$ foi modificada para incluir todos os processos sem-falha, fazendo com que o emissor envie a mensagem a todos os demais processos diretamente utilizando enlaces ponto-a-ponto. A estratégia NATREE constrói uma árvore utilizando inundação. A árvore é criada utilizando a própria mensagem TREE de *broadcast*. Cada processo que recebe a mensagem pela primeira vez se junta à árvore e retransmite a mensagem aos demais processos do sistema, exceto àquele do qual ele recebeu a mensagem. Se um processo já está na árvore e recebe uma outra cópia da mensagem ele envia uma mensagem de NACK e não retransmite mais a mensagem. Uma vez criada a árvore, as demais mensagens são enviadas somente pelas arestas da árvore. Em caso de falha, uma nova árvore é criada a partir do processo fonte utilizando o mesmo procedimento de inundação.

O mecanismo de detecção de falhas utilizado está descrito em Ruoso (2013). Trata-se de um algoritmo de diagnóstico distribuído e adaptativo com baixa latência e reduzido número de mensagens quando comparado a outras soluções existentes. A latência de diagnóstico varia de $\log_2 n$ a $\log_2^2 n$ rodadas.

### 7.1. Parâmetros de Simulação

Uma vez que não existem mecanismos auxiliares de *broadcast* e *multicast*, quando um processo precisa enviar uma mensagem para mais de um destinatário ele deve utilizar primitivas SEND sequencialmente. Assim, para cada mensagem, $t_s$ unidades de tempo são utilizadas para enviar a mensagem e $t_r$ unidades para recebê-la, além do atraso de transmissão $t_t$. Estes intervalos são computados para cada cópia da mensagem enviada.

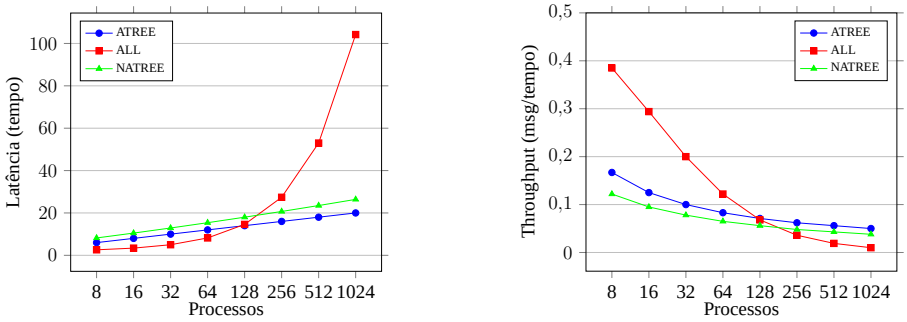
Para avaliar o desempenho de soluções de *broadcast*, duas métricas são frequentemente utilizadas [Boichat e Guerraoui 2005]: (1) *throughput*, dado pelo total de mensagens de *broadcast* completos durante um intervalo de tempo; (2) a latência para entregar a mensagem de *broadcast* a todos os processos corretos.

Os algoritmos propostos foram avaliados em diferentes cenários variando o número de processos e a quantidade de processos falhos. Os parâmetros de comunicação foram definidos em $t_s = t_r = 0,1$ e $t_t = 0,8$. O intervalo de testes do detector foi definido em $5,0$ unidades de tempo. Um processo é considerado falho se não responder ao teste após $4 * (t_s + t_r + t_t)$ unidades de tempo.

### 7.2. Resultado dos Experimentos

Os experimentos foram realizados em duas etapas. Inicialmente, foram utilizados cenários sem falhas com sistemas de 8 a 1024 processos. Além disso, a latência e a quantidade de mensagens enviadas por cada aplicação foram analisadas pontualmente em um sistema com 512 processos. Em seguida, cenários com falhas foram gerados aleatoriamente para um sistema com 512 processos contendo de 1 a 8% de processos falhos.

**Experimentos em cenários sem-falhas.** Nos cenários sem-falhas, como não há retransmissões, os algoritmos de *broadcast* de melhor-esforço e confiável propostos possuem o mesmo comportamento. A Figura. 4 mostra a latência e o *throughput* considerando que uma única mensagem é enviada pelo processo $p_0$. O caminho mais longo em um VCube com $n$ processos é $\log_2 n$. Portanto, quando $n$ é pequeno, o tempo para enviar a mensagem pelo caminho mais longo é maior que o tempo para enviar as $n - 1$ mensagens sequencialmente pela estratégia ALL. Considerando que o tempo de envio de cada mensagem é $t_s = 0,1$, o intervalo entre o envio da mensagem TREE e a recepção do ACK correspondente pelo caminho mais longo da árvore é $2 \log_2 n(t_s + t_r + t_t)$. Já na estratégia ALL, para enviar $n - 1$ mensagens são utilizadas $(n - 2)t_s + t_t + tr$ unidades de tempo.



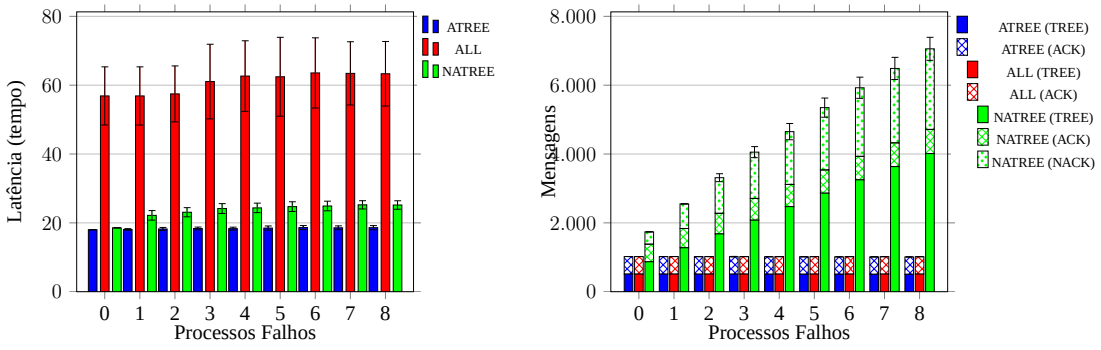

(a) Latência (b) Throughput

**Figura 4. Broadcast de melhor-esforço em uma execução sem-falhas.**

Assim, embora ALL seja mais eficiente para sistemas pequenos, seu desempenho cai rapidamente em sistemas maiores. A solução com árvore não-autonômica NATREE apresenta comportamento semelhante a ATREE, exceto pela latência extra para aguardar as mensagens de ACK/NACK durante a configuração da árvore. Em cenários sem-falha, o desempenho das duas soluções passa a ser o mesmo depois após a árvore estar completa.

O *throughput* foi calculado como $1/latency$, visto que apenas um processo envia uma única mensagem. Assim como a latência, na solução ALL o desempenho é melhor para sistemas pequenos, até 128 processos, mas diminui rapidamente quando $n$ aumenta. Estes resultados indicam a escalabilidade da estratégia hierárquica proposta.

**Experimentos em cenários com falhas.** Nos cenários com falhas foram utilizados sistemas com 512 processos com variações na quantidade de processos falhos. Falhas foram geradas em um intervalo de 1 a $\log_2 n$. Para cada variação de falhas foram gerados 100 diferentes cenários com falhas distribuídas aleatoriamente usando uma distribuição normal. Diferente dos testes sem falha nos quais uma única mensagem é enviada, nos experimentos com falha os processos enviam 10 mensagens de *broadcast*. A média da latência está representada no gráfico da Figura 5(a). A solução ALL tem latência maior que ATREE porque precisa enviar todas as mensagens sequencialmente, como já justificado anteriormente. Na solução NATREE, o resultado é similar a ALL visto que, para cada falha, a árvore precisa ser reconstruída por inundação. ALL é mais eficiente para cenários sem-falha ou com poucas falhas, mas degrada rapidamente quando a quantidade de falhas aumenta. Em relação à quantidade de mensagens, para ATREE e ALL o resultado é muito semelhante, conforme registrado na Figura 5(b). No caso de NATREE, a reconstrução da árvore a partir da raiz aumenta consideravelmente o total de mensagens enviadas. Estes resultados demonstram a eficiência da solução proposta na criação e manutenção autonômica da árvore, neste caso utilizada pelo *broadcast*.




(a) Latência (b) Total de mensagens

**Figura 5. Broadcast confiável com $n = 512$ e diferente número de falhas.**

## 8. Conclusão

Este trabalho apresentou uma solução para criação autonômica de árvores geradoras mínimas em sistemas distribuídos sujeitos a falhas. Os processos do sistema são organizados em uma topologia lógica, denominada VCube. As árvores são geradas dinamicamente a partir de um processo fonte e reconfiguradas localmente quando processos falham. Dois algoritmos de *broadcast* foram implementados utilizando a estratégia de árvore proposta, uma solução de *broadcast* de melhor-esforço e uma outra abordagem

confiável. Resultados de simulação demonstraram a eficiência das soluções propostas, destacando a escalabilidade delas.

Como trabalhos futuros, pretende-se estender o VCube para falhas *crash* com recuperação. Outro caminho a ser seguido é considerar a modificação do modelo para tratar falhas de particionamento de rede. Além disso, espera-se implementar o algoritmo de árvore geradora como um serviço para aplicações arbitrárias sobre uma rede.

## Referências


Avresky, D. (1999). Embedding and reconfiguration of spanning trees in faulty hypercubes. *Parallel and Distributed Systems, IEEE Transactions on*, 10(3):211–222.

Boichat, R. e Guerraoui, R. (2005). Reliable and total order broadcast in the crash-recovery model. *J. Parallel Distrib. Comput.*, 65(4):397–413.

Dahan, S., Philippe, L. e Nicod, J. (2009). The distributed spanning tree structure. *Parallel and Distributed Systems, IEEE Transactions on*, 20(12):1738–1751.

Dalal, Y. (1987). A distributed algorithm for constructing minimal spanning trees. *Software Engineering, IEEE Transactions on*, SE-13(3):398–405.

Duarte Jr., E. P. e Nanya, T. (1998). A hierarchical adaptive distributed system-level diagnosis algorithm. *IEEE Trans. Comput.*, 47(1):34–45.

Elkin, M. (2006). A faster distributed protocol for constructing a minimum spanning tree. *J. Comput. Syst. Sci.*, 72(8):1282–1308.

England, D., Veeravalli, B. e Weissman, J. (2007). A robust spanning tree topology for data collection and dissemination in distributed environments. *IEEE Trans. Parallel Distr. Syst.*, 18(5):608–620.

Eugster, P. T., Guerraoui, R., Handurukande, S. B., Kouznetsov, P. e Kermarrec, A.-M. (2003). Lightweight probabilistic broadcast. *ACM Trans. Comput. Syst.*, 21(4):341–374.

Flocchini, P., Mesa Enriquez, T., Pagli, L., Prencipe, G. e Santoro, N. (2012). Distributed minimum spanning tree maintenance for transient node failures. *IEEE Trans. Comput.*, 61(3):408–414.

Freiling, F. C., Guerraoui, R. e Kuznetsov, P. (2011). The failure detector abstraction. *ACM Comput. Surv.*, 43:9:1–9:40.

Gallager, R. G., Humblet, P. A. e Spira, P. M. (1983). A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5:66–77.

Gärtner, F. C. (2003). A survey of self-stabilizing spanning-tree construction algorithms. Technical report, Swiss Federal Institute of Technology (EPFL).

Guerraoui, R. e Rodrigues, L., editores (2006). *Introduction to Reliable Distributed Programming*. Springer-Verlag, Berlin, Germany.

Hadzilacos, V. e Toueg, S. (1993). Distributed systems. chapter Fault-tolerant broadcasts and related problems, páginas 97–145. ACM Press, New York, NY, USA, 2 edition.

Kephart, J. e Chess, D. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Kruskal Jr., J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. Anais do *American Mathematical Society*, páginas 48–50.

Leitão, J., Pereira, J. e Rodrigues, L. (2007). HyParView: A membership protocol for reliable gossip-based broadcast. Anais do *DSN*, páginas 419–429.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401.

Rodrigues, L. A. (2013). Fault-tolerant broadcast algorithms for the virtual hypercube topology. Anais do *Student Forum - DSN-W'43*, páginas 1–4.

Ruoso, V. K. (2013). Uma estratégia de testes logarítmica para o algoritmo hi-adsd. Master's thesis, Universidade Federal do Paraná.

Urbán, P., Défago, X. e Schiper, A. (2002). Neko: A single environment to simulate and prototype distributed algorithms. *Journal of Inf. Science and Eng.*, 18(6):981–997.

# Reconfiguração Modular de Sistemas de Quóruns*

**Eduardo Adilio Pelinson Alchieri**[1] **, Alysson Neves Bessani**[2] **,**
**Fabíola Greve**[3] **, Joni da Silva Fraga**[4]

[1]CIC, Universidade de Brasília, Brasília - Brasil
[2]LaSIGE, Universidade de Lisboa, Lisboa - Portugal
[3]DCC, Universidade da Bahia, Bahia - Brasil
[4]DAS, Universidade Federal de Santa Catarina, Florianópolis - Brasil

***Abstract.*** *Quorum systems are useful tools for implementing consistent and available storage in the presence of failures, especially in modern distributed systems such as clouds and MANETs. This paper proposes* FREESTORE*, a set of fault-tolerant protocols that emulates a register in dynamic asynchronous systems in which processes are able to join/leave the servers set by reconfigurations. The reconfiguration protocols presented are modular and loosely coupled with read/write protocols, making it easy to adapt static fault-tolerant registers to dynamic environments. By an analytical study, we show that* FREESTORE *is more efficient (in terms of communication steps) than previous solutions.*

***Resumo.*** *Sistemas de quóruns são úteis na implementação consistente e confiável de sistemas de armazenamento de dados em presença de falhas, sobretudo em sistemas distribuídos modernos, tais como nuvens e MANETs. Este artigo propõe o* FREESTORE*, um conjunto de protocolos capazes de emular um registrador em sistemas dinâmicos assíncronos, onde processos podem entrar e sair do sistema, durante sua execução, através de reconfigurações. A reconfiguração proposta é modular e fracamente acoplada aos protocolos de leitura e escrita, facilitando a adaptação de registradores estáticos para funcionamento em ambientes dinâmicos. Uma análise detalhada mostra que essas características tornam o* FREESTORE *mais eficiente (em termos de passos de comunicação) do que outras soluções encontradas na literatura.*

## 1. Introdução

Os sistemas de quóruns [Gifford 1979] são abstrações fundamentais usadas para garantir consistência e disponibilidade de dados armazenados de forma replicada em um conjunto de servidores. Além de serem blocos básicos de construção para protocolos de sincronização (ex.: consenso), o grande atrativo destes sistemas está em seu poder de escalabilidade e balanceamento de carga, uma vez que as operações não precisam ser executadas por todos os servidores do sistema, mas apenas por um quórum dos mesmos. Atualmente, num contexto de computação em nuvem, quóruns são fundamentais para implementação das garantias de confiabilidade [Vogels 2009].

Sistemas de quóruns foram inicialmente estudados em ambientes estáticos, onde não é permitida a entrada e saída de servidores durante a execução do sistema [Gifford 1979, Attiya et al. 1995, Malkhi and Reiter 1998]. Esta abordagem não é adequada para sistemas que permanecerão em execução por um longo tempo, uma vez que,

*Este trabalho é parcialmente suportado pela Fundação de Ciência e Tecnologia (FCT) de Portugal através dos projetos SITAN (PTDC/EIA-EIA/113729/2009) e LaSIGE (PEst-OE/EEI/UI0408/2014).

dispondo de um quantidade suficiente de tempo, um adversário pode comprometer um número maior de servidores do que o tolerado e então quebrar as propriedades do sistema. Outra limitação é que estes protocolos não permitem que um administrador, em tempo de execução, adicione máquinas no sistema (para suportar um aumento na carga de processamento) ou troque máquinas antigas. Além disso, estes protocolos não são adequados para sistemas distribuídos modernos, desenvolvidos para redes dinâmicas, móveis ou auto-organizáveis (e.x., MANETs, P2P, Nuvens), onde, pela sua própria natureza, o conjunto de processos que compõem o sistema sofre modificações durante a execução.

O processo de modificar o conjunto de processos que compõem o sistema chama-se *reconfiguração*. Algumas propostas para sistemas de quóruns reconfiguráveis [Gilbert et al. 2010, Martin and Alvisi 2004] utilizam consenso [Lamport 1998] para garantir que todos os processos concordem com o conjunto de processos que fará parte do sistema. Entretanto, é conhecido que o consenso torna-se impossível num ambiente assíncrono sujeito a falhas de servidores e, para resolvê-lo, abstrações de sincronia precisam ser incorporadas ao sistema, como os detectores de falhas [Chandra and Toueg 1996]. Ocorre que a manutenção da consistência de um sistema de quóruns estático pode ser feita sem a necessidade de acordo e, portanto, num sistema totalmente assíncrono. Até pouco tempo, ainda não se sabia se seria possível implementar reconfigurações sem necessidade de acordo. O *DynaStore* [Aguilera et al. 2011] responde a essa questão apresentando um conjunto de algoritmos que implementam um sistema de quóruns dinâmico sem o auxilio de consenso. Outra proposta recente que segue esta abordagem, mas para sistemas Bizantinos, é o QUINCUNX [Alchieri et al. 2012].

Este artigo apresenta o FREESTORE, um conjunto de algoritmos que permite atualizar o conjunto de servidores do sistema durante sua execução e implementa um registrador atômico [Lamport 1986] tolerante a falhas por parada (*crash*) e livre de espera [Herlihy 1991]. Dois conjuntos de protocolos são propostos: *(i)* protocolos de leitura e escrita – que podem ser adaptados a partir de qualquer registrador estático (ex., o clássico ABD [Attiya et al. 1995]) e *(ii)* protocolos de reconfiguração – que implementam o processo de reconfiguração do conjunto de servidores presentes no sistema.

O FREESTORE melhora o estado da arte na área em pelo menos três aspectos: *modularidade*, *eficiência* e *simplicidade*. A modularidade é conseguida através da separação entre protocolos de reconfiguração e protocolos de leitura e escrita. Tal desacoplamento permite que o FREESTORE seja muito mais eficiente, em termos de passos de comunicação, do que outras propostas baseadas [Gilbert et al. 2010] ou não [Aguilera et al. 2011] em consenso, tanto nos protocolos de leitura e escrita quanto na reconfiguração. Finalmente, consideramos a estratégia empregada no protocolo de reconfiguração do FREESTORE mais simples de entender do que outras propostas, elucidando o processo de reconfiguração de sistemas de quóruns.

Resumidamente, as principais contribuições deste artigo são as seguintes: (i) apresentação do FREESTORE, um conjunto de algoritmos modular e eficiente para reconfiguração de sistemas de quóruns; (ii) apresentação do primeiro protocolo tolerante a falhas por parada para sistemas de armazenamento em ambientes dinâmicos no qual os protocolos de reconfiguração são desacoplados dos protocolos de leitura e escrita, facilitando a adaptação de registradores estáticos para funcionamento em ambientes dinâmicos; (iii) discussão sobre as principais diferenças entre utilizar ou não consenso em

reconfigurações; e (iv) apresentação de uma comparação detalhada do FREESTORE com outras soluções encontradas na literatura.

## 2. Definições Preliminares

### 2.1. Modelo de Sistema

Consideramos um sistema distribuído completamente conectado composto pelo conjunto universo de processos $U$, que é dividido em dois subconjuntos distintos: um conjunto infinito de servidores $\Pi = \{s_1, s_2, ...\}$; e um conjunto infinito de clientes $C = \{c_1, c_2, ...\}$. Clientes acessam o sistema de armazenamento implementado por um subconjunto dos servidores (uma *visão*) executando operações de leitura e escrita (R/W). Cada processo do sistema (cliente ou servidor) possui um identificador único e está sujeito a falhas por parada (*crash*). A chegada dos processos no sistema segue o modelo de chegadas infinitas com concorrência desconhecida mas finita [Aguilera 2004]. Além disso, consideramos que os processos estão conectados através de canais confiáveis, os quais são implementáveis através de SSL/TLS [Dierks and Allen 1999].

Consideramos um sistema distribuído assíncrono, onde não existem limites para o tempo de transmissão de mensagens ou processamentos locais nos processos. No entanto, cada servidor do sistema tem acesso a um relógio local usado para iniciar reconfigurações. Estes relógios não são sincronizados e não existe qualquer limite (*bounds*) para seus desvios (*drifts*), sendo portanto apenas contadores.

### 2.2. Dinamicidade e Propriedades do Sistema

Durante a execução, uma sequência de visões é instalada no sistema através de reconfigurações causadas por entradas e saídas de servidores. A seguir são apresentadas algumas definições e propriedades satisfeitas pelo FREESTORE.

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

**Visões.** Uma visão $v$ é composta por um conjunto de *updates* (representados por $v.entries$) e seu *membership* (representado por $v.members$) é definido como $v.members = \{i \in \Pi : \langle +, i\rangle \in v.entries \wedge \langle -, i\rangle \notin v.entries\}$. Por simplicidade, usamos $i \in v$ para simbolizar $i \in v.members$. Note que um processo $i$ pode entrar e sair do sistema uma única vez, mas esta condição pode ser relaxada na prática adicionando-se um número de época em cada *update*.

Dizemos que uma visão $v$ é *instalada em um servidor correto* caso o mesmo considere $v$ sua *visão atual* e passa a executar as operações de clientes nesta visão. Neste caso, dizemos que $v$ foi *instalada no sistema*. Em qualquer tempo real $t$, definimos $V(t)$ como sendo *a visão mais atual* (veja definição abaixo) instalada no sistema. Consideramos que $V(t)$ permanece *ativa* desde o tempo de sua instalação até que todos os servidores corretos de outra visão mais atual $V(t'), t' > t$, instalam $V(t')$.

**Comparando visões.** Comparamos duas visões $v_1$ e $v_2$ através de suas $entries$. Usamos a notação $v_1 \subset v_2$ e $v_1 = v_2$ como abreviação de $v_1.entries \subset v_2.entries$ e $v_1.entries = v_2.entries$, respectivamente. Caso $v_1 \subset v_2$, então $v_2$ é *mais atual* do que $v_1$.

***Bootstrapping*.** Assumimos uma visão inicial não vazia $V(0)$ conhecida por todos os processos do sistema. Durante a inicialização do sistema, cada servidor $i \in V(0)$ recebe a visão inicial $v_0 = \{u_0\}$, onde $u_0 = \{\langle +, j\rangle : j \in V(0)\}$.

**Visões vs. operações de R/W.** Em qualquer tempo real $t$, operações de r/w são executadas apenas em $V(t)$. Quando um servidor $i$ requisita um *join* no sistema, estas operações permanecem desabilitadas em $i$ até a ocorrência de um evento *enable operations*. Após isso, $i$ permanecerá apto a responder estas operações até que $i$ solicite o *leave* do sistema, o que acontece com a ocorrência do evento *disable operations*.

**Definição 1 (Propriedades do FREESTORE)** *As seguinte propriedades são satisfeitas:*

- ***Armazenamento (Segurança)**: Os protocolos de r/w satisfazem as propriedades de segurança de um registrador de r/w atômico [Lamport 1986].*
- ***Armazenamento (Terminação)**: Todas operações de r/w executadas por clientes corretos terminam.*
- ***Reconfiguração – Join (Segurança)**: Se um servidor $j$ instala a visão $v$ tal que $i \in v$, então o servidor $i$ executou a operação $join$ ou é membro da visão inicial.*
- ***Reconfiguração – Leave (Segurança)**: Se um servidor $j$ instala a visão $v$ tal que $i \notin v \wedge (\exists v' : i \in v' \wedge v' \subset v)$, então o servidor $i$ executou a operação $leave$.*
- ***Reconfiguração – Join (Terminação)**: O evento enable operations terminará por ocorrer em todo servidor correto que executa a operação $join$.*
- ***Reconfiguration – Leave (Terminação)**: O evento disable operations terminará por ocorrer em todo servidor correto que executa a operação $leave$.*

Para lidar com a dinamicidade do sistema, algumas suposições são necessárias. As suposições 1 e 2 são adaptações diretas dos resultados para sistemas de quóruns estáticos [Attiya et al. 1995] e devem ser satisfeitas enquanto a visão $v$ estiver ativa. A Suposição 3 diz que um servidor correto que está deixando o sistema participará do protocolo de reconfiguração. Já a Suposição 4 garante que operações de r/w sempre terminam, i.e., garante que um cliente lento reiniciará sua operação, devido a reconfigurações, um número finito de vezes e, então, completará sua operação[1]. Esta suposição é necessária em todas as propostas para sistemas de quóruns dinâmicos [Aguilera et al. 2011, Gilbert et al. 2010, Martin and Alvisi 2004].

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

**Suposição 2 (Tamanho dos quóruns)** *Para cada visão $v$, consideramos quóruns de tamanho $v.q = \lceil \frac{|v.members|+1}{2} \rceil$.*

**Suposição 3 (Saídas graciosas)** *Um servidor correto $i \in V(t)$ que requisita um leave no tempo $t$ permanece no sistema até saber que uma visão mais atual $V(t'), t' > t, i \notin V(t')$ foi instalada no sistema.*

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

## 3. Geradores de Sequências de Visões

*Geradores de sequências de visões* ($\mathcal{G}$) são módulos locais usados pelos servidores para gerar sequências de novas visões para reconfigurar o sistema. Este módulo captura os requisitos de sincronia necessários para a reconfiguração do sistema. Note que estes geradores geram sequências de visões (e não uma única visão), pois o principal desafio dos protocolos de reconfiguração é garantir a convergência para uma sequência de visões geradas durante a reconfiguração [Aguilera et al. 2010, Aguilera et al. 2011, Gilbert et al. 2010].

[1] Apenas o número de *updates* requisitados concorrentemente com uma operação de r/w deve ser finito.

Para cada visão $v$, cada servidor $i \in v$ associa um gerador $\mathcal{G}_i^v$ para gerar sequências de visões para atualizar o sistema. Um servidor $i$ interage com o gerador através de duas primitivas: (1) $\mathcal{G}_i^v.gen\_view(seq)$, chamada por $i$ para propor uma nova sequências de visões $seq$ para atualizar $v$; e (2) $\mathcal{G}_i^v.new\_view(seq)$, um *callback* invocado pelo gerador para informar $i$ sobre uma nova sequência $seq$ gerada. Como outros oráculos distribuídos (ex., detectores de falhas [Chandra and Toueg 1996]), os geradores podem implementar estas operações de diferentes formas, de acordo com os diferentes ambientes nos quais irão executar (ex., síncrono ou assíncrono). No entanto, neste trabalho introduzimos geradores que satisfazem as seguintes propriedades.

**Definição 2 (GERADORES)** *Um gerador $\mathcal{G}_i^v$ (associado à visão $v$ no processo $i$) satisfaz as seguintes propriedades:*

- ***Exatidão**: existem duas variantes de exatidão:*
  - ***Exatidão Forte**: se $i$ receber o upcall $\mathcal{G}_i^v.new\_view(seq_i)$, então todos os servidores $j \in v$ recebem upcalls $\mathcal{G}_j^v.new\_view(seq_j)$ tal que $seq_i = seq_j$.*
  - ***Exatidão Fraca**: se $i$ receber o upcall $\mathcal{G}_i^v.new\_view(seq_i)$, então todos os servidores $j \in v$ recebem upcalls $\mathcal{G}_j^v.new\_view(seq_j)$ tal que $seq_i \subseteq seq_j$ ou $seq_j \subseteq seq_i$.*
- ***Não trivialidade**: para todas upcalls $\mathcal{G}_i^v.new\_view(seq_i)$, para qualquer visão $w \in seq_i$, temos que $v \subset w$.*
- ***Terminação**: para qualquer invocação $\mathcal{G}_i^v.gen\_view(seq)$, existirá uma upcall $\mathcal{G}_i^v.new\_view(seq_i)$, a menos que $i$ falhe.*

As propriedades de *Exatidão* e *Não trivialidade* estão relacionadas com a segurança, enquanto que *Terminação* está relacionada com a vivacidade do processo de geração de sequências de visões. Qualquer gerador deve atender as propriedades de Não trivialidade e Terminação para garantir que apenas visões atualizadas estarão nas sequências que sempre serão geradas. Usando as duas variantes de Exatidão definimos dois geradores: $\mathcal{P}$, gerador *perfeito* que satisfaz *Exatidão Forte*; e $\mathcal{L}$, gerador *vivo* que satisfaz apenas *Exatidão Fraca*. A implementação de $\mathcal{P}$ requer consenso, enquanto $\mathcal{L}$ pode ser implementado sem primitivas fortes de sincronização.

### 3.1. Gerador de Sequências de Visões Perfeito – $\mathcal{P}$

Geradores perfeitos garantem que uma única sequência de visões é gerada em todos os servidores que iniciaram o gerador para atualizar uma visão $v$. Nossa implementação de $\mathcal{P}$ (Algoritmo 1) é baseada no algoritmo de consenso Paxos [Lamport 1998] que requer um sistema parcialmente síncrono [Dwork et al. 1988]. Considerando o algoritmo do Paxos, qualquer servidor de $v$ pode ser proponente (invocando $Paxos_v.propose(seq)$), mas todos os servidores de $v$ são aceitantes e aprendizes (recebem um *upcall* $Paxos_v.learn(seq')$).

**Algoritmo 1** $\mathcal{P}$ associado a $v$ - servidor $i \in v$.

**procedure** $\mathcal{G}_i^v.gen\_view(seq)$
1) **if** $\forall w \in seq : v \subset w$ **then** //somente visões atualizadas são propostas
2) $Paxos_v.propose(seq)$ //inicializa um consenso entre os membros de $v$

**upon** $Paxos_v.learn(seq')$
3) $\mathcal{G}_i^v.new\_view(seq')$ // ... informa a decisão para o processo.

### 3.2. Gerador de Sequências de Visões Vivo – $\mathcal{L}$

Nossa implementação para $\mathcal{L}$ (Algoritmo 2) não utiliza consenso, pois não é necessário que a mesma sequência seja gerada em todos os processos, sendo implementável em um sistema assíncrono. Desta forma, duas ou mais sequências podem ser geradas em diferentes servidores, mas a propriedade de intersecção dos quóruns garante que uma sequência gerada estará contida em qualquer outra sequência gerada posteriormente. A ideia principal do algoritmo é fazer os processos atualizarem as suas propostas até que um quórum convirja para uma sequência contendo todas as propostas (ou uma combinação delas).

**Algoritmo 2** $\mathcal{L}$ associado a $v$ - servidor $i \in v$.

**functions:** Funções auxiliares e predicados
$most\_updated(seq) \equiv w\colon (w \in seq) \wedge (\nexists w' \in seq\colon w \subset w')$
**variables:** Conjuntos utilizados no protocolo
$\texttt{SEQ} \leftarrow \emptyset$ // sequência proposta por $i$
$\texttt{LCSEQ} \leftarrow \emptyset$ // última sequência convergida conhecida por $i$

**procedure** $\mathcal{G}_i^v.gen\_view(seq)$
1) **if** $\texttt{SEQ} = \emptyset \wedge \forall w \in seq : v \subset w$ **then**
2) $\quad \texttt{SEQ} \leftarrow seq$
3) $\quad \forall j \in v, send\langle$SEQ-VIEW, SEQ$\rangle$ to $j$

**upon receipt of** $\langle$SEQ-VIEW, $seq\rangle$ from $j$
4) **if** $\exists w \in seq\colon w \notin \texttt{SEQ}$ **then**
5) $\quad$ **if** $\exists w, w'\colon w \in seq \wedge w' \in \texttt{SEQ} \wedge w \not\subset w' \wedge w' \not\subset w$ **then**
6) $\quad\quad w \leftarrow most\_updated(\texttt{SEQ})$
7) $\quad\quad w' \leftarrow most\_updated(seq)$
8) $\quad\quad \texttt{SEQ} \leftarrow \texttt{LCSEQ} \cup \{w.entries \cup w'.entries\}$
9) $\quad$ **else** // todas as visões podem ser agrupadas em uma sequência
10) $\quad\quad \texttt{SEQ} \leftarrow \texttt{SEQ} \cup seq$
11) $\quad \forall k \in v, send\langle$SEQ-VIEW, SEQ$\rangle$ to $k$

**upon receipt of** $\langle$SEQ-VIEW, SEQ$\rangle$ from $v.q$ servers in $v$
12) $\texttt{LCSEQ} \leftarrow \texttt{SEQ}$
13) $\forall k \in v, send\langle$SEQ-CONV, SEQ$\rangle$ to $k$

**upon receipt of** $\langle$SEQ-CONV, $seq\rangle$ from $v.q$ servers in $v$
14) $\mathcal{G}_i^v.new\_view(seq)$

Neste algoritmo, cada servidor $i$ pertencente a uma visão $v$ utiliza: *(i)* uma função $most\_updated$ para extrair a visão mais atual de uma sequência (i.e., a visão que não está contida em nenhuma outra visão da sequência); *(ii)* uma variável SEQ para armazenar sua sequência proposta; e *(iii)* uma variável LCSEQ para armazenar a última sequência convergida em $i$. Quando o gerador é inicializado em um servidor $i \in v$, $i$ verifica se já fez uma proposta ou se a sequência a ser proposta contém apenas visões atualizadas (linha 1), para então fazer sua proposta (linhas 2-3). No entanto, diferentes servidores de $v$ podem propor diferentes sequências que devem ser combinadas em uma sequência composta por todas as propostas. Quando um servidor $i \in v$ recebe uma proposta de um servidor $j$, $i$ verifica se a proposta contém algum visão que ainda não está contida em sua proposta (linha 4). Note que pode acontecer de $i$ ainda não ter feito nenhuma proposta. Nestes casos, $i$ atualiza sua proposta de acordo com dois casos excludentes:

CASO 1: [Existem *visões conflitantes* na sequência proposta por $i$ e na sequência recebida (linhas 5-8)]. Para garantir que os servidores convirjam para uma mesma sequência e manter a relação ("$\subset$") entre as visões, $i$ define sua proposta como sendo uma nova sequência contendo LCSEQ mais uma visão representando a união entre as propostas conflitantes.

CASO 2: [A sequência proposta por $i$ (caso exista) e a sequência recebida *podem ser organizadas em uma nova sequência* (linhas 9-10)]. Então, a nova sequência a ser proposta por $i$ é formada por todas as visões conhecidas por $i$.

Note que em ambos os casos é garantido que a nova proposta sempre contém

sequências previamente convergidas em pelo menos um quórum, i.e., sequências já geradas (ver abaixo). Quando $i$ recebe a mesma proposta de um quórum de servidores em $v$, $i$ *converge* para esta sequência atualizada `SEQ` (linhas 12-13). Quando $i$ descobre que um quórum de processos convergiu para uma sequência $seq$, $i$ *gera* $seq$ (linha 14).

O ponto principal deste algoritmo é garantir que se um quórum de processos converge para uma sequência $seq$ (i.e., executaram as linhas 12-13), $seq$ é gerada (linha 14) e será uma subsequência de qualquer outra sequência gerada posteriormente, garantindo *Exatidão Fraca*. Isso é garantido pela propriedade de intersecção dos quóruns, i.e., pelo menos um servidor correto deve participar da geração de ambas sequências e aplicará as regras dos Casos 1 e 2 descritos acima.

Servidores de $v$ podem gerar no máximo $|v.members| - v.q + 1$ sequências, tal que $seq_1 \subset seq_2 \subset ... \subset seq_{(|v.members|-v.q)} \subset seq_{(|v.members|-v.q+1)}$: $seq_1$ é obtida pela união das propostas de um quórum de servidores em $v$ e as outras $|v.members| - v.q$ sequências são construídas adicionando-se (cumulativamente) a $seq_1$ diferentes propostas de cada um dos $|v.members| - v.q$ servidores que não participaram da geração de $seq_1$.

A propriedade de *Terminação* é garantida pelo fato de que *(i)* cada servidor faz apenas uma proposta inicial (linhas 1-3); *(ii)* os servidores atualizam suas propostas até que um quórum concorde com uma proposta; e *(iii)* sempre existe pelo menos um quórum de servidores corretos em $v$.

## 4. Protocolo de Reconfiguração do FREESTORE

Cada servidor que executa o algoritmo de reconfiguração do FREESTORE utiliza um gerador associado a sua visão atual $cv$ para processar um lote de requisições de atualizações (*joins* e *leaves*) que atualizará o sistema de $cv$ para uma nova visão $w$. Como estas requisições são processadas periodicamente em lotes, mais do que um *update* pode ser processado em cada execução do algoritmo. Além disso, estes protocolos também permitem que operações de r/w sejam executadas em paralelo com reconfigurações.

O Algoritmo 3 descreve como um servidor $i$ executa reconfigurações. A seguir, descrevemos como os geradores são inicializados e discutimos o comportamento do algoritmo quando utiliza um gerador tanto da classe $\mathcal{L}$ (Seção 4.2) quanto da $\mathcal{P}$ (Seção 4.3).

### 4.1. Inicialização do Gerador de Sequências de Visões

O Algoritmo 3 (linhas 1-7) descreve como um servidor $i$ processa requisições de *updates* e inicializa o gerador associado a sua visão atual $cv$. Um servidor $j$ que deseja entrar no sistema primeiramente deve encontrar a visão atual $cv$ e executar a operção $join$ (linhas 1-2), enviando a tupla $\langle +, j \rangle$ para os membros de $cv$. Para sair do sistema o processo é similar (operação $leave$ – linhas 3-4). Quando $i$ recebe estas requisições de $j$, $i$ verifica se a visão de $j$ está desatualizada para então enviar sua $cv$ para $j$ (omitido do algoritmo por simplicidade). Caso contrário, $i$ verifica se ainda não processou a requisição de $j$ para então armazenar a requisição em `RECV`, que será processada na próxima reconfiguração. Finalmente, $i$ envia um *acknowledgment* para $j$ (linhas 5-6).

Por questões de desempenho, um relógio local é utilizado para execuções periódicas de reconfigurações (e não a cada $join/leave$). Um servidor $i \in cv$ inicia a reconfiguração de $cv$ (i.e., inicia o gerador associado a $cv$) quando ocorrer o *timeout* de $cv$ e existir *updates* pendentes (caso contrário, o *timeout* para $cv$ é redefinido). O gerador é inicializado com uma sequência contendo uma visão que representa a $cv$ de $i$ atualizada

com os *updates* pendentes (linha 7). Este algoritmo garante que requisições recebidas por um quórum de servidores em $cv$ serão executadas na reconfiguração seguinte.

### 4.2. Reconfiguração usando $\mathcal{L}$

Esta seção descreve o comportamento das reconfigurações quando o Algoritmo 3 é inicializado com um gerador $\mathcal{L}$ que pode gerar diferentes sequências de visões em diferentes servidores. Nestes casos, este algoritmo garante que uma única sequência de visões é instalada. A ideia principal é escolher apenas algumas visões vindas de $\mathcal{L}$ e definir uma sequência de visões a ser instalada até que uma única visão mais atual seja instalada.

**Visão Geral.** Depois que $\mathcal{L}$ gerou uma sequência, as visões desta sequência são ativadas, uma a uma, até que a visão mais atual da sequência seja instalada. Note que diferentes servidores podem gerar diferentes sequências, mas qualquer sequência gerada será uma subsequência de outra sequência gerada posteriormente.

Dada uma sequência $seq : w_1 \rightarrow \ldots \rightarrow w_k \rightarrow w$ gerada para atualizar uma visão $v$, apenas $w$ é instalada no sistema. As outras $k$ visões $(w_1, \ldots, w_k)$ apenas são utilizadas como intermediárias e são chamadas de *visões auxiliares*. Caso um servidor conheça toda esta sequência, estas visões não são instaladas pois existe uma visão mais atual $w$ para ser instalada. No entanto, é importante perceber que alguns servidores podem conhecer apenas uma subsequência de $seq$ como $seq' : w_1 \rightarrow \ldots \rightarrow w_j$, $j \leq k$, e instalar $w_j$. Neste caso, estes servidores acabarão atualizando suas visões de $w_j$ para $w$.

**Algoritmo 3** Reconfiguração - servidor $i$.

**functions:** Funções auxiliares e predicados
$least\_updated(seq) \equiv w\colon (w \in seq) \wedge (\nexists w' \in seq\colon w' \subset w)$
**variables:** Conjuntos e Gerador utilizado no protocolo
$cv \leftarrow v_0$ // visão mais atual conhecida por $i$
RECV $\leftarrow \emptyset$ // conjunto de *updates* recebidos

**procedure** $join()$
1) $\forall j \in cv, send\langle$RECONFIG, $\langle +, i\rangle, cv\rangle$ to $j$
2) **wait** for $cv.q$ $\langle$REC-CONFIRM$\rangle$ replies from servers in $cv$

**procedure** $leave()$
3) $\forall j \in cv, send\langle$RECONFIG, $\langle -, i\rangle, cv\rangle$ to $j$
4) **wait** for $cv.q$ $\langle$REC-CONFIRM$\rangle$ replies from servers in $cv$

**upon receipt of** $\langle$RECONFIG, $\langle *, j\rangle, cv\rangle$ from $j \wedge \langle *, j\rangle \notin cv$
5) RECV $\leftarrow$ RECV $\cup \{\langle *, j\rangle\}$
6) $send\langle$REC-CONFIRM$\rangle$ to $j$

**upon** ($timeout$ for $cv$) $\wedge$(RECV $\neq \emptyset$)
7) $\mathcal{G}_i^{cv}.gen\_view(\{cv \cup$ RECV$\})$

**upon** $\mathcal{G}_i^{ov}.new\_view(ov, seq)$ //gerador associado a $ov$ gerou uma nova sequência para atualizar $ov$ (geralmente, $ov = cv$)
8) $w \leftarrow least\_updated(seq)$ //a próxima visão na sequência $seq$
9) *R-multicast*($ov \cup w$,$\langle$INSTALL-SEQ, $w, seq, ov\rangle$)

**upon** *R-delivery*($ov \cup w$,$\langle$INSTALL-SEQ, $w, seq, ov\rangle$)
10) **if** $i \in ov$ **then** //$i$ é membro da visão anterior
11) **if** $cv \subset w$ **then** *stop the execution of r/w operations* // enfileira estas operações enquanto trasfere o estado
12) $\forall j \in w, send\langle$STATE-UPDATE, $\langle val, ts\rangle$, RECV$\rangle$ to $j$ //$i$ envia seu estado para os servidores de $w$
13) **if** $cv \subset w$ **then** //$w$ é mais atual do que $cv$ e o sistema reconfigurará de $cv$ para $w$
14) **if** $i \in w$ **then** //$i$ está na nova visão
15) **wait** for $\langle$STATE-UPDATE, $*, *\rangle$ messages from $ov.q$ servers in $ov$
16) $\langle val, ts\rangle \leftarrow \langle val_h, ts_h\rangle$, pair with highest timestamp among the ones received above //$i$ atualiza seu estado...
17) RECV $\leftarrow$ RECV $\cup$ {update requests from STATE-UPDATE messages} $\setminus\ w.entries$
18) $cv \leftarrow w$ //... e atualiza sua visão atual para $w$
19) **if** $i \notin ov$ **then** *enable operations* //$i$ esta entrando no sistema
20) $\forall j \in ov \setminus cv, send\langle$VIEW-UPDATED, $cv\rangle$ to $j$ //informa os servidores de $ov \setminus cv$, os quais podem deixar o sistema
21) **if** $(\exists w' \in seq\colon cv \subset w')$ **then** //existem visões mais atuais do que $cv$ em $seq$...
22) $seq' \leftarrow \{w' \in seq\colon cv \subset w'\}$ //... seleciona estas visões...
23) $\mathcal{G}_i^{cv}.gen\_view(seq')$ //... e as propõe (volta para o Agoritmo 2)
24) **else**
25) *resume the execution of r/w operations* in $cv = w$ and start a timer for $cv$ //$w$ **é instalada**
26) **else** //$i$ está saindo do sistema
27) *disable operations*
28) **wait** for $\langle$VIEW-UPDATED, $w\rangle$ messages from $w.q$ servers in $w$ and then **halt** //$i$ pode deixar o sistema

A propriedade de Exatidão Fraca garante que nenhuma operação de escrita executada em uma visão de $seq'$ (caso instalada) é perdida. Para isso, é realizada uma "leitura em cadeia" onde os servidores de $v$ transferem seus estados para os servidores de $w_1$, os quais transferem seus estados para os servidores de $w_2$ e assim por diante, até que os servidores de $w$ recebem o valor armazenado no sistema a partir dos servidores de $w_k$.

**Reconfiguração.** O Algoritmo 3 (linhas 8-28) apresenta a parte principal do protocolo de reconfiguração do FREESTORE. Este algoritmo utiliza a função $least\_updated$ para extrair a visão menos atualizada de uma sequência (i.e., a visão que está contida em todas as outras visões da sequência).

Quando o gerador associado a uma visão antiga $ov$ gerar uma sequência de visões $seq$, a visão menos atualizada $w \in seq$ é a primeira a ser inicializada, sendo proposta na mensagem INSTALL-SEQ enviada para os servidores de $ov \cup w$ (linhas 8-9). Para garantir que todos receberão esta mensagem, usamos uma primitiva de *multicast* confiável [Hadzilacos and Toueg 1994]. Esta primitiva pode ser implementada eficientemente em sistemas assíncronos através da retransmissão de mensagens no receptor antes da entrega (i.e., em um passo de comunicação) [Hadzilacos and Toueg 1994].

Os últimos passos importantes do algoritmo são os seguintes. Primeiro, os servidores param de executar operações de r/w (linha 11), as quais serão executadas quando a visão mais atual da sequência é instalada (linha 25), o que garante que nenhuma operação de r/w é executada em visões auxiliares. Segundo, os servidores de $ov$ enviam seus estados (geralmente o valor do registro e um *timestamp* [Attiya et al. 1995]) para os servidores de $w$ (linha 12). Nestas mensagens também são enviados os conjuntos `RECV` e os servidores coletam estes *updates*, evitando a perda de *updates* requisitados durante reconfigurações. Terceiro, caso $w$ não seja a visão mais atual de $seq$ ($w$ é uma visão auxiliar), uma sequência para atualizar $w$ é proposta e os geradores associados a $w$ são executados (linhas 21-23). Finalmente, um servidor que está deixando o sistema somente sai quando for garantido que a visão mais atual foi inicializada em um quórum de servidores (linha 28), garantindo a vivacidade do protocolo.

O FREESTORE satisfaz linearizabilidade [Herlihy and Wing 1990] devido a três pontos principais: (1) as visões instaladas no sistema formam uma única sequência; (2) durante a instalação de uma visão, o estado do registrador é atualizado através de uma "leitura em cadeia" nas visões anteriores da sequência, sendo que a última visão instalada sempre estará presente nesta sequência e nenhuma escrita é executada em visões auxiliares, garantindo que nenhuma escrita é perdida durante as reconfigurações; e (3) após uma visão $v$ ser desinstalada (i.e., uma visão mais atual do que $v$ ser instalada), operações de r/w não são mais executadas em $v$.

**Exemplo de Execução.** A Figura 1 ilustra um execução do protocolo de reconfiguração do FREESTORE usando $\mathcal{L}$ onde os servidores convergem para uma única visão, mesmo com propostas conflitantes. Considere a visão inicial $v_0 = \{1, 2, 3\}$. No exemplo, ① os servidores $1, 2, 3$ receberam o $join$ do processo $4$ antes de inicializar seus geradores, enquanto que o $join$ do processo $5$ foi recebido apenas pelo processo $1$. Consequentemente, os geradores associados com $v_0$ ($L(v_0)$) são inicializados nos servidores $2, 3$ com uma sequência contendo a visão $v_1 = \{1, 2, 3, 4\}$ e no servidor $1$ com a sequência contendo a visão $v_2 = \{1, 2, 3, 4, 5\}$ (linha 7). A propriedade de Exatidão Fraca de $L(v_0)$ garante que ② os servidores $2, 3$ geram a sequência $seq_1^{v_0} = v_1$ e após, os servidores $1, 2, 3$ geram a

sequência $seq_2^{v_0} = v_1 \rightarrow v_2$ ($seq_1^{v_0} \subset seq_2^{v_0}$).

A Figura mostra que ② os servidores $3, 4 \in v_1$ receberam INSTALL-SEQ para $seq_1^{v_0}$ e ③ instalaram $v_1$ (linhas 11-31) antes de receber INSTALL-SEQ para $seq_2^{v_0}$. Adicionalmente, ③ os servidores $3, 4$ inicializam seus geradores associados com $v_1$ ($L(v_1)$) e, ③ como receberam o $join$ do servidor $6$, $L(v_1)$ é inicializado com uma sequência contendo a visão $v_3 = \{1, 2, 3, 4, 6\}$ (linha 7). Por outro lado, os servidores $1, 2$ recebem INSTALL-SEQ para $seq_2^{v_0}$ antes de receber estas mensagens para $seq_1^{v_0}$. Consequentemente, ③ $v_1$ será uma visão auxiliar nos servidores $1, 2$ que iniciarão $L(v_1)$ com uma sequência contendo $v_2$ (linhas 22-23).

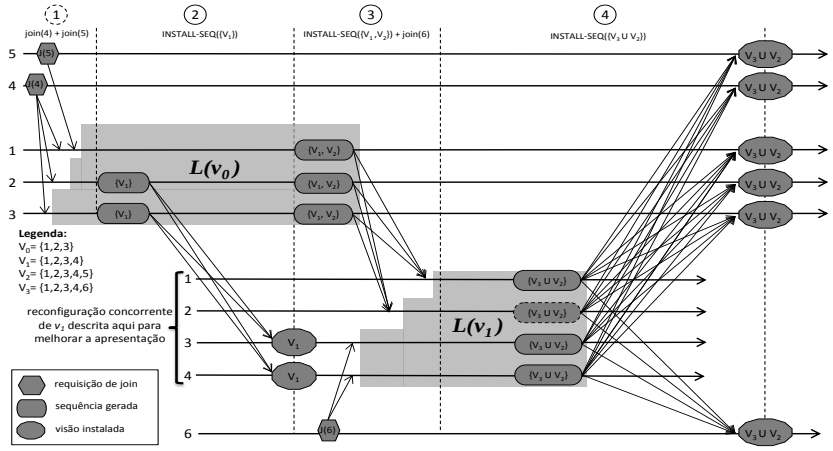



**Figura 1. Convergência para uma única visão.**

Neste cenário, os processos de $v_1$ estão com propostas conflitantes ($v_2 \not\subset v_3$ e $v_3 \not\subset v_2$) que devem ser organizadas em uma única sequência, por $L(v_1)$, para ser instalada após $v_1$. Dependendo da ordem de recebimento das mensagens durante a execução de $L(v_1)$ (i.e., a composição do primeiro quórum de SEQ-VIEW recebido pelo processos de $v_1$), esta sequência poderá ser: (1) $v_2$, (2) $v_3$, (3) $v_2 \rightarrow \{v_3 \cup v_2\}$, (4) $v_3 \rightarrow \{v_3 \cup v_2\}$ ou (5) $\{v_3 \cup v_2\}$. Entretanto, a propriedade de Exatidão Fraca de $L(v_1)$ garante que os servidores não instalarão tanto $v_3$ quanto $v_2$, i.e., existirá uma sequência única de visões instaladas. Em nosso exemplo, como tanto $\{1, 2\}$ quanto $\{3, 4\}$ não formam um quórum de $v_1$, o conflito deve ser resolvido antes de qualquer sequência ser gerada. Consequentemente, a única sequência que poderá ser gerada é a (5). Assim, ④ $L(v_1)$ gera uma sequência $seq_1^{v_1} = \{v_3 \cup v_2\}$ e todos os servidores instalam a visão $v_3 \cup v_2$ (linhas 11-31).

**Otimização.** É possível reduzir um passo de comunicação no protocolo de reconfiguração (Algoritmo 3) quando utilizado com $\mathcal{L}$ (Algoritmo 2). Para isso, simplesmente basta usar as mensagens SEQ_CONV de $\mathcal{L}$ para informar os servidores que uma nova sequência deve ser instalada, eliminando as mensagens INSTALL_SEQ.

### 4.3. Reconfiguração usando $\mathcal{P}$

Caso um gerador desta classe seja usado com o protocolo de reconfiguração do FREESTORE (Algoritmo 3), todos os geradores irão gerar a mesma sequência de visões (Exatidão Forte) que conterá uma única visão $w$, uma vez que todos propõem uma única visão (linha 7). Isso fará com que o sistema reconfigure diretamente da visão atual $cv$ para $w$.

## 5. Protocolos de Leitura e Escrita

Esta seção discute como protocolos para sistemas de quóruns estáticos podem ser adaptados para sistemas dinâmicos através do FREESTORE, o que é simplificado pelo fato

de as reconfigurações e as operações de r/w serem desacopladas. Basicamente, existem duas modificações principais. Primeiro, cada processo deve manipular uma variável $cv$ para armazenar a visão mais atual conhecida e todas as mensagens do protocolo de r/w devem conter $cv$. Os servidores rejeitam qualquer operação enviada para uma visão antiga, respondendo com a visão atual. Os clientes então atualizam $cv$ e reiniciam a fase da operação que está sendo executada (ver abaixo). Segundo, antes de acessar o sistema um cliente $c$ deve obter a visão atual do sistema, o que pode ser implementado através de um serviço de diretório [Aguilera et al. 2010] onde os servidores armazenam as visões. Estes problemas são intrínsecos de protocolos para sistemas dinâmicos [Martin and Alvisi 2004, Aguilera et al. 2011, Gilbert et al. 2010].

Neste trabalho estendemos o clássico protocolo ABD [Attiya et al. 1995] para tolerar múltiplos escritores. A seguir, são apresentadas apenas os aspectos principais dos protocolos de r/w, os quais são executados em fases. Cada fase corresponde a um acesso a um quórum de servidores de $cv$. O protocolo de leitura funciona da seguinte forma:

- (*1ª Fase*) O leitor requisita um conjunto de tuplas $\langle val, ts\rangle$ de um quórum de servidores de $cv$ ($val$ é o valor armazenado pelos servidores e $ts$ é o *timestamp* associado) e seleciona a tupla com o maior *timestamp* $\langle val_h, ts_h\rangle$. A operação termina e retorna $val_h$ caso todas as tuplas coletadas sejam iguais, o que acontece em execuções sem concorrência com escritas ou falhas.
- (*2º Fase*) Caso contrário, o cliente executa uma fase de *write-back* no sistema e aguarda por confirmações até ser garantido que um quórum de servidores de $cv$ receberam $\langle val_h, ts_h\rangle$ e então retorna $val_h$.

O protocolo de escrita funciona de forma similar:

- (*1ª Fase*) O escritor obtém um conjunto de *timestamps* de um quórum de servidores de $cv$ e escolhe o maior deles, $ts_h$. O *timestamp* de escrita $ts$ é definido incrementando-se $ts_h$ e concatenando-se o identificador do escritor nos *bits* menos significativos (para resolver conflitos) [Malkhi and Reiter 1998];
- (*2º Fase*) O escritor envia a tupla $\langle val, ts\rangle$ para os servidores de $cv$, escrevendo $val$ com *timestamp* $ts$, e aguarda por um quórum de confirmações.

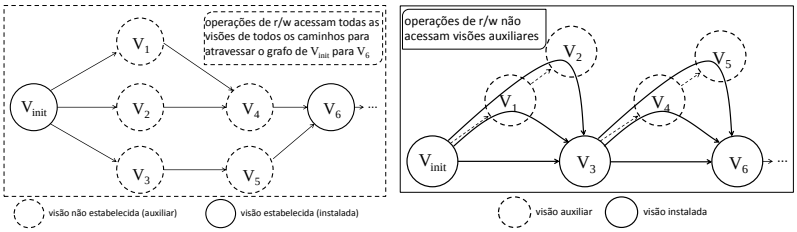
O desacoplamento entre os protocolos de r/w e as reconfigurações possibilita a execução destas operações em paralelo com reconfigurações e simplifica a adaptação de qualquer implementação estática para ambientes dinâmicos.

## 6. Análise das Soluções para Reconfiguração de Sistemas de Quóruns

### 6.1. DynaStore vs. FREESTORE

Esta seção discute as principais diferenças entre as abordagens que não usam consenso.

**Estratégia de Convergência.** No DynaStore, reconfigurações geram um grafo de visões a partir do qual é possível identificar um sequência de visões estabelecidas (instaladas – Figura 2(a)). Visões não estabelecidas funcionam como auxiliares, mas são acessadas nas operações de r/w. Para qualquer visão estabelecida $v$, o número máximo de visões geradas é da ordem de $|v|$, i.e., cada membro de $v$ pode gerar uma nova visão e combinações também são possíveis. Por outro lado, no FREESTORE sequências geradas por diferentes geradores são organizadas em uma única sequência a ser instalada (Figura 2(b)). Para qualquer visão $v$, nossa implementação de $\mathcal{L}$ limita o número de sequências geradas em $|v| - v.q + 1$, compostas por no máximo $|v|$ visões.




(a) DynaStore (b) FreeStore

**Figura 2. Dynastore vs. Freestore: convergência e operação.**

**Caso Normal de Execução.** Para cada visão $v$ (estabelecida ou não), o DynaStore associa um *weak snapshot object* $wso_v$ que armazena as atualizações e é implementado por um conjunto $S_v$ de $|v|$ (uma para cada membro de $v$) registradores SRMR (*single-writer multi-reader*), que são executados nos próprios servidores de $v$. Durante uma operação de r/w, $wso_v$ deve ser acessado duas vezes para verificar se existem atualizações para $v$. Cada um destes acessos envolve duas leituras em cada registrador de $S_v$. Assim, para executar um operação de r/w, são necessários $4 * |v|$ (4 em sequência, $|v|$ em paralelo) acessos a um quórum (dois passos de comunicação). Por outro lado, o *overhead* introduzido pelo FREESTORE nas operações de r/w é a verificação da visão utilizada pelo cliente, i.e., verificar se esta visão está atualizada através da comparação de dois conjuntos (Seção 2.2).

**Concorrência entre R/W e Reconfigurações.** O DynaStore gera um grafo de visões para reconfigurar o sistema e uma operação de r/w deve percorrer este grafo para encontrar a visão mais atual. Para isso, cada aresta do grafo deve ser acessada. Para cada visão $v$ (estabelecida ou não), é necessário acessar seu *weak snapshot object*, o que demanda $2 * |v|$ (2 em sequência, $|v|$ em paralelo) acessos a um quórum. Além disso, enquanto o sistema está convergindo para alguma visão, as operações de r/w não terminam. Por outro lado, no FREESTORE as operações de r/w não acessam visões auxiliares. Além disso, enquanto o sistema está convergindo, as operações de r/w são executadas concorrentemente na última visão instalada $v$. No entanto, após uma sequência ser obtida para atualização de $v$, as operações de r/w apenas terminam após a instalação da última visão desta sequência.

**Passos de Comunicação para R/W.** A Tabela 1 analisa o número de passos de comunicação necessários para executar uma operação de r/w. Para um cliente $c$ que possui uma visão atualizada $v$, cada operação de r/w executada no DynaStore necessita dos seguintes passos: 2 acessos ao $wso_v$, resultando em 8 passos de comunicação para executar 4 leituras em cada registrador de $S_v$, verificando se $v$ está atualizada; e 4 passos de comunicação para ler/escrever o valor de/para $v$. No FreeStore, uma operação de r/w necessita dos mesmos passos de comunicação usados no protocolo de r/w base (Seção 5).

A Tabela 1 também mostra os resultados para quando $c$ está com a visão $v$ desatualizada. Neste caso, o DynaStore precisa de 6 passos de comunicação para acessar cada aresta do grafo de visões. Para simplificar esta análise, consideramos que somente uma aresta deve ser acessada para encontrar a visão atualizada $w$ (melhor caso). Assim, $c$ necessita de 6 passos de comunicação para encontrar $w$, 12 passos de comunicação para executar a operação de r/w em $w$ e 1 passo de comunicação para notificar os processos sobre a nova visão. No FREESTORE, para qualquer caso, 2 passos adicionais são necessários para receber $w$ a partir dos servidores de $v$.

| | DynaStore | FreeStore |
|---|---|---|
| | Visão Atualizada | |
| Leitura | 12 | 2/4 |
| Escrita | 12 | 4 |
| | Visão Desatualizada | |
| Leitura | 19 | 4/6 |
| Escrita | 19 | 6 |

| Soluções Livres de Consenso | Melhor Caso | Pior Caso |
|---|---|---|
| DynaStore | 23 | $18 * \|v\| + 5$ |
| FREESTORE com $\mathcal{L}$ | 4 | $7 * \|v\| - 2 * v.q - 1$ |
| FREESTORE com $\mathcal{L}$ (otimizado) | 3 | $5 * \|v\| - v.q - 1$ |
| **Soluções Baseadas em Consenso** | **Melhor Caso** | **Pior Caso** |
| RAMBO | 7 | 7 |
| FREESTORE com $\mathcal{P}$ | 5 | 5 |

**Tabela 1. Custos das operações de leitura e escrita (esquerda) e reconfigurações (direita).**

## 6.2. Reconfiguração Livre de Consenso vs. Baseada em Consenso.

Esta seção analisa os custos envolvidos na reconfiguração de uma visão $v$. Os resultados para as soluções livres de consenso (DynaStore [Aguilera et al. 2011] e FREESTORE com $\mathcal{L}$) e para duas solução que utilizam consenso para reconfigurar o sistema (RAMBO [Gilbert et al. 2003] e FREESTORE com $\mathcal{P}$) são apresentados na Tabela 1. São analisados os custos para o melhor caso, quando todos os processos propõem os mesmos *updates*, e para o pior caso, quando cada processo propõe diferentes *updates* para reconfigurar o sistema. Na análise das soluções livres de consenso, consideramos que nenhuma reconfiguração é inicializada paralelamente por processos de outras visões. Finalmente, consideramos uma execução síncrona do protocolo de consenso Paxos [Lamport 1998] para as soluções baseadas em consenso, o que requer três passos de comunicação.

Os custos para o melhor caso são definidos como segue: *(i)* DynaStore - somente um processo de $v$ escreve um *update* no *weak snapshot object* associado a $v$ ($wso_v$), desta forma são necessários 4 passos para escrever em $wso_v$, 6 passos para ler o valor do registrador de $v$, 6 passos para atravessar o grafo de visões e alcançar a nova visão $w$, 6 passos para escrever o valor do registrador em $w$ e 1 passo para notificar os outros processos sobre $w$, totalizando 23 passos; *(ii)* FREESTORE com $\mathcal{L}$ - são necessários 4 passos, i.e., 1 SEQ_VIEW + 1 SEQ_CONV + 1 INSTALL_SEQ + 1 STATE_UPDATE; *(iii)* FREESTORE com $\mathcal{L}$ (otimizado) - como o anterior, mas as mensagens INSTALL_SEQ não são mais necessárias; *(iv)* RAMBO - 3 passos para executar um consenso, 2 passos para ler o valor da configuração (visão) antiga e 2 passos para escrever o valor na nova configuração (atualizando-a), totalizando 7 passos; e *(v)* FreeStore com $\mathcal{P}$ - 3 passos para executar um consenso mais 1 INSTALL_SEQ e 1 STATE_UPDATE, totalizando 5 passos.

Os custos para o pior caso são definidos como segue: *(i)* DynaStore - os processos de $v$ escrevem seus *updates* nos *weak snapshot objects* formando um grafo com um único caminho de $v$ para a visão mais atualizada composta pelo união de todos os *updates* (somente no final deste processamento uma operação de r/w termina), este caminho contém $|v|$ visões e, para atravessá-lo, um processo gasta 18 passos para cada visão, 4 passos para escrever seus *updates* e 1 passo para notificar os processos sobre uma visão mais atual, totalizando $18 * |v| + 5$ passos; *(ii)* FreeStore com $\mathcal{L}$ - os geradores associados com $v$ geram no máximo $|v| - v.q + 1$ sequências contendo no máximo $|v|$ visões, para isso são necessárias $|v|$ SEQ_VIEW, $|v| - v.q + 1$ SEQ_CONV, $|v| - v.q + 1$ INSTALL_SEQ e 1 STATE_UPDATE, já os geradores associados com as outras $|v| - 1$ visões recebem estas sequências e terminam em 4 passos (um para cada mensagem acima citada) demandando mais $4(|v|-1)$ passos, totalizando $7*|v|-2*v.q-1$ passos; *(iii)* FreeStore com $\mathcal{L}$ (otimizado) - como o anterior, mas as mensagens INSTALL_SEQ não são mais necessárias. As soluções baseadas em consenso apresentam os mesmos custos para ambos os cenários.

## 7. Conclusões

Este artigo apresentou uma nova abordagem para reconfigurar sistemas de quóruns, a qual evidencia as diferenças entre o uso ou não de consenso. O principal resultado é um protocolo mais eficiente (em termos de passos de comunicação tanto para operações de r/w quanto para reconfigurações) do que outras soluções encontradas na literatura e que desacopla as operações de r/w das reconfigurações. Outro resultado interessante é que no melhor caso, reconfigurações que não utilizam consenso são mais baratas do que as que utilizam este protocolo de sincronização. Acreditamos que explorar diferentes implementações de geradores (e diferentes classes) é um importante caminho para trabalhos futuros que desejam investigar o problema da reconfiguração.

## Referências

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

Aguilera, M. K., Keidar, I., Malkhi, D., Martin, J.-P., and Shraer, A. (2010). Reconfiguring replicated atomic storage: A tutorial. *Bulletin of the EATCS: The Distributed Computing Column*.

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

Alchieri, E., Bessani, A., Fraga, J., and Greve, F. (2012). Memória compartilhada em sistemas bizantinos dinâmicos. In *Anais XIII Workshop de Teste e Tolerância a Falhas*.

Attiya, H., Bar-Noy, A., and Dolev, D. (1995). Sharing memory robustly in message-passing systems. *Journal of the ACM*, 42(1):124–142.

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

Dierks, T. and Allen, C. (1999). The TLS Protocol Version 1.0 (RFC 2246). IETF Request For Comments.

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

Gifford, D. (1979). Weighted voting for replicated data. In *7th ACM Symposium on Operating Systems Principles*.

Gilbert, S., Lynch, N., and Shvartsman, A. (2003). Rambo ii: Rapidly reconfigurable atomic memory for dynamic networks. *Proceedings of the 33th International Conference on Dependable Systems and Networks*.

Gilbert, S., Lynch, N., and Shvartsman, A. (2010). Rambo: A robust, reconfigurable atomic memory service for dynamic networks. *Distributed Computing*, 23(4).

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

Herlihy, M. (1991). Wait-free synchronization. *ACM Transactions on Programing Languages and Systems*, 13(1).

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

Lamport, L. (1986). On interprocess communication (part II). *Distributed Computing*, 1(1):203–213.

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

Malkhi, D. and Reiter, M. (1998). Secure and scalable replication in Phalanx. In *17th Symposium on Reliable Distributed Systems*, pages 51–60.

Martin, J.-P. and Alvisi, L. (2004). A Framework for Dynamic Byzantine Storage. In *34th International Conference on Dependable Systems and Networks*.

Vogels, W. (2009). Eventually consistent. *Commun. ACM*, 52(1):40–44.

# Validação Atômica Não-Bloqueante com Faltas Bizantinas

**Aldelir Fernando Luiz[1,4], Lau Cheuk Lung[2], Miguel Correia[3], Valdir Stumm Júnior[4]**

[1]Departamento de Automação e Sistemas – Universidade Federal de Santa Catarina

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

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

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

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

***Abstract.*** *In this work we present the Byzantine Fault-Tolerant Non-Blocking Weak Atomic Commitment (BFT-NBWAC), a new proposal to solve the Non-Blocking Atomic Commitment (NBAC) problem, where the participants of protocol may fail in a Byzantine way. Our approach is based on virtualization technology, in which we specify a distributed system's architecture that makes possible the detection, masking and confining the faulty behavior of the transactions' participants.*

***Resumo.*** *Neste trabalho apresentamos o BFT-NBWAC (Byzantine Fault-Tolerant Non-Blocking Weak Atomic Commitment), uma proposta inovadora para resolver um problema de acordo conhecido na literatura como Validação Atômica Não-Bloqueante (NBAC), em um ambiente onde os participantes de uma transação podem falhar de maneira bizantina. A abordagem é baseada em um conceito novo denominado "máquinas virtuais gêmeas", em que são exploradas algumas vantagens fornecidas pela tecnologia de virtualização para especificar uma arquitetura de sistema distribuído, na qual é possível detectar, mascarar e confinar o comportamento faltoso exibido pelos participantes da transação.*

## 1. Introdução

A noção de transações é essencial para a especificação e implementação de sistemas computacionais, para os quais a consistência acerca da execução de um conjunto de operações sobre dados deve ser rigorosamente preservada. Não obstante, assegurar a terminação de uma transação de maneira consistente em um sistema distribuído não é uma tarefa trivial. Para tal, é requerido o estabelecimento de um acordo entre os participantes envolvidos na transação, a fim de validar (do inglês, *commit*) ou anular (do inglês, *abort*) as operações que a compõem (atomicidade), de modo a torná-las permanentes (durabilidade) ou descartá-las do sistema. O problema que define a validação atômica de transações em sistemas distribuídos é conhecido na literatura como **Validação Atômica Não-Bloqueante** (do inglês, *Non-Blocking Atomic Commitment* – NBAC) [Babaoğlu and Toueg 1993].

O NBAC pode ser visto como um **problema de acordo** em sistemas distribuídos [Pease et al. 1980, Hadzilacos 1990, Guerraoui 1995], e como tal, está sujeito às mesmas restrições teóricas e aos aspectos práticos presentes na concepção de sistemas tolerantes a faltas. Assim, ao longo dos anos um grande número de soluções para o NBAC

foram propostas na literatura, algumas no intuito de circunscrever propriedades e ampliar a resiliência a despeito de faltas de parada (p. ex. *crash*) [Guerraoui et al. 1995, Guerraoui 1995, Guerraoui et al. 1996, Park et al. 2013], e outras para prover otimizações [Guerraoui and Schiper 1995, Abdallah and Pucheral 1999, Greve and Narzul 2002]. Um aspecto particularmente interessante destes estudos é que alguns deles demonstraram as condições necessárias para a resolução do NBAC (p.ex. modelos de sistema, ambientes computacionais, modelos de falhas, detectores de falhas, etc.), além de terem evidenciado as situações nas quais o problema é solúvel.

Embora o NBAC seja um problema bem estabelecido na literatura, tanto em cenários com ou sem faltas de parada, até o momento ele permanece sem uma solução em se tratando de faltas bizantinas [Lamport et al. 1982]. A impossibilidade desta advém de um importante resultado teórico, o qual demonstrou que os problemas de acordo baseados na propriedade de **acordo uniforme** não podem ser resolvidos com faltas bizantinas [Charron-Bost and Schiper 2004]. Isto decorre do simples fato de que o modelo bizantino não assegura a decisão para um processo faltoso, pois nada é previsível a respeito do comportamento bizantino. Este resultado é estendido para outros problemas de acordo especificados a partir da propriedade **uniforme**, como é o caso do NBAC. Particularmente, o NBAC não é solúvel porque mesmo que um participante bizantino reúna as condições necessárias para validar uma transação, ele pode optar por anulá-la propositadamente para deturpar o protocolo. Ou ainda, se o protocolo decidir pela validação, não se pode garantir que o processo bizantino realizará as ações pós-decisão (p.ex. tornar as escritas permanentes).

Neste sentido, este trabalho apresenta uma abordagem prática para resolver a **Validação Atômica Não-Bloqueante** em um ambiente onde os participantes de uma transação podem falhar de maneira bizantina. Para isso, a solução explora uma arquitetura de sistema distribuído baseada em um modelo de falhas híbrido e realista, na qual é realizada uma separação dos participantes da transação em ativos e passivos. Esta separação é realizada a fim de prover os requisitos e as condições necessárias para assegurar as propriedades do NBAC, a despeito de faltas dos participantes. No caso, os participantes ativos (que conduzem o NBAC) podem falhar de maneira bizantina, enquanto os passivos falham apenas por parada. Ao nosso conhecimento, este é o primeiro trabalho da literatura a abordar o NBAC com um modelo híbrido de falhas, de tal maneira que os participantes ativos da transação possam falhar de maneira bizantina. Deste modo, o presente trabalho representa uma contribuição no âmbito das áreas de algoritmos distribuídos tolerantes a faltas, e de arquiteturas e ambientes para a concepção de sistemas distribuídos tolerante a faltas.

O restante deste artigo está organizado da seguinte maneira: a Seção 2 apresenta uma breve descrição do problema da validação atômica não-bloqueante em sistemas distribuídos; a Seção 3 apresenta algumas definições e conceitos preliminares; na Seção 4 é apresentada a solução proposta por este trabalho; a Seção 5 descreve alguns resultados obtidos; e por fim, a Seção 6 descreve as conclusões acerca do trabalho.

## 2. Caracterização do Problema da Validação Atômica Não-Bloqueante

Tipicamente, uma transação distribuída acessa dados residentes em vários locais (p. ex. sítios) e envolve um conjunto de processos - os participantes, que cooperam entre si para a execução das operações que compõem a unidade lógica de processamento nos sítios [Babaoğlu and Toueg 1993]. Usualmente, um participante desempenha dois papéis em seu local de atuação (ou sítio); o de **gestor de transação** (*Transaction Manager* – TM) e o

de **gestor de dados** (*Data Manager* – DM) - também denominado por **gestor de recursos** (*Resource Manager* – RM) [Babaoğlu and Toueg 1993]. O TM tem como atribuições receber a(s) operação(ões) que lhe foi(ram) solicitada(s), e realizar a cooperação com os demais sítios através de seus TMs, no decurso da transação. Já o DM é o responsável por executar a(s) operação(ões) sobre o(s) dado(s) a que ela(s) faz(em) referência naquele sítio [Babaoğlu and Toueg 1993].

Ao término da transação distribuída, a atomicidade acerca das operações realizadas depende de um acordo entre os participantes engajados na transação. Não obstante, eles devem decidir de maneira única e irrevogável sobre o resultado final da transação, que será pela sua validação (*COMMIT*) - se tudo ocorreu conforme o esperado, ou pela anulação (*ABORT*) - caso algo anormal tenha ocorrido. Este acordo é obtido através de um protocolo de validação atômica, que tem por finalidade assegurar a concordância de todos os participantes quanto ao resultado final da transação, o qual será determinado de acordo com as condições locais de cada participante. Com isso, ao concluir o processamento da transação, cada participante emite um voto (SIM ou NÃO), que declara a sua capacidade em garantir a consistência e a durabilidade da sua parte na transação. E a partir dos votos recebidos é calculado um resultado, que normalmente é *COMMIT* se todos votaram SIM ou, do contrário, o resultado é *ABORT*. Formalmente, a problema da validação atômica não-bloqueante (NBAC) é definido pelas propriedades de [Babaoğlu and Toueg 1993, Guerraoui 1995]:

- **Acordo Uniforme**: todos os processos que decidem obtém um mesmo valor de decisão;
- **Validade Uniforme**: se um participante decide *COMMIT*, então todos os participantes votaram SIM;
- **Integridade**: um participante não pode reverter sua decisão após ela ter sido tomada;
- **Terminação**: todos os participantes corretos chegam a um valor de decisão;
- **Não-Trivialidade**: se todos os participantes votaram SIM e não há falhas, então a decisão é *COMMIT*.

Rachid Guerraoui em um de seus trabalhos obteve um importante resultado teórico, no qual demonstrou que não é possível resolver o NBAC em ambientes sujeitos a uma simples falta de parada [Guerraoui 1995]. Tal resultado decorre da propriedade **Não-Trivialidade**, que não pode ser plenamente satisfeita na maioria dos sistemas distribuídos que admitem falhas [Guerraoui 1995], isto é, em razão das falhas não poderem ser verificadas de maneira segura mesmo com o uso de detectores de falhas não confiáveis [Chandra and Toueg 1996]. Com isso, o autor definiu uma nova propriedade com exigências mais fracas, a qual deu origem a um novo problema denominado **Validação Atômica Fraca Não-Bloqueante** (ou *Non-Blocking Weak Atomic Commitment* - NB-WAC). O NB-WAC é especificado a partir das mesmas propriedades do NBAC, com exceção da **Não-Trivialidade**, que é enfraquecida levando em conta aspectos mais adequados, realistas e suficientes para os sistemas transacionais. Assim, ao invés de se basear em "falhas", considera-se apenas "suspeitas", e a propriedade passa a ser definida por [Guerraoui 1995]:

- **Não-Trivialidade**: se todos os participantes votaram SIM e não há suspeitas de falhas, então a decisão é *COMMIT*.

Note que, diferente do NBAC [Babaoğlu and Toueg 1993] o NB-WAC [Guerraoui 1995] pode ser reduzido ao consenso, já que os resultados obtidos para o consenso com detectores de falhas não-confiáveis podem ser estendidos ao NB-WAC. Neste

ensejo, apresentamos uma contribuição inédita para resolver o NB-WAC em um cenário onde alguns dos participantes da transação podem falhar de maneira bizantina. Em nossa proposta, devido às atribuições inerentes aos papéis de TM e DM definidos no âmbito do NBAC [Babaoğlu and Toueg 1993], consideramos o TM como participante "ativo" e o DM como participante "passivo". A partir daí, são especificados: (*i*) uma arquitetura de sistema, por meio de uma abordagem baseada em tecnologia de virtualização [Smith and Nair 2005]; e (*ii*) um protocolo que é executado sobre tal arquitetura. Neste modelo/sistema, os sítios são especificados a partir de uma abordagem denominada **máquinas virtuais gêmeas** [Dettoni et al. 2013], onde cada sítio é composto por um conjunto de máquinas virtuais, que passam a atuar como réplicas do participante ativo (o TM); e um sistema anfitrião que passa a atuar apenas como participante passivo (o DM). Tal abordagem previne que o comportamento bizantino exibido por um participante ativo faltoso afete a execução do protocolo, de maneira que apenas as faltas de parada dos sítios são percebidas.

## 3. Conceitos Preliminares

Nesta seção se apresenta os principais conceitos e aspectos teóricos deste trabalho, os quais fundamentam a proposição da solução para o NB-WAC com faltas bizantinas.

### 3.1. Visão Geral da Solução Proposta

Para facilitar a compreensão acerca da abordagem adotada para a proposição deste trabalho, na Figura 1 é ilustrada a especificação da arquitetura elaborada para prover os requisitos necessários à resolução da validação atômica fraca não-bloqueante com faltas bizantinas.

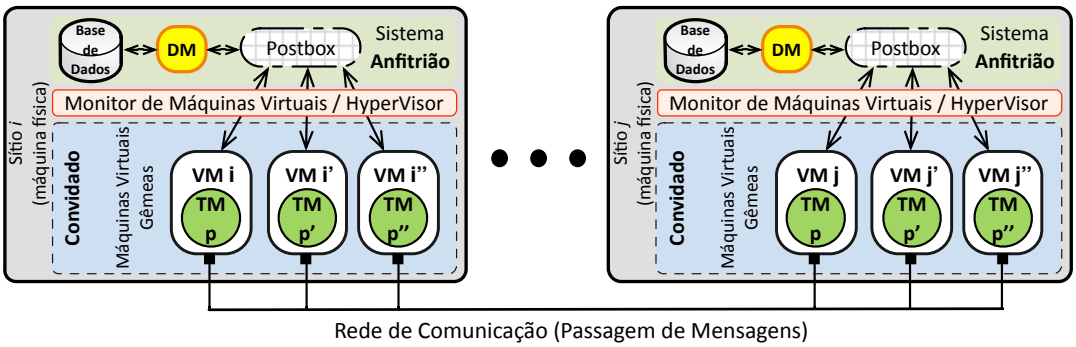



**Figura 1. Arquitetura proposta para especificar o protocolo BFT–NBWAC.**

Conforme se pode verificar na Figura 1, cada sítio engajado na transação e envolvido com o protocolo BFT–NBWAC é organizado respeitando a especificação original do NBAC (vide Seção 2), onde as entidades TM e DM estão presentes. No entanto, devido a natureza das atribuições definidas para o TM e DM na especificação do NBAC [Babaoğlu and Toueg 1993], neste trabalho define-se o TM como o participante "ativo" da transação e o DM como participante "passivo". Portanto, em nosso protocolo estes são implementados por processos distintos, que são executados separadamente e em diferentes níveis da arquitetura, diferente de todos os trabalhos verificados na literatura [Guerraoui et al. 1995, Guerraoui and Schiper 1995, Guerraoui 1995, Guerraoui et al. 1996, Abdallah and Pucheral 1999, Park et al. 2013]. Como uma das tarefas do TM é cooperar com os demais sítios engajados na transação, e a do DM é executar as operações sobre os dados, concluímos que não há a necessidade do DM em participar do acordo realizado para a terminação da transação, mas apenas os TMs é que devem fazê-lo.

Sendo assim, no BFT–NBWAC apenas o TM participa do acordo para terminação da transação, sendo executado de maneira independente do DM, que atua isoladamente na

execução das operações da transação (p.ex. escritas e leituras na base de dados). Embora as atribuições do TM e DM sejam complementares, esta separação é um meio factível para resolver o NB-WAC com faltas Bizantinas. E no âmbito deste trabalho, esta separação é concretizada por meio de virtualização [Smith and Nair 2005], em que um sítio é então representado por um **sistema anfitrião** – que implementa o DM e a base de dados utilizada no protocolo, e portanto, pode falhar apenas por parada (*crash*); e por $2f_p + 1$ [1] **sistemas convidados** – que implementam réplicas de um TM (i.e. o participante ativo) para o sítio e podem falhar de maneira bizantina. Note que, como o NB-WAC não é solúvel em um ambiente puramente bizantino [Charron-Bost and Schiper 2004], a separação proposta permite resolvê-lo a partir de um ambiente onde as falhas manifestadas são híbridas e isoladas.

Em suma, a replicação do TM é um meio factível para mascarar a(s) falta(s) bizantina(s) do participante ativo do BFT-NBWAC que ocorrem em nível de sítio. Assim, o comportamento faltoso de um TM fica confinado ao seu sítio local, e, se manifestado externamente, torna-se facilmente detectável. Na prática isto é realizado com o uso de um esquema de assinaturas de limiar [Shoup 2000], a fim de assegurar a integridade da mensagem e a autenticidade do participante ativo do sítio emissor. Com isso, sempre que uma mensagem tiver de ser enviada para outros sítios, pelo menos $f + 1$ réplicas de TM do emissor devem assinar aquela mensagem a partir do esquema de assinatura de limiar. E se porventura um participante ativo faltoso tentar deturpar o protocolo, o fará isoladamente e será detectado e desconsiderado pelos participantes ativos dos demais sítios envolvidos no protocolo.

### 3.2. Modelo de Sistema

A arquitetura da Figura 1 é especificada a partir de um modelo híbrido, onde são admitidas diferentes hipóteses acerca do modelo de falhas das entidades que a compõem. O sistema é composto por um universo de processos $\mathcal{U}$, sendo este dividido em alguns subconjuntos. O subconjunto $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ denota os sítios do sistema distribuído. Cada sítio $s_i$ é composto por um sistema **anfitrião** (*host*), sobre o qual é executado um Monitor de Máquinas Virtuais (ou *Virtual Machine Monitor* – VMM). Para cada sítio $s_i$, o VMM mantém um conjunto de **máquinas virtuais gêmeas**, que são também conhecidas como sistemas **convidados** (*guests*), tal que $\forall s_i \in \mathcal{S} : \exists\ \mathcal{G}_{s_i} = \{g_1, g_2, ..., g_n\}$. O subconjunto $\mathcal{S}$ tem cardinalidade superior a $f_s$, logo $|\mathcal{S}| \geq f_s + 1$; e os subconjuntos $\mathcal{G}_{s_i}$ têm cardinalidade $\forall s_i \in \mathcal{S} : |\mathcal{G}_{s_i}| \geq 2f_p + 1$.

Cada sistema **anfitrião** mantém uma base de dados, a qual é utilizada nas transações, sendo também o responsável por executar o processo que atua como gestor de dados (DM) no protocolo NB-WAC. Não obstante, o anfitrião mantém um monitor de máquinas virtuais, sobre o qual são executadas como sistemas **convidados** as máquinas virtuais gêmeas. Cada máquina virtual gêmea executa apenas uma réplica do processo que atua no papel de gestor da transação (TM) daquele sítio, de modo que a atuação destas réplicas se dá de maneira conjunta, para representar um único TM em cada sítio. Devido a separação das entidades em níveis distintos da arquitetura, assumimos que: (*i*) até $f_p \leq \lceil \frac{n-1}{2} \rceil$ máquinas virtuais gêmeas, **por sistema anfitrião** (no caso, os TMs), podem desviar arbitrariamente de suas especificações e então falhar de maneira bizantina; (*ii*) o sistema anfitrião pode falhar apenas por parada (*crash*), sendo que não mais de $f_s$ anfitriões (i.e. sítios) falham. Assumimos que o sistema anfitrião pode apresentar vulnerabilidades, porém, o VMM fornece isolamento entre os sistemas anfitrião e convidado(s), de tal forma que vulnerabilida-

[1] O termos $f_s$ e $f_p$ denotam o limite de faltas dos *s*ítios e dos *p*articipantes de um sítio, respectivamente.

des não podem ser exploradas através das máquinas virtuais, nem entre máquinas virtuais [Popek and Goldberg 1974].

Consideramos a existência de um esquema criptográfico de assinaturas de limiar ($n$, $k$) [Shoup 2000] baseado no algoritmo RSA [Rivest et al. 1978], onde existem $n$ chaves parciais (ou segredos) e $n$ chaves de verificação, de tal forma que pela combinação de pelo menos $k$ segredos é possível gerar uma assinatura RSA, verificável a partir da chave pública correspondente. Assumimos que o sistema anfitrião - que não é suscetível a faltas bizantinas, atua como distribuidor dos segredos aos processos TMs, sendo que ele gera uma chave privada, a qual fica em sua posse e é dividida em $n$ segredos que são distribuídos a cada uma das máquinas gêmeas, e uma chave pública. A chave pública é usada para verificar a autenticidade e integridade das mensagens assinadas, tanto por meio da chave privada usada pelo sistema anfitrião (p.ex. o DM), como daquelas geradas a partir do esquema de limiar.

A respeito das hipóteses temporais, assumimos o modelo baseado no sincronismo terminal (*eventually synchronous system*) [Dwork et al. 1988], devido às características realistas do mesmo, isto é, ele é assíncrono em grande parte do tempo, mas há períodos de estabilidade, nos quais os tempos são limitados, porém desconhecidos. O protocolo se baseia no uso de temporizadores para detectar falhas de parada dos sítios. Porém, os relógios não são sincronizados, embora os desvios entre eles sejam pequenos - uma suposição válida, considerando as tecnologias atuais. O sistema anfitrião fornece abstrações de memória compartilhada para comunicação entre os processos de um sítio, estas denominadas por *PostBox* [Stumm Júnior et al. 2010]. Na *PostBox* as escritas são *append-only* e as leituras seguem a ordem *FIFO*. Um processo pode escrever apenas em sua *PostBox*, tendo o direito de leitura nas *PostBoxes* dos demais processos do mesmo sítio. A comunicação entre os processos é realizada de duas maneiras: (*i*) entre as máquinas virtuais gêmeas de um sítio (*intra-sítio*); e (*ii*) entre os sítios (*inter-sítio*). As comunicações *inter-sítio* ocorrem via canais ponto-a-ponto confiáveis e autenticados, e a comunicação *intra-sítio* ocorre através das *PostBoxes*.

## 4. Validação Atômica Fraca Não-Bloqueante tolerante a Faltas bizantinas

Esta seção descreve a integração da arquitetura apresentada na Seção 3 (vide Figura 1) e sua relação com a proposição do protocolo que resolve o problema da NB-WAC com participantes ativos sujeitos a faltas bizantinas.

### 4.1. Princípio de Funcionamento do BFT-NBWAC

A definição dos problemas NBAC e NB-WAC [Babaoğlu and Toueg 1993, Guerraoui 1995] frisam a terminação, e não a execução a transação. Logo, o protocolo apresentado nesta seção é um protocolo para a terminação de transações distribuídas. Por esta razão, consideramos a existência de um suporte de nível superior para a execução da transação, de tal forma que a terminação da transação é realizada por meio do protocolo BFT-NBWAC. A Figura 2 ilustra as fases e os passos de comunicação realizados pelo protocolo BFT-NBWAC, onde o problema é resolvido a partir de três fases e quatro passos de comunicação *inter-sítio*. Note que toda comunicação *intra-sítio* é precedida por uma comunicação *inter-sítio* - esta última ocorre entre as réplicas de TM de um sítio, e entre os TMs e DM do mesmo sítio, ambas a partir das *PostBoxes*.

No caso da Figura 2, o sítio $S_1$ atua como o líder da transação e é o responsável por conduzir o protocolo de terminação. Na mesma figura, $f_p = 1$, onde cada sítio mantém $2f_p + 1$ réplicas de TM - ali ilustrados por $p$, $p'$ e $p''$. Na primeira fase, o líder solicita a declaração

de voto dos demais participantes sobre validar as operações realizadas na transação, a fim de verificar se eles estão preparados para tal. Na fase seguinte, os sítios não-líderes votam de acordo com a condição em que se encontram, e enviam seus votos ao sítio líder. O voto de um sítio é: *SIM* - se ele está pronto para validar as atualizações; ou *NÃO* caso não esteja pronto ou suspeite que houve falha do líder. Ainda nesta fase, o líder coleta os votos recebidos, avança para a fase três e calcula uma estimativa de resultado para a transação, baseada nas declarações recebidas dos sítios, incluindo a sua. Na sequência, o líder envia a estimativa aos demais sítios, os quais após o recebimento definem seus resultados locais a partir desta. Finalmente, todos os sítios enviam os resultados através de difusão confiável (*reliable multicast*) para subsidiar a decisão pela validação ou anulação da transação.

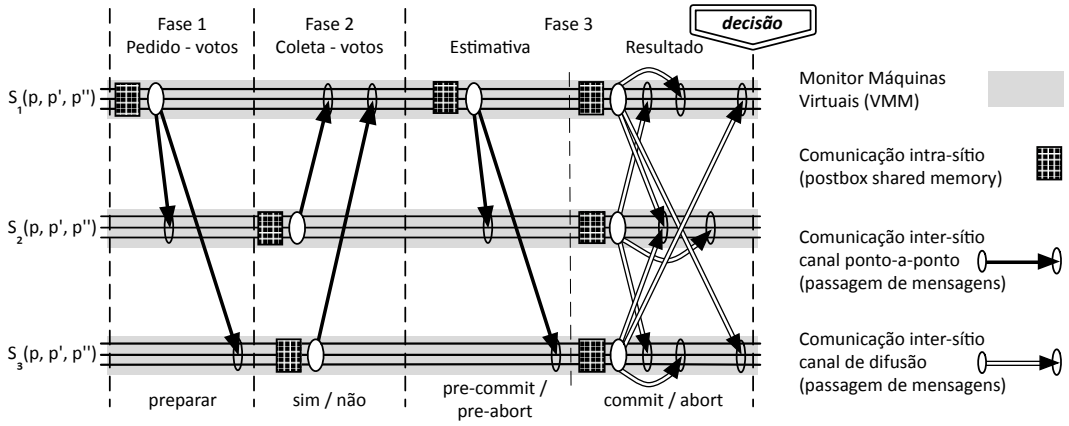



**Figura 2. Fases e passos do protocolo BFT-NBWAC.**

Note que todas as fases da Figura 2 são necessárias para respeitar as condições e propriedades do NB-WAC (vide Seção 2), onde uma decisão pela validação é legítima apenas se **todos** os sítios votaram *sim*. E para não permitir equívocos na tomada de decisão, o protocolo se orienta da seguinte maneira: (*i*) na fase 1, se um sítio não puder validar ou não tiver recebido o pedido de voto do líder em um tempo o suficiente, ele vota *não*; do contrário, ele vota *sim*; (*ii*) na fase 2, se porventura o líder não receber **todos** os votos dos sítios não-líderes, ele calculará a estimativa como *pre-abort*; ou do contrário, como *pre-commit*; (*ii*) na fase 3, o resultado será subsidiado pela estimativa recebida do líder, sendo que, caso algum sítio não receba a estimativa e suspeite de falha no líder, ele determinará seu resultado local como *abort*, o que incorrerá na anulação da transação pelo protocolo. Em todos os casos, observa-se que uma decisão pela validação ocorrerá somente se todos os sítios votaram *sim*, e em cada fase do protocolo não houve suspeita(s) de falha(s).

### 4.2. Base Algorítmica do BFT-NBWAC

Esta seção apresenta os detalhes do protocolo BFT-NBWAC, que é especificado a partir dos algoritmos das Figuras 3, 4 e 5. Cabe ressaltar que estes algoritmos descrevem apenas a implementação dos participantes ativos do NB-WAC (i.e., TMs), que compreendem as entidades responsáveis pela execução do protocolo de terminação [Babaoğlu and Toueg 1993]. Neste caso, o participante passivo (o DM) é apenas consultado pelos TMs para verificar se as operações da transação foram completamente executadas, e sem erros. O algoritmo da Figura 3 descreve as variáveis utilizadas pelo protocolo e também a tarefa principal do BFT-NBWAC, que é invocada pelos sítios participantes ao término da transação. Esta tarefa especifica o protocolo que coordena a terminação da transação, cujo propósito é obter um resultado final único e uniforme para a transação, a despeito da ocorrência de faltas bizantinas em alguns dos participantes. É digno de nota que o algoritmo que especifica a comunicação *intra-sítio* entre os TMs, e entre TMs e DM, foi omitido devido a restrições de espaço.

Os algoritmos das Figuras 3, 4 e 5 referenciam o algoritmo que implementa esta comunicação, a partir da função *intrasite_validate*. Note que sempre que há a necessidade de um sítio em se comunicar com outros sítios, esta função é invocada naqueles algoritmos. Ademais, esta função também implementa um esquema de assinaturas de limiar [Shoup 2000], que é usado para prover a integridade e autenticidade das mensagens oriundas dos participantes ativos de um sítio, sendo essencial para o funcionamento do protocolo. Além das garantias mencionadas, o uso deste esquema impede um processo faltoso de forjar assinaturas sobre mensagens espúrias. E se porventura um processo faltoso insistir no envio de mensagens inválidas, elas serão descartadas no receptor, pois toda a recepção ou entrega (para difusão confiável) de mensagens no protocolo é condicionada à verificação da assinatura, a partir da chave pública do sítio emissor (vide algoritmos das Figuras 3, 4 e 5).

**Declaration of Variables:**

1: $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ /* Sítios que participam das transações distribuídas */
2: $\mathcal{P}[|\mathcal{S}|] = \emptyset$ /* Processos gêmeos de cada sítio $S_i$ */
3: $vote : \{yes, no\}$ /* Votos possíveis para a terminação de uma transação */
4: $outcome : \{commit, abort\}$ /* Resultados possíveis para a terminação do protocolo */
5: $estimate : \{pre\text{–}commit, pre\text{–}abort\}$ /* Estimativas possíveis para uma decisão */

**Local variables:** /* Variáveis inicializadas a cada rodada do protocolo */

6: $task \leftarrow \perp$
7: $vote \leftarrow \perp$
8: $estimate \leftarrow \perp$
9: $outcome \leftarrow \perp$
10: $decision \leftarrow \perp$
11: $leader \leftarrow \perp$

**procedure** $Atomic\_Commit(T_{id})$ /* Procedimento que inicia o protocolo de validação atômica não-bloqueante */
12: $leader \leftarrow S_k : k = T_{id} \bmod |\mathcal{S}|$ /* Define o coordenador para conduzir o protocolo BFT-NBWAC */
13: **if** $p_j \in leader$ **then**
14: $task \leftarrow TaskLeader$ /* Se for o coordenador, inicia a tarefa de coordenador */
15: **fork** $task$
16: **else**
17: $task \leftarrow TaskNonLeader$ /* Se não for o coordenador, inicia a tarefa normal */
18: **fork** $task$
19: $task.join()$ /* Todos os processos aguardam o término da tarefa iniciada de acordo com o papel do sítio */

/* Phase 3 : Outcome */

20: **if** $decision = \perp$ **then**
21: $\langle S_i, \texttt{OUTCOME}, T_{id}, outcome\rangle \leftarrow partial\_sign(p_j, \langle S_i, \texttt{OUTCOME}, T_{id}, outcome\rangle)$
22: $\langle S_i, \texttt{OUTCOME}, T_{id}, outcome\rangle \leftarrow intrasite\_validate(\langle S_i, \texttt{OUTCOME}, T_{id}, outcome\rangle_{\sigma})$
23: $R\text{-}multicast((\bigcup_{S_j \in S} \mathcal{P}[S_j]), \langle S_i, \texttt{OUTCOME}, T_{id}, outcome\rangle_{\sigma_{f+1}})$
24: **wait until** [($R\text{-}deliver(\langle S_i, \texttt{OUTCOME}, T_{id}, outcome\rangle_{\sigma_{f+1}})$ from each $S_j \in \mathcal{S}$ : $verifysig(S_j, \langle S_i, \texttt{OUTCOME}, T_{id}, outcome\rangle_{\sigma_{f+1}})$) **or** ($\Delta_{S_j}$ expires)]

/* Decide */

25: $\forall S_j \in \mathcal{S}$ *such that* $\langle -, \texttt{OUTCOME}, T_{id}, -\rangle_{\sigma_{f+1}}$ *was delivered* $\Rightarrow$ $proposal[S_j] \leftarrow \{delivered(\langle S_j, \texttt{OUTCOME}, T_{id}, outcome\rangle)$ from $S_j\}$
26: **if** $(|(\bigcup_{S_j \in S} proposal[S_j])| = |\mathcal{S}|) \wedge (\nexists\, propose \in (\bigcup_{S_j \in S} proposal[S_j]) : \forall S_j \Rightarrow proposal[S_j] = abort)$ **then**
27: $decision \leftarrow commit$
28: **else**
29: $decision \leftarrow abort$
30: $decide(S_i, decision)$

**upon** $R\text{-}deliver(\langle S_i, \texttt{DECISION}, T_{id}, abort\rangle_{\sigma_{f+1}})$ from some $p_k$ of $S_j \in \mathcal{S} : verifysig(S_j, \langle S_i, \texttt{DECISION}, T_{id}, abort\rangle_{\sigma_{f+1}})$
31: $decision \leftarrow abort$ /* Decisão antecipada caso algum sítio tenha declarado o voto como *não* */
32: $task.terminate()$ /* Encerra a tarefa iniciada de acordo o papel do participante */

**Figura 3. Tarefa principal do protocolo BFT-NBWAC – sítio $s_i$ – processo $p_j$.**

O algoritmo especificado pela Figura 3 é iniciado quando um participante conclui sua parte no processamento da transação, onde invoca o procedimento *Atomic_Commit()* para realizar a validação atômica da transação em lide. No caso, um processo $p_j$ que implementa uma réplica de TM do sítio $s_i$, primeiramente inicializa as variáveis para aquela rodada do protocolo nos valores padrões (linhas 6 a 11), e posteriormente define o líder da transação baseado nos identificadores dos sítios e da transação em questão (linha 12). Se o processo

pertence ao sítio definido como líder, ele inicia uma tarefa para executar as atividades de líder no protocolo BFT-NBWAC (linhas 13 a 15), cujo algoritmo é especificado na Figura 4. De outro modo, se o processo for apenas um participante, ele inicia uma tarefa para realizar as atividades de não-líder no protocolo (linhas 16 a 18), especificado pelo algoritmo da Figura 5. O código entre as linhas 19-30 compreende à última fase do protocolo - a de decisão, a qual é comum a todos os participantes independente do papel exercido. Após o lançamento das tarefas de acordo com os papéis dos sítios, os processos aguardam o término de suas respectivas tarefas e retornam para a tarefa principal (função *task.join()* da linha 19 - Figura 3), a fim de obter uma decisão e concluir a rodada do protocolo. No que segue, o protocolo será explicado de acordo com as fases do protocolo ilustradas pela Figura 2.

**TaskLeader:** /* Apenas o participante líder da transação executa a tarefa - Phase 1 */

1: $\langle S_i, \texttt{REQUEST-VOTE}, T_{id}\rangle \leftarrow partial\_sign(p_j, \langle S_i, \texttt{REQUEST-VOTE}, T_{id}\rangle)$
2: $\langle S_i, \texttt{REQUEST-VOTE}, T_{id}\rangle \leftarrow intrasite\_validate(\langle S_i, \texttt{REQUEST-VOTE}, T_{id}\rangle_\sigma)$
3: $send(p_j, \langle S_i, \texttt{REQUEST-VOTE}, T_{id}\rangle_{\sigma_{f+1}})$ to $\forall p_k \in ((\bigcup_{S_j \in S} \mathcal{P}[S_j]) \setminus \mathcal{P}[S_i])$

/* Phase 2 */

4: **wait until** $[(received(\langle S_j, \texttt{VOTE}, T_{id}, vote\rangle_{\sigma_{f+1}})$ from some $p_k$ of each $S_j \in (\mathcal{S} \setminus \{S_i\})$ :
$verifysig(S_j, \langle S_j, \texttt{VOTE}, T_{id}, vote\rangle_{\sigma_{f+1}}))$ **or** $(\Delta_{S_j}$ expires)]
5: **for each** $S_j \in (\mathcal{S} \setminus \{S_i\})$ **do**
6: $\quad msgs[S_j] \leftarrow \{received(\langle S_j, \texttt{VOTE}, T_{id}, vote\rangle_{\sigma_{f+1}})$ from some $p_j : p_j \in S_j\}$
7: $\quad vote[S_j] \leftarrow compute\_site\_vote(msgs[S_j])$
8: $vote \leftarrow query\_Data\_Manager(T_{id})$
9: $\langle S_i, \texttt{VOTE}, T_{id}, vote\rangle \leftarrow partial\_sign(p_j, \langle S_i, \texttt{VOTE}, T_{id}, vote\rangle)$
10: $msgs[S_i] \leftarrow intrasite\_validate(\langle S_i, \texttt{VOTE}, T_{id}, vote\rangle_\sigma)$
11: $vote[S_i] \leftarrow msgs[S_i]$

/* Phase 3 : Estimate */

12: **if** $(|(\bigcup_{S_j \in S} vote[S_j])| = |\mathcal{S}|) \wedge (\nexists\, vote \in (\bigcup_{S_j \in S} vote[S_j]) : \forall S_j \Rightarrow vote[S_j] = no)$ **then**
13: $\quad estimate \leftarrow pre\text{–}commit$
14: **else**
15: $\quad estimate \leftarrow pre\text{–}abort$
16: $\langle S_i, \texttt{ESTIMATE}, T_{id}, estimate, msgs[]\rangle \leftarrow partial\_sign(p_j, \langle S_i, \texttt{ESTIMATE}, T_{id}, estimate, msgs[]\rangle)$
17: $\langle S_i, \texttt{ESTIMATE}, T_{id}, estimate, msgs[]\rangle \leftarrow intrasite\_validate(\langle S_i, \texttt{ESTIMATE}, T_{id}, estimate, msgs[]\rangle_\sigma)$
18: $send(p_j, \langle S_i, \texttt{ESTIMATE}, T_{id}, estimate, msgs[]\rangle_{\sigma_{f+1}})$ to $\forall p_k \in ((\bigcup_{S_j \in S} \mathcal{P}[S_j]) \setminus \mathcal{P}[S_i])$
19: $outcome \leftarrow (estimate = pre\text{–}commit)\ ?\ commit : abort$

**Figura 4. Tarefa executada pelos TMs do sítio líder – sítio $s_i$ – processo $p_j$.**

**Fase 1** O sítio líder dá início ao protocolo, sendo que os demais sítios aguardam a manifestação do líder para iniciá-lo. Cada processo réplica de TM do sítio líder, quando adentra na primeira fase do protocolo solicita aos participantes dos demais sítios, através de seus respectivos TMs, uma declaração de intenção em validar as operações da transação. Para tanto, cada processo TM do líder envia uma mensagem `REQUEST-VOTE` assinada através do esquema de limiar, que é gerada por pelo menos $f+1$ TMs daquele sítio (linhas 1 a 3 - Figura 4). Os processos TMs dos sítios não-líderes - que estão à espera da manifestação do líder (linha 1 - Figura 5), quando recebem uma mensagem `REQUEST-VOTE` contendo uma assinatura válida do sítio líder ou; de outro modo, quando algum TM de um sítio não-líder não recebe a manifestação inicial do sítio líder em um tempo suficiente, um temporizador é expirado; e após um destes eventos a fase 2 do protocolo é iniciada.

**Fase 2** Nesta fase, os TMs líderes aguardam pelos votos solicitados aos TMs não-líderes na fase anterior (linha 4 - Figura 4), enquanto os TMs não-líderes, ao iniciar esta fase do protocolo fazem o seguinte: (*i*) se o temporizador expirou e não receberam o pedido de voto do sítio líder, eles passam a suspeitar de falha do líder, e declaram seu voto como *não* (linhas 2 e 3 - Figura 5) - note que, isso pode ocorrer apenas se o sítio líder falhar por parada, pois todas as réplicas corretas do TM líder enviaram a mensagem na fase anterior; (*ii*) se recebem uma mensagem com o pedido de voto assinada corretamente pelo sítio líder, eles consultam ao DM de seu sítio local, a fim de verificar se sua parte da transação logrou êxito (linhas 4 e

5 - Figura 5). Esta consulta ocorre através de comunicação *intra-sítio* entre os TMs e o DM de um mesmo sítio, em que a função *query_Data_Manager* retorna *SIM* se houve êxito; ou *NÃO* caso algo impeça a transação de ser validada. E baseado no retorno do DM, os TMs dos sítios não-líderes preparam suas declarações de voto através da mensagem VOTE, com os identificadores do sítio e da transação para a qual o voto está sendo emitido, bem como o voto propriamente dito. A mensagem é submetida ao processo de validação *intra-sítio* já mencionado e, uma vez assinada ela é enviada ao líder (linhas 6 a 9 - Figura 5), e os TMs não-líderes passam a aguardar por uma estimativa de resultado proveniente do sítio líder, para então dar sequência à próxima fase do protocolo (linha 14 - Figura 5).

**TaskNonLeader:** /* Todos os participantes não-líderes da transação executam a tarefa - Phase 1 */

1: **wait until** [($received(\langle S_j, \texttt{REQUEST-VOTE}, T_{id}\rangle_{\sigma_{f+1}})$ from some $p_k$ of $S_j : S_j = leader \wedge verifysig(msg.S_j, \langle msg\rangle_\sigma)$) **or** ($\Delta_{S_j}$ expires)]

/* Phase 2 */

2: **if** $\Delta_{S_j}$ expired **then**
3: $vote \leftarrow no$
4: **else**
5: $vote \leftarrow query\_Data\_Manager(T_{id})$
6: **if** $vote = yes$ **then**
7: $\langle S_i, \texttt{VOTE}, T_{id}, vote\rangle \leftarrow partial\_sign(p_j, \langle S_i, \texttt{VOTE}, T_{id}, vote\rangle)$
8: $\langle S_i, \texttt{VOTE}, T_{id}, vote\rangle \leftarrow intrasite\_validate(\langle S_i, \texttt{VOTE}, T_{id}, vote\rangle_\sigma)$
9: $send(p_j, \langle S_i, \texttt{VOTE}, T_{id}, site\_vote\rangle_{\sigma_{f+1}})$ to $\forall p_k \in leader$
10: **else**
11: $\langle S_i, \texttt{DECISION}, T_{id}, abort\rangle \leftarrow partial\_sign(p_j, \langle S_i, \texttt{DECISION}, T_{id}, abort\rangle)$
12: $\langle S_i, \texttt{DECISION}, T_{id}, abort\rangle \leftarrow intrasite\_validate(\langle S_i, \texttt{DECISION}, T_{id}, abort\rangle_\sigma)$
13: $R\text{-}multicast((\bigcup_{S_j \in S} \mathcal{P}[S_j]), \langle S_i, \texttt{DECISION}, T_{id}, abort\rangle_{\sigma_{f+1}})$

/* Phase 3 : Estimate */

14: **wait until** [($received(\langle S_j, \texttt{ESTIMATE}, T_{id}, estimate, msgs[]\rangle_{\sigma_{f+1}})$ from some $p_k$ of $S_j$ : $S_j = leader \wedge verifysig(S_j, \langle S_j, \texttt{ESTIMATE}, T_{id}, estimate, msgs[]\rangle_{\sigma_{f+1}})$) **or** ($\Delta_{S_j}$ expires)]
15: **if** $\Delta_{S_j}$ expired **then**
16: $outcome \leftarrow abort$
17: **else**
18: $msg \leftarrow \{received(\langle S_j, \texttt{ESTIMATE}, T_{id}, estimate, msgs[]\rangle_{\sigma_{f+1}})$ from $p_k \in leader\}$
19: $outcome \leftarrow (msg.estimate = pre\text{–}commit)\ ?\ commit : abort$
20: **for all** $S_j \in \mathcal{S}$ **do**
21: **if** $\neg(verifysig(S_j, \langle msg.msgs[S_j]\rangle)) \vee \neg(compute\_site\_vote(msg.msgs[S_j]) = yes)$ **then**
22: $outcome \leftarrow abort$
23: **break**

**Figura 5. Tarefa executada pelos TMs dos sítios não-líderes – sítio $s_i$ – processo $p_j$.**

Um TM do sítio líder, ao receber as mensagens contendo a declaração de voto de cada um dos sítios não-lideres, armazena as mensagens recebidas e computa os respectivos votos (linhas 5 a 7 - Figura 4). E como o TM líder também é participante da transação, ele também consulta seu DM local, para subsidiar sua declaração de voto. Nos mesmos moldes, a mensagem de voto do sítio líder é assinada pelo esquema de limiar pelos TMs líderes (linhas 8 e 9 - Figura 4); que posteriormente armazenam a mensagem assinada junto aos demais votos recebidos (conjuntos $msgs[S_i]$ e $vote[S_i]$ – linhas 10 e 11 da Figura 4).

**Fase 3** Ao adentrar nesta fase, o primeiro passo dos TMs líderes é verificar se eles receberam os votos de todos os participantes e se dentre os que foram recebidos não houve nenhuma declaração *não* - ambas as condições necessárias para decidir pela validação (linha 12 - Figura 4). Se estas condições forem satisfeitas, é um indício de que **todos** estão de acordo e não houve falhas, e então a transação pode ser validada. Neste caso, cada TM líder calcula a estimativa como *pre-commit*; do contrário, se uma das condições não for satisfeita a estimativa será *pre-abort*. Na seqüência, a mensagem de estimativa é validada junto aos demais TMs líderes e posteriormente é enviada aos sítios não-líderes (linhas 16 a 18 - Figura 4). Note que junto à estimativa, o líder envia aos demais sítios todas as mensagens que subsidiaram a mesma. De posse da estimativa, o líder determina o seu resultado para a transação e o guarda para a etapa final do protocolo (linha 19 - Figura 4). De outro modo, os TMs

não-líderes iniciam esta fase em duas situações: quando receberem a estimativa do líder; ou quando do esgotamento do temporizador, caso não tenham recebido a estimativa dentro de um período estabelecido pelo protocolo (linha 14 - Figura 5). A partir do recebimento da estimativa, os TMs não-líderes determinam seus resultados como: *commit* se recebeu *pre-commit*; *abort* se recebeu *pre-abort*. E de maneira conservadora, após definirem os resultados os mesmos TMs verificam as mensagens que resultaram naquela estimativa (p.ex. os votos), as quais vieram junto à mensagem da estimativa (linhas 20 a 23 - Figura 5). Se tal verificação constatar o recebimento de alguma mensagem inválida, ou uma declaração de voto *não*, o resultado será alterado para *abort*. Se porventura o temporizador se esgotar e a estimativa não tiver sido recebida, eles passam a suspeitar de falha do líder e declaram implicitamente seu resultado como *abort* (linhas 15 e 16 - Figura 5).

Quando os TMs de todos os sítios concluem suas tarefas executadas para o protocolo, eles retornam ao algoritmo da tarefa inicial (Figura 3), o qual ficou bloqueado até a conclusão das respectivas tarefas (função *task.join()* da linha 19 - Figura 3). O código das linhas 20 a 30 compreende à última etapa do protocolo, a **decisão**, que é executada por todos os TMs. Primeiramente eles verificam se até aquele ponto não houve nenhuma decisão para a transação, e em não tendo havido, cada TM pega o resultado que obteve na fase anterior e envia aos TMs dos demais sítios através de *difusão confiável* [Hadzilacos and Toueg 1994] (linhas 20 a 23 - Figura 3). Logo após, os TMs de cada sítio aguardam pela entrega das mensagens contendo os resultados obtidos por todos os demais sítios (linha 24 - Figura 3).

Note que a mensagem do resultado de um sítio não será entregue somente no caso do sítio falhar por parada, e antes do envio, pois a mensagem é enviada através de difusão confiável. Ao concluir o passo da linha 24, os TMs verificam se as mensagens dos resultados de cada sítio foram entregues, e se dentre elas não há nenhuma que visa a anulação da transação (linha 26 - Figura 3). Neste caso, uma decisão legítima será pela validação (*commit*), já que ambas as condições necessárias para a validação da transação foram satisfeitas. Do contrário, se uma das condições não for satisfeita a decisão será pela anulação (*abort*). Ao invocar a função *decide*, cada TM dos sítios insere o valor de decisão em sua *PostBox* para orientar o DM a realizar as alterações ou descartá-las da base de dados, que o faz de acordo com a decisão verificada na *PostBox* de pelo menos $f + 1$ TMs daquele sítio. Isto é o que garante a não-reversão da decisão e o acordo uniforme no sítio, a despeito de faltas.

É importante salientar que o protocolo admite decisões antecipadas, tal como previsto na especificação do NBAC [Babaoğlu and Toueg 1993]. Esta admissão é plausível, pois de acordo com as propriedades do NBAC, a anulação pode ser vista como a única decisão aceitável e legítima. Em nosso protocolo tal condição é especificada nas linhas 10 a 13 do algoritmo da Figura 5, onde um participante decide de maneira unilateral pela anulação da transação, se por alguma razão não puder validá-la. No caso, os TMs daquele sítio enviam aos demais sítios uma mensagem `DECISION` através da primitiva de *difusão confiável*, para assegurar a entrega desta pelos demais sítios. A entrega da mensagem é vista nas linhas 31 e 32 da tarefa principal do protocolo (algoritmo da Figura 3), onde os participantes, independentemente do papel que estão a desempenhar, decidem de maneira definitiva pela anulação e encerram as respectivas tarefas que haviam iniciado. Por conseguinte, cada participante retorna para a tarefa principal e efetiva a decisão recebida (linhas 19, 20 e 30 - Figura 3).

Salientamos que as provas de correção para os algoritmos apresentados foram omitidas devido à restrições de espaço. Todavia, as provas e também o algoritmo que especifica a comunicação *intra-sítio* estão disponíveis na versão estendida do artigo [Luiz et al. 2013].

## 5. Avaliação do Protocolo

No intuito de verificar a eficiência da solução proposta, realizamos uma comparação analítica dos resultados obtidos para o BFT-NBWAC e de alguns protocolos de validação atômica encontrados na literatura. Esta avaliação é realizada nos mesmos moldes dos trabalhos relacionados [Guerraoui and Schiper 1995, Guerraoui et al. 1996, Abdallah and Pucheral 1999, Greve and Narzul 2002], onde as medidas de desempenho verificadas são aquelas recorrentes na literatura, pela análise de cenários favoráveis (p. ex. sem falhas), onde todos os processos votam *sim*. A Tabela 1 descreve os resultados verificados para o BFT-NBWAC, para o clássico 3PC (*Three Phase Commit*) para os trabalhos denominados: DNB-AC [Guerraoui and Schiper 1995], MD3PC [Guerraoui et al. 1996], NB-2PC [Greve and Narzul 2002] e ANB-CLL [Abdallah and Pucheral 1999].

**Tabela 1. Comparação entre alguns protocolos de validação atômica.**

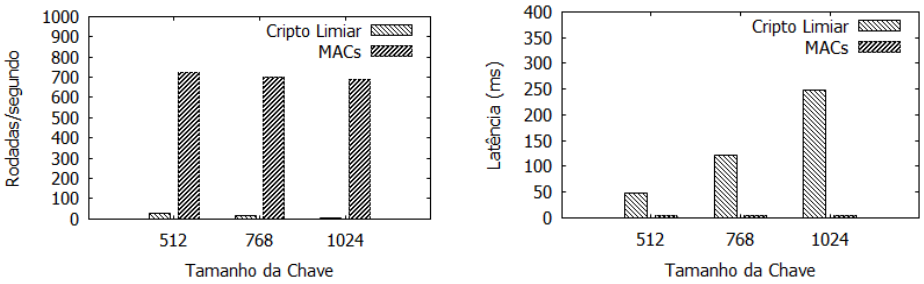
| Protocolo | # Passos | # Mensagens | Resiliência | # Participantes | # Processos | Faltas |
|---|---|---|---|---|---|---|
| **3PC** | 5 | $5n$ | $-$ | $-$ | $-$ | parada |
| **DNB-AC** | 3 | $2n^2+n$ | $f<n/2$ | $2f+1$ | $2f+1$ | parada |
| **MD3PC** | 3 | $3nf+3n$ | $f<n/2$ | $2f+1$ | $2f+1$ | parada |
| **NB-2PC** | 3 | $2nf+3n$ | $f<n$ | $f+1$ | $f+1$ | parada |
| **ANB-CLL** | 2 | $n^2+n$ | $f<n/2$ | $2f+1$ | $2f+1$ | parada |
| **BFT-NBWAC** | 4 | $3(n-1)+n^2$ | $f<n$ | $f_s+1$ | $2f_pf_s+2f_p+f_s+1$ | bizantina |

Para os resultados exibidos na Tabela 1, dois pontos merecem destaque. O ANB-CLL obtém a terminação em apenas dois passos, já que ele elimina a fase de votação, por se basear nas propriedades locais dos protocolos de serialização dos participantes, para obter o resultado da transação. Em decorrência disso, o protocolo tem restrições de uso apenas em ambientes onde é possível conhecer a organização local dos participantes. O NB-2PC apresenta a melhor resiliência, em razão dele permitir decisões antecipadas quando o ambiente reúne condições favoráveis para tal. No caso, a decisão é delegada a um subconjunto de processos $Set_{NB} \leq N$ previamente escolhido (p.ex. os mais confiáveis). Notadamente, os protocolos avaliados apresentam maior eficiência em relação ao BFT-NBWAC, primeiro por tolerarem apenas faltas de parada; e segundo, porque todos eles são especificados a partir de um serviço de *consenso* subjacente, onde o acordo é resolvido com o auxílio de um detector de falhas da classe $\diamond\mathcal{S}$. E isto implica que as mensagens e passos incidentes do acordo realizado através do *consenso*, não são refletidos no cômputo das medidas destes trabalhos.

Por outro lado, o BFT-NBWAC requer um número menor de participantes do que alguns protocolos avaliados, devido aos demais considerarem o sítio como participante. Em nosso caso, como um participante é implementado no sítio por $2f+1$ réplicas, um número maior de processos é requerido para tolerar as faltas bizantinas. Um ponto que também merece destaque é que, diferente do que ocorre nos demais trabalhos, onde o custo associado à difusão confiável usada para propagar a decisão do protocolo é omitido, o número de mensagens apresentado para o BFT-NBWAC representa o número total de mensagens, considerando a difusão confiável. Já o número de passos requerido decorre do fato de que em nosso protocolo o acordo é baseado no líder, pois para que os processos possam decidir seguramente é necessário que todos tenham ciência do voto de todos. Se optássemos por não fazer desta maneira, um número maior de mensagens seria requerido para o protocolo.

E finalmente, a fim de realizarmos uma prova de conceito, implementamos um protótipo simplificado do BFT-NBWAC. Para tal, adotamos a linguagem Java e as bibliotecas *ThreshSig* (*http://threshsig.sourceforge.net*) e *Chronicle* (*https://github.com/peter-lawrey/Java-Chronicle*) para implementar o esquema de assinaturas de limiar e as *PostBoxes*, respectivamente. Para os canais de comunicação, utilizamos os *SocketChannels* dis-

poníveis na API Java NIO. Os experimentos foram realizados em um ambiente de rede local, composto por máquinas com as seguintes configurações: três máquinas com 8GB de RAM; 1 Interface Ethernet Gigabit, sendo 1 (uma) com processador Intel® Core™I5-2410M 2.30GHz; 1 (uma) com processador Intel® Core™I3-2310M 2.10GHz e 1 (uma) com processador Intel® Core™2 Duo P8600 2.40GHz. As máquinas foram interligadas em uma rede cabeada a partir de um roteador Cisco® DPC3925. Em cada uma das máquinas foi instalado o sistema operacional Debian 7.0 (”wheezy”) em nível de anfitrião, sobre o qual foram instalados o *VMware Workstation* 10 como VMM e o MySQL 5.5.33 como base de dados. Em cada VMM foram criadas 3 máquinas virtuais (ou $2f + 1$, onde $f = 1$) com o mesmo sistema operacional (Debian 7.0), sendo reservados 1.5GB de RAM para cada uma.




(a) Vazão do protocolo. (b) Latência média.

**Figura 6. Avaliação experimental do protocolo BFT-NBWAC.**

Os experimentos foram realizados a partir da especificação original do BFT-NBWAC e de uma implementação onde o esquema de assinaturas de liminar foi substituído por autenticadores - códigos de autenticação de mensagens (MACs), ambos com variação no tamanho das chaves. O intuito da avaliação foi o de analisar o custo incidente da rodada (*rounds*) do acordo para a terminação da transação. Para tanto, cada sítio foi carregado com um arquivo contendo 10.000 transações identificadas e não concorrentes, a partir das quais as instâncias do acordo foram iniciadas nos sítios, com um intervalo de 10ms. Observamos que o maior impacto no desempenho do BFT-NBWAC foi causado pelo mecanismo de assinaturas utilizado (Figura 6(a)). O uso de assinaturas de limiar incidiu em uma degradação demasiada do desempenho. Conforme os resultados reportados na Figura 6(b), o tempo médio consumido por uma assinatura de limiar é drasticamente maior em ordens de magnitude do que o consumido por códigos de autenticação de mensagens. Em nossos experimentos, o tempo médio consumido para a geração de uma assinatura de limiar em relação aos código de autenticação de mensagens foi de aproximadamente 15 vezes maior para chaves de 512 bits, e de aproximadamente 45 e 60 vezes para chaves de 768 bits e 1024 bits, respectivamente.

## 6. Considerações Finais

Este artigo apresentou uma arquitetura e um protocolo para resolução do problema da validação atômica não-bloqueante em sistema distribuídos, a fim de oferecer uma solução em cenários onde os participantes da transação podem falhar de maneira bizantina. A solução foi desenvolvida a partir de uma abordagem pragmática e realista, com o uso de tecnologia de virtualização, sendo adotado um modelo híbrido de faltas, com diferentes hipóteses para cada nível da arquitetura do sistema. À primeira vista, acreditamos que os resultados trazidos por este trabalho representam um passo inicial para que soluções práticas, voltadas para ambiente reais, possam vir a ser desenvolvidas. Também, por ser realista e baseada em tecnologia moderna, a ideia proposta pode ser vista como um avanço para a área de algoritmos distribuídos tolerante a faltas. Ademais, a proposta é passível de adaptação para uma arquitetura modular, com vista a resolver outros problemas de computação distribuída.

## Referências


Abdallah, M. and Pucheral, P. (1999). A low-cost non-blocking atomic commitment protocol for asynchronous systems. In *Proceedings of the 11th International Conference on Parallel and Distributed Computing and Systems*, pages 911–918.

Babaoğlu, O. and Toueg, S. (1993). Non-blocking atomic commitment. In Mullender, S., editor, *Distributed systems*, pages 147–168. 2 edition.

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

Charron-Bost, B. and Schiper, A. (2004). Uniform consensus is harder than consensus. *Journal of Algorithms*, 51(1):15–37.

Dettoni, F., Lung, L. C., Correia, M., and Luiz, A. F. (2013). Byzantine fault-tolerant state machine replication with twin virtual machines. In *Proceedings of the 8th Symposium on Computers and Communications*, pages 398–403.

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

Greve, F. and Narzul, J.-P. L. (2002). Um protocolo de validação atômica não-bloqueante eficiente. In *Anais do XX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 309–323.

Guerraoui, R. (1995). Revisiting the relationship between non-blocking atomic commitment and consensus. In *Proceedings of the 9th International Workshop on Distributed Algorithms*, pages 87–100.

Guerraoui, R., Larrea, M., and Schiper, A. (1995). Non blocking atomic commitment with an unreliable failure detector. In *Proceedings of the 14th Symposium on Reliable Distributed Systems*, pages 41–50.

Guerraoui, R., Larrea, M., and Schiper, A. (1996). Reducing the cost for non-blocking in atomic commitment. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 692–697.

Guerraoui, R. and Schiper, A. (1995). The decentralized non-blocking atomic commitment protocol. In *Proceedings of the 7th Symposium on Parallel and Distributeed Processing*, pages 2–9.

Hadzilacos, V. (1990). On the relationship between the atomic commitment and consensus problems. In Simons, B. and Spector, A., editors, *Fault-Tolerant Distributed Computing*, volume 448 of *Lecture Notes in Computer Science*, pages 201–208.

Hadzilacos, V. and Toueg, S. (1994). A modular approach to the specification and implementation of fault-tolerant broadcasts and related problems. Technical Report TR 94-1425, Department of Computer Science, Cornell University.

Stumm Júnior, V., Lung, L. C., Correia, M., Fraga, J. S., and Lau, J. (2010). Intrusion tolerant services through virtualization: A shared memory approach. In *Proceedings of the 24th International Conference on Advanced Information Networking and Applications*, pages 768–774.

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

Luiz, A. F., Lung, L. C., Correia, M., and Stumm Júnior, V. (2013). Validação atômica não-bloqueante com faltas bizantinas. Relatório Técnico, PGEAS/DAS/UFSC. Disponível em `http://www.das.ufsc.br/~aldelir/report2013-01.pdf`.

Park, S.-H., Lee, J.-Y., and Yu, S.-C. (2013). Non-blocking atomic commitment algorithm in asynchronous distributed systems with unreliable failure detectors. In *Proceedings of the 10th International Conference on Information Technology*, pages 33–38.

Pease, M., Shostak, R., and Lamport, L. (1980). Reaching agreement in the presence of faults. *Journal of ACM*, 27(2):228–234.

Popek, G. J. and Goldberg, R. P. (1974). Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7):412–421.

Rivest, R. L., Shamir, A., and Adleman, L. M. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Shoup, V. (2000). Practical threshold signatures. In *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques*, pages 207–220.

Smith, J. E. and Nair, R. (2005). The architecture of virtual machines. *Computer*, 38(5):32–38.

# Lidando com Transações Interativas em um STM Tolerante a Faltas Bizantinas

**Tulio Alberton Ribeiro**[1], **Lau Cheuk Lung**[1], **Hylson Vescovi Netto**[1,2]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
88040-900 – Trindade – Florianópolis – Santa Catarina – Brasil

[2]Instituto Federal Catarinense (IFC), Campus Blumenau – Santa Catarina – Brasil

{tulio.ribeiro,hylson.vescovi}@posgrad.ufsc.br, lau.lung@inf.ufsc.br

***Abstract.*** *Recently, researchers have shown an increased interest in concurrency control using distributed Software Transactional Memory (STM). However, there has been little discussion about certain types of fault tolerance, such as Byzantine Fault Tolerance (BFT), for kind of systems. The focus of this paper is on tolerating byzantine faults on optimistic processing of interactive and declared transactions using STM. The result is an algorithm named Mesobi. The processing of a transaction runs with an optimistic approach, benefiting from the high probability of messages being delivered in order when using Reliable Multicast on a local network (LAN). The protocol performs better when messages are delivered ordered. In case of a malicious replica or out-of-order messages, the Byzantine protocol is initiated.*

***Resumo.*** *Pesquisadores têm mostrado aumento de interesse em utilizar Memória Transacional em Software (STM) como mecanismos de controle de concorrência. Ao utilizar STM, o programador não necessita tratar com mecanismos explícitos de controle de concorrência como algoritmos de exclusão mútua. Entretanto, existem poucas discussões sobre tolerância a falta quando o assunto é STM, principalmente tolerância a faltas Bizantinas. O foco deste artigo é tolerância a faltas Bizantinas no contexto de STM utilizando uma abordagem otimista. Propomos o Mesobi, um protocolo que executa transações interativas (on-line) e pré-declaradas em simultâneo utilizando um mecanismo otimista no processamento das mesmas. O protocolo tem bom desempenho quando operando de forma otimista. Em caso de réplicas maliciosas ou mensagens fora de ordem, o protocolo Bizantino precisa ser iniciado.*

## 1. Introdução

Os sistemas de Memória Compartilhada Distribuída (DSM - Distributed Shared Memory) abstraem os mecanismos de comunicação remota do desenvolvedor da aplicação [Protic et al. 1996]. As DSM podem ser descritas como um espaço de endereçamento virtual que é compartilhado por um conjunto de processadores [Li and Hudak 1989]. Nesses sistemas, os processadores acessam cada endereço de memória como se fosse local. As DSM por sua vez, permitem o compartilhamento coerente de dados através de acesso uniforme às escritas e leituras (sincronização baseada em travas - *locks*). Como uma alternativa para a sincronização baseada em travas, surgiram as Memórias Transacionais em Software (STM - Software Transactional Memory). As STM permitem acesso uniforme

às escritas e leituras através de transações. As transações em STM são semelhantes as transações em bancos de dados, porém, sem a propriedade durabilidade.

Pesquisadores têm mostrado um aumento de interesse no controle de concorrência utilizando STM, como pode ser visto no trabalhos $D^2STM$ [Couceiro et al. 2009], *RAM-DUR* [Sciascia and Pedone 2012], *SPECULA* [Peluso et al. 2012], *OSARE* [Palmieri et al. 2011], *SCert* [Carvalho et al. 2011], *AGGRO* [Palmieri et al. 2010], *STR* [Romano et al. 2010], Granola [Cowling and Liskov 2012] e *Zhang* [Zhang and Zhao 2012]. Ao utilizar STM, os programadores não precisam lidar com mecanismos explícitos de controle de concorrência (como monitores, *locks* ou semáforos). Ao invés disso, apenas precisam delinear quais objetos necessitam ser tratados como concorrentes, através do uso de transações. Com isso, é possível focar mais na lógica da aplicação do que nos mecanismos explícitos de controle de concorrência, que é feito pela STM.

A maioria dos trabalhos citados acima, tratam apenas tolerância a faltas de parada (*crash*). Entre eles, *AGGRO*, *SPECULA*, *OSARE* e *STR* utilizam a alta probabilidade das mensagens serem entregues em ordem pela rede [Pedone and Schiper 1998, Kemme et al. 2003], mas necessitam que a ordem final de entrega de mensagens seja definida para concluir sua execução. Por sua vez, somente o trabalho de [Zhang and Zhao 2012] menciona tolerância a faltas Bizantinas no contexto de STM. O modelo proposto por Zhang não utiliza a alta probabilidade das mensagens serem entregues em ordem pela rede, e aborta transações somente leitura.

Este trabalho apresenta o Mesobi, uma arquitetura de STM tolerante a faltas Bizantinas que executa, em simultâneo, tanto transações interativas (ou online) quanto transações pré-declaradas utilizando um mecanismo otimista no processamento das mesmas. O protocolo proposto no Mesobi é baseado na execução paralela de transações não conflitantes, semelhante ao proposto por [Kotla and Dahlin 2004]. O presente trabalho é uma evolução do OB-STM [Ribeiro et al. 2013] que lida somente com transações pré-declaradas.

Transações interativas são um recurso útil para aplicações que seguem uma lógica dependente de resultados de operações anteriores. Em ambientes onde é necessária a confidencialidade das operações a lógica do negócio deve estar junto ao cliente. Existem algumas abordagens que permitem a execução confidencial de algumas operações, como média, desvio padrão e regressão; porém operações mais complexas ainda não podem ser feitas na réplica [Naehrig et al. 2011]. Lidar com transações interativas é um problema difícil de se resolver quando se considera a possibilidade de faltas bizantinas no sistema. Processos bizantinos podem de forma maliciosa induzir ao erro processos corretos, através de comportamento arbitrário. O comportamento arbitrário ou malicioso, pode causar a perda de consistência ao enviar respostas diferentes a processos corretos. Protocolos que não consideram tolerância a faltas maliciosas podem ficar estagnados na presença de processos maliciosos; por exemplo, não enviando ou atrasando respostas para outras réplicas que estejam esperando pelas mesmas. Torna-se então importante um protocolo que permita a interação segura entre clientes e réplicas, garantindo que as transações serão corretamente executadas a despeito de faltas arbitrárias.

O Mesobi se beneficia da alta probabilidade das mensagens serem entregues em

ordem pela rede usando IP-Multicast [Kemme et al. 2003, Pedone and Schiper 1998]; somente em caso de mensagens fora de ordem ou réplicas maliciosas é que o protocolo Bizantino inicia, não necessitando que a ordem final seja definida para a execução das transações no caso otimista. O Mesobi foi construído utilizando a JVSTM e preserva suas propriedades *weak atomicity* e *opacity*. A propriedade *weak atomicity* garante atomicidade entre transações, e a propriedade *opacity* garante que todas as transações observam um estado consistente do sistema. A JVSTM é uma biblioteca que suporta controle de concorrência com múltiplas versões *(MVCC - MultiVersioning Concurrency Control)*, foi criada por [Cachopo and Rito-Silva 2006] e oferece um excelente desempenho em operações somente leitura.

O artigo está organizado como segue: Na seção 2 estão os trabalhos relacionados. A seção 3 descreve o modelo e definições do sistema. Na seção 4 é detalhado o protocolo e seu funcionamento. Na seção 5 encontram-se os algoritmos utilizados. Na seção 6 estão as avaliações e resultados encontrados. Por fim, as conclusões na seção 7.

## 2. Trabalhos relacionados

Nesta seção, sintetiza-se brevemente os trabalhos relevantes para a construção do Mesobi. A tabela 1 resume as principais características dos trabalhos relevantes em STM. As abordagens $D^2STM$ e $Zhang$ utilizam *Atomic Broadcast* (AB) desde o início de sua execução, não levando em conta uma possível ordenação das mensagens pela rede. Inicialmente a abordagem do Mesobi é otimista, ou seja, não há a necessidade de um mecanismo de entrega de mensagens atômico para execução de transações, quando operando de forma otimista.

Nos trabalhos $AGGRO$, $SPECULA$, $OSARE$ e $STR$ existe execução otimista das transações, mas diferente do Mesobi, é necessário que a ordem final de entrega das mensagens seja definida para que as transações possam ser confirmadas. Mais especificamente, o modelo utilizado em $STR$ é baseado na completa exploração de todas ordens distintas plausíveis recebidas pelo *Optimistic Atomic Broadcast* (OAB). O Mesobi utiliza uma abordagem semelhante à utilizada por $STR$, $AGGRO$, $SPECULA$ e $SCert$ onde transações são executadas localmente logo que são recebidas, sem necessidade de troca de mensagens inter-réplica. É interessante notar que as abordagens $STR$, $AGGRO$ e $OSARE$ bloqueiam a *thread* que executou a transação, impossibilitando que código não transacional seja executado. No $SCert$ novas transações podem ser bloqueadas temporariamente antes de serem executadas.

| | A-B | Byzantine. | Réplica. | A-R | Modelo |
|---|---|---|---|---|---|
| D2STM | Sim | Não | >1 | Não | Pré-declarado |
| RAM-DUR | Sim | Não | >1 | Não | Interativo |
| AGGRO | Sim[1] | Não | >1 | Não | Interativo |
| STR | Idem AGGRO | Não | >1 | Não | Interativo |
| SPECULA | Idem AGGRO | Não | >1 | Sim | Interativo |
| SCert | Idem AGGRO | Não | >1 | Sim | Interativo |
| OSARE | Idem AGGRO | Não | >1 | Não | Interativo |
| Granola | Não | Não | (2f+1)*c [2] | Não | Pré-declarado |
| Zhang | Sim | Sim | 3f+1; 2f+1 | Sim | Pré-declarado |
| **Mesobi** | **Adaptive**[3] | **Sim** | **3f+1** | **Sim/Não**[4] | Ambos |

**Tabela 1. Trabalhos relacionados e proposta.**

[1]Precisa esperar pela ordem final de entrega de mensagens para terminar.

Semelhante ao Mesobi, $STR$, $AGGRO$, $OSARE$, $RAM-DUR$ e $SCert$ não necessitam do conhecimento *a-priori* dos dados acessados por uma transação para sua execução (transações interativas, ver seção 3.1). O Mesobi utiliza para transações interativas mecanismo semelhante ao $RAM-DUR$ para verificação de conflitos entre transações, onde os conjuntos de leitura e escritas são enviados para cada réplica. Diferente de $RAM-DUR$, ao invés de utilizar AB como mecanismo de validação, Mesobi utiliza um modelo de certificação total, melhor detalhado na seção 4.1. Adicionalmente, o Mesobi trata da execução simultânea de transações pré-declaradas e interativas, sendo o cliente quem define por qual utilizar. Nosso trabalho utiliza mecanismo semelhante ao *Granola* para certificação entre transações, onde é necessário que todos repositórios respondam positivamente a uma requisição de *commit*. Nosso modelo também utiliza mecanismo semelhante quando transações interativas necessitam confirmar em paralelo: o desempate é feito priorizando a transação com menor identificador.

Dentre os trabalhos citados, somente Zhang menciona tolerância a faltas Bizantinas no contexto de STM. Sua abordagem utiliza $3f+1$ réplicas para consenso e $2f+1$ réplicas para execução. No trabalho de Zhang, transações somente leitura podem ser abortadas, não se utiliza a alta probabilidade em que as mensagens são entregues pela rede utilizando IP-Multicast e o conceito de transações interativas não é considerado.

## 3. Modelo e Definições

Esta seção descreve as definições básicas do sistema, premissas consideradas para o protocolo e sua arquitetura.

### 3.1. Definições básicas do sistema e premissas

Consideramos um sistema distribuído assíncrono clássico, consistindo de um conjunto arbitrário de clientes (não infinito) não Bizantinos C = $\{c_1, c_2, ..., c_n\}$ e um finito conjunto de réplicas R = $\{r_1, r_2, ..., r_n\}$ [Guerraoui and Rodrigues 2006]. Assumimos em nosso modelo um nível parcial de sincronia, onde o sistema se comporta de forma assíncrona em grande parte do tempo mas existem períodos de estabilidade [Dwork et al. 1988]. As réplicas se comunicam via passagem de mensagem e podem falhar de acordo com o modelo de faltas Bizantinas [Lamport and Fischer 1982] em até $f$ réplicas. A cardinalidade do conjunto de réplicas consiste de $\mid R \mid \geq 3f+1$.

Uma réplica é dita faltosa ou Bizantina quando se desvia de suas especificações, uma réplica faltosa pode parar de enviar mensagens, enviar mensagens fora de ordem, omitir o envio ou recebimento de mensagens, atrasar mensagens e corromper mensagens. Todas as mensagens são assinadas e trafegam em um canal confiável.

O protocolo utiliza transações pré-declaradas e interativas. Em transações pré-declaradas, o cliente necessita enviar todo código transacional para as réplicas [5]. Nas transações interativas, o cliente interage com as réplicas, enviando operação por operação. As transações interativas têm prioridade sobre as transações pré-declaradas, sendo que em caso de confirmação simultânea as transações interativas são confirmadas.

[2]Existem 2f+1 réplicas dentro de cada repositório, onde *c* é o número de repositórios.

[3]*Reliable Multicast* quando otimista e *Total Order Byzantine Multicast* quando não otimista.

[4]Aborta read-only somente para transações interativas.

[5]Na prática *templates* das transações são criados nas réplicas e apenas os parâmetros são enviados.

### 3.2. Arquitetura

A figura 1 provê uma visão em alto nível da arquitetura composta em cada réplica Mesobi. O componente Mesobi recebe as solicitações do cliente e faz a comunicação inter-réplica. Todas as solicitações ao serem recebidas pelas réplicas são analisadas através do analisador de transações paralelas (*PTA*); este módulo é divido em duas partes, uma específica para transações interativas denominado *Gerente TI* e outra específica para transações pré-declaradas denominado *Gerente TD*. Após serem analisadas, as transações são encaminhadas para os executores de transações caso não possuam conflitos (JVSTM), mais detalhes sobre conflitos podem ser vistos na seção 4.2. Os executores JVSTM são compostos por *threads* e cada transação é vinculada a uma *thread*. Após o término da transação, o certificador otimista faz a verificação de possibilidade de confirmar a transação de maneira otimista, caso isso seja possível, a transação é confirmada. Caso contrário, o protocolo Bizantino precisa ser executado.

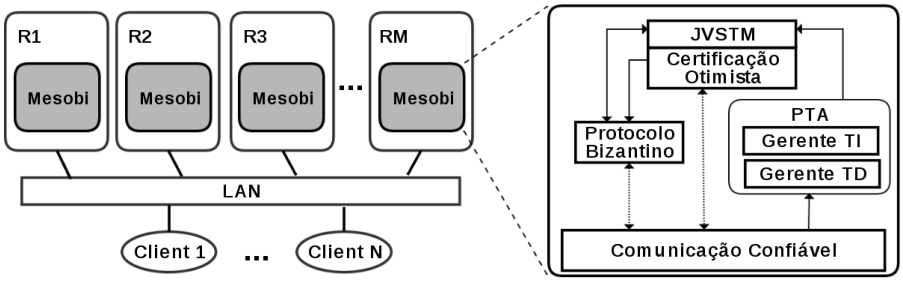



**Figura 1. Arquitetura de uma réplica Mesobi.**

## 4. Protocolo Mesobi

Para melhor compreensão, será explicado o fluxo de operação (figura 2) junto com os algoritmos (seção 5). Inicialmente o cliente envia uma solicitação às réplicas (etapa TX.Begin, figura 2). A solicitação do cliente pode ser de dois tipos, pré-declarada ($T_i$) ou interativa ($TX_i$). Cada cliente pode apenas enviar uma solicitação por vez, independente se pré-declarada ou interativa. As transações pré-declaradas e interativas são definidas em forma de tuplas, e necessariamente precisam conter: $T = \{T_i.B, T_i.Op_1, ..., T_i.Op_n, T_i.C\}$ para as pré-declaradas e $TX = \{TX_i.B, TX_i.Op_1, ..., TX_i.Op_n, TX_i.C\}$ para as interativas, onde $B \equiv$ Begin, $Op \equiv$ Operação e $C \equiv$ Commit.

A execução otimista e não otimista das transações pré-declaradas, na ausência de transações interativas, segue fluxo idêntico ao protocolo OB-STM [Ribeiro et al. 2013]. O Mesobi é uma evolução do protocolo OB-STM onde transações interativas e pré-declaradas concorrem pela execução, sendo que as transações interativas tem prioridade sobre as pré-declaradas. O Mesobi utiliza o conceito *WiP - WorkInProgress*, semelhante ao utilizado por [Palmieri et al. 2010], diferindo no quesito propagação de atualização de objetos. Uma transação está em andamento (WiP) se a mesma já foi iniciada e ainda não foi concluída. A área sombreada na figura 2b representa o PBFT [Castro and Liskov 1999], utilizado na fase de resolução bizantina (mensagens fora de ordem ou réplicas maliciosas).

No caso de transações interativas, o cliente necessita efetuar alguns passos a mais. Inicialmente o cliente solicita às réplicas a intenção de iniciar uma transação (etapa TX.Begin figura 2, linha 1 algoritmo 2). Cada réplica retorna ao cliente o seu *timestamp*. O *timestamp* ($ts$) utilizado segue o modelo proposto por [Bazzi and Ding 2004], para evitar o crescimento arbitrário do espaço de endereçamento por réplicas maliciosas. O cliente espera por $2f + 1$ respostas das réplicas contendo o $ts$ em questão. Com o $ts$ de $2f + 1$ réplicas, é possível definir o valor que será considerado pelo gerenciador de contenção caso exista concorrência entre transações interativas. A figura 2 detalha a interação cliente-réplicas e réplica-réplica. Após a definição do $ts$ pelo cliente (etapa TX.Begin figura 2), a primeira operação ($TX_i.Op$) pode ser enviada às réplicas (etapa TX.Op figura 2, linha 6 algoritmo 2). A operação somente é executada se não houver conflitos com outras transações em execução ou esperando para ser executada (linha 12 do algoritmo 2). A cada operação executada pelas réplicas um novo $ts$ é retornado ao cliente para determinar a ordem das operações; essa ordem de operação é utilizada pelo gerenciador de contenção como critério de desempate.

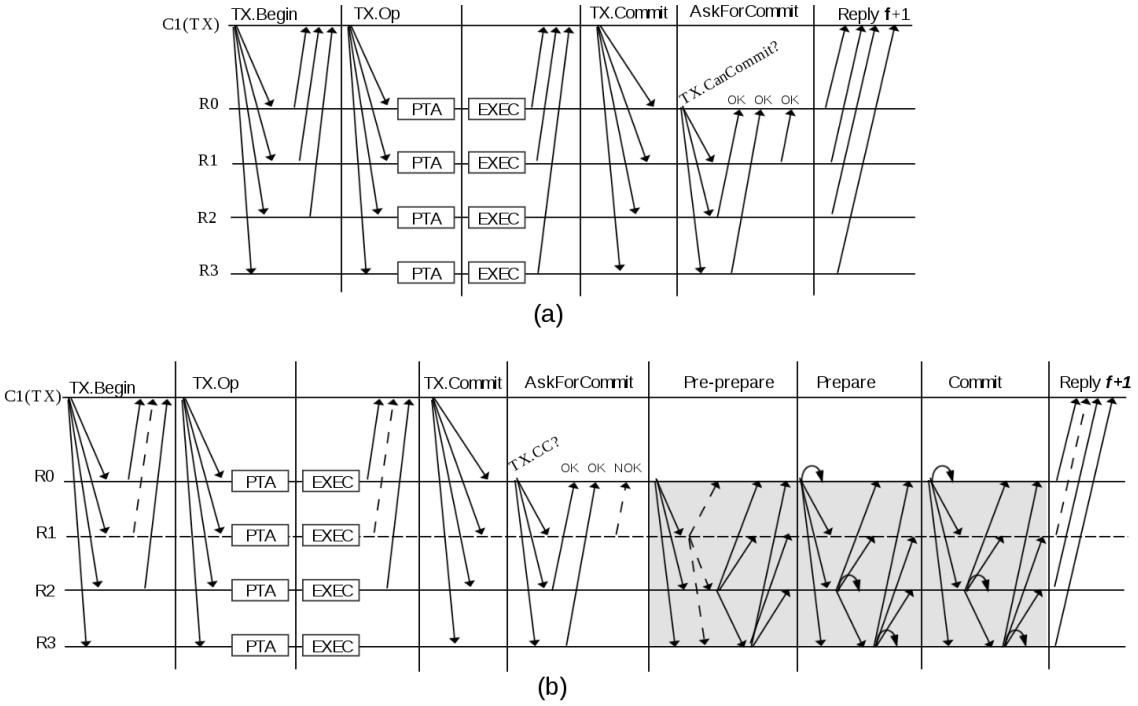



**Figura 2. Transações interativas: fluxo de operação otimista (a) e não otimista (b).**

Após o cliente terminar o envio de operações, o cliente precisa enviar o comando de término da transação ($TX_i.C$) (etapa TX.Commit figura 2). Ao receber do cliente a solicitação de confirmação (linha 20 algoritmo 2) cada réplica envia para o conjunto de réplicas no sistema ($\prod^{R}$) uma mensagem de permissão de confirmação da transação em questão (etapa AskForCommit, figura 2). Cada réplica necessita receber a confirmação de todas outras réplicas (linha 39 algoritmo 3) para confirmar a transação. Caso receba uma não-permissão (mensagem NOK!, etapa AskForCommit, figura 2b) devido a uma réplica maliciosa ou transações fora de ordem, o protocolo Bizantino precisa ser iniciado (etapa Pre-prepare, figura 2b, linha 42 algoritmo 3). O protocolo Bizantino, para evitar

a sobrecarga imposta pelo consenso, quando iniciado, define a ordem das transações pré-declaradas e interativas existentes nos *buffers* $\prod^{T}$ e $\prod^{TX}$. Os critérios definidos para confirmar ou abortar uma transação serão explicados a seguir.

### 4.1. Histórico, premissas e snapshot

Nessa seção será detalhado o tratamento das transações quando executadas no caso normal (sem falhas e entrega ordenada das mensagens) e no caso com falhas (mensagens fora de ordem ou réplicas maliciosas). Para ambas abordagens, as definições abaixo se mantêm.

O princípio da teoria de seriabilidade [Bernstein and Goodman 1985] garante que a execução do histórico $\mathcal{H}$ de transações em todos os processos replicados é equivalente à execução de $\mathcal{H}$ em um ambiente não replicado. O histórico de execução $\mathcal{H}$ (também conhecido como *schedule*) é definido sobre um conjunto de transações $T = \{T_1, T_2, ..., T_n\}$ e especifica um conjunto de operações (podendo ser operações entrelaçadas) dessas transações. Cada transação é composta por uma *tupla*: $T = \{B, Op_1, ..., Op_n, C\}$, e nas transações interativas as operações podem ser enviadas uma-a-uma. Formalmente, um histórico completo $\mathcal{H}^{c}_{T}$ definido sobre um conjunto de transações $T = \{T_1, T_2, ..., T_n\}$ é um histórico de ordem parcial $\mathcal{H}^{p}_{T} = \{\Sigma_T, \prec_{\mathcal{H}}\}$ onde:

(1) $\Sigma_T = \cup_{i=1}^{n} \Sigma_i$. (2) $\prec_{\mathcal{H}} \supseteq \cup_{i=1}^{n} \prec T_i$. (3) Para quaisquer duas operações conflitantes $Op_i, Op_j \in \Sigma_T$, ou $Op_i \prec_{\mathcal{H}} Op_j$, ou $Op_j \prec_{\mathcal{H}} Op_i$.

A primeira condição declara que o histórico global de execução das operações é a união da execução das operações individuais. A segunda, define a relação de ordenação do histórico como um super-conjunto das ordenações de operações individuais (a ordem das operações dentro de cada transação é mantida). A terceira, define a ordem em que as operações conflitantes são executadas no histórico.

A ordem serial imposta pelo Mesobi é garantida através das seguintes premissas: (i) toda transação pré-declarada que não apresentar conflito com transações interativas e pré-declaradas (em execução ou na fila de execução) pode ser executada de imediato (transações sem conflito a ordem de execução não é importante); (ii) transações pré-declaradas que tenham conflitos com operações de transações interativas não podem ser executadas de imediato; (iii) operações de transações interativas são executadas somente se não houver conflitos com transações pré-declaradas em execução ou que estejam marcadas como *WiP*. As verificações das premissas (i) e (ii) encontram-se no algoritmo 1, na função PTA (linha 1). A premissa (iii) é verificada no algoritmo 2, linha 12. O Mesobi utiliza *weak snapshot isolation*, que permite transações somente leitura sejam feitas em um *snapshot* que reflete um estado correto (*committed*) dos dados. O modelo *snapshot isolation* que o Mesobi utiliza possui algumas variações. O *snapshot* é capturado no momento em que ocorre a primeira operação ($TX_i.Op$) emitida pelo cliente nas transações interativas; o momento considerado consiste no momento sem conflitos, onde a operação poderá ser executada. Operações pré-declaradas somente leitura não são abortadas, pois retornam os dados do último *snapshot* confirmado. Essas situações são exemplificadas na figura 3.

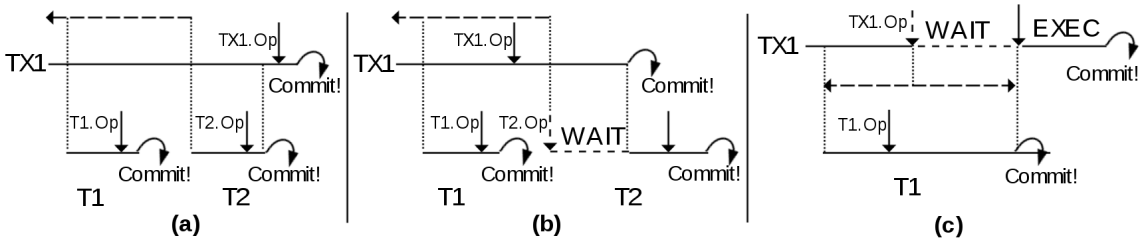



**Figura 3. Transações interativas e pré-declaradas: (a) sem conflito. (b) com conflito ($TX_1.Op \prec T_2.B$). (c) com conflito ($T_1.Op \prec TX_1.Op$), $TX_1$ precisa esperar $T_1$ confirmar. O símbolo $\downarrow$ deve ser interpretado como a operação $z = z + 1$;**

### 4.2. Caso otimista

A figura 3 detalha as relações de conflitos entre transações interativas e declaradas. No detalhe (a), não existem conflitos entre as transações, pois não existem operações entrelaçadas (visão local). Formalmente, $\mathcal{H}_T^c = \{T_1.Op, T_1.C, T_2.Op, T_2.C, TX_1.Op, TX_1.C\}$[6]. É importante lembrar que o *snapshot* somente é capturado quando a operação $(TX_1.Op)$ for recebida pela réplica; nesse caso, como $(TX_1.Op)$ não possui conflitos, o *snapshot* é capturado. No detalhe (b), existe conflito entre as operações $(TX_1.Op)$ e $(T_2.Op)$. Nesse caso, à transação $T_2$ não será permitido executar de imediato, pois viola a premissa (ii). Somente após o término (*commit*|*abort*) da transação $TX_1$ é que $T_2$ será executada. No detalhe (c), existe conflito entre as operações $(T_1.Op)$ e $(TX_1.Op)$. A operação $(TX_1.Op)$ não poderá ser executada de imediato, pois viola a premissa (iii). Somente após o término (*commit*|*abort*) da transação $T_1$ é que $(TX_1.Op)$ será executada. Ressalta-se que no primeiro momento em que a operação $(TX_1.Op)$ foi recebida pela réplica havia conflito (seta pontilhada), e com isso o *snapshot* não pode ser capturado. Somente após o (*commit*|*abort*) da transação $T_1$ é que o *snapshot* realmente será capturado. É importante ressaltar que essas propriedades se mantêm no Mesobi, mesmo quando a ordem em que as transações são recebidas pelas réplicas forem diferentes. Isso se dá devido ao protocolo Bizantino, que em caso de mensagens fora de ordem, ordena as mesmas. O caso não otimista será melhor detalhado na seção 4.3.

### 4.3. Caso não otimista

Conforme a figura 4, suponha que a ordem de chegada das operações nas réplicas seja: $R_{0,1} = \{T_1.B, T_2.B, TX_1.Op\}$ e $R_{2,3} = \{T_2.B, T_1.B, TX_1.Op\}$[7]. A transação $T_2$ nas réplicas $R_{0,1}$ e a transação $T_1$ nas réplicas $R_{2,3}$ não poderão ser executadas de imediato pois não atendem a premissa (i). Os históricos de operações das réplicas $R_{0,1}$ são diferentes das réplicas $R_{2,3}$, o que viola a seriabilidade. Para fins de simplicidade, definiremos que a réplica $R_0$ terminou a execução de $T_1$ primeiro. Após o término, uma mensagem chamada *AskForCommit* (AFC) é enviada para cada réplica solicitando permissão para confirmar $T_1$. Cada réplica ao receber a mensagem AFC, verifica se a transação pode ser confirmada. Devido às ordens de execução distintas entre as réplicas $R_{0,1}$ e $R_{2,3}$,

[6]As operações de *Begin* das transações interativas foram omitidas pois não influenciam na análise de conflito.

[7]Para as transações pré-declaradas o *Begin* é importante na análise de conflito, pois os conjuntos de leitura e escrita já são conhecidos.

onde $\mathcal{H}^{p}_{R_{0,1}} = \{T_1.Op\}$ e $\mathcal{H}^{p}_{R_{2,3}} = \{T_2.Op\}$, uma mensagem de retorno contendo uma não-permissão de confirmação será gerada e devido a mensagem de não confirmação, o protocolo Bizantino será iniciado. O protocolo Bizantino definirá a ordem de execução ou confirmação das transações através de trocas de mensagens utilizando *total order deliver* (*TO-Deliver* - PBFT). Após a definição da ordem pelo protocolo Bizantino, as réplicas corretas seguirão a definição imposta.

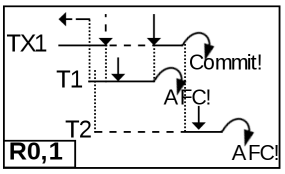
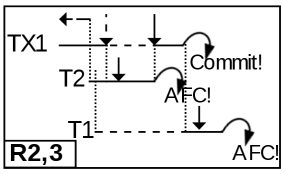
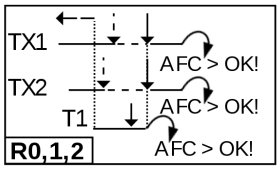



**Figura 4. Transações pré-declaradas e interativas. Com conflito de dados.**



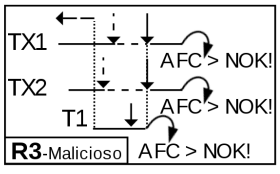




**Figura 5. Transações pré-declaradas e interativas. Com réplica maliciosa.**

A figura 5 retrata o caso com réplica maliciosa, a ordem de recebimento das transações é a mesma em todas as réplicas: $R_{0,1\ ,2\ ,3} = \{T_1.B, TX_2.Op, TX_1.Op\}$. Nesse exemplo transações interativas e declaradas são conflitantes; conforme as premissas (i) e (ii), a transação $T_1.B$ não possui conflitos com as operações ($TX_1.Op$) e ($TX_2.Op$) nem com outras transações declaradas, isso permite com que $T_1$ seja executada. Porém, enquanto $T_1$ está executando, as operações das transações $TX_1$ e $TX_2$ chegam. Com isso as operações da transação $TX_1$ e $TX_2$ devem esperar, conforme premissa (iii). Após o término da execução da transação $T_1$, caso não existisse réplica maliciosa (R3), o protocolo deveria terminar, mas ao solicitar permissão para confirmar a transação $T_1$, as réplicas solicitantes recebem uma mensagem (NOK!); com isso o protocolo Bizantino deve ser iniciado. O protocolo Bizantino define a ordem e as réplicas corretas seguem sua imposição.

As relações de conflitos entre transações interativas e pré-declaradas são detectadas em função das seguintes condições: para as operações interativas, o ponto importante são as operações e para as pré-declaradas o início (*Begin*). Então, conflitos acontecem se: $T_i.B \prec TX_i.Op$ ou $TX_i.op \prec T_i.B$; nesses casos as premissas (i, ii e iii) devem ser seguidas.

## 5. Algoritmos

Essa seção contém o algoritmo que é executado em cada réplica; o algoritmo do cliente foi omitido por restrição de espaço. Os algoritmos foram divididos para melhor enquadramento no texto e melhor compreensão. A tabela 2 detalha as variáveis utilizadas. No algoritmo 1 é feita análise de conflitos entre transações interativas e pré-declaradas e execução das transações pré-declaradas. O algoritmo 2 trata a execução das transações interativas. A verificação de possibilidade de confirmação das transações entre réplicas e confirmação das mesmas caso possível, é detalhada no algoritmo 3. Por fim, o algoritmo 4 define a ordem de confirmação das transações caso mensagens estejam fora de ordem ou existam réplicas maliciosas; além disso re-executa transações, aborta transações e define qual será a próxima a ser executada. Os algoritmos são citados na seção 4 onde são explicados em paralelo com a figura 3.

**Tabela 2. Variáveis.**

| | | | |
|---|---|---|---|
| 1: $\prod^{R}$ | ▷ Replica Set | 8: $\prod^{TXOp}$ | ▷ Interactive Transactions Operation Set |
| 2: $\prod^{Reply}$ | ▷ Reply Set | 9: $\prod^{TXCommit}$ | ▷ Interactive Transactions Commit Set |
| 3: $\prod^{T}$ | ▷ Pre-Declared Transactions Set | 10: $\prod^{XWS}$ | ▷ Interactive Transactions Write Set |
| 4: $\prod^{WS}$ | ▷ Pre-Declared Transactions Write Set | 11: $\prod^{XRS}$ | ▷ Interactive Transactions Read Set |
| 5: $\prod^{RS}$ | ▷ Pre-Declared Transactions Read Set | 12: f | ▷ Number of tolerated faults |
| 6: BCO | ▷ Pre-Declared Buffer Commit Order | 13: timeout | ▷ Max time to wait for response of replicas |
| 7: $\prod^{TXBegin}$ | ▷ Interactive Transactions Begin Set | | |

**Algoritmo 1** Transações pré-declaradas.

1: **function** PTA($T_i, C_i$) ▷ Make conflict analysis among all transactions
2: **if** ($size(\prod^{T})$> 1 ∨ $size(\prod^{TXBegin})$> 0) **then**
3: **for all** $Tj \in \prod^{T}$ minus $T_i$ **do**
4: **if** (¬ ( (WS($T_i$) ∩ RS($T_j$) = ∅) ∧ (WS($T_j$) ∩ RS($T_i$) = ∅) ∧ (WS($T_i$) ∩ WS($T_j$) = ∅) )) **then**
5: exit ATP
6: **for all** $TX_i.Op \in \prod^{TXOp}$ **do**
7: **if** (¬ ( (WS($T_i$) ∩ XRS($TX_i$) = ∅) ∧ (WS($TX_i$) ∩ RS($T_i$) = ∅) ∧ (WS($T_i$) ∩ XWS($TX_i$) = ∅) )) **then**
8: exit ATP
9: Call **EXEC-T**($T_i, C_i$)

10: **upon:** receive($\langle REQUEST, T_i, C_i \rangle \sigma$) from client
11: $\prod^{T} \leftarrow \prod^{T} \cup (T_i, C_i)$
12: $\prod^{WS} \leftarrow \prod^{WS} \cup$ (getWriteSet($T_i$), $C_i$)
13: $\prod^{RS} \leftarrow \prod^{RS} \cup$ (getReadSet($T_i$), $C_i$)
14: **Call PTA**($T_i$)
15:

16: **function** Exec-T($T_i$, $C_i$)
17: **set** WS($T_i$) and RS($T_i$) as WorkInProgress
18: Executes $T_i$ atomically
19: **set** $T_i$ as Commiting
20: ReliableMulticast(⟨ASKFORCOMMIT⟩,
21: $T_i, \prod^{R}, C_i, WS, RS$)

**Algoritmo 2** Transações interativas.

1: **upon:** receive($\langle REQUEST - B, TX_i.Begin, C_i \rangle \sigma$) from client
2: $\prod^{TXBegin} \leftarrow \prod^{TXBegin} \cup (TX_i.Begin, C_i)$
3: **set** $TX_i$ as running
4: SendReliable(⟨RSEQ-NUMBER⟩, $C_i, R_i, RTS$) ▷ Return Replica TS (RTS) to client
5:
6: **upon:** receive($\langle REQUEST - OP, TX_i.Op, TX_i.CTS, C_i \rangle \sigma$) from client ▷ CTS - Client Timestamp
7: $\prod^{TXOp} \leftarrow \prod^{TXOp} \cup (TX_i.Op, C_i)$
8: $\prod^{XWS} \leftarrow \prod^{XWS} \cup$ (getWriteSet($TX_i.Op$), $TX_i.CTS$, $C_i$)
9: $\prod^{XRS} \leftarrow \prod^{XRS} \cup$ (getReadSet($TX_i.Op$), $TX_i.CTS$, $C_i$)
10: initExec ← true
11: **for all** $T_j \in \prod^{T} \wedge$ WiP($T_j$) **then**
12: **if** (¬ ( ((WS($T_j$) ∩ (XRS($TX_i.Op$))=∅) ∧ (RS($T_j$) ∩ (XWS($TX_i.Op$))=∅ ) ∧ (WS($T_j$) ∩ (XWS($TX_i.Op$))=∅ )) )) **then**
13: initExec ← false
14: stop **for all**
15: **if** (initExec)
16: EXEC-TX ($TX_i.Op$)
17: **else**
18: Set $TX_i.Op$ as waiting until be informed of a commit.
19:
20: **upon:** receive(⟨ REQUEST-C, $TX_i.Commit, C_i$ ⟩$\sigma$) from client
21: ReliableMulticast($\langle ASKFORCOMMIT - TX, TX_i.Commit, \prod^{R}, XWS, XRS \rangle \sigma$)
22:
23: **function** EXEC-TX($TX_i.Op$)
24: $Result \leftarrow$ Execute ($TX_i.Op$) ▷ Execute an operation in an interactive transaction
25: SendReliable($\langle RSEQ - NUMBER, C_i, R_i, RTS, Result \rangle \sigma$)

## 5.1. Otimização

Existe uma otimização que pode ser feita para melhorar o desempenho e diminuir o número de mensagens trafegadas pela rede. A fase de certificação de uma transação (linha 21 do algoritmo 1) pode ser melhorada através da seguinte verificação: (i) cada réplica

**Algoritmo 3** Certificação e confirmação.

1: **upon:** receive($\langle ASKFORCOMMIT, T_i, R_i, C_i \rangle \sigma$) from replica
2: **if** ($T_i$ can commit) **then** ▷ verification of conflicts, resembling lines 4 and 7 algorithm 1
3: SendReliable($\langle R - ASKFORCOMMIT, T_i, R_i, C_i, "OK!" \rangle \sigma$)
4: **else**
5: SendReliable($\langle R - ASKFORCOMMIT, T_i, R_i, C_i, "NOK!" \rangle \sigma$)
6:
7: **upon:** receive($\langle ASKFORCOMMIT - TX, TX_i, R_i, C_i \rangle \sigma$) from replica
8: **if** ($TX_i$ can commit) **then** ▷ verification of conflicts , resembling lines 4 and 7 algorithm 1
9: SendReliable($\langle R - ASKFORCOMMIT - TX, TX_i, R_i, C_i, "OK!" \rangle \sigma$)
10: **else**
11: SendReliable($\langle R - ASKFORCOMMIT - TX, TX_i, R_i, C_i, "NOK!" \rangle \sigma$)

12: **function** COMMIT-T($T_i$)
13: Commit transaction $T_i$!
14: $\prod^{T} \leftarrow \prod^{T} \setminus T_i$
15: $\prod^{WS} \leftarrow \prod^{WS} \setminus$ (getWriteSet($T_i$), $C_i$)
16: $\prod^{RS} \leftarrow \prod^{RS} \setminus$ (getReadSet($T_i$), $C_i$)
17: Informs waiting TX.Ops a Transaction was committed
18: SendReliable($\langle$REPLY$\rangle$, $Ri$, $Reply$)
19: Call **NEXT-T-EXEC**()
20:
21: **function** COMMIT-TX($TX_i$)
22: Commit transaction $TX_i$!
23: $\prod^{TXBegin} \leftarrow \prod^{TXBegin} \setminus TX_i.Begin$
24: $\prod^{XWS} \leftarrow \prod^{XWS} \setminus$ (getWriteSet($TX_i$), $C_i$)
25: $\prod^{XRS} \leftarrow \prod^{XRS} \setminus$ (getReadSet($TX_i$), $C_i$)
26: $\prod^{TXCommit} \leftarrow \prod^{TXCommit} \cup TX_i$
27: SendReliable($\langle$REPLY$\rangle$, $Ri$, $Reply$)
28: Call **NEXT-T-EXEC**()
29:
30: **upon:** receive($\langle R - ASKFORCOMMIT, T_i, R_i, C_i, Reply \rangle \sigma$) from replica
31: acceptCommit ← WaitForAcceptance(timeout)
32: **if** (acceptCommit = 3f) **then**
33: Call **COMMIT-T** ($T_i$)
34: **else**
35: TO-Deliver($\langle PRE-PREPARE, R_i, Order, OrderTX, v, n, d \rangle \sigma$)
36:
37: **upon:** receive($\langle R - ASKFORCOMMIT - TX, TX_i, R_i, C_i, Reply \rangle \sigma$) from replica
38: acceptCommit ← WaitForAcceptance(timeout)
39: **if** (acceptCommit = 3f) **then**
40: Call **COMMIT-TX** ($TX_i$)
41: **else**
42: TO-Deliver($\langle PRE-PREPARE, R_i, Order, OrderTX, v, n, d \rangle \sigma$)

**Algoritmo 4** Protocolo Bizantino - PBFT.

1: **upon:** receive($\langle PRE - PREPARE, Ri, Order, OrderTX, v, n, d \rangle \sigma$) from replicas
2: **if** (PRE-PREPARE was accepted) **then**
3: **for all** ($R \in \prod^{R}$ minus his own replica) **do**
4: TO-Deliver($\langle PREPARE, Ri, Order, OrderTX, v, n, d \rangle \sigma$)
5:
6: **upon:** receive($\langle PREPARE, Ri, Order, OrderTX, v, n, d \rangle \sigma$) from replicas
7: **if** (PREPARE was accepted) **then**
8: **for all** ($R \in \prod^{R}$ minus his own replica) **do**
9: TO-Deliver($\langle COMMIT - ORDER, Ri, Order, OrderTX, v, n, d \rangle \sigma$)
10:
11: **upon:** receive($\langle COMMIT - ORDER, Ri, Order, OrderTX, v, n, d \rangle \sigma$) from replicas
12: **if** (COMMIT-ORDER was accepted) **then**
13: BCO ← Order
14: Set all $T \in$ BCO as Work in Progress (WiP)
15: **for all** ($TX_i \in OrderTX$) **do**
16: **if** ($TX_i$ Can Commit) **then**
17: Call **COMMIT-TX**($TX_i$)
18: **else**
19: Call **ABORT-TX**($TX_i$)
20: Call **NEXT-T-EXEC**()

21: **function** ABORT-TX($TX_i$)
22: Abort transaction $TX_i$!
23: $\prod^{TXBegin} \leftarrow \prod^{TXBegin} \setminus TX_i.Begin$
24: $\prod^{XWS} \leftarrow \prod^{XWS} \setminus$ (getWriteSet($TX_i$), $C_i$)
25: $\prod^{XRS} \leftarrow \prod^{XRS} \setminus$ (getReadSet($TX_i$), $C_i$)
26: SendReliable($\langle$REPLY$\rangle$, $Ri$, $Reply$)
27: Call **NEXT-T-EXEC**()
28:
29: **function** RE-EXEC-T($T_i$)
30: Begins $T_i$ re-execution
31: BCO ← BCO $\setminus T_i$
32: **Call Commit-T** ($T_i$)
33:
34: **function** NEXT-T-EXEC( )
35: **if** (BCO $> 0$) **then**
36: Call **RE-EXEC-T**(getFirstElement(BCO))
37: **else**
38: Call **ATP**(getFirstElement($\prod^{T}$))

ao receber uma mensagem *AskForCommit* armazena o identificador da réplica solicitante; (ii) antes de enviar uma mensagem *AskForCommit*, a réplica verifica se já recebeu uma mensagem *AskForCommit* sobre a mesma solicitação feita por outra réplica. Caso essas premissas sejam verdadeiras, menos mensagens serão trocadas. Outro ponto que pode ser verificado, é a possibilidade de na fase certificação total (AskForCommit), a certificação ser feita utilizando um conjunto de transações (*batch*) ao invés de uma a uma.

## 6. Avaliação

As avaliações foram feitas utilizando cinco computadores conectados por uma rede Local (LAN) 10/100. Dentre estes, quatro foram utilizados para processamento nas réplicas, com sistema operacional Linux Ubuntu Server 3.8.0-33-generic x86_84 Intel(R) Core I7 com oito núcleos 1.6GHz, com 12GB de memória e capacidade de 2 *threads* por núcleo. Um computador para simular os clientes, rodando sobre o sistema operacional Slackware Linux 2.6.37.6-smp 2 SMP i686 Intel(R) Core(TM)2 com quatro núcleos 3.00GHz GenuineIntel GNU/Linux, com 3GB de memória e capacidade de uma *thread* por núcleo. A representação do cliente no sistema se fez através de *threads*.

Os testes foram executados utilizando 100 requisições por cliente, sendo que cada requisição contém apenas uma operação. As operações consistem em incrementar um contador compartilhado, foram criados dois contadores *A* e *B*. Nos gráficos, quando dito operações conflitantes, significa que transações interativas possuem conflitos entre si, transações pré-declaradas (para fins de brevidade, nos gráficos será chamado declaradas) possuem conflitos entre si e que ambas transações possuem conflitos entre si (declaradas e interativas). Quando dito sem conflitos, significa que transações interativas possuem conflitos entre si, transações pré-declaradas possuem conflitos entre si mas não existem conflitos entre os dois tipos de transações (declaradas e interativas).

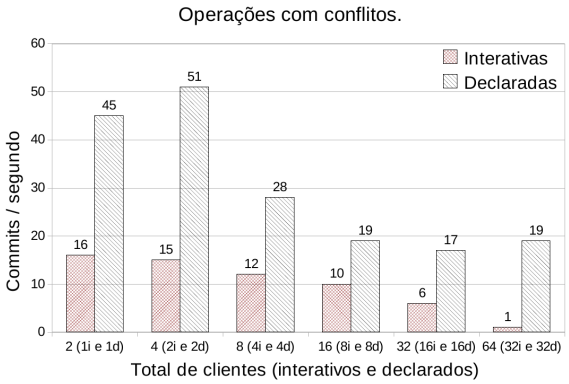



**Figura 6. Transações pré-declaradas e interativas, com conflito de dados.**

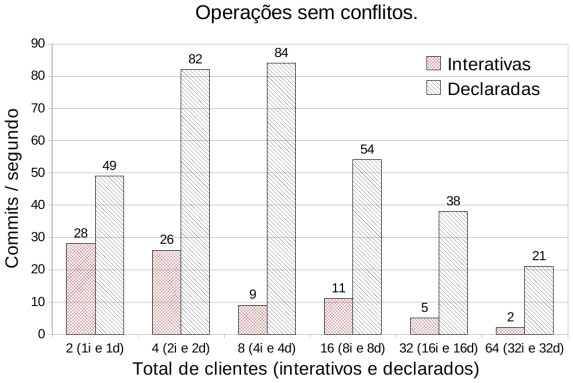



**Figura 7. Transações pré-declaradas e interativas, sem conflito de dados.**

A figura 6 apresenta a taxa de *commits* por segundo, em um cenário com conflitos (transações interativas e declaradas alteram o mesmo dado). Percebe-se que as transações declaradas alcançam um número maior de *commits* do que as transações interativas. Isto se deve ao fato de que cada transação declarada consiste de apenas uma mensagem, enquanto cada transação interativa é composta de, no mínimo, três mensagens (*begin, op-*

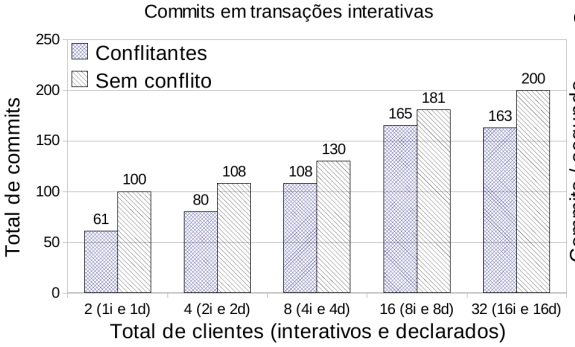



**Figura 8. Total de commits em transações interativas.**

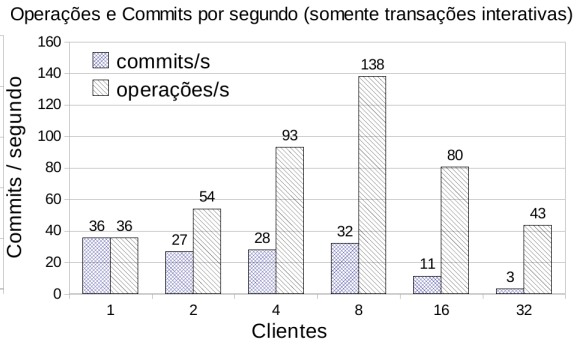



**Figura 9. Transações interativas com conflito.**

*eration* e *commit*). Considerando então a regra de espera[8], podemos dizer que transações interativas esperam mais vezes do que transações declaradas. Exemplificando, quando uma transação interativa está executando, duas ou três declaradas podem ser colocadas em espera, e serão executadas em sequência (pois estão na fila de espera) logo que a interativa terminar. Quando uma declarada está executando, uma interativa será colocada em espera, e será executada quando a declarada terminar. Observamos ainda na figura 6 que em situações de alta disputa (32 clientes interativos), é mínima a taxa de efetivação de operações, sendo este um ponto de saturação.

A figura 7 demonstra novamente um melhor resultado para transações declaradas, em relação às transações interativas, que mantém seu ponto de saturação por volta de 32 clientes. A baixa taxa de confirmações por parte de transações interativas se dá pela quantidade de mensagens trocadas para finalização do protocolo.

Na figura 8 pode-se notar que, considerando apenas o total de *commits* realizados por transações interativas, existe um incremento de solicitações efetivadas, tanto em situações de conflitos quanto em situações sem disputa. A figura 9 apresenta a execução de transações interativas sem concorrência com transações declaradas. Nota-se que a concorrência entre as transações interativas faz com que apenas metade das solicitações sejam efetivadas, para dois clientes, na unidade de tempo de um segundo. No cenário com 32 clientes solicitando transações interativas, apenas três conseguem efetivar suas solicitações, num total de 43 operações realizadas na unidade de tempo considerada.

## 7. Conclusão

Existem sistemas nos quais é fundamental a execução de tarefas de forma atômica. Memória transacional em software disponibiliza uma abstração denominada transação para realizar tarefas de acordo com a semântica atômica. Dentre os tipos de transações, consideram-se as interativas e as pré-declaradas. Transações interativas são importantes pois permitem em tempo de execução modificar a lógica da aplicação com base nos resultados obtidos; ao mesmo tempo, transações pré-declaradas são úteis pois permitem certa flexibilidade em relação ao momento em que serão executadas. Neste artigo apresentamos o Mesobi, um protocolo que permite a execução de transações interativas e pré-declaradas

[8]Essa regra é exemplificada na seção 4.1: transações que chegam podem precisar esperar pelo término da execução de outra transação conflitante em execução.

no mesmo ambiente. A implementação de um protótipo foi realizada e os resultados demonstram que é viável a execução simultânea de ambos os tipos de transações, sendo que as transações pré-declaradas apresentam uma melhor escalabilidade em relação às interativas.

## Referências


Bazzi, R. A. and Ding, Y. (2004). Non-skipping timestamps for byzantine data storage systems. In *Distributed Computing*, pages 405–419. Springer.

Bernstein, P. A. and Goodman, N. (1985). Serializability theory for replicated databases. *Journal of Computer and System Sciences*, 31(3):355–374.

Cachopo, J. and Rito-Silva, A. (2006). Versioned boxes as the basis for memory transactions. *Science of Computer Programming*, 63(2):172–185.

Carvalho, N., Romano, P., and Rodrigues, L. (2011). Scert: Speculative certification in replicated software transactional memories. In *Proceedings of the 4th Annual International Conference on Systems and Storage*, page 10. ACM.

Castro, M. and Liskov, B. (1999). Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186.

Couceiro, M., Romano, P., Carvalho, N., and Rodrigues, L. (2009). D2stm: Dependable distributed software transactional memory. In *Dependable Computing, 2009. PRDC'09. 15th IEEE Pacific Rim International Symposium on*, pages 307–313. IEEE.

Cowling, J. and Liskov, B. (2012). Granola: low-overhead distributed transaction coordination. In *Proceedings of the 2012 USENIX conference on Annual Technical Conference*, pages 21–21. USENIX Association.

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

Guerraoui, R. and Rodrigues, L. (2006). *Reliable Distributed Programming*, volume 138. Springer.

Kemme, B., Pedone, F., Alonso, G., Schiper, A., and Wiesmann, M. (2003). Using optimistic atomic broadcast in transaction processing systems. *Knowledge and Data Engineering, IEEE Transactions on*, (4).

Kotla, R. and Dahlin, M. (2004). High throughput Byzantine fault tolerance. *International Conference on Dependable Systems and Networks, 2004*, pages 575–584.

Lamport, L. and Fischer, M. (1982). Byzantine generals and transaction commit protocols. Technical report, Technical Report 62, SRI International.

Li, K. and Hudak, P. (1989). Memory coherence in shared virtual memory systems. *ACM Transactions on Computer Systems (TOCS)*, 7(4):321–359.

Naehrig, M., Lauter, K., and Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? *Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11*, page 113.

Palmieri, R., Quaglia, F., and Romano, P. (2010). AGGRO: Boosting STM Replication via Aggressively Optimistic Transaction Processing. *2010 Ninth IEEE International Symposium on Network Computing and Applications*, pages 20–27.

Palmieri, R., Quaglia, F., and Romano, P. (2011). OSARE: Opportunistic Speculation in Actively REplicated Transactional Systems. *SRDS*, (257784):59–64.

Pedone, F. and Schiper, A. (1998). Optimistic Atomic Broadcast. *Distributed Computing. Springer Berlin Heidelberg*, (95):318–332.

Peluso, S., Fernandes, J., Romano, P., Quaglia, F., and Rodrigues, L. (2012). SPECULA: Speculative Replication of Software Transactional Memory. *SRDS*, pages 91–100.

Protic, J., Tomasevic, M., and Milutinovic, V. (1996). Distributed shared memory: Concepts and systems. *Parallel & Distributed Technology: Systems & Applications, IEEE*, 4(2):63–71.

Ribeiro, T. A., Lung, L. C., and Netto, H. V. (2013). OB-STM: An Optimistic Approach for Byzantine Fault Tolerance in Software Transactional Memory. *Simpósio Brasileiro de Engenharia de Sistemas Computacionais - SBESC*.

Romano, P., Palmieri, R., Quaglia, F., Carvalho, N., and Rodrigues, L. (2010). An Optimal Speculative Transactional Replication Protocol. *International Symposium on Parallel and Distributed Processing with Applications*, pages 449–457.

Sciascia, D. and Pedone, F. (2012). RAM-DUR: In-Memory Deferred Update Replication. *2012 IEEE 31st Symposium on Reliable Distributed Systems*, pages 81–90.

Zhang, H. and Zhao, W. (2012). Concurrent Byzantine Fault Tolerance for Software-Transaction-Memory Based Applications. *International Journal of Future Computer and Communication*, 1(1).


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 8*

## *Projeto e Análise de Redes e Serviços Avançados*

# Filas com Prioridades: Novas Metodologias de Análise

**Paulo Henrique de Aguiar Rodrigues**[1], **Daniel Sadoc Menasché**[1]

[1]Departamento de Ciência da Computação
Universidade Federal do Rio de Janeiro (UFRJ)
Av. Athos da S. Ramos, 149 – Blc C – 21.941-909
Rio de Janeiro – RJ – Brasil

aguiar@ufrj.br, sadoc@dcc.ufrj.br

***Abstract.*** *Priority queues are one of the fundamental tools to analyze the performance of communications and computer networks, and scheduling in computing systems. Although priority queues have been analyzed for more than 5 decades, their treatment in the literature usually requires a significant background. In this paper, we present a novel methodology to derive closed form solutions for the mean number of users in a queue with priorities. The methodology is significantly simpler than the other ones found in literature, and, we believe, brings novel insights about the system.*

***Resumo.*** *Filas com prioridades são uma das principais ferramentas analíticas para a análise de desempenho de redes de comunicação e de computadores, e disciplinas de escalonamento em sistemas computacionais. A análise destas filas tem sido alvo de estudo há mais de 5 décadas, mas através de derivações complexas e de difícil compreensão. Neste artigo, apresentamos uma nova abordagem para a determinação da média do número de clientes de alta e baixa prioridade em uma fila com prioridades. A abordagem baseia-se em princípios básicos de teoria de filas, e é significativamente mais simples e intuitiva que as abordagens prévias, encontradas na literatura.*

## 1. Introdução

Modelos de filas com prioridades podem ser usados para analisar políticas de QoS em aplicações multimídia [Markhasin 2008], escalonamento de processos em servidores [Gittins and Nash 1977, Wierman 2007], *data centers* [Liu et al. 2013] e aplicações *MapReduce* [Lin et al. 2013], e gerenciamento de tarefas administrativas [Gupta 2013]. Sistemas inteligentes envolvendo controladores, como *Software Defined Networks* (SDN) e OpenFlow, podem fazer uso de prioridades para melhorar o desempenho [McKeown et al. 2008]. Rádios cognitivos constituem outro território fértil para a aplicação de prioridades na decisão das ações de usuários primários e secundários. Dessa forma, filas com prioridades constituem uma ferramenta fundamental de análise, sendo pesquisadas há mais de 5 décadas [Jaiswal 1968, Harchol-Balter 2013].

Dada a importância do uso de filas com prioridade na avaliacão de desempenho de diferentes cenários modernos, é surpreendente que não haja nenhuma abordagem intuitiva para o entendimento e derivação de propriedades básicas das mesmas. Mais além, os livros texto padrões de teoria de filas [Kleinrock 1976, Harchol-Balter 2013, Stewart 2009,

Le Boudec 2010] não tratam os casos com preempção sem continuidade, onde o serviço interrompido de menor pioridade recomeça depois de onde parou, devido à dificuldade da análise. Nosso objetivo é preencher esta lacuna, apresentando metodologia simples para derivar o tempo médio experimentado pelos usuários em diferentes cenários.

Num sistema com prioridade sem preempção, o usuário de baixa prioridade, após entrar em serviço, não pode ser mais interrompido. Este cenário é amplamente estudado em livros-texto (e.g., [Harchol-Balter 2013]), em vídeo [Menasché 2013] e analisado em [Rodrigues 2013a]. Por isso, o artigo abordará apenas cenários com preempção.

Neste artigo apresentamos duas abordagens gerais, que podem ser adaptadas para análise de inúmeros tipos de filas com prioridades, incluindo os casos com preempção e sem continuidade. A abordagem padrão para a análise de filas com prioridades com preempção sem continuidade consiste na obtenção da transformada do número de clientes no sistema, em função do tempo e de condições iniciais. As análises resultam em transformadas de Laplace multidimensionais (assunto em geral não coberto em livros texto elementares) e expressões de difícil interpretação mesmo para um especialista na área, sem qualquer aspecto intuitivo. Nossas abordagens, em contraste, dependem apenas de transformadas com um único parâmetro, sendo de fácil derivação e interpretação, e constituem uma contribuição importante para o entendimento de filas com prioridades.

O restante deste artigo está organizado da seguinte forma. Na seção 2 apresentamos o sistema a ser considerado. Nas seções 3 e 4 derivamos as duas abordagens propostas. A seguir, em cada uma das seções subsequentes, especializamos os resultados para cenários de especial interesse, a saber, filas com prioridades com preempção e continuidade (o serviço continua de onde parou, vide seção 5), e filas com prioridades com preempção e sem continuidade, com reamostragem ( aloca um novo serviço a cada reentrada no servidor, vide seção 6) e sem reamostragem (repete o mesmo serviço a cada reentrada no servidor, vide seção 7). A seção 8 discute trabalhos relacionados e a seção 9 traz a conclusão e considerações finais.

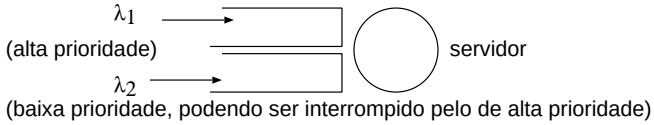



**Figura 1. Cenário geral. Clientes do tipo 1 tem prioridade sobre clientes do tipo 2 e dentro de cada classe a disciplina de atendimento é sempre FCFS.**

## 2. Sistema com Prioridade

O sistema considerado é apresentado na Figura 1. Clientes de alta e baixa prioridades chegam a um servidor único, segundo fluxos Poisson com taxas $\lambda_1$ e $\lambda_2$, respectivamente. Seja $\lambda = \lambda_1 + \lambda_2$. Clientes de alta prioridade (tipo 1) demandam serviço que é caracterizado pela variável aleatória $X_1$ (pdf $f_{X_1}(x)$) , com transformada de Laplace (T.L.) $X_1^\star(s)$, enquanto o serviço dos que clientes de baixa prioridade (tipo 2) é dado pela variável aleatória $X_2$ (pdf $f_{X_2}(x)$) , com T.L. $X_2^\star(s)$. Em cada classe, o atendimento é sempre na ordem de chegada (FCFS), implicando que, em cenários com preempção (Seções 5-7), o cliente interrompido seja sempre o próximo da classe a ser atendido.

As métricas de interesse são o tempo de espera na fila de espera (sem contar o tempo de serviço) dos clientes do tipo 1 e 2, $W_1$ e $W_2$, respectivamente, com médias (ou esperanças) dadas por $E[W_1]$ e $E[W_2]$. O tempo de permanência dos usuários do tipo 1 e tipo 2 no sistema é dado por $T_1$ e $T_2$, respectivamente, onde $T_i = W_i + X_i$, $i = 1, 2$. A partir dos tempos médios e usando o resultado de Little, pode-se obter o número médio na fila de espera e no sistema.

## 3. Metodologia geral I: analisando os ciclos

Para os clientes do tipo 1, seja $E[X_{1r}]$ a vida residual do serviço e $\rho_1 = \lambda_1 E[X_1]$, o número médio destes clientes no servidor, grandeza também conhecida como utilização. Como $E[W_1]$ independe da presença dos clientes do tipo 2 para sistemas com preempção, tem-se diretamente pela fórmula de Pollaczek-Khinchin para valor médio em M/G/1 que, para $\rho_1 < 1$, $E[W_1] = \rho_1 E[X_{1r}]/(1 - \rho_1)$ e $E[T_1] = E[W_1] + E[X_1]$.

Para calcular o tempo médio de espera dos clientes do tipo 2, vamos acompanhar $C_2$, um freguês típico do tipo 2, através do sistema desde a sua chegada. Quando $C_2$ chega ao sistema, ele pode encontrar uma dentre três situações:

- sistema vazio (evento $V$)
- sistema ocupado com período ocupado iniciado por cliente do tipo 2 (evento $O_2$)
- sistema ocupado com período ocupado iniciado por cliente do tipo 1 (evento $O_1$)

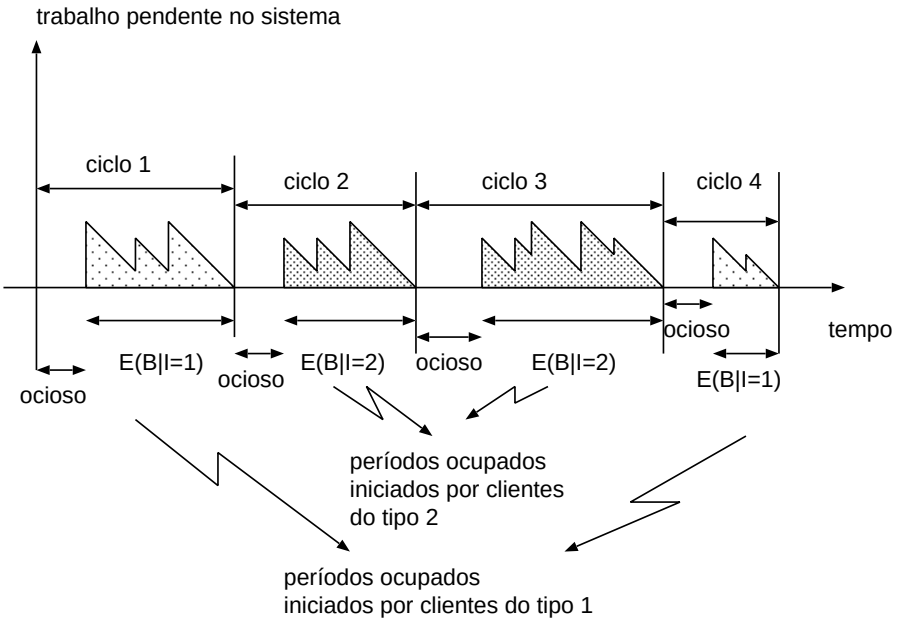



**Figura 2. Ciclos do sistema**

A probabilidade de um período ocupado iniciar com uma chegada do tipo $i$ é

$$P(I = i) = \frac{\lambda_i}{\lambda_1 + \lambda_2} = \frac{\lambda_i}{\lambda}, i = 1, 2 \quad (1)$$

O tempo médio que o servidor fica ocioso, contado desde o instante que o servidor fica vazio até ocorrer uma chegada, é $1/\lambda$.

Seja $E[B|I=i]$ a duração média de um período ocupado iniciado por um cliente do tipo $i$. Usando a teoria da renovação, e observando os ciclos ilustrados na Figura 2, percebemos que a probabilidade de uma chegada encontrar o servidor vazio é,

$$P(V) = \frac{1/\lambda}{E[B] + 1/\lambda} = \frac{1}{\lambda_2 E[B|I=2] + \lambda_1 E[B|I=1] + 1}$$

De forma semelhante, a probabilidade de uma chegada encontrar o servidor ocupado com um período ocupado iniciado por um cliente do tipo $i$ é,

$$P(O_i) = \frac{\lambda_i E[B|I=i]}{\lambda_2 E[B|I=2] + \lambda_1 E[B|I=1] + 1}$$

Dessa forma, $E[W_2] = E[W_2|V]P(V) + E[W_2|O_1]P(O_1) + E[W_2|O_2]P(O_2)$ e, como $E[W_2|V] = 0$, temos

$$E[W_2] = \frac{E[W_2|O_1]\lambda_1 E[B|I=1] + E[W_2|O_2]\lambda_2 E[B|I=2]}{\lambda_2 E[B|I=2] + \lambda_1 E[B|I=1] + 1} \quad (2)$$

Inicialmente, vamos deduzir as médias condicionadas ao evento $O_2$, quando $C_2$ encontra no sistema o cliente do mesmo tipo que iniciou o período ocupado.

Seja $D_2$ o tempo entre o instante que $C_2$ entra em serviço pela primeira vez até sair do sistema. O tempo até $C_2$ entrar no servidor pode ser imaginado com o tempo de espera em uma fila virtual M/G1, na qual clientes do tipo 2 chegam com taxa $\lambda_2$ e são servidos por um tempo $D_2$, formado pelo tempo de serviço $X_2$ e pelos serviços de todos os clientes do tipo 1 que interrompem o cliente do tipo 2 até que ele saia do sistema. Quando o cliente é interrompido, assume-se na fila virtual que ele continua em serviço. Para $\rho_2^{\#} = \lambda_2 E[D_2] < 1$, o tempo médio de espera será na fila virtual dado pela fórmula da M/G/1, $E[W_2^{\#}] = \frac{\rho_2^{\#} E[D_2^r]}{1-\rho_2^{\#}}$, e a vida residual do serviço dada por $E[D_2^r] = \frac{E[D_2^2]}{2E[D_2]}$.

Assumindo $N_2^{\#} =$ o número de clientes na fila virtual, temos

$$E[W_2|O_2] = E[W_2^{\#}|N_2^{\#} > 0] = \frac{E[W_2^{\#}]}{P(N_2^{\#} > 0)} = \frac{E[D_2^r]}{1-\rho_2^{\#}}, \;\; \rho_2^{\#} < 1 \quad (3)$$

O período ocupado médio iniciado por um cliente do tipo 2 é igual ao período ocupado médio da fila virtual descrita acima e então

$$E[B|I=2] = \frac{E[D_2]}{1-\rho_2^{\#}}, \rho_2^{\#} < 1 \quad (4)$$

A seguir, vamos analisar o evento $O_1$, onde $C_2$ encontra um período ocupado iniciado por um freguês do tipo 1. Considerando $B_1$ como o período ocupado de uma fila M/G/1 alimentada apenas por fregueses do tipo 1, $E[B|I=1]$ pode ser visto como o período ocupado médio da fila virtual acima modificada, onde o primeiro serviço em cada período ocupado é dado por $B_1$ e os demais são dados por $D_2$.

Sabendo que $E[B_1] = \frac{E[X_1]}{1-\rho_1}, \rho_1 < 1$, temos

$$E[B|I=1] \;\; = \;\; \frac{E[B_1]}{1-\rho_2^{\#}} = \frac{E[X_1]}{(1-\rho_1)(1-\rho_2^{\#})}, \rho_2^{\#} < 1 \quad (5)$$

Resta-nos calcular $E[W_2|O_1]$. Seja $W_0$ o tempo de espera na fila M/G/1 virtual modificada descrita acima e seja $N_0$ o número de clientes nessa fila. $E[W_0]$ pode ser obtido via cadeia embutida (vide [Harchol-Balter 2013] ou [Rodrigues 2013a, pg 87]). Então,

$$E[W_2|O_1] = \frac{E[W_0]}{P(N_0 > 0)} = \frac{E[W_0]}{1 - P(N_0 = 0)} = \frac{\frac{\lambda_2 E[D_2^2]}{2(1-\rho_2^{\#})} + \frac{\lambda_2 E[B_1^2] - \lambda_2 E[D_2^2]}{2(1-\rho_2^{\#} + \lambda_2 E[B_1])}}{\frac{\lambda_2 E[B_1]}{1-\lambda_2 E[D_2]+\lambda_2 E[B_1]}} \quad (6)$$

Substituindo (4), (3), (5) e (6) em (2), obtemos $E[W_2] = E[W_2^{\#}] + \rho_1 E[B_1^r]$, onde é conhecido que $E[B_1^r] = \frac{E[B_1^2]}{2E[B_1]} = \frac{E[X_1^2]}{2(1-\rho_1)^2 E[X_1]}$. Consequentemente,

$$E[T_2] = E[D_2] + E[W_2] = E[D_2] + E[W_2^{\#}] + \rho_1 E[B_1^r] \quad (7)$$

Analisando $E[T_2]$, o termo $E[D_2]+E[W_2^{\#}]$ representa o tempo médio na fila virtual, onde os serviços são sempre do tipo $D_2$. Todavia, nas situacões em que $C_2$ chega ao sistema e encontra um serviço iniciado por um freguês do tipo 1, o período ocupado residual dos fregueses do tipo 1 pode ser visto como um tempo de férias do servidor. A teoria clássica de filas mostra que $E[T_{ferias}] = E[T_{M/G/1}]$ + vida residual das férias, o que justifica nossa interpretação e justifica como escrever a equação de $E[T_2]$ sem qualquer dedução. Como já observado, a condição de estabilidade para a classe 2 é dada por $\rho_2^{\#} < 1$.

### 3.1. Comparação com resultados da literatura

De acordo com as equações (3.21) do cap. IV do livro *Priority Queues*, Jaiswal e eq (6.6) do cap. III do mesmo livro, temos,

$$E[N_2] = \lambda_2 E[D_2] + \frac{\lambda_1 \lambda_2 E[X_1^2]}{2(1-\rho_1)^2} + \frac{\lambda_2^2 E[D_2^2]}{2(1-\lambda_2 E[D_2])}$$

Fazendo uma simples manipulação álgebrica da expressão de $E[N_2]$ acima obtemos

$$\begin{aligned} E[N_2] &= \lambda_2 \left( E[D_2] + \lambda_1 \frac{E[B_1^2]}{2(1+\lambda_1 E[B_1])} + \frac{\lambda_2 E[D_2^2]}{2(1-\lambda_2 E[D_2]} \right) \\ &= \lambda_2 \Big( E[D_2] + \rho_1 E[B_1^r] + E[W_2^{\#}] \Big) \end{aligned}$$

demonstrando, por Little, que obtemos o mesmo resultado que Jaiswal para $E[T_2]$.

## 4. Metodologia geral II: foto do sistema no instante de chegada

Apresentamos agora a segunda metodologia para análise de filas com prioridades. A metodologia consiste em fazer uma tirada de foto virtual do sistema no instante de chegada do cliente típico ao sistema. Nessa foto, o cliente típico registra o número médio de pessoas de cada tipo que ele encontra no sistema e que devem ser servidas antes dele. A partir desses valores e considerando o tempo de serviço de cada pessoa em média, ele pode identificar o trabalho médio pendente até que ele possa entrar no servidor (quando se obter $E[W_2]$ para sistemas não preemptivos) ou sair do sistema (quando se procura obter $E[T_2]$ para sistemas preemptivos), imaginando-se que não ocorram mais chegadas ao sistema. Caso esteja-se considerando um sistema sem preempção, calcula-se o tempo

médio até entrar-se no servidor. Caso esteja-se considerando um sistema com preempção, calcula-se o tempo médio até a saida do sistema. Em ambos os casos, em primeiro passo, não se leva em conta o impacto dos períodos ocupados gerados pelos clientes de mais alta prioridade que chegarem, enquanto os clientes encontrados pelo cliente típico e ele próprio estiverem ainda no sistema. Finalmente, expande-se o tempo de espera para levar em conta estes períodos ocupados usando a expressão do período ocupado modificado (quando o tempo do serviço inicial é diferente de todos os outros) e obtém-se a solução final para o problema. Resumimos na Tabela 1 as principais ideias envolvidas em tal abordagem, focando nos clientes do tipo 2 (menos prioritários). A Tabela 2 resume as definições das variáveis envolvidas nas deduções analíticas.

| tipo de fila | métrica de interesse | trab. pendente na chegada | per. ocup. modificado |
|---|---|---|---|
| sem preempção | $E[W]$, tempo em fila de espera | $E[W_{02}]$ | $E[W_2] = \frac{E[W_{02}]}{1-\rho_1}$ |
| com preempção | $E[T]$, tempo no sistema | $E[T_{02}]$ | $E[T_2] = \frac{E[T_{02}]}{1-\rho_1}$ |

**Tabela 1. Metodologia da foto para análise de filas FCFS com duas prioridades**

| variável | descrição |
|---|---|
| $T_1$ | tempo no sistema do cliente de alta prioridade |
| $T_2$ | tempo no sistema do cliente de baixa prioridade |
| $W_2^{\#}$ | tempo de espera do cliente de baixa prioridade na fila de espera virtual |
| $B_1^r$ | vida residual do período ocupado dos clientes de alta prioridade |
| $D_2$ | tempo de serviço dos clientes de baixa prioridade na fila virtual |
| $D_2^r$ | vida residual de $D_2$ |
| $\rho_1$ | utilização do servidor por clientes de alta prioridade, $\rho_1 = \lambda_1 E[X_1]$ |
| $\rho_2$ | utilização do servidor por clientes de baixa prioridade, $\rho_2 = \lambda_2 E[X_{t2}]$ |
| $\rho_2^{\#}$ | utilização do servidor virtual, $\rho_2^{\#} = \lambda_2 E[D_2]$ |
| $X_{t2}$ | tempo no servidor desde que cliente entra pela primeira vez, até sair do sistema, $E[X_{t2}] = (1-\rho_1)E[D_2]$ |
| $X_{t2}^{\#}$ | vida residual de $X_{t2}$ |

**Tabela 2. Tabela de notação. A fila virtual refere-se a uma fila M/G/1 com taxa de chegadas $\lambda_2$ e tempo médio de serviço $1/\mu_2^{\#} = E[D_2]$.**

## 5. Preempção com continuidade: após uma interrupção, o serviço do menos prioritário recomeça de onde parou

Neste sistema, os clientes do tipo 2 podem ter o serviço interrompido pela chegada de clientes mais prioritários do tipo 1. Quando não mais existirem clientes prioritários no sistema, o cliente interrompido retorna e recomeça o serviço de onde parou. Um exemplo ilustrativo deste cenário envolve tarefas que podem ser interrompidas por um rejuvenescimento do monitor de máquina virtual (VMM) no servidor. VMM pode esperar que o processo em serviço termine (nesse caso, temos prioridade sem preempção) ou então salvar o estado atual e reiniciar o processo a partir do ponto de interrupção, quando temos prioridade com preempção e continuidade [Machida et al. 2014]. Queremos obter $E[T_2]$.

### 5.1. Usando a metodologia geral I: ciclos

No sistema com preempção com continuidade, $D_2$, o serviço virtual, é formado pelo serviço $X_2$ do freguês típico $C_2$ e da soma dos períodos ocupados $B_{1,k}$ formados pelos serviços do tipo 1, a cada interrupção de serviço de $C_2$. Sendo $K$ o número de interrupções sofridas por $C_2$, a teoria indica que $K(z) = X_2^\star(\lambda_1 - \lambda_1 z)$.

O tempo médio na fila espera virtual é dado por

$$E[W_2^\#] = \frac{\rho_2^\# E[X_2^{r\#}]}{1-\rho_2^\#} = \frac{\lambda_2 E[D_2]E[D_2^r]}{1-\lambda_2 E[D_2]} = \frac{\lambda_2 E[D_2]E[D_2^r]}{1-\lambda_2 E[D_2]} = \frac{\lambda_2 E[D_2^2]/2}{1-\lambda_2 E[D_2]} \quad (8)$$

Dado $D_2^\star(s)$, $E[D_2] = -D_2^{'\star}(0)$ e $E[D_2^2] = D_2^{''\star}(0)$. Conhecido $K(z)$ e $E[B_{1,k}] = E[B_1] = \frac{E[X_1]}{1-\rho_1}$, $\rho_1 < 1$, e dado que $D_2 = \sum_{j=0}^{K} B_{1,j} + X_2$, $D_2^\star(s)$ pode ser obtida através de condicionamento duplo, visto que $X_2$ e $K$ são dependentes. Temos

$$\begin{aligned} D_2^\star(s) &= \int_{x_2=0}^{\infty} e^{-sx_2} \sum_{k=1}^{K} E[e^{-sB_1}]^k P(K=k|x_2 < X_2 < x_2 + dx_2) f_{X_2}(x_2) dx_2 \\ &= \int_{x_2=0}^{\infty} e^{-sx_2} K(B_1^\star(s)|x_2 < X_2 < x_2 + dx_2) f_{X_2}(x_2) dx_2 \\ &= \int_{x_2=0}^{\infty} e^{-sx_2} e^{-\lambda_1(1-B_1^\star(s))x_2} f_{X_2}(x_2) dx_2 = X_2^\star(\lambda_1 - \lambda_1 B_1^\star(s) + s) \end{aligned}$$

Logo,

$$E[D_2] = -X_2^{'\star}(0)(1-\lambda_1 B_1^{'\star}(0)) = \frac{E[X_2]}{1-\rho_1} \quad (9)$$

$$E[D_2^2] = X_2^{''\star}(0)(1+\lambda_1 E[B_1])^2 - \lambda_1 X_2^{'\star}(0) B_1^{'\star}(0) = \frac{E[X_2^2]}{(1-\rho_1)^2} + \frac{\rho_2 E[X_1^2]}{(1-\rho_1)^3} \quad (10)$$

Usando (9) e (10) em (7), obtém-se o tempo médio de espera $E[T_2]$. A estabilidade da fila 2 requer $\rho_2^\# = \lambda_2 E[D_2] = \frac{\rho_2}{1-\rho_1} < 1$, implicando em $\rho_1 + \rho_2 < 1$.

### 5.2. Usando a metodologia geral II: fotos

Com esta metodologia, para a obtenção de $E[T_2]$, identificamos o trabalho médio pendente na chegada de $C_2$ e aplicamos o conceito do período ocupado modificado, como indicado na seção 4. Como o atendimento é FCFS em cada classe e o servidor está sempre ocupado quando o sistema não esta vazio (sistema conservativo), o trabalho médio pendente $E[T_{02}]$ pode ser interpretado como o tempo médio de espera em uma fila M/G/1 alimentada pelos dois tipos de clientes mais o serviço $E[X_2]$ do próprio $C_2$. Então, $E[T_{02}] = E[X_2] + \frac{\rho_1 E[X_{1r}] + \rho_2 E[X_{2r}]}{1-\rho_1-\rho_2}$, $\rho_1 + \rho_2 < 1$.

Lembrando que $E[T_2] = \frac{E[T_{02}]}{1-\rho_1}$, obtém-se

$$E[T_2] = \frac{E[X_2](1-\rho_1-\rho_2) + \rho_1 E[X_{1r}] + \rho_2 E[X_{2r}]}{(1-\rho_1)(1-\rho_1-\rho_2)}, \quad \rho_1 + \rho_2 < 1 \quad (11)$$

É fácil mostrar que a solução acima é igual à obtida usando a metodologia geral via ciclos.

## 6. Preempção e reamostragem da classe menos prioritária: um novo serviço $X_2$ é alocado a cada reentrada no servidor

Neste sistema, a cada reentrada da classe menos prioritária no servidor, um novo serviço $X_2$ é alocado. Como exemplo de aplicação, em redes de rádios cognitivos, quando o usuário primário entra em cena os usuários secundários são imediatamente interrompidos. Considere um canal altamente dinâmico (*fading* varia muito em função do tempo) [Cabric et al. 2004]. Quando o canal fica disponível novamente para os usuários secundários, o canal já mudou de estado, o que implica em um novo tempo de serviço para a classe 2, menos prioritária, caracterizando o modelo premptivo com reamostragem.

### 6.1. Usando metodologia geral I: ciclos

Seja $D_2$ o tempo que um cliente leva desde que entra no servidor pela primeira vez até sair do sistema. Seja $Y$ uma variável exponencialmente distribuida com taxa $\lambda_1$ e que representa o intervalo entre chegadas dos clientes mais prioritários. Quando o serviço de $C_2$, o freguês típico do tipo 2, é interrompido pela j-ésima vez, o serviço interrompido é dado por $R_j = Y|Y < X_2$. Após a interrupção, segue um período ocupado $B_1$. Somente ao final de $B_1$, $C_2$ retorna ao servidor, iniciando um novo serviço $X_2$. O tempo entre retornos ao servidor após a $j$-ésima interrupção é dado por $Z_j = R_j + B_1$. Em sua última passagem pelo servidor, $C_2$ não sofre interrupção e o serviço é dado por $V = X_2|X_2 < Y$.

No que segue, seja $N$ o número de interrupções sofridas por $C_2$. A probabilidade de um serviço $X_2$ não ser interrompido por uma chegada do fluxo 1 é dada por $X_2^\star(\lambda_1)$ e a probabilidade de ser interrompido é $1 - X_2^\star(\lambda_1)$. É imediato então verificar que a variável $N$ terá uma distribuição geométrica com parâmetro $X_2^\star(\lambda_1)$.

Então $D_2 = \sum_{j=0}^{N} Z_j + V$, com $Z_0 = 0$. Temos

$$E[D_2] = E[N](E[[Y|Y < X_2] + E[B_1]) + E[X_2|X_2 < Y] \tag{12}$$

e as esperanças condicionais acima são facilmente obtidas por condicionamento [Rodrigues 2013a]. Pode-se mostrar que $D_2^\star(s) = N(Z_i^\star(s))V^\star(s)$ e

$$D_2''^\star(s) = N''(1)(Z_i'^\star(s))^2 + 2N'(1)Z_i'^\star(s)V'^\star(s) + V''^\star(s) + N'(1)Z_i''^\star(s) \tag{13}$$

$E[D_2^2]$ será dado por $D_2''^\star(0)$. Usando $E[D_2]$ e $E[D_2^2]$ em (7), obtem-se $E[T_2]$.

### 6.2. Usando metodologia geral II: fotos

#### 6.2.1. Tempo de permanência efetiva no servidor

Seja $E[X_{t2}]$ o tempo médio que um cliente do tipo 2 passa no servidor desde o instante em que entra no servidor pela primeira vez até finalmente sair. Assim, a utilização da classe 2 será dada por $\rho_2 = \lambda_2 E[X_{t2}]$. Sendo $D_2$ o tempo entre entrar em serviço pela primeira vez até sair do sistema e que durante este tempo $\lambda_1 E[D_2]$ clientes do tipo 1 chegaram e foram atendidos antes de $C_2$, então $E[X_{t2}] = E[D_2] - \lambda_1 E[D_2]E[X_1] = (1 - \rho_1)E[D_2]$, $E[D_2]$ dado por (12). Embora existam outras formas de deduzir a relacão entre as duas grandezas, esta é a mais simples. Da relacão temos $\rho_2 = (1 - \rho_1)\rho_2^\#$ e $\rho_2^\# < 1$ implica em $\rho_1 + \rho_2 < 1$.

### 6.2.2. Tempo de permanência no sistema

Ao chegar, $C_2$ encontra na média $E[N_{q1}]$ clientes do tipo 1 na fila de espera, cada um deles requerendo $E[X_1]$ de serviço, e $\rho_1$ clientes deste tipo no servidor, com $E[X_{1r}]$ de serviço residual. Então, o trabalho pendente do tipo 1 na chegada é $E[N_{q1}]E[X_1] + \rho_1 E[X_{1r}] = E[W_1]$, fórmula básica de M/G/1. Do tipo 2, são encontrados $E[N_{q2}] = \lambda_2 E[W_2]$ clientes na fila de espera, cada um deles requerendo $E[X_{t2}]$ de serviço, e $\rho_2 = \lambda_2 E[X_{t2}]$ clientes em serviço, com um servico residual $E[X_{t2r}]$.

Observando que $E[W_2] = E[T_2] - E[X_{t2}]$ e que o servíco de $C_2$ será também $E[X_{t2}]$, temos $E[T_{02}] = E[W_1] + \rho_2 E[T_2] - \rho_2 E[X_{t2}] + \rho_2 E[X_{t2r}] + E[X_{t2}]$. Como todo este tempo é expandido pelas chegadas prioritárias, temos $E[T_2] = \frac{E[T_{02}}{1-\rho_1}$. Então,

$$E[T_2] = \frac{E[W_1] + \rho_2 E[X_{t2r}] + (1-\rho_2)E[X_{t2}]}{1-\rho_1-\rho_2}, \rho_1 + \rho_2 < 1 \qquad (14)$$

Para o cálculo de $E[T_2]$, a variável $E[X_{t2r}]$ tem que ser obtida, o que será feito a seguir.

### 6.2.3. Obtenção de $E[X_{t2r}]$

Nosso objetivo é calcular $E[X_{t2r}]$, a vida residual do serviço do tipo 2 na chegada de $C_2$. A Figura 3 mostra a ocupacão do servidor no tempo, quando apenas clientes do tipo 2 são considerados. O servidor passa por ciclos, onde cada ciclo é formado por um serviço do tipo 2, seja ele interrompido ou não por uma chegada do tipo 1. Uma chegada $C_2$ pode encontrar um dos dois tipos de serviço 2 em andamento. Para cálculo dos ciclos, seja $Y$ uma variável exponencial com taxa $\lambda_1$. A duração de um ciclo não interrompido

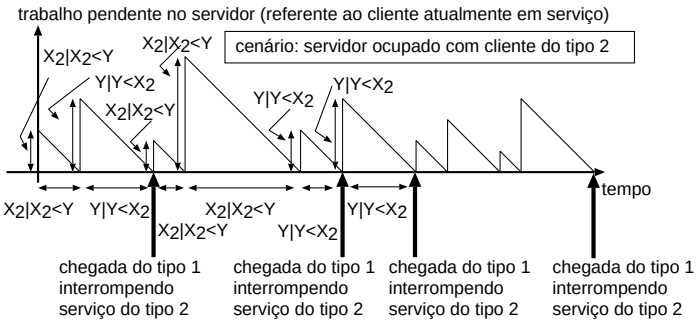



**Figura 3. Ciclos de serviço dos clientes do tipo 2**

é $X_2|X_2 < Y$ e a duração de um ciclo interrompido é $Y|Y < X_2$. Para cada ciclo não interrompido temos, em média, $\frac{1-X_2^\star(\lambda_1)}{X_2^\star(\lambda_1)}$ ciclos interrompidos. Note que a fração de ciclos não interrompidos é $X_2^\star(\lambda_1)$. Será preciso calcular as vidas residuais médias dos ciclos acima, de modo que teremos que calcular a média e o segundo momento das variáveis condicionais. Os momentos condicionais podem ser obtidos a partir de transformadas (veja deduções em [Rodrigues 2013a]), como se segue

$$E[e^{-sY}|Y < X_2] = \frac{\lambda_1(1 - X_2^\star(\lambda_1 + s))}{(\lambda_1 + s)(1 - X_2^\star(\lambda_1))}, \quad E[e^{-sX_2}|X_2 < Y] = \frac{X_2^\star(s + \lambda_1)}{X_2^\star(\lambda_1)}$$

As vidas residuais médias de $(X_2|X_2 < Y)$ e $(Y|Y < X_2)$ são por definição

$$E[(X_2|X_2 < Y)_r] = \frac{E[X_2^2|X_2 < Y]}{2E[X_2|X_2 < Y]}, \;\; E[(Y|Y < X_2)_r] = \frac{E[Y^2|Y < X_2]}{2E[Y|Y < X_2]}$$

Seja $q$ a probabilidade de se encontrar o servidor ocupado com um cliente do 2 que não tem seu serviço interrompido. Usando a teoria da renovação, temos que

$$q \;\; = \;\; \frac{E[X_2|Y > X_2]}{E[X_2|Y > X_2] + E[Y|Y < X_2]\frac{1-X_2^\star(\lambda_1)}{X_2^\star(\lambda_1)}}$$

Seja $R_0$ a vida residual do primeiro subserviço do cliente do tipo 2 encontrado no servidor.

$$E[R_0] = qE[(X_2|Y > X_2)_r] + (1-q)E[(Y|Y < X_2)_r] \tag{15}$$

onde o primeiro termo se refere ao serviço que não é interrompido e o segundo termo se refere ao serviço que é interrompido. Expandido os termos, temos

$$E[R_0] = \frac{X_2''^\star(0|X_2 < Y)X_2^\star(\lambda_1) + Y''^\star(0|Y < X_2)(1 - X_2^\star(\lambda_1))}{2(-X_2'^\star(0|X_2 < Y)X_2^\star(\lambda_1) - Y'^\star(0|Y < X_2)(1 - X_2^\star(\lambda_1)))} \tag{16}$$

Após efetuarmos os algebrismos, pode-se mostrar que $E[R_0] = \frac{1}{\lambda_1} - \frac{X_2'^\star(\lambda_1)}{X_2^\star(\lambda_1)-1}$.

Sabendo que com probabilidade $(1-q)$ o serviço será interrompido e o cliente levará novamente $X_{t2}$ em serviço até sair, o tempo total que o cliente que foi encontrado no servidor irá levar efetivamente em serviço antes de finalmente sair do sistema é

$$E[X_{t2r}] = E[R_0] + E[X_{t2}](1-q) \tag{17}$$

Em [Menasché and Rodrigues 2013] mostramos o código Matlab (pacote simbólico) que demonstra que substituindo (17) em (14) obtém-se (7).

## 7. Preempção sem reamostragem: repete o mesmo serviço de baixa prioridade anteriormente interrompido a cada reentrada no servidor

Um exemplo prático deste cenário ocorre em redes de rádios cognitivos, nas quais a condição do canal permanece praticamente inalterada ao longo do tempo. Nestas condições, o tempo de transmissão ou tempo de serviço de um pacote depende apenas do seu tamanho, sendo modelado como um sistema com preempção (perda da transmissão por interferência) e sem reamostragem, pois retransmite integralmente o mesmo pacote.

Seja $D_2$ o tempo que um cliente leva desde que entra no servidor pela primeira vez até sair do sistema. Seja $Y$ uma variável exponencialmente distribuida com taxa $\lambda_1$ e que representa o intervalo entre chegadas dos clientes mais priotários. Quando o serviço de $C_2$, o freguês típico do tipo 2, é interrompido, o serviço interrompido é dado por $R_i = Y|Y < X_2$. Após a interrupção, segue um período ocupado $B_1$. Somente ao final de $B_1$, $C_2$ retorna ao servidor, repetindo o mesmo serviço $X_2$. O tempo entre retornos ao servidor após a $i$-ésima interrupção pode ser representado como $Z_i = R_i + B_1$. Em sua última passagem pelo servidor, $C_2$ não sofre interrupção e o serviço recebido é dado $X_2$. Seja $N$ o número de interrupções sofridas por $C_2$.

Neste cenário, $D_2 = \sum_{i=0}^{N}(R_i + B_1) + X_2$. As variáveis $N$, $R_i$ e $X_2$ são dependentes. Condicionando em $x_2 < X_2 < x_2 + dx_2$, no entanto, possibilita o tratamento do problema. A probabilidade de um serviço $X_2, x_2 < X_2 < x_2 + dx_2$ não ser interrompido por uma chegada do fluxo 1 é dada por $p = e^{-\lambda_1 x_2}$ e a probabilidade do serviço ser interrompido é $1 - p$. Consequentemente, pode-se ver que a variável $N|x_2 < X_2 < x_2 + dx_2$ é geometricamente distribuída com transformada Z da pmf dada por

$$N(z|x_2 < X_2 < x_2 + dx_2) = \frac{p}{1 - z(1-p)} = \frac{e^{-\lambda_1 x_2}}{1 - z(1 - e^{-\lambda_1 x_2})} \tag{18}$$

Sabendo que $X_2^\star(s|x_2 < X_2 < x_2 + dx_2) = e^{-sx_2}$ e condicionando, temos

$$\begin{aligned} D_2^\star(s) &= \int_{x_2=0}^{\infty} D_2^\star(s|x_2 < X_2 < x_2 + dx_2) f_{X_2}(x_2) dx_2 \\ &= \int_{x_2=0}^{\infty} N(R_i^\star(s) B_1^\star(s)|x_2 < X_2 < x_2 + dx_2) e^{-sx_2} f_{X_2}(x_2) dx_2 \\ &= \int_{x_2=0}^{\infty} \frac{p}{1 - R_i^\star(s|x_2 < X_2 < x_2 + dx_2) B_1^\star(s)(1-p)} e^{-sx_2} f_{X_2}(x_2) dx_2 \end{aligned} \tag{19}$$

Como $R_i^\star(s|x_2 < X_2 < x_2 + dx_2) = \left(\frac{\lambda_1}{\lambda_1 + s}\right)\left(\frac{1 - e^{-(s+\lambda_1)x_2}}{1 - e^{-\lambda_1 x_2}}\right)$, obtemos finalmente

$$D_2^\star(s) = \int_{x_2=0}^{\infty} \frac{e^{-(\lambda_1+s)x_2}}{1 - \left(\frac{\lambda_1}{\lambda_1+s}\right) B_1^\star(s)(1 - e^{-(\lambda_1+s)x_2})} f_{X_2}(x_2) dx_2 \tag{20}$$

Finalmente, derivando (20) podemos obter a média e o segundo momento de $D_2$,

$$E[D_2] = \left(\frac{1}{\lambda_1} + E[B_1]\right)(X_2^\star(-\lambda_1) - 1) \tag{21}$$

$$\begin{aligned} E[D_2^2] &= 2\left(\frac{1}{\lambda_1} + E[B_1]\right)^2 E\left[(e^{\lambda_1 X_2} - 1)^2\right] + \left[E[B_1^2] + \frac{2E[B_1]}{\lambda_1} + \frac{2}{\lambda_1^2}\right] \\ &\quad \times (X_2^\star(-\lambda_1) - 1) - 2\left(\frac{1}{\lambda_1} + E[B_1]\right)(-X_2'^\star(-\lambda_1)) \end{aligned} \tag{22}$$

As expressões acima para os momentos podem ser encontradas na página 89 de [Jaiswal 1968]. Usando (21) e (22) em (7), obtém-se o tempo médio de espera $E[T_2]$. A condição de estabilidade continua sendo $\rho_2^{\#} = \lambda_2 E[D_2] < 1$.

## 8. Trabalhos relacionados

A literatura sobre filas com prioridades é vasta [Jaiswal 1968, Harchol-Balter 2013, Gaver Jr 1962, Simatos et al. 2013]. Entretanto, não conhecemos nenhum trabalho anterior que tenha apresentado os resultados sobre filas com prioridades da forma que fazemos neste artigo, utilizando duas abordagens simples. Os livros texto sobre teoria de filas [Kleinrock 1976, Harchol-Balter 2013, Stewart 2009, Le Boudec 2010] não consideram a análise de filas com prioridades com preempção e sem continuidade. Livros [Jaiswal 1968] e artigos [Gaver Jr 1962, Gaver Jr 1959], por outro lado, não apresentam a intuição mostrada neste trabalho. Vale destacar que [Jaiswal 1968, Gaver Jr 1962,

Gaver Jr 1959] apresentam uma análise menos intuitiva, mas ao mesmo tempo mais geral, dado que os autores derivam também distribuições. Na literatura existem inúmeros exemplos numéricos ilustrando a aplicabilidade dos resultados apresentados neste artigo (e.g., [Jaiswal 1968, pg. 97-98]), e, por limitação de espaço, focamos estritamente em nossa contribuição, que consiste nas novas metodologias para análise de filas com prioridades.

Em engenharia de sistemas, visa-se soluções rápidas e eficientes para tomada de decisões importantes. Muitas vezes, estas decisões precisam ser tomadas com poucos dados, já que por motivos de privacidade ou dificuldade de coleta torna-se impeditivo ter acesso a uma grande quantidade de dados. Assim sendo, médias muitas vezes são suficientes para guiar a tomada de decisões. Ter uma intuição sobre os fatores que afetam as médias é de grande valia na construção de heurísticas e soluções, mesmo que sem a variância [Ben-Haim 2012].

A metodologia apresentada neste trabalho é ilustrada em dois vídeos disponibilizado no Youtube [Menasché 2013]. Ela pode ser facilmente estendida para a análise de filas com reentrada, como apresentado na apostila [Rodrigues 2013a] e nos slides [Rodrigues 2013b].

## 9. Conclusão e Considerações Finais

Neste trabalho apresentamos duas novas abordagens intuitivas para obtenção de valores médios (número médio de clientes, tempo no sistema) em sistemas de filas com prioridades. Consideramos duas filas com prioridades, a fila 1 com prioridade sobre a fila 2. A extensão dos resultados aqui apresentados para o caso de múltiplas filas é natural. Por exemplo, para analisar o caso de 3 filas, basta considerar primeiro a interação entre a primeira e a segunda filas, ignorando a terceira. Em seguida, analisa-se a terceira fila levando em conta o fato de que, os usuários da terceira fila, ao serem interrompidos terão que esperar por um período ocupado composto por usuários das filas 1 e 2.

A avaliação numérica dos resultados apresentados nesse artigo pode ser facilmente executada usando a ferramenta de modelagem e análise Tangram II [de Souza e Silva and Leão 2000], comprovando a corretude das fórmulas obtidas. No site [Menasché and Rodrigues 2013] disponibilizamos os modelos do Tangram II associados aos diferentes cenários apresentados neste artigo. Caso os tempos de serviço sejam caracterizados por variáveis exponenciais, pode-se resolver a cadeia de Markov associada ao problema de via métodos diretos ou iterativos. Caso contrário, os modelos precisam ser resolvidos via simulação. Dessa forma, os resultados aqui apresentados contribuem para um melhor entendimento sobre filas com prioridades, que constituem peça chave no estudo de redes de computadores, além de servirem para a obtenção de limitantes (*bounds*) simples para métricas de interesse como o tempo médio no sistema [Rodrigues 2013a].

## Referências

Ben-Haim, Y. (2012). Doing our best: Optimization and the management of risk. *Risk Analysis*, 32(8):1326–1332.

Cabric, D., Mishra, S. M., and Brodersen, R. W. (2004). Implementation issues in spectrum sensing for cognitive radios. In *Asilomar Conference on Signals, systems and computers*, pages 772–776. IEEE.

de Souza e Silva, E. and Leão, R. M. (2000). The tangram-ii environment. In *Computer Performance Evaluation. Modelling Techniques and Tools*, pages 366–369. Springer.

Gaver Jr, D. (1962). A waiting line with interrupted service, including priorities. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 73–90.

Gaver Jr, D. P. (1959). Imbedded markov chain analysis of a waiting-line process in continuous time. *The Annals of Mathematical Statistics*, pages 698–720.

Gittins, J. and Nash, P. (1977). Scheduling, queues and dynamic allocation indices. In *Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pages 191–202. Springer.

Gupta, D. (2013). Queueing models for healthcare operations. In *Handbook of Healthcare Operations Management*, pages 19–44. Springer.

Harchol-Balter, M. (2013). *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press.

Jaiswal, N. K. (1968). *Priority queues*, volume 50. Academic Press New York.

Kleinrock, L. (1976). *Queueing systems: volume 2: computer applications*, volume 82. John Wiley & Sons New York.

Le Boudec, J.-Y. (2010). *Performance evaluation of computer and communication systems*. EPFL Press.

Lin, M., Zhang, L., Wierman, A., and Tan, J. (2013). Joint optimization of overlapping phases in mapreduce. *Performance Evaluation*, 70(10):720–735.

Liu, Z., Wierman, A., Chen, Y., Razon, B., and Chen, N. (2013). Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Performance Evaluation*, 70(10):770–791.

Machida, F., Nicola, V. F., and Trivedi, K. S. (2014). Job completion time on a virtualized server with software rejuvenation. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 10(1):10.

Markhasin, A. (2008). Analysis and optimization of multiservice mobile systems. *Optoelectronics, Instrumentation and Data Processing*, 44(5):477–485.

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*, 38(2):69–74.

Menasché, D. S. (2013). Vídeos de alunos sobre filas com prioridades. `http://www.dcc.ufrj.br/~sadoc/ad20122/` (clique na caixa sobre filas com prioridades).

Menasché, D. S. and Rodrigues, P. A. (2013). Filas com prioridades. `http://www.dcc.ufrj.br/~sadoc/prioridade/`.

Rodrigues, P. H. A. (2013a). Apostila sobre avaliação de desempenho. `http://www.dcc.ufrj.br/~sadoc/ad20132/apostila.pdf`.

Rodrigues, P. H. A. (2013b). Slides sobre avaliação de desempenho. `http://www.voip.nce.ufrj.br/cursos/images/files/mab515/aulas/adaula17.pdf`.

Simatos, F., Bouman, N., and Borst, S. (2013). Lingering issues in distributed scheduling. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 141–152. ACM.

Stewart, W. J. (2009). *Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling*. Princeton University Press.

Wierman, A. (2007). *Scheduling for today's computer systems: Bridging theory and practice*. PhD thesis, Carnegie Mellon University.

## 10. Apêndice

Para facilitar a comparação da solução apresentada neste artigo com a literatura e apreciar a simplicidade das deduções aqui apresentadas, indicamos neste apêndice mais detalhes sobre a abordagem adotada em [Jaiswal 1968] para calcular o número médio de clientes.

Segundo [Jaiswal 1968, pg 61], considere um período ocupado do sistema, que inicia-se em $t = 0$. Nesse período ocupado, seja $p(m_1, m_2, x_1, x_2, t)dx_1dx_2$ a probabilidade de, no instante $t$, haver $m_1$ e $m_2$ tarefas de alta e baixa prioridade, respectivamente, no sistema, a quantidade de serviço gasto com a tarefa do tipo 1 no servidor estar no intervalo $(x_1, x_1 + dx_1)$, e a quantidade de tempo que a tarefa do tipo 2 gastou no servidor até a última preempção estar no intervalo $(x_2, x_2 + dx_2)$. Analogamente, seja $q(m_1, m_2, x_2, t)dx_2$ a probabilidade de, no instante $t$, haver $m_1$ e $m_2$ tarefas de alta e baixa prioridade, respectivamente, no sistema, e a quantidade de serviço gasto com a tarefa do tipo 2 no servidor estar no intervalo $(x_2, x_2 + dx_2)$. Note que $p(m_1, m_2, x_1, x_2, t)dx_1dx_2$ e $q(m_1, m_2, x_2, t)dx_2$ denotam os eventos em que uma tarefa do tipo 1 e do tipo 2 estão no servidor, respectivamente. Afunção geradora da probabilidade conjunta do tamanho das filas 1 e 2 em $t$, dado que um período ocupado iniciou-se em $t = 0$, é dada por,

$$\begin{aligned} M_{1,2}(z_1, z_2, t) &= \sum_{m_1=0}^{\infty}\sum_{m_2=0}^{\infty}\int_{x_1=0}^{\infty}\int_{x_2=0}^{\infty} z_1^{m_1} z_2^{m_2} p(m_1, m_2, x_1, x_2, t)dx_1dx_2 \\ &\quad + \sum_{m_1=0}^{\infty}\sum_{m_2=1}^{\infty}\int_{x_2=0}^{\infty} z_1^{m_1} z_2^{m_2} q(m_1, m_2, x_1, x_2, t)dx_2 \end{aligned} \tag{23}$$

A derivação das quantidades de interesse é feita então a partir de (23). Para tal, as funções $p$ e $q$ são escritas em termos da probabilidade de que, no instante $t$, um cliente de baixa prioridade entre em serviço com $m_2 - 1$ clientes de baixa prioridade em fila. Então, denotando esta probabilidade no instante $t$ por $H(m_2, t)$, segundo [Jaiswal 1968, pg 60],

$$H(m_2, t) = \frac{\lambda_2}{\lambda_1 + \lambda_2} H(m_2, t|②) + \frac{\lambda_1}{\lambda_1 + \lambda_2} H(m_2, t|①) \tag{24}$$

onde ① e ② denotam o evento de o período ocupado ter se iniciado com um serviço de alta e baixa prioridade, respectivamente. Dessa forma, estabelece-se uma conexão entre a dervição de [Jaiswal 1968] e aquela apresentada na seção 3 deste artigo (vide eq.(1)). A derivação alternativa apresentada neste artigo é complementar à descrita acima, e traz novos *insights* sobre o problema.

# Heurísticas para o Projeto Topológico de Redes Heterogêneas Resilientes com QoS

**Bráulio Antônio Mesquita Souza**[1], **Daniel Ludovico Guidoni**[1], **Fernanda Sumika Hojo de Souza**[1], **Geraldo Robson Mateus**[2]

[1]Departamento de Ciência da Computação – Universidade Federal de São João del-Rei

[2]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais

{braulio,guidoni,fsumika}@ufsj.edu.br, mateus@dcc.ufmg.br

***Resumo.*** *O projeto de infraestrutura de redes é um ponto fundamental a fim de atender o vasto número de requisitos dos usuários de serviços de telecomunicações, os quais incluem capacidades de largura de banda, tolerância a falhas, baixos níveis de atraso, segurança, entre outros, sendo que a provisão destes irá impactar diretamente na qualidade de serviço e satisfação do usuário. Considerando o atual cenário de redes, a existência de diferentes classes de usuários (de acordo com seus requisitos) e a disponibilidade de diversas tecnologias (com variadas capacidades e consequentemente custos proporcionais), é possível projetar topologias de forma a atender as demandas existentes mas ao mesmo tempo otimizar os custos implicados por tal infraestrutura. Neste trabalho, são propostas duas heurísticas para o problema de projeto de redes heterogêneas resilientes com qualidade de serviço. Resultados dos experimentos demonstram a eficiência dos métodos propostos em comparação com outros métodos da literatura.*

***Abstract.*** *Network design is a key challenge in order to supply a large set of requirements from telecommunication services customers, which include bandwidth capacity, fault tolerance, low levels of delay, security, among others, so that the fulfilment of these requirements will directly impact on quality of service and customer satisfaction. Considering the current network scenario, the existence of different classes of customers (according to their requirements) and the availability of many technologies (with different capacities and consequently proportional costs), it is possible to design topologies to meet the demanding requirements but also optimize costs implied by such infrastructure. In this paper, we propose two heuristics for the problem of resilient heterogeneous network design with quality of service. Experimental results show the efficiency of the proposed method compared with other methods from the literature.*

## 1. Introdução

O advento de novas tecnologias tem proporcionado a difusão de uma série de novos serviços em termos de telecomunicações. Consequentemente, é observada uma crescente expansão na demanda dos usuários, que a cada dia têm se tornado mais exigentes e dependentes de serviços como telefonia celular, Internet, voz sobre IP (*VoIP - Voice over Internet Protocol*), entre outros. Os provedores de serviços por sua vez, precisam lidar com questões de disponibilidade, capacidade, tolerância a falhas e segurança. Sendo assim, a

mera existência de tecnologias avançadas não implica em altos padrões de qualidade, se as redes de comunicação não forem projetadas da forma adequada.

Os meios de comunicação de redes atuais incluem conexões cabeadas e sem fio, que evoluíram significativamente nos últimos anos. Os principais meios de conexões cabeadas são par trançado, cabo coaxial e cabo de fibra óptica. Na busca por meios mais flexíveis, surgiram as conexões de rede sem fio, em que podemos destacar a transmissão por microondas para conexão via rádio ou via satélite. A escolha dos meios de comunicação a serem instalados deve ser realizada em função do tipo de serviço desejado e do orçamento disponível para investir na instalação. Cada uma das tecnologias apresenta vantagens e desvantagens em termos de capacidade, qualidade e custos. Dessa forma, é inevitável que serviços de maior qualidade acarretem custos mais elevados aos usuários da rede.

Em função das diferentes demandas provenientes dos diversos usuários de serviços de telecomunicações, mostra-se necessária uma organização hierárquica de maneira que os provedores de serviço possam suprir as necessidades das diferentes classes de usuários da melhor forma possível. Neste contexto, qualidade de serviço (*QoS - quality of service*) [Xiao and Ni 1999] surge como forma de prover garantias especiais a classes de usuários na rede. QoS se refere à habilidade de uma rede em entregar resultados previsíveis. Estes resultados indicam o desempenho da rede em termos de disponibilidade, vazão, atraso e taxa de falhas, por exemplo.

Resiliência é um dos aspectos chave que indicam qualidade de serviço nos dias de hoje. Falhas como interferência, rompimento de fibra ou queda de energia são situações comuns em cenários de redes, que podem levar à perda da conexão em um determinado enlace da rede. Sob condições de falha, mecanismos de proteção possibilitam a rede manter garantia nos serviços. Tais mecanismos, podem estar presentes desde o nível topológico até de protocolo. No nível topológico, por exemplo, uma rede 2-conexa [West 2001] mostra-se robusta contra falha em único enlace. Já foi constatado que falhas em um único enlace representam $70\%$ das falhas em redes [Iannaccone et al. 2002].

Um outro ponto importante ligado à QoS diz respeito ao atraso no envio de pacotes na rede. Novamente, este requisito pode ser tratado desde o nível topológico até o algorítmico, onde o roteamento será responsável por prover as garantias desejadas. Em nível topológico, o atraso pode ser estimado de forma aproximada, através do número de saltos que um pacote percorre até alcançar seu destino. Assim, muitas vezes o chamado diâmetro da rede [Chung 1986] tem sido utilizado como uma aproximação que limita o atraso máximo permitido.

Diante destas considerações, é possível perceber que um bom projeto topológico da rede pode representar uma etapa crucial na provisão de serviços de qualidade. Sem uma infraestrutura adequada, os algoritmos de roteamento podem não ser capazes de atender certos requisitos das aplicações. A necessidade de determinismo na resiliência e no diâmetro de redes justifica o projeto de redes que cumpram esse requisito. Entretanto, este problema é NP-difícil [Kerivin and Mahjoub 2005]. Neste cenário, técnicas de otimização desempenham papel fundamental visando a melhorar o desempenho das redes de telecomunicações, prover serviços mais confiáveis e de baixo custo. Este trabalho tem como objetivo contribuir com novas soluções para o problema de projeto de

redes heterogêneas com qualidade de serviço em termos de resiliência e diâmetro da rede. Duas heurísticas são propostas para o problema, demonstrando a viabilidade de aplicação prática, em função dos baixos tempos computacionais exigidos e da alta qualidade das soluções, em comparação com um algoritmo baseado em programação linear inteira.

As seções seguintes, estão organizadas como explicado a seguir. Na Seção 2 são descritos os trabalhos relacionados ao problema em estudo. Na Seção 3 é apresentada a definição formal do problema e sua formulação matemática. Na Seção 4 são propostas as duas heurísticas para o problema enquanto os resultados dos experimentos e a discussão são apresentados na Seção 5. Finalmente, a Seção 6 conclui o trabalho.

## 2. Trabalhos Relacionados

O problema de projeto de redes resilientes (também conhecido na literatura como *SNDP - survivable network design problem*) e suas variações constituem um tema bastante relevante, sendo foco de estudo de muitos pesquisadores na área. Um *survey* em métodos para o SNDP pode ser visto em [Kerivin and Mahjoub 2005] e [Abd-El-Barr 2009].

Prover resiliência tem um significado amplo, podendo ser abordado através de diferentes métricas. Como apresentado por [Vasconcelos and Salles 2012], a resiliência está relacionada à quantidade de redundâncias na rede, bem como as capacidades destas redundâncias. Métricas para avaliar a robustez de uma rede podem incluir: tamanho do maior componente conexo (*LCC - largest connected component*) após uma falha ou ataque, tamanho médio do menor caminho entre os vértices (*ASPL - average shortest path length*) e diâmetro, que corresponde ao maior dentre todos os caminhos mínimos da rede. Uma outra forma de medir resiliência destacada é baseada na k-conectividade da rede. Os autores propoem uma nova métrica de resiliência, que leva em consideração diferentes características da topologia da rede, como número de vértices, número de enlaces disponíveis, suas disposições, capacidades, ocupações e carga.

No estudo realizado por [Couto et al. 2012], a robustez intra "centros de dados"(*data centers*) baseada na topologia da rede é avaliada, através de métricas de LCC e diâmetro. Os autores destacam que a análise de robustez considerando apenas a topologia da rede é de grande importância, por ser independente da aplicação e dos algorimos de roteamento e engenharia de tráfego das arquiteturas.

As abordagens de solução para o SNDP e suas variações são divididas em exatas e heurísticas. As abordagens exatas são geralmente baseadas em técnicas de Programação Linear Inteira Mista (PLIM) [Abd-El-Barr 2009, Chimani et al. 2010]. O tratamento de instâncias de maior porte demandam o uso de técnicas heurísticas uma vez que estas mostram-se mais adequadas em termos de esforço computacional.

Em [Rosenberg 2005], é apresentada um solução hierárquica para projetar topologias de redes com restrições de diâmetro, grau máximo dos vértices e tolerância a falha em um único vértice, minimizando o número de arcos a serem instalados. O método, que consiste em um procedimento de clusterização através de programação dinâmica gerando redes de grande escala a partir de instâncias menores resolvidas à otimalidade ou próxima dela, foi utilizado para planejar duas redes de alta velocidade da AT&T.

Um algoritmo genético foi proposto em [Morais et al. 2011], para o projeto topológico de redes ópticas resilientes com despesas de capital mínimas. Neste trabalho,

os autores consideram que as topologias devem ser resilientes contra falha única de enlaces. Os resultados dos experimentos demonstraram a viabilidade de uso do algoritmo pela qualidade das soluções e tempo de processamento.

O uso de uma métrica de resiliência não só baseada em conectividade é estudada em [Konak and Smith 2005]. Através de um algoritmo genético híbrido, busca-se maximizar a resiliência da rede, considerando todos os estados possíveis da combinação dos estados operacionais dos componentes individuais da rede.

Até onde conhecemos, não foi encontrado método na literatura que tratou o problema de resiliência contra falha de um único enlace combinado com restrições de diâmetro e diferentes classes de usuários e tecnologias a serem instaladas nos enlaces, foco do nosso estudo.

## 3. Definição Formal do Problema e Formulação Matemática

O problema de projeto de redes heterogêneas resilientes com qualidade de serviço pode ser definido a partir de um grafo $G = (V, E)$, onde $V$ é o conjunto de vértices, representando os usuários/roteadores, e $E$ é o conjunto de arestas, que representam as ligações entre tais usuários. Seja $V = \{V_1, V_2, ..., V_t\}$ o conjunto de usuários/roteadores agrupados por classes em um conjunto $T$ numeradas de 1 até $t$, de acordo com a tecnologia requerida. Sejam $\{c_{ij}^t : \forall\{i,j\} \in E, \forall t \in T\}$ os custos associados a cada aresta $\{i,j\}$ se a tecnologia $t$ for empregada, tal que $c_{ij}^t \geq c_{ij}^{t'}$ se $t < t'$. Seja $D$ o diâmetro máximo permitido para a comunicação dos vértices e considere todos os possíveis pares de vértices da rede.

A solução para o problema consiste em encontrar um subconjunto de arestas $S \subseteq E$, tal que cada aresta esteja associada a uma única tecnologia e o subgrafo de $G$ induzido por $(V, S)$ possa prover duas rotas disjuntas em arestas para cada par de vértices com tamanho máximo $D$. As tecnologias das arestas são definidas de acordo com as rotas que atravessam a aresta, ou seja, rotas que conectam dois vértices da classe $t$ devem atravessar apenas arestas de tecnologia $t$, ao passo que a tecnologia exigida para um par de vértices de classes diferentes é determinada pela tecnologia da classe mais baixa. O subgrafo resultante corresponderá ao projeto topológico da rede, provendo qualidade de serviço em termos de resiliência e diâmetro máximo, além de tecnologias de acordo com a classe do usuário.

Este problema pode ser formulado como um problema de Programação Linear Inteira (PLI) baseado na abordagem de fluxo em redes [Ahuja et al. 1993]. Para formular o problema, usamos dois conjuntos de variáveis de decisão: $(i)$ $y_{ij}^t \in \{0,1\}$ assumindo valor 1 se a aresta $\{i,j\}$ é instalada com tecnologia $t$ e portanto inclusa na solução (0, caso contrário); $(ii)$ $w_{ij}^{dpq} \in \{0,1\}$ indicando o fluxo que passa no arco $(i,j)$ para ligar o vértice de origem $p$ ao vértice de destino $q$ no $d$-ésimo salto de uma rota. Seja $A$ o conjunto de arcos de $G$, tal que cada aresta $\{i,j\} \in E$ está associada ao par $(i,j) \in A$ e $(j,i) \in A$. Considere ainda que $A_i^-$ e $A_i^+$ denotam os conjuntos de arcos chegando e deixando o vértice $i$, respectivamente. Assim, o problema pode ser escrito como:

$$\min \left[ \sum_{\{i,j\} \in E} \sum_{t \in T} c_{ij}^t y_{ij}^t \right] \tag{1}$$

sujeito ao seguinte conjunto de restrições:

Restrições de conectividade através de duas rotas, tomadas para todos os pares origem-destino $(p, q) \in V$:

$$\sum_{j \in A_p^+} w_{pj}^{1pq} \geq 2 \qquad \forall p, q \in V \tag{2}$$

$$\sum_{d=1}^{D} \sum_{i \in A_q^-} w_{iq}^{dpq} \geq 2 \qquad \forall p, q \in V \tag{3}$$

$$\sum_{i \in A_j^-} w_{ij}^{dpq} - \sum_{l \in A_j^+} w_{jl}^{d+1pq} = 0 \qquad \forall p, q \in V, \forall j \in V, j \neq p \neq q, d = 1, ..., D-1 \tag{4}$$

Restrições para que cada aresta esteja associada a uma única tecnologia:

$$\sum_{t \in T} y_{ij}^{t} \leq 1 \qquad \forall \{i, j\} \in E \tag{5}$$

Restrições de acoplamento entre variáveis:

$$\sum_{d=1}^{D} w_{ij}^{dpq} \leq \sum_{t' \in T : t' \leq t(pq)} y_{\{ij\}}^{t'} \qquad \forall p, q \in V, \forall (i, j) \in A / \{i, j\} \in E \tag{6}$$

Restrições gerais:

$$w_{ij}^{dpq} \in \{0, 1\} \qquad \forall p, q \in V, \forall (i, j) \in A, d = 1, ..., D \tag{7}$$

$$y_{ij}^{t} \in \{0, 1\} \qquad \forall \{i, j\} \in E, \forall t \in T \tag{8}$$

A função objetivo (1) minimiza os custos de instalação para prover conectividade entre os usuários da rede respeitando os requisitos contratados. Restrições para garantir a conectividade através de duas rotas para cada par de vértices ou usuários $(p, q)$ são dadas por (2)-(4). Inequações (2) garantem que o fluxo possa sair do vértice $p$ com destino $q$ através de duas arestas distintas indexadas pelo primeiro salto ($d = 1$) de cada rota. Inequações (3) garantem a existência de duas rotas para o vértice $q$ partindo de $p$, independente do número de saltos usados para alcançá-lo, contanto que menos do que $D$. Restrições (4) garantem a conectividade nas rotas passando por vértices intermediários, de forma que ao deixar um vértice, o índice de saltos seja maior em uma unidade do que na sua chegada. Inequações (5) asseguram que em cada aresta seja instalada apenas uma tecnologia. Inequações (6) garantem que as classes de tecnologias demandadas pelos usuários sejam respeitadas nas rotas que conectam esses vértices. Note que as inequações (5) em conjunto com as inequações (6) impõem que as duas rotas para cada par de usuários $(p, q)$ sejam disjuntos em arestas. Formulação (1)-(8) tem um número de restrições na ordem de $O(n^4)$ e variáveis na ordem de $O(n^5)$. Portanto, é esperado que apenas instâncias de pequeno porte sejam passíveis de solução através deste método.

## 4. Heurísticas

Heurísticas são procedimentos que buscam boas soluções a um custo computacional aceitável sem, no entanto, fornecer qualquer garantia de otimalidade ou mesmo de proximidade da solução ótima [Gendreau and Potvin 2010]. O uso de heurísticas tem sido muito explorado nas últimas décadas por se mostrar bastante eficiente em problemas reais.

As heurísticas podem ser classificadas em três tipos: heurísticas construtivas, que se baseiam na construção de uma solução inicial para o problema, elemento a elemento; heurísticas de refinamento ou busca local, que buscam a melhora de uma solução a partir de alterações de seus elementos; e metaheurísticas, que são procedimentos de mais alto-nível que conduzem heurísticas mais simples à melhores soluções, escapando de ótimos locais. Neste trabalho, são propostas duas adaptações de metaheurísticas ao problema que utilizam uma heurística construtiva e uma heurística de busca local, desenvolvidas especificamente para o problema em questão.

A heurística construtiva proposta neste trabalho para construção de uma solução inicial é baseada na geração de rotas entre os pares origem-destino, a fim de atender suas demandas de comunicação. O algoritmo utilizado para geração das rotas é uma versão do algoritmo de Lawler [Lawler 1976] para resolver o problema do caminho mínimo com restrição de saltos (a fim de garantir o limite imposto pelo diâmetro $D$), baseado em programação dinâmica.

Um pseudo-código da heurística construtiva proposta é apresentada no Algoritmo 1. O conjunto $S$ das arestas da solução inicia-se vazio (linha 2). A geração das rotas para os pares origem-destino é feita iniciando-se por aqueles que demandam um custo mais alto de comunicação, isto é, os pares cujos vértices são de classe mais prioritária. Assim, os pares são ordenados de acordo com suas classes (linha 3). A fim de obter uma diversidade na geração de soluções iniciais, os pares origem-destino são embaralhados antes da busca por suas rotas, porém, pares origem-destino de classes mais prioritárias vêm antes daquelas com classes menos prioritárias.

Para cada par origem-destino, é realizada a tentativa de encontrar duas rotas disjuntas em arestas (laço entre as linhas 4 e 35). Se a primeira rota não é encontrada para algum dos pares, significa que o problema não tem solução viável e um erro é retornado (linhas 6 a 8). Para garantir a segunda rota disjunta em enlaces, as arestas utilizadas na $rota1$ são removidas do grafo na busca pela $rota2$ (linha 9). Se as duas rotas são encontradas com sucesso, as arestas de ambas são adicionadas ao conjunto $S$ (linhas 11 a 16). Caso contrário, significa que não foi possível encontrar a $rota2$ disjunta em enlaces da $rota1$. Isso não quer dizer que não existam duas rotas disjuntas para esse par, uma vez que a escolhas das arestas da $rota1$ possam ter impossibilitado a geração da $rota2$. Sendo assim, as arestas da $rota1$ são removidas uma a uma do grafo, e novas rotas são geradas até que ambas sejam encontradas (linhas 17 a 34).

Para diminuir custos e aproveitar as arestas da rede previamente escolhidas, as arestas de rotas já calculadas têm seus custos igualados à zero, pois uma vez que estes enlaces fazem parte da rede outros pares origem-destino poderão utilizá-lo sem custo adicional (linhas 14 e 28). Ao final do algoritmo, o conjunto $S$ é retornado (linha 36). O custo da solução é calculado a partir das arestas em $S$ e dos custos associados à tecnologia a ser aplicada a cada aresta, de acordo com as rotas que passam por tais arestas, isto é, o custo relativo à tecnologia demandada pelas rotas.

O procedimento de busca local proposto utiliza o conceito de vizinhança, que é definida em [Hertz and Widmer 2003] como um conjunto de soluções $s'$ que podem ser obtidas a partir de pequenas modificações em uma solução $s$, onde os vizinhos obtidos com menores modificações geram uma vizinhança mais próxima, enquanto os vizi-

nhos obtidos com uma maior modificação na solução corrente formam uma vizinhança mais distante. Neste artigo foram utilizadas estruturas de vizinhanças que consistem na remoção de arestas da solução de partida. Definimos como $N^1$ e $N^2$ as vizinhanças que removem 1 e 2 arestas da solução corrente, respectivamente. Após a remoção das arestas, é possível que haja inviabilidade na solução. Portanto, faz-se necessário o re-cálculo de rotas para os pares origem-destino que utilizavam tais arestas, que é realizado através do mesmo método utilizado na heurística construtiva. A busca local aplicada é conhecida como "melhor aprimorante", na qual todos os vizinhos de uma solução são avaliados e o melhor deles passa a ser a solução corrente. O algoritmo realiza quantas iterações sejam necessárias até que um ótimo local seja atingido, ou seja, não exista nenhum vizinho de melhora.

### 4.1. Metaheurística VNS

Busca em Vizinhança Variável (VNS, do inglês *Variable Neighborhood Search*) [Hansen and Mladenovic 2001] é uma metaheurística que adota a ideia de troca de vizinhanças. Um pseudo-código desta metaheurística é apresentado no Algoritmo 2.

Ao buscar por uma solução melhor, a partir de uma solução inicial $s$ (linha 2), o algoritmo VNS gera uma solução aleatória $s'$ pertencente à uma vizinhança $N^k$, da solução corrente (linha 6). Aplica-se busca local à $s'$ (linha 7), obtendo um ótimo lo-

**Algoritmo 1** Heurística Construtiva

```
1: função GERASOLUÇÃOINICIAL(G(V,E))
2:     S = ∅
3:     Ordenar pares origem-destino por classes
4:     para cada par origem-destino faça
5:         Encontrar primeira rota entre os vértices
6:         se não encontrar a primeira rota então
7:             retorna Erro: "problema inviável"
8:         fim se
9:         Remover do grafo, as arestas da primeira rota
10:        Encontrar a segunda rota entre os vértices
11:        se encontrar a segunda rota então
12:            para cada aresta e ∈ rota1 ∪ rota2 faça
13:                S = S + e
14:                custo(e) = 0
15:            fim para
16:            Próximo par origem-destino
17:        senão
18:            para cada aresta e ∈ primeirarota faça
19:                E = E − e
20:                Encontrar primeira rota
21:                se encontrar a primeira rota então
22:                    Remover do grafo, as arestas da primeira rota
23:                    E = E + e
24:                    Encontrar a segunda rota
25:                    se encontrar a segunda rota então
26:                        para cada aresta e ∈ rota1 ∪ rota2 faça
27:                            S = S + e
28:                            custo(e) = 0
29:                        fim para
30:                        Próximo par origem-destino
31:                    fim se
32:                fim se
33:            fim para
34:        fim se
35:    fim para
36:    retorna S
37: fim função
```

cal $s''$, o qual é aceito caso $f(s'') < f(s)$ (linha 9), onde $f(.)$ representa a função de avaliação, que indica o custo de uma solução. Se o procedimento de busca local falha na obtenção de uma solução mais promissora, isto é, quando $f(s'') \geq f(s)$, utiliza-se uma estrutura de vizinhança mais distante para gerar $s'$ (linha 12). O número máximo de vizinhanças é dado por $r$. Por outro lado, se o procedimento de busca local encontra uma melhor solução, volta-se a utilizar a estrutura de vizinhança mais próxima (linha 10). Este processo é repetido até que seja alcançada a última estrutura de vizinhança (laço entre linhas 5 e 14) e, então reinicia-se a busca utilizando a primeira estrutura de vizinhança (linha 4) até que o critério de parada seja satisfeito (laço entre linhas 3 e 15). Foi adotado como critério de parada, o número de iterações sem melhora do algoritmo. Os parâmetros utilizados foram: $r = 2$, ou seja, foram utilizadas as duas vizinhanças relacionadas anteriormente e o número máximo de iterações sem melhora igual a $10$. Valores maiores deste parâmetro não demonstraram melhoria nos resultados.

### 4.2. Metaheurística ILS

Outra metaheurística abordada neste artigo é a Busca Local Iterativa (ILS, do inglês *Iterated Local Search*) [Lourenço et al. 2002] que consiste na busca por melhores soluções em um espaço de vizinhança e na aplicação de perturbações à solução corrente. Esta metaheurística é ilustrada no Algoritmo 3.

Uma solução inicial é gerada (linha 2) e submetida a uma busca local (linha 3). Partindo desta solução, o algoritmo ILS aplica uma perturbação na solução corrente (linha 5) e submete a solução perturbada à um método de busca local (linha 6), aceitando o resultado da busca quando este for promissor (linhas 7 e 8). A perturbação ideal é aquela que não seja muito pequena e nem muito grande pois, caso a perturbação seja muito pequena o método de busca local acaba retornando à solução anterior à perturbação. Por outro lado, uma perturbação muito grande na solução torna a busca no espaço de soluções muito aleatória.

Utilizando o método de busca local com remoção de uma aresta, a perturbação aplicada à solução remove três arestas da mesma, as quais são escolhidas aleatoriamente, recalculando as rotas para os pares origem-destino necessários. O critério de parada também foi estabelecido como o número de iterações sem melhora, fixado em $10$. Valores maiores deste parâmetro não demonstraram melhoria nos resultados.

**Algoritmo 2** VNS

```
1: função VNS
2:     s = GeraSoluçãoInicial(G(V,E));
3:     enquanto critério de parada não satisfeito faça
4:         k = 1;
5:         enquanto k ≤ r faça
6:             gere um vizinho qualquer s' ∈ N^k(s);
7:             s'' = BuscaLocal(s');
8:             se f(s'') < f(s) então
9:                 s = s'';
10:                k = 1;
11:            senão
12:                k = k + 1;
13:            fim se
14:        fim enquanto
15:    fim enquanto
16:    retorna s;
17: fim função
```

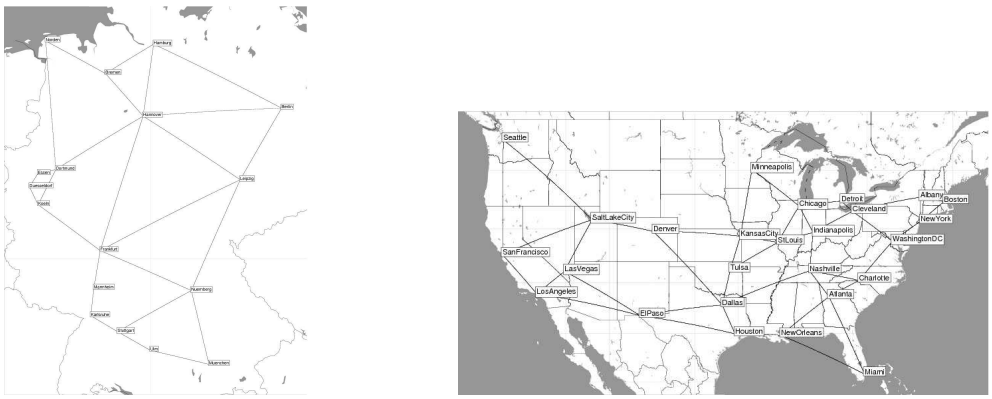
**Algoritmo 3** ILS

```
1: função ILS(s_0)
2:     s_0 = GeraSoluçãoInicial(G(V,E));
3:     s = BuscaLocal(s_0);
4:     enquanto critério de parada não satisfeito faça
5:         s' = Perturbacao(s);
6:         s'' = BuscaLocal(s');
7:         se f(s'') < f(s) então
8:             s = s'';
9:         fim se
10:    fim enquanto
11:    retorna s;
12: fim função
```

## 5. Resultados Experimentais

Nesta seção, são apresentados os resultados dos experimentos realizados para o problema, obtidos através de um algoritmo *branch-and-bound* baseado na formulação matemática apresentada na Seção 3 e através das duas metaheurísticas propostas na Seção 4. Todos os exmperimentos deste estudo foram conduzidos em uma máquina Intel Core i7 com 6 núcleos de 3.4GHz e 32GB de RAM, sob o sistema operacional Linux. Para o algoritmo *branch-and-bound*, o *solver* CPLEX versão 12.51 [ILOG 2013] foi utilizado, com um limite de tempo de 3600s.

Foram utilizadas duas topologias de redes reais (`nobel-germany` e `janos-us`) disponíveis na SNDlib (http://sndlib.zib.de/) [Orlowski et al. 2010] (Figura 1). A rede `nobel-germany` é composta por 17 vértices e 26 arestas, enquanto a rede `janos-us` possui 26 vértices e 42 arestas. Estas redes proporcionam um cenário de interesse, visto que suas estruturas topológicas são bastante representativas com relação a estruturas de telecomunicações reais.




(a) nobel-germany (b) janos-us

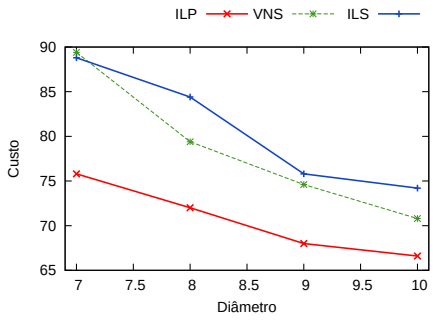
**Figura 1. Redes SNDlib**

Para criar diferentes instâncias de cada topologia para os experimentos, um gerador aleatório foi utilizado, com os seguintes parâmetros: número de classes de usuários ou tecnologias ($T$) e número de vértices em cada classe. Uma classe é atribuída a cada vértice aleatoriamente e custos baseados na distância euclidiana entre os vértices são gerados para cada aresta e tecnologia de forma proporcional. Foram geradas instâncias com $T = 2$ (ou seja, duas tecnologias de comunicação). A quantidade de vértices na classe

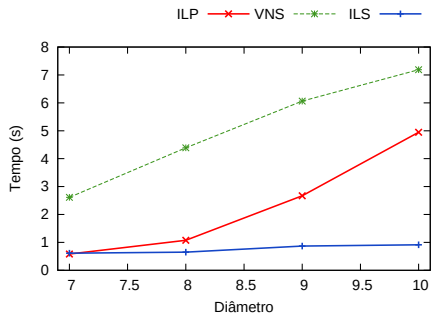
mais prioritária $P \in \{2, 4, 8\}$. O diâmetro da rede é definido como $D \in \{7, 8, 9, 10\}$ para a rede `nobel-germany` e $D \in \{9, 10, 11, 12\}$ para `janos-us` (diferentes em função do menor valor de $D$ necessário para garantir uma solução viável em cada uma das redes). Foram geradas 5 instâncias aleatórias com as mesmas características, nas quais os vértices prioritários estarão posicionados de forma diferente em cada um delas.

As Figuras 2 e 3 e as Tabelas 1 e 2 apresentam os resultados em termos de custo da solução e tempo computacional em função do diâmetro para as redes `nobel-germany` e `janos-us` e cada um dos algoritmos: ILP (*branch-and-bound*), VNS e ILS. Cada dado sumariza a média das 5 instâncias aleatórias; no caso das metaheurísticas, são ainda sumarizadas 5 execuções do algoritmo para cada uma das 5 instâncias.

A Figura 2 apresenta os resultados para a topologia `nobel-germany`, com quantidade de vértices na classe prioritária $P = 8$, variando o diâmetro máximo $D$ permitido na rede. É possível observar na Figura 2(a) que o algoritmo ILP apresentou os menores valores em termos de custo (uma vez que alcançou os valores ótimos) seguido pelo VNS e ILS. Os tempos computacionais apresentados em 2(b) demonstram que todos os algoritmos encontraram soluções para o problema em tempos próximos e bastante reduzidos. Isso se deve ao fato da topologia ser de pequena escala.




(a) Custo da solução

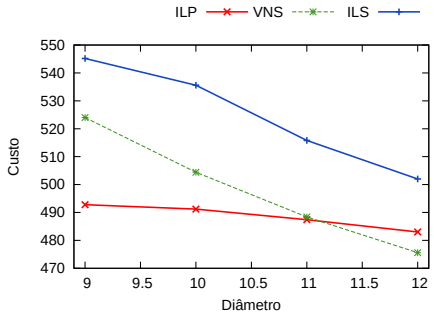



(b) Tempo computacional

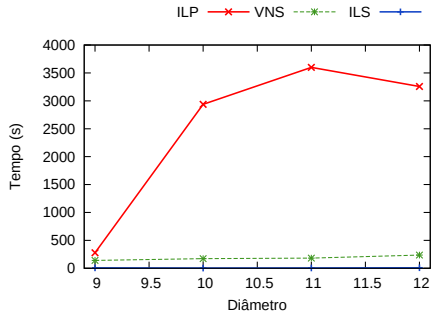
**Figura 2. Topologia `nobel-germany` com $P = 4$**

A Figura 3 apresenta os resultados para a topologia `janos-us`, com $P = 4$. Na Figura 3(a), o ILP nem sempre apresentou os melhores resultados em termos de custo, pois se tratando de uma topologia de maior porte, a solução ótima só foi encontrada para todos os casos quando $D = 9$. Assim, o algoritmo VNS foi capaz de encontrar soluções melhores do que o ILP, com o aumento do valor de $D$. Observando a Figura 3(b), é possível notar que os tempos computacionais do ILP são significativamente superiores aos tempos das metaheurísticas VNS e ILS.

As Tabelas 1 e 2 sumarizam os resultados para todos os parâmetros e algoritmos. A coluna $P$ indica o número de vértices pertencentes à classe mais prioritária, enquanto $D$ corresponde ao diâmetro permitido para comunicação do vértices. As colunas seguintes apresentam os resultados em termos de custo da solução, para cada um dos algoritmos. Valores sublinhados indicam que valor é otimo, enquanto os valores em negrito indicam a heurística com melhor desempenho (tempo e qualidade da solução). As colunas GAP (%) indicam a qualidade das soluções heurísticas, em função do valor ótimo provido pelo ILP. A fórmula é dada por $gap = (sol_h - sol_o)/sol_h$, onde $sol_h$ corresponde à solução




(a) Custo da solução




(b) Tempo computacional

**Figura 3. Topologia `janos-us` com $P = 4$**

**Tabela 1. Resultados sumarizados para a topologia `nobel-germany`**

| | | Custo | | | | | Tempo | | | # opt |
|---|---|---|---|---|---|---|---|---|---|---|
| **P** | **D** | **ILP** | **VNS** | **GAP (%)** | **ILS** | **GAP (%)** | **ILP** | **VNS** | **ILS** | **ILP** |
| 2 | 7 | 51.4 | **53.0** | 3.02 | 53.4 | 3.74 | **0.61** | 2.32 | 1.28 | 5 |
| | 8 | 51.0 | **52.0** | 1.92 | 52.8 | 3.41 | 1.46 | 4.15 | **0.63** | 5 |
| | 9 | 48.2 | **50.8** | 5.11 | 51.8 | 6.94 | 3.73 | 6.08 | **0.72** | 5 |
| | 10 | 47.4 | **50.0** | 5.20 | 51.0 | 7.05 | 7.67 | 5.44 | **0.94** | 5 |
| 4 | 7 | 75.8 | 89.4 | 15.21 | **88.8** | 14.64 | 0.58 | 2.61 | **0.61** | 5 |
| | 8 | 72.0 | **79.4** | 9.31 | 84.4 | 14.69 | 1.07 | 4.38 | **0.65** | 5 |
| | 9 | 68.0 | **74.6** | 8.84 | 75.8 | 10.29 | 2.67 | 6.06 | **0.87** | 5 |
| | 10 | 66.6 | **70.8** | 5.93 | 74.2 | 10.24 | 4.94 | 7.19 | **0.91** | 5 |
| 8 | 7 | 94.6 | **105.8** | 10.5 | 108.8 | 13.05 | 0.61 | 2.51 | **0.44** | 5 |
| | 8 | 90.0 | **99.0** | 9.09 | 99.2 | 9.27 | 1.23 | 4.38 | **0.78** | 5 |
| | 9 | 82.0 | **87.4** | 6.17 | 94.8 | 13.50 | 5.14 | 5.82 | **0.75** | 5 |
| | 10 | 79.0 | **83.2** | 5.04 | 90.2 | 12.41 | 10.14 | 6.77 | **0.75** | 5 |
| **Média** | | 68.83 | **74.61** | 7.12 | 77.10 | 9.93 | 3.32 | 4.81 | **0.78** | 5 |

heurística e $sol_o$ corresponde ao valor da solução ótima. Nos casos em que o ótimo não foi alcançado para todas as instâncias, o *gap* não é calculado (símbolo '-' na tabela). Os tempos de cada algoritmo são apresentados, seguidos pelo número de soluções ótimas encontradas pelo ILP, destacando que cada linha representa a média de 5 instâncias com as mesmas características.

Pela Tabela 1, os resultados demonstram que a metaheurística VNS obteve melhores resultados do que a metaheurística ILS, porém próximos, com tempos computacionais também próximos. Por se tratar de uma instância de pequeno porte, todos os algoritmos foram capazes de encontrar soluções de qualidade em baixos tempos computacionais, sendo que as metaheurísticas VNS e ILS apresentaram *gaps* médios de $7.12\%$ e $9.93\%$, respectivamente. Na Tabela 2, observa-se que os valores ótimos foram encontrado apenas para $D = 9$. Novamente, a metaheurítica VNS apresentou resultados de qualidade superior ao ILS, sendo que para valores de $D$ e $P$ mais elevados, as duas metaheurísticas superaram o ILP em alguns casos. O melhor desempenho em termos de tempos computacionais foi do ILS, seguido pelo VNS, que ainda apresenta tempos aceitáveis. O ILP atingiu o limite de tempo imposto em várias instâncias, não se mostrando adequado para instâncias de maior porte.

**Tabela 2. Resultados sumarizados para a topologia `janos-us`**

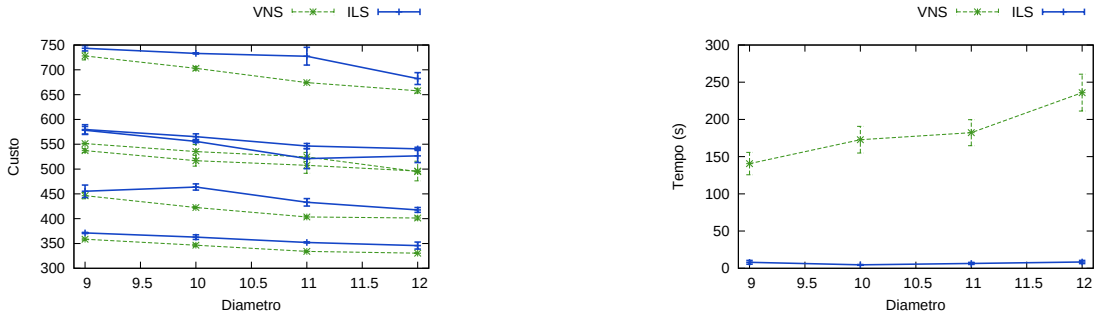
| | | Custo | | | | | Tempo | | | # opt |
|---|---|---|---|---|---|---|---|---|---|---|
| **P** | **D** | **ILP** | **VNS** | **GAP (%)** | **ILS** | **GAP (%)** | **ILP** | **VNS** | **ILS** | **ILP** |
| 2 | 9 | 372.6 | **395.8** | 5.86 | 407.4 | 8.54 | 116.15 | 109.47 | **4.99** | 5 |
| | 10 | 365.2 | **387.4** | - | 399.8 | - | 2844.87 | 174.75 | **3.75** | 3 |
| | 11 | 376.8 | **377.8** | - | 386.8 | - | 3453.31 | 163.52 | **5.14** | 1 |
| | 12 | 380.8 | **370.2** | - | 384.0 | - | 3035.15 | 188.72 | **5.01** | 1 |
| 4 | 9 | 492.8 | **524.0** | 5.95 | 545.2 | 9.61 | 277.60 | 140.64 | **8.02** | 5 |
| | 10 | 491.2 | **504.4** | - | 535.6 | - | 2937.34 | 172.77 | **4.59** | 2 |
| | 11 | 487.4 | **488.4** | - | 515.8 | - | 3600.00 | 182.25 | **6.51** | 0 |
| | 12 | 483.0 | **475.6** | - | 502.0 | - | 3258.01 | 236.03 | **8.47** | 1 |
| 8 | 9 | 558.8 | **602.6** | 7.26 | 626.2 | 10.76 | 166.50 | 118.39 | **7.40** | 5 |
| | 10 | 563.4 | **573.4** | - | 610.2 | - | 3072.61 | 162.10 | **6.99** | 2 |
| | 11 | 692.6 | **551.4** | - | 592.6 | - | 3600.00 | 184.50 | **8.51** | 0 |
| | 12 | 596.8 | **550.2** | - | 565.8 | - | 2484.39 | 204.17 | **8.44** | 2 |
| **Média** | | 488.45 | **483.43** | 6.35 | 505.95 | 9.63 | 2403.66 | 169.80 | **6.48** | **2.25** |

Algumas observações gerais podem ainda ser destacadas para todos os métodos. O aumento no valor do número de vértices prioritários ($P$) eleva o custo da solução, enquanto o aumento no valor de $D$ diminui esse custo, visto que ao permitir rotas mais longas, mais arestas podem ser aproveitadas, sem a necessidade de instalações adicionais para garantir o atraso máximo. Em relação aos tempos, o aumento no valor de $D$ implicou em um leve aumento dos tempos, principalmente para as metaheurísticas. Isso pode ser explicado pela maior possibilidade de rotas alternativas a serem pesquisadas, em função de um diâmetro maior.

Com o objetivo de avaliar a robustez das abordagens propostas, foi avaliada a dispersão dos resultados em torno da média através de um intervalo de confiança de $90\%$ para as $5$ execuções dos algoritmos em cada uma das instâncias, como mostra a Figura 4. Na Figura 4(a), os resultados das $5$ instâncias com características iguais (exceto pelo posicionamento dos vértices de cada nível) são apresentadas para o VNS e ILS. Os dados não foram sumarizados para as $5$ instâncias, uma vez que a posição dos vértices podem tornar os valores de custo em cada instância bastante variados. Como o objetivo é avaliar a robustez do algoritmo em suas diferentes execuções para uma mesma instância, os intervalos de confiança foram apresentados separadamente. É possível observar que a dispersão dos valores é pequena, sendo possível afirmar que o VNS apresenta melhor desempenho (em termos de qualidade de solução) do que o ILS por seus intervalos não estarem sobrepostos na maior parte dos dados. Na Figura 4(b), são apresentados os tempos computacionais sumarizados para as $5$ execuções de cada uma das $5$ instâncias nos dois algoritmos, pois o tempo computacional não é influenciado significativamente pela posição dos vértices. Em termos de tempo computacional, podemos afirmar que o desempenho do ILS é superior ao VNS. Resultados similares foram encontrados para a rede `nobel-germany` e demais valores de $P$.

## 6. Conclusões

Neste trabalho foram apresentadas duas heurísticas para o problema de projeto de redes heterogêneas com qualidade de serviço em termos de resiliência e diâmetro da rede. O problema em estudo é NP-difícil e soluções exatas para o mesmo nem sempre são




(a) Custo da solução (b) Tempo computacional

**Figura 4. Análise da topologia `janos-us` com $P = 4$**

aplicáveis a redes de tamanho real. Os resultados dos experimentos confirmam este fato, mostrando a importância de soluções heurísticas para o problema. De forma geral, as duas heurísticas obtiveram resultados próximos, apresentando soluções com no máximo $9.93\%$ de *gap* (em média) em relação ao valor ótimo (nos casos em que este é conhecido). Além disso, as heurísticas foram capazes de prover soluções para redes reais de médio porte, com baixo esforço computacional, caso em que o método por programação inteira apresenta tempos muito elevados. É importante destacar ainda que as heurísticas propostas podem ser aplicadas a outros cenários, para instâncias de maior porte, em função dos baixos tempos computacionais observados.

Como trabalhos futuros, outras estruturas de vizinhança para a busca local podem ser exploradas, além do uso de mais de uma vizinhança na etapa de perturbação do ILS. A combinação ILS e VNS deve ser analisada para obter melhores resultados. Pretende-se ainda aplicar as heurísticas propostas neste trabalho no contexto de virtualização de redes, onde redes com características diferenciadas podem ser executadas em paralelo utilizando a mesma infraestrutura. Através da estrutura hierárquica, é possível explorar topologia modular, balanceamento de carga, redistribuição de recursos de acordo com a necessidade de cada uma das redes, entre outros.

## Agradecimentos

Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

## Referências

Abd-El-Barr, M. (2009). Topological network design: A survey. *Journal of Network and Computer Applications*, 32(3):501–509.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, USA.

Chimani, M., Kandyba, M., Ljubic, I., and Mutzel, P. (2010). Orientation-based models for 0, 1, 2-survivable network design: theory and practice. *Mathematical Programming*, 124(1-2):413–439.

Chung, F. (1986). Diameters of communication networks. In *Proceedings of the Symposium on Applied Mathematics of the American Mathematical Society*, volume 34, pages 1–18.

Couto, R. S., Campista, M. E. M., and Costa, L. H. M. K. (2012). Uma avaliação da robustez intra data centers baseada na topologia da rede. In *Anais do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 610–623, Ouro Preto, MG, Brasil.

Gendreau, M. and Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition.

Hansen, P. and Mladenovic, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Hertz, A. and Widmer, M. (2003). Guidelines for the use of meta-heuristics in combinatorial optimization. *European Journal of Operational Research*, 151(2):247–252.

Iannaccone, G., Chuah, C.-n., Mortier, R., Bhattacharyya, S., and Diot, C. (2002). Analysis of link failures in an ip backbone. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurement*, IMW '02, pages 237–242, New York, NY, USA. ACM.

ILOG, I. (2013). Cplex optimizer. [Online; acessado em 20-Mar-2013].

Kerivin, H. and Mahjoub, A. R. (2005). Design of survivable networks: A survey. *Netw.*, 46(1):1–21.

Konak, A. and Smith, A. E. (2005). Designing resilient networks using a hybrid genetic algorithm approach. In Beyer, H.-G. and O'Reilly, U.-M., editors, *GECCO*, pages 1279–1285. ACM.

Lawler, E. (1976). *Combinatorial optimization: networks and matroids*. Holt, Rinehart and Winston, New York.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2002). Iterated local search. In *Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science*, pages 321–353. Kluwer Academic Publishers.

Morais, R., Pavan, C., Pinto, A. N., and Agra, C. (2011). Genetic algorithm for the topological design of survivable optical transport networks. *IEEE/OSA Journal of Optical Communications and Networking*, 3(1):17–26.

Orlowski, S., Wessäly, R., Pióro, M., and Tomaszewski, A. (2010). Sndlib 1.0—survivable network design library. *Netw.*, 55(3):276–286.

Rosenberg, E. (2005). Hierarchical topological network design. *IEEE/ACM Transactions on Networking*, 13(6):1402–1409.

Vasconcelos, M. F. and Salles, R. M. (2012). Emprego de resiliência na gerência de redes de computadores. In *Anais do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 596–609, Ouro Preto, MG, Brasil.

West, D. B. (2001). *Introduction to Graph Theory (2nd Edition)*. Prentice Hall.

Xiao, X. and Ni, L. M. (1999). Internet QoS: A big picture. *IEEE Network*, 13:8–18.

# Definição, Modelagem e Aplicações de Camadas de Sensoriamento Participativo

**Thiago H. Silva**[1], **Pedro O. S. Vaz de Melo**[1], **Jussara M. Almeida**[1], **Aline C. Vianna,**[2] **Juliana Salles,**[3] **Antonio A. F. Loureiro**[1]

[1]DCC-UFMG, Belo Horizonte, Brasil
[2]INRIA, França
[3]Microsoft Research, Redmond, EUA

***Abstract.*** *Data from different participatory sensor networks can be represented as sensing layers, each one enabling the access of data related to a certain aspect of predefined geographical regions. The joint analysis of multiple sensing layers may be useful to develop new applications. The purposes of this paper are three-fold. First, we define the concept of sensing layers. Then we define a framework for integrating multiple sensing layers, which can be applied to more sophisticated services, for example, identification of causes of traffic jams. Finally, we propose two applications that illustrate the proposed framework and the potential of using multiple sensing layers.*

***Resumo.*** *Dados de diferentes redes de sensores participativos podem ser representados como camadas de sensoriamento, cada uma possibilitando o acesso a observações relacionadas a certo aspecto de regiões geográficas pré-definidas. A análise conjunta de múltiplas camadas de sensoriamento pode ser útil no desenvolvimento de novas aplicações. Com isso, os objetivos deste trabalho são três. Primeiro, definir o conceito de camadas de sensoriamento. Em seguida, definir um arcabouço para integração de múltiplas camadas de sensoriamento, que pode ser aplicado em serviços mais sofisticados, por exemplo, identificação de causas de congestionamentos. Finalmente, nós propomos duas aplicações que ilustram o uso do arcabouço proposto e o potencial da utilização de múltiplas camadas de sensoriamento.*

## 1. Introdução

Mídias sociais baseadas em localização, tais como o Foursquare[1], Instagram[2] ou Waze[3], são bastante populares hoje em dia, graças à popularização dos *smartphones* e ao acesso à Internet por meio desses dispositivos. Por exemplo, o Foursquare registrou 40 milhões de usuários em setembro de 2013[4]. Esses aplicativos podem ser vistos como redes de sensores participativos (RSPs) [Burke et al. 2006, Silva et al. 2013a]. Nesse tipo de rede o dispositivo móvel do usuário é uma peça fundamental. Indivíduos carregando esses dispositivos são capazes de sensoriar o ambiente e fazer observações relevantes. Assim, cada nó em uma RSP consiste de um usuário com o seu dispositivo móvel [Silva et al. 2013a]. Cada RSP fornece acesso aos dados relacionados a certo aspecto de uma região geográfica pré-definida ( por exemplo, condições de tráfego, imagens de locais, condição do clima, etc.), e pode ser representada como uma camada de sensoriamento (ou simplesmente camada).

Uma camada de sensoriamento permite o estudo em larga escala de cada aspecto monitorado em (quase) tempo real, além de fornecer dados históricos sobre padrões observados

[1]http://www.foursquare.com. Todos os websites foram acessados pela última vez em 05/11/2013.
[2]http://www.instagram.com.
[3]http://www.waze.com.
[4]https://foursquare.com/about.

durante longos períodos de tempo. Isso nos ajuda a compreender melhor a dinâmica de cidades e o comportamento social urbano em diferentes regiões do mundo, bem como reagir mais rapidamente a mudanças inesperadas nos padrões observados.

A exploração de forma independente de camadas de sensoriamento pode ser muito útil. Por exemplo, uma camada de sensoriamento contendo informações de trânsito pode possibilitar a identificação em tempo real de rodovias com acidentes e buracos, cuja detecção é difícil com sensores tradicionais, mas torna-se mais factível quando os usuários participam do processo de sensoriamento. Tal detecção abre oportunidades para vários serviços, por exemplo, auxiliar carros inteligentes na correta identificação de problemas na estrada [Silva et al. 2013d]. De fato, trabalhos anteriores já analisaram diferentes camadas de sensoriamento, tais como check-ins [Cranshaw et al. 2012], fotos de lugares [Silva et al. 2013b] e alertas de tráfego [Silva et al. 2013d] e sugeriram aplicações no projeto de serviços diversos.

A análise conjunta de múltiplas camadas de sensoriamento também pode ser extremamente útil na construção de outras aplicações mais sofisticadas. Por exemplo, sabemos que uma queixa comum dos habitantes das grandes cidades é congestionamento. Com isso, uma aplicação que emerge naturalmente é uma que possui o objetivo de inferir as causas de congestionamento, passo fundamental para tratar o problema. Esta não é uma tarefa fácil de realizar, e o resultado pode variar de lugar para lugar. No entanto, a análise conjunta de diferentes camadas de sensoriamento na cidade poderia contribuir para essa aplicação. Por exemplo, poderíamos cruzar as informações fornecidas pelas camadas: alertas de tráfego, derivadas do Waze; check-ins, derivados do Foursquare; e foto de lugares, derivados do Instagram. A primeira camada fornece dados próximos do tempo real sobre onde estão acontecendo congestionamentos, a segunda fornece dados sobre os tipos de lugares localizados nas áreas dos congestionamentos, com isso é possível entender melhor as áreas de interesse, por exemplo, identificando que é uma área comercial. Finalmente, através da análise da camada fotos de lugares nós podemos obter evidência visual do que acontece em tempo real próximo das áreas durante os congestionamentos. Ao analisar conjuntamente dados destas três camadas podemos detectar, por exemplo, carros bloqueando cruzamentos, e inferir as possíveis causas disso. Obviamente, outras camadas podem também ser utilizadas, tal como a condição do clima, camada derivada de sistemas como o Weddar[5].

As contribuições deste trabalho podem ser divididas em três: (1) definição do conceito de camadas de sensoriamento; (2) proposta de um arcabouço para integração de múltiplas camadas de sensoriamento, que pode ser aplicado em serviços mais sofisticados do que serviços baseados em uma única camada; (3) proposta de aplicações que ilustram o uso do arcabouço apresentado e o potencial da utilização de múltiplas camadas de sensoriamento.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 descreve os trabalhos relacionados. A Seção 3 define o emergente conceito de camadas de sensoriamento. A Seção 4 propõe um arcabouço para trabalhar com camadas de sensoriamento. A Seção 5 define algumas operações que podem ser aplicadas em camadas de sensoriamento. A Seção 6 apresenta algumas aplicações propostas que ilustram o potencial da utilização de camadas de sensoriamento. Finalmente, a Seção 7 apresenta as conclusões e os trabalhos futuros.

## 2. Trabalhos Relacionados

Como os dados fornecidos por RSPs podem ser muito complexos, um passo fundamental antes de qualquer investigação é a caracterização dos dados, a fim de entender seus desafios e utilidade. As propriedades de RSPs derivadas de diferentes aplicações, tais como o Foursquare, um serviço

[5]http://www.weddar.com.

de compartilhamento de localização, e o Instagram, um serviço de compartilhamento de fotos, já foram investigadas na literatura [Noulas et al. 2011a, Crandall et al. 2009, Vasconcelos et al. 2012, Long et al. 2012, Cheng et al. 2011, Cho et al. 2011]. Em particular, nós analisamos as propriedades das RSPs derivadas das aplicações mencionadas acima, bem como uma RSP derivada do Waze, como redes de sensores, analisando mais de 30 milhões de registros (fotos, check-ins ou alertas) [Silva et al. 2013b, Silva et al. 2013a, Silva et al. 2013d]. Dentre os resultados, nós mostramos a escala planetária dessas redes, bem como a frequência altamente desigual do compartilhamento de dados, no espaço e no tempo, o que é altamente correlacionado com a rotina das pessoas. Alguns dos estudos anteriores também exploraram as propriedades das redes de sensores participativos para construir várias aplicações e serviços em diferentes áreas. Essas aplicações variam da extração de regiões distintas de uma cidade que refletem padrões de atividade coletiva similares, até a identificação automática de pontos de interesse [Cranshaw et al. 2012, Noulas et al. 2011b, Frias-Martinez et al. 2012, Quercia et al. 2012, Silva et al. 2013b, Sakaki et al. 2010].

Este estudo diferencia dos anteriores, incluindo os nossos, porque todos eles estudam diferentes camadas de sensoriamento de forma independente. Em contraste, o presente estudo objetiva propor um arcabouço de integração para permitir a análise e exploração de várias camadas simultaneamente. Ele foi motivado por um estudo anterior [Silva et al. 2013c] onde realizamos uma comparação entre as RSPs derivados do Instagram e Foursquare com o objetivo de entender se os dados de um sistema podem complementar dados de outro, ou se eles são compatíveis para o estudo da dinâmica de cidades e comportamento social urbano. Encontramos indícios de que estes sistemas podem completar um ao outro, no caso de identificação de pontos de interesse. Este resultado chamou nossa atenção para o potencial de utilização conjunta de camadas de sensoriamento.

Existem também estudos que consideram diferentes fontes de dados para entender melhor a dinâmica de cidades. Por exemplo, em [Bollen et al. 2011] os autores investigaram se os estados de humor coletivos derivados de mensagens compartilhadas no Twitter estão correlacionados com os índices da bolsa de valores Dow Jones (*Dow Jones Industrial Average*) ao longo do tempo. Os resultados indicam que é possível obter previsões relativamente precisas das variações diárias para cima e para baixo do fechamento da bolsa de valores estudada. Em [Sagl et al. 2012] os autores analisaram o comportamento humano coletivo utilizando dados de telefonia móvel correlacionados com dados meteorológicos. Os autores identificaram *insights* interessantes sobre a dinâmica humana coletiva no contexto de condições meteorológicas.

O presente trabalho também difere dessas propostas, pois aqui nós propomos um arcabouço para a integração de múltiplas camadas de sensoriamento, que pode ser aplicado em serviços mais sofisticados. Por exemplo, em qualquer cidade é provável encontrar vários locais onde as pessoas realizam com maior frequência uma determinada atividade, por exemplo uma área de bares e restaurantes onde as pessoas se encontram para socializar. Esses locais poderiam ser identificados com a ajuda da camada check-ins. As informações fornecidas por outras camadas poderiam ajudar os usuários na escolha das melhores áreas de interesse. Por exemplo, um usuário poderia usar as informações fornecidas pela camada de alertas de tráfego para identificar, dentre todas as opções, a área com menor número de problemas de trânsito no momento, e utilizar a camada fotos de lugares para visualizar o perfil dos estabelecimentos das áreas e das pessoas que os frequentam.

## 3. O Conceito de Camadas de Sensoriamento

Nós sensores em uma RSP são compostos por usuários, cada um com um dispositivo portátil. Assim, quando nos referimos a um usuário, estamos nos referindo a um nó sensor em um RSP. Perceba que o sensoriamento depende da vontade das pessoas participarem no processo de sensoriamento, assim um usuário pode ser contribuinte ou não em uma determinada janela de tempo. Em estudos anteriores [Silva et al. 2013b, Silva et al. 2013a, Silva et al. 2013d] apresentamos alguns exemplos de RSPs que oferecem dados sobre vários aspectos de uma região geográfica. Isto ajuda a ilustrar a ubiquidade e diversidade dos dados que podem estar disponíveis. Este universo de "dados ubíquos" pode ser complexo de entender e lidar, abrindo oportunidades de pesquisa para apresentação de propostas para facilitar essas tarefas.

Uma camada de sensoriamento representa dados, com seus atributos correspondentes, provenientes de uma determinada fonte de dados. Exemplos de fontes de dados incluem: dados disponibilizados através de *web services*, por exemplo a condição do tempo fornecida pela empresa "The Weather Channel"[6]; dados de redes de sensores sem fio tradicionais; e dados de redes de sensores participativos. Discutimos o conceito de camadas de sensoriamento para RSPs. No entanto todos os conceitos discutidos aqui podem ser utilizados para outras fontes de dados associadas a regiões geográficas pré-definidas, com as adaptações necessárias.

A Figura 1 ilustra essa ideia, exibindo quatro camadas diferentes para Nova York: **alertas de tráfego** (obtida, por exemplo, a partir do Waze); **check-ins** (obtida, por exemplo, a partir do Foursquare); **condição do clima** (obtida, por exemplo, a partir do Weddar), e **fotos de lugares** (obtida, por exemplo, a partir do Instagram). Tal como ilustrado, uma camada de sensoriamento é representada por um plano, onde as observações de certo aspecto de uma região geográfica pré-definida estão dispostas. A cada observação (em cada camada) estão associados dados relativos a: tempo (i.e., quando a observação foi feita), espaço (localização geográfica), pessoa (usuário), e dados específicos de cada camada (*especialidade*).

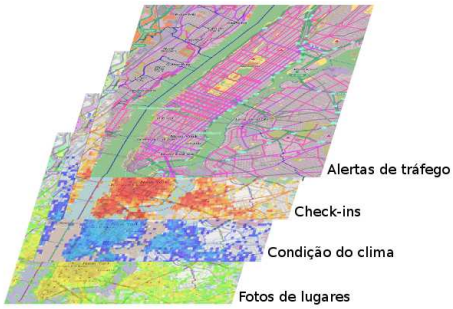



**Figura 1. Camadas de sensoriamento para a cidade de Nova York.**

A Figura 2 complementa a definição de RSP apresentada em [Silva et al. 2013b], agora incorporando o conceito de camadas de sensoriamento. Esta figura ilustra três camadas de sensoriamento diferentes, a saber: *fotos de lugares*; *alertas de tráfego*; e *check-ins*. Essa figura ilustra ainda outras camadas derivadas da camada fotos de lugares. Como podemos ver, existem dois tipos de camadas de sensoriamento: **primitiva** e **contextual**.

Uma camada de sensoriamento primitiva é composta por dados básicos provenientes diretamente do sorvedouro/fonte, descrevendo aspectos específicos de localizações geográficas no espaço, tempo e usuários contribuintes. Cada sensor (usuário + dispositivo portátil) em uma RSP pode produzir um fluxo de dados. Neste contexto, as aplicações provêm um fluxo contínuo de dados. Como representado na Figura 2, esses dados devem ser coletados (por exemplo,

[6]http://www.weather.com

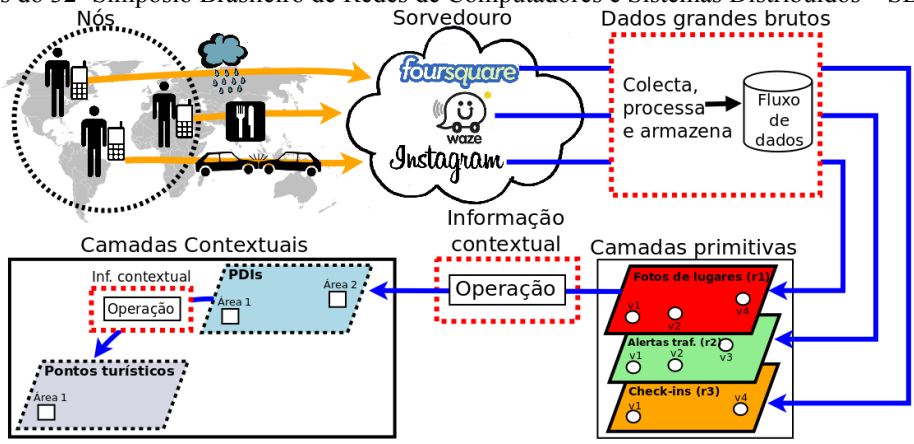



**Figura 2. Visão geral de redes de sensores participativos com o conceito de camadas de sensoriamento.**

usando uma API) e processados, passo que inclui as tarefas de análise e padronização dos dados, uma vez que todos os registros de todas as camadas primitivas devem possuir a mesma estrutura. O último passo é o armazenamento dos dados. Essas etapas não buscam extrair contexto [Dey and Abowd 2000] (ou conhecimento) a partir dos dados medidos, mas sim organizá-los. Com isso, camadas primitivas oferecem dados básicos que podem ser utilizados para a inferência de contexto.

Para ilustrar este processo, considere os dados provenientes de um RSP derivada do Foursquare. No Foursquare, os usuários podem, dentre outras ações, realizar check-ins em lugares e deixar dicas sobre locais visitados. Destes dados mencionados, podemos extrair pelo menos duas camadas primitivas, a saber: check-ins, contendo os check-ins realizados pelos usuários (check-ins podem ser usados para descobrir locais populares, por exemplo), e dicas de lugares, contendo dicas (*tips*), por exemplo "este restaurante tem uma comida incrível", enviadas por usuários sobre determinados lugares. A geração de camadas primitivas é representada na Figura 2 por uma caixa nomeada "dados grandes brutos", e depende de operações específicas de cada sistema. Considere check-ins do Foursquare, por exemplo. Uma maneira de obtê-los é através do Twitter, como explicado em [Silva et al. 2013a]. Isso significa que temos que coletar, analisar e processar *tweets*. A codificação de ferramentas para realizar estas etapas varia de acordo com o sistema ou aplicação. Posteriormente, nós devemos criar uma estrutura para representar os dados de interesse associados com um certo local onde foram compartilhados, assim representando uma camada primitiva. Em uma camada primitiva, cada dado tem os seguintes atributos:

- $t$ - instante em que o dado foi compartilhado;
- $a$ - localização (coordenadas GPS), onde o dado foi compartilhado. Cada dado tem uma coordenada GPS. Como esta informação está sujeita a erros, consideramos que a localização dos dados é representada por uma área. Essa área pode ser, por exemplo, um círculo com raio de $x$ metros a partir da coordenada GPS;
- $s$ - especialidade da camada;
- $u$ - id do usuário que compartilhou o dado.

Uma ou várias camadas primitivas podem ser processadas (possivelmente após uma fusão) para construir camadas contextuais. Uma camada contextual representa contexto ou conhecimento obtido através da transformação dos dados disponíveis em uma ou mais camadas primitivas. Pontos de interesse (PDI) em uma cidade, obtidos a partir dos dados compartilhados no Instagram, conforme descrito em [Silva et al. 2013b], representam um exemplo de camada contextual. Em uma camada contextual $c$ o contexto inferido, possui os seguintes atributos:

- $w_t$ - um vetor que descreve uma janela de tempo contendo os dados da camada primitiva utilizados para inferir um contexto;
- $a$ - um vetor com uma ou mais áreas associadas com o contexto. Por exemplo, um PDI representa um contexto, nesse caso $a$ é um vetor contendo a área do PDI.
- $s$ - especialidade (o próprio contexto);
- $O$ - grupo de ids dos usuários que contribuíram os dados usados na geração do contexto.

Cada camada tem uma variável $h$, onde $h = 0|1$, representando os estados ativos e inativos de uma camada. A lista abaixo representa alguns exemplos de camadas, primitivas e contextuais, que estão disponíveis hoje:

1. Check-ins (**primitiva**, exemplo de fonte: Foursquare);
2. Dicas sobre locais (**primitiva**, exemplo de fonte: Foursquare) ;
3. Alertas de tráfego (**primitiva**, exemplo de fonte: Waze) ;
4. Fotos de lugares (**primitiva**, exemplo de fonte: Instagram);
5. *City Image* (**contextual**, derivada da camada check-ins);
6. Pontos de interesse (**contextual**, derivada, por exemplo, da camada fotos de lugares);
7. Pontos turísticos (**contextual**, derivada, por exemplo, da camada pontos de interesse);

O contexto referente à camada *city image* é um resumo da dinâmica de uma área geográfica com base em movimentos realizados pelos usuários (camada detalhada em [Silva et al. 2012]). O contexto da camada PDI refere-se às áreas que atraem mais atenção de moradores e visitantes. Dentre os PDIs temos os pontos turísticos, contexto referente à camada pontos turísticos. As camadas pontos de interesse e pontos turísticos são detalhadas em [Silva et al. 2013b].

## 4. Formalização do Modelo de Camadas de Sensoriamento

Seja $U = \{u_1, u_2...u_n\}$ um conjunto de usuários (nós sensores), e seja $P = \{p_1, p_2, ...p_n\}$ um conjunto de RSPs. Cada usuário $u_i \in U$ pode compartilhar dados ilimitados em qualquer RSP $p_k \in P$. Cada $j$-ésimo dado compartilhado $d_j^{p_k}$ em uma RSP $p_k$ possui o formato $d_j^{p_k} =< t, m >$, em que $t$ refere-se ao instante em que o usuário $u_i$ compartilhou o dado em $p_k$ e $m$ é uma tupla contendo os atributos deste dado. A tupla $m$ é composta pelos atributos presentes em todos os dados provenientes de uma RSP, ou seja, $m = (a, u, s)$, onde $a$ é a área[7] do local onde o dado foi compartilhado, $s$ é a especialidade, e $u$ refere-se ao usuário $u_i \in U$ que compartilhou o dado.

Os dados compartilhados em $p_k \in P$ podem ser vistos como um fluxo de dados $B^{p_k}$. Definimos que um fluxo de dados $B^{p_k}$ é composto por todos os $n$ dados compartilhados pelos usuários $U$ em uma RSP $p_k$ em um determinado intervalo de tempo. Desta forma, $B^{p_k} =< d_1^{p_k}, d_2^{p_k}, ..., d_n^{p_k} >$, e $B^{p_k}$ representa uma camada primitiva $r_{p_k}$. A Tabela 1 mostra exemplos de dados presentes em camadas primitivas que foram compartilhados nas três RSPs derivadas do Foursquare, Waze e Instagram, ilustradas na Figura 2. Observe que dados no mesmo fluxo podem ter o mesmo tempo, pois podem ter sido compartilhados por dois usuários simultaneamente.

Para trabalhar com camadas precisamos representá-las em um *plano de trabalho*, que contém uma ou mais camadas. Esse plano de trabalho representa uma combinação dos dados das camadas que desejamos trabalhar. Como efetuar essa combinação dos dados depende da funcionalidade da camada, do que ela captura. A abstração usada para representar uma combinação é um dicionário de dados $M$, que representa uma coleção de pares $chave : valor$. Definimos que a função responsável por realizar esse passo chama-se $COMBINATION(\mathcal{F}, relation())$, em

[7] Lembre-se de que um ponto GPS é representado por uma área, devido a erros de GPS.

**Tabela 1. Fluxo de dados descrevendo a atividade de usuários em três RSP diferentes: Foursquare, Waze e Instagram.**

| Fluxo de dados de uma RSP derivada do Foursquare | | | |
|---|---|---|---|
| | Atributos ($m$) | | |
| Tempo ($t$) | Área ($a$) | Usuário ($u$) | Especialidade ($s$) |
| T1 | $a_1$ | 1 | “Times Square” |
| T1 | $a_1$ | 2 | “Times Square” |
| T2 | $a_2$ | 1 | “Fifth Av.” |
| T3 | $a_4$ | 1 | “Statue of Liberty” |

| Fluxo de dados de uma RSP derivada do Waze | | | |
|---|---|---|---|
| | Atributos ($m$) | | |
| Tempo ($t$) | Área ($a$) | Usuário ($u$) | Especialidade ($s$) |
| T1 | $a_1$ | 3 | “Congestionamento” |
| T2 | $a_3$ | 2 | “Acidente” |
| T2 | $a_3$ | 3 | “Controle policial” |

| Fluxo de dados de uma RSP derivada do Instagram | | | |
|---|---|---|---|
| | Atributos ($m$) | | |
| Tempo ($t$) | Área ($a$) | Usuário ($u$) | Especialidade ($s$) |
| T1 | $a_1$ | 3 | “dados da foto” |
| T3 | $a_4$ | 1 | “dados da foto” |

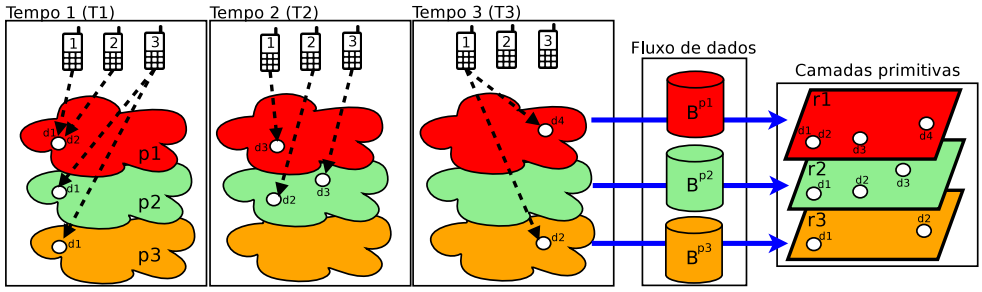



**Figura 3. Ilustração de compartilhamento de dados em três RSP ao longo do tempo, resultando em camadas primitivas.**

que $\mathcal{F}$ é um subconjunto de $\mathcal{B}$, ou $\mathcal{F} \subseteq \mathcal{B}$, e $relation()$ é uma função que define a relação entre os dados dos fluxos $B^{p_k}$ contidos em $\mathcal{F}$. A função $relation()$ define as chaves do dicionário $M$, bem como os dados que essas chaves irão referenciar, que são as demais observações do dado $d_i^{p_k}$ não utilizadas como chave. A operação $COMBINATION$ resulta em um dicionário $M$.

Nós ilustramos aqui dois tipos de relações utilizadas para agregar os dados: (1) por locais e (2) por usuários. Para simplificar, vamos ilustrar a combinação apenas para as camadas primitivas, mas a mesma ideia também é válida para camadas contextuais. Para ilustrar um plano de trabalho contendo os dados combinados por localidade, vamos considerar a atividade representada na Figura 3. O plano de trabalho $M_1$ representa essa atividade, onde $M_1 = \{a_1 : \{d_1^{r1'}, d_2^{r1'}, d_1^{r2'}, d_1^{r3'}\}, a_2 : \{d_3^{r1'}, d_2^{r2'}\}, a_3 : \{d_3^{r2'}\}, a_4 : \{d_4^{r1'}, d_2^{r3'}\}\}$. Nesse plano de trabalho uma chave $k_i$ é representada por $a_i$, que é uma área única dentre todas as áreas de todos os dados compartilhados nas camadas consideradas: $r_1$, $r_2$ e $r_3$. Já $d_i^{rj'}$ refere-se às observações não utilizadas como chave do dado $d_i$ da camada primitiva $r_j$, ou seja $< t, u, s >$. Assim, todas as áreas únicas tornaram-se uma chave no plano de trabalho $M_1$. Os dados compartilhados em um mesmo local foram agrupados e são indexados pela chave que representa o local. O plano

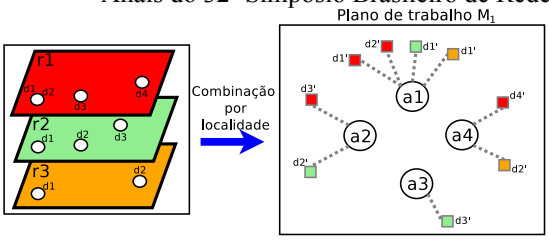



**Figura 4. Combinação por localidade.**

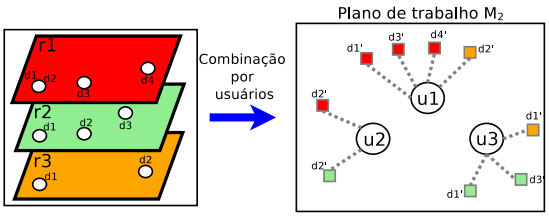



**Figura 5. Combinação por usuários.**

de trabalho $M_1$, como descrito, está representado na Figura 4. Veja que o plano de trabalho representa dados que foram compartilhados em todas as camadas consideradas. A cor do símbolo que representa um dado $d'_i$ indica de qual camada ele foi extraído, por exemplo, a cor vermelha é referente à camada $r_1$.

A Figura 5 ilustra uma combinação por usuário. Neste caso, é criado um plano de trabalho contendo chaves que representam os usuários. A figura mostra o plano de trabalho $M_2$, que foi criado considerando as atividades mostradas na Figura 3. O conteúdo desse plano de trabalho é: $M_2 = \{u_1 : \{d_1^{r1'}, d_3^{r1'}, d_4^{r1'}, d_2^{r3'}\}, u_2 : \{d_2^{r1'}, d_2^{r2'}\}, u_3 : \{d_1^{r2'}, d_1^{r3'}, d_3^{r2'}\}\}$. Todos os usuários únicos tornaram-se uma chave em $M_2$. Os dados foram combinados e cada grupo foi apontado pela chave que representa o usuário que efetuou o compartilhamento dos dados no grupo.

## 5. Operações

Na Seção 4, dentre outras coisas, ilustramos como representar camadas de sensoriamento em um plano de trabalho, por exemplo, por localidade ($M_1$) ou usuários ($M_2$). Os planos de trabalho gerados são estruturas básicas que podem ser facilmente manipuladas. De posse de um plano de trabalho, como o $M_1$ ou o $M_2$ mostrado nas figuras 4 e 5, podemos aplicar operações para derivar outras estruturas e também extrair conhecimento. Exemplos de operações são:

- **uGRAPH** (*undirected graph*): *Espera* um plano de trabalho. *Resulta* em um grafo não direcionado. Essa operação cria, a partir de um plano de trabalho, um grafo $G = (V, E)$ não direcionado, que consiste em um conjunto de nós $V$ e um conjunto de arestas $E$. Cada chave $k_i$ no plano de trabalho representa um nó $v_i \in V$, e os dados que são indexados por $k_i$ são atributos de $v_i$. Uma aresta $e = (v_i, v_j)$ pode ser adicionada conectando os nós $v_i$ e $v_j$ dependendo da análise desejada, que é expressa através de algumas operações específicas, que descrevemos a seguir. Inicialmente, $E = \varnothing$. Todas as variáveis do plano de trabalho são incorporadas no grafo gerado;
- **dGRAPH** (*directed graph*): *Espera* um plano de trabalho. *Resulta* em um grafo direcionado. A operação cria um grafo *direcionado* respeitando a operação **uGRAPH**;
- **CNG** (*change*): *Espera* um plano de trabalho, um $ID$ da camada e um status ($0$ ou $1$). *Resulta* em um plano de trabalho com as modificações impostas pela mudança de $h$. Essa operação altera o status atual de uma camada através de $h$. Se o status informado é igual a $0$ então a camada em questão é desabilitada. Essa camada pode ser reabilitada posteriormente.
- **dEDGE** (*directed edge*): *Espera* um grafo direcionado resultante de um plano de trabalho combinado por localidade. *Resulta* em um grafo com arestas direcionadas. A operação cria uma aresta direcionada do nó $v_i$ para o nó $v_j$ se e somente se pelo menos um mesmo usuário compartilhou um dado, em qualquer camada, no local representado pelo nó $v_j$ logo após o compartilhamento de um dado, também em qualquer camada, em uma localização representada pelo nó $v_i$. O peso de uma aresta representa o número

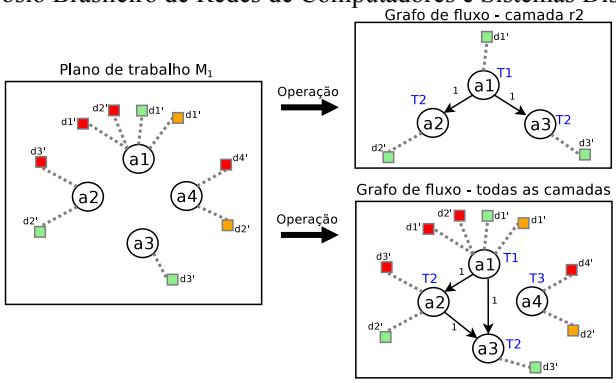



**Figura 6. Ilustração da criação de grafo de fluxo considerando uma única e múltiplas camadas.**

total de transições realizadas a partir de $v_i$ para $v_j$ considerando transições de todos os usuários.

- **DEL**: *Espera* um grafo $G$ e um inteiro $t$. *Resulta* em um subgrafo derivado de $G$. A operação apaga arestas $e_i \in E$ ($E$ é o conjunto de arestas de $G$), com peso $w_i < t$.

Todas estas operações mencionadas, com as devidas adaptações, são adequadas para camadas primitivas e contextuais. Existem ainda operações específicas para criar camadas contextuais, como as operações mostradas a seguir:

- **fPOIS** (*find POIs*): *Espera* um plano de trabalho representando uma camada de sensoriamento. *Resulta* em um plano de trabalho contendo PDIs. A operação identifica pontos de interesse (PDIs), que são obtidos aplicando o algoritmo especificado no trabalho [Silva et al. 2013b] em áreas geográficas desejadas.
- **fSIGHTS** (*find sights*): *Espera* um plano de trabalho contendo PDIs. *Resulta* em um grafo contendo pontos turísticos. A operação identifica pontos turísticos a partir de um grafo $G^{PDIs} = (V, E)$, onde os nós $v_i \in V$ são as áreas $a$ dos PDIs identificados em uma determinada região geográfica pré-definida (algoritmo detalhado em [Silva et al. 2013b]).

Para dar um exemplo de uso de operações, vamos demonstrar como gerar grafos de fluxo, que grafos que mapeiam os locais onde um mesmo usuário contribuiu dados, capturando assim a movimentação ou transição dos usuários em uma área geográfica. É particularmente interessante ilustrar a criação de grafos de fluxo, pois usamos esse tipo de grafo nas aplicações descritas na Seção 6. A Figura 3 representa três usuários que compartilharam dados em diferentes RSPs, $p_1$ (vermelha), $p_2$ (verde) e $p_3$ (laranja) em três intervalos diferentes de tempo ($T1$, $T2$ e $T3$). Dados compartilhados pelos usuários em cada intervalo são apontados com setas tracejadas. Observe que os usuários não participam necessariamente em múltiplos sistemas todo o tempo, além disso podem contribuir para diferentes camadas no mesmo intervalo de tempo.

Após um determinado tempo, pode-se modelar os dados com o intuito de extrair conhecimento de diferentes maneiras. Por exemplo, na Figura 6 a parte nomeada "Grafo de fluxo - camada $r2$" mostra um grafo de fluxo que chamamos de $G^{flow}_{r_2}$. Para a geração desse grafo vamos considerar o plano de trabalho $M_1$ como explicado acima (combinado por localidades). Primeiramente, escondemos as camadas $r1$ e $r3$: $M_1^{'} \leftarrow$ **CNG**$(M_1, r1, 0)$ e $M_1^{''} \leftarrow$ **CNG**$(M_1^{'}, r3, 0)$. Em seguida geramos um grafo direcionado: $G \leftarrow$ **dGRAPH**$(M_1^{''})$. Por último aplicamos a operação **dEDGE** em $G$: $G^{flow}_{r_2} \leftarrow$ **dEDGE**$(G)$. Neste caso, temos um grafo de fluxo que representa dados de uma única camada. Com este grafo podemos extrair diversas informações valiosas, por exemplo, as trajetórias regulares em uma cidade, como demonstrado em [Silva et al. 2012].

Outra possibilidade de análise é considerar diferentes camadas ao mesmo tempo. Na Figura 6, a parte nomeada "Grafo de fluxo - todas as camadas" mostra um grafo que chamamos de $G^{flow2}$. Vamos considerar também o plano de trabalho $M_1$, usado acima. Com o intuito de obter $G^{flow2}$ primeiramente criamos um grafo direcionado $G$: $G \leftarrow$ **dGRAPH**$(M_1)$. Depois disso aplicamos **dEDGE** em $G$: $G^{flow2} =$ **dEDGE**$(G)$. No grafo resultante, os nós representam dados compartilhados no mesmo local, em qualquer camada. As arestas conectam os nós $v_i \rightarrow v_j$ se pelo menos um mesmo usuário compartilhou um dado no local representado pelo nó $v_j$ logo após compartilhar um dado no local representado pelo nó $v_i$.

## 6. Aplicações do Arcabouço de Camadas de Sensoriamento

Nesta seção discutimos duas aplicações que ilustram o potencial do arcabouço proposto.

### 6.1. Identificação de Pontos Turísticos

Em [Silva et al. 2013c] mostramos indícios de que o uso integrado de múltiplas camadas, especificamente uma derivada do Instagram e outra do Foursquare, poderia melhorar a precisão da identificação de pontos turísticos em uma cidade. Com base nisso, nesta seção discutimos uma aplicação que identifica pontos turísticos considerando múltiplas camadas simultaneamente, destacando as melhorias sobre a estratégia apresentada em [Silva et al. 2013c], que considera apenas uma camada.

Nesta análise, consideramos os datasets do INSTAGRAM (1.855.235 fotos) e FOURSQUARE (4.548.941 check-ins) que foram coletadas diretamente do Twitter[8], uma vez que fotos do Instagram e check-ins do Foursquare não são publicamente disponíveis por *default*. Ambos datasets foram coletados durante o mesmo período. Cada conteúdo (foto ou check-in) consiste de coordenadas GPS (latitude e longitude) e do instante em que foi compartilhado. Neste trabalho consideramos os dados compartilhados na área de Belo Horizonte representada pela Figura 7, que cobre praticamente toda a cidade.

A camada fotos de lugares ($r_1$) é representada pelo dataset INSTAGRAM, e a camada check-ins ($r_2$) pelo dataset FOURSQUARE. Nosso intuito é trabalhar com esses dados conjuntamente. Para isso, agregamos os dados por localização, produzindo um plano de trabalho representado pelo plano de trabalho $M_1$. Inicialmente identificamos os pontos turísticos para $r_1$. Com isso em mente desabilitamos a camada $r_2$ de $M_1$ obtendo $M_{r_1}$: $M_{r_1} \leftarrow$ **CNG**$(M_1, r2, 0)$. Depois disso aplicamos **fPOIS** em $M_{r_1}$ para gerar $M_{r_1}^{pdis}$, plano de trabalho contendo os pontos de interesse. No plano de trabalho resultante $M_{r_1}^{pdis}$, as chaves são as áreas dos PDIs identificados. No cenário ilustrado pela Figura 2 temos apenas duas chaves em um plano de trabalho, representadas por "área 1" e "área 2".

Cada PDI em $M_{r_1}^{pdis}$ representa uma área $a$ popular em uma determinada região geográfica, por exemplo uma cidade. A popularidade é identificada através do volume de contribuição de dados realizada pelos usuários $O$[9]. Ou seja, um PDI representa a atividade realizada por um grupo de usuários $O$ em uma janela de tempo $w_t$. Note que, a especialidade $s$, neste caso particular, é a própria área. Utilizamos o plano de trabalho $M_{r_1}^{pdis}$ para a extração de pontos turísticos com o auxílio da operação **fSIGHTS**. Primeiramente é criado um grafo direcionado (no exemplo $G_{r_1}^{pois}$), a partir de um plano de trabalho (no exemplo $M_{r_1}^{pois}$), usando a operação **dGRAPH**. Em seguida a operação mapeia as transições realizadas pelos usuários entre os PDIs. Para isso, é aplicada a operação **dEDGE** em $G_{r_1}^{pdis}$ obtendo $G_{r_1}^{pdis-flow}$.

[8]http://www.twitter.com.

[9]Relembre a notação na Seção 3.


**Figura 7. Todos os pontos turísticos identificados com Foursquare e Instagram datasets.**

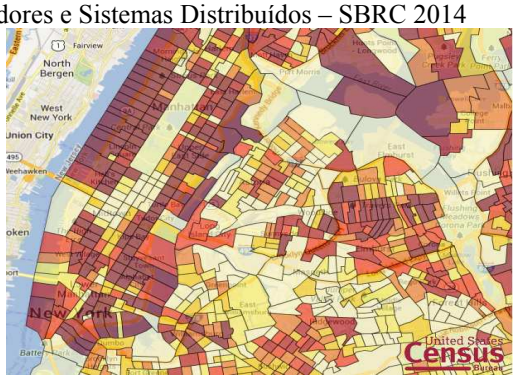



**Figura 8. Exemplos de setores de Nova York que têm o rendimento mediano fornecido pelo U.S. Census Bureau.**

Depois disso, são selecionadas transições populares que ligam dois nós $v_i \rightarrow v_j$. Para isso, é utilizada a operação **DEL** no grafo $G_{r_1}^{pois-flow}$ usando um parâmetro $t$ (calculado como mostrado em [Silva et al. 2013b]). De acordo com a conjectura considerada no algoritmo da operação **fSIGHTS**, as transições populares selecionadas conectam os pontos turísticos, que estão representados no grafo $G_{r_1}^{sights} = (V', E')$. Nesse grafo os nós $v_i \in V'$ são as áreas $a$ dos pontos turísticos identificados, representados em $M_{r_1}^{sights}$.

Em seguida, identificamos os pontos turísticos $M_{r_2}^{sights}$ para $r_2$. Primeiramente reativamos a camada $r_2$ e desativamos $r_1$ em $M_1$, utilizando a operação **CNG**, obtendo assim o plano de trabalho $M_{r_2}$. Os próximos passos são realizados de maneira semelhante ao que foi apresentado para $r_1$. Depois disso agrupamos as camadas contendo os pontos turísticos para as camadas $r_1$ e $r_2$, $M_{r_1}^{sights}$ e $M_{r_2}^{sights}$, no plano de trabalho $M_{total}^{sigths}$ (por exemplo utilizando uma operação **MERGE** que agrupa dois planos de trabalho). Este plano de trabalho contém todos os pontos turísticos identificados, que são mostrados pela Figura 7. O ponto turístico apontado por uma seta vermelha (Mercado Central), foi identificado apenas pelo Foursquare. Os pontos turísticos apontados por uma seta azul (Igreja da Pampulha, Lagoa da Pampulha e área de lazer 2) não foram identificados com Foursquare. Todos os locais são muito relevantes, note com isso o potencial para a complementação de resultados usando ambos os sistemas.

## 6.2. Análise Econômico-Cultural de Regiões

A aplicação descrita nesta seção permite a realização de várias análises econômico-culturais, porém neste trabalho vamos focar em apenas duas. O objetivo da primeira análise é correlacionar o sentimento geral expressado nas *dicas*[10] para todos os locais de um determinado setor $a_i$, com a renda mediana dos habitantes desse setor. Já o objetivo da segunda análise é estudar o movimento dos usuários nos setores considerados, levando em consideração a renda típica desses setores. Essa segunda análise possibilita o estudo de segregação social em uma cidade.

Para ilustrar essa aplicação, consideramos dois conjuntos de dados derivados do Foursquare. O primeiro, nomeado CHECKINS-NY, é composto por 34.677 check-ins realizados na cidade de Nova York, em uma semana de Abril de 2012. Os dados desse dataset foram coletados diretamente do Twitter e cada check-in consiste de uma coordenada GPS (latitude e longitude), de um ID do local onde o check-in foi realizado, do instante em que foi compartilhado e da categoria do local. O segundo dataset, nomeado DICAS-NY, contém todas as dicas contribuídas até 01/2013, em todos os locais únicos do dataset CHECKINS-NY. As dicas foram coletadas por meio da API do Foursquare, e cada dica contém um ID do local onde ela foi

[10]veja o artigo [Vasconcelos et al. 2012] para saber mais sobre dicas no Foursquare.

compartilhada, o instante de tempo do compartilhamento, e o conteúdo textual da dica. Consideramos apenas dicas em Inglês. Definimos que uma dica está na língua Inglesa se pelo menos metade das palavras constam em um dicionário contendo as principais palavras dessa língua. Isso resultou em: 157.197 dicas (2.531 descartadas).

O dataset DICAS-NY é utilizado para representar uma camada primitiva chamada dicas de locais ($r_1$). A camada $r_1$ é composta por um fluxo de dados $B_i$. Cada dado no fluxo tem a forma: $< t, (a, u, s) >$. Um exemplo de especialidade $s$ desta camada é: "Este lugar é incrível, eu recomendo o fígado com jiló.". A camada contextual renda $c_1$ é composta por um dataset $d_j$ para diferentes setores $j$ da cidade de Nova York. Essa camada foi derivada de uma camada primitiva contento informações individuais dos habitantes de determinados setores, que participaram do censo de Nova York. A área de cada setor é pré-definida pelo censo de Nova York. Cada especialidade em $d_j$ possui a renda mediana dos habitantes de um setor específico. A Figura 8 representa alguns exemplos de setores considerados em Nova York. Todos os dados são referentes ao *2006-2010 American Community Survey*. A forma de $d_1$ é $< t =$2006-2010,$a$=[área setor 1],$O$="habitantes do setor 1 que participaram do censo", $s$="renda mediana em US$ para o setor 1"$>$. Note que apesar dessa fonte de dados não ser proveniente de um usuário com seu dispositivo móvel, também pode ser considerada uma camada contextual sobre uma região geográfica pré-definida. Isso mostra a flexibilidade do modelo proposto.

Para a primeira análise, inicialmente devemos agregar os dados das camadas $r_1$ e $c_1$. O método escolhido é a combinação por localidades, método descrito na Seção 4. Este processo de combinação resulta em um plano de trabalho $M_1$, onde as chaves são as áreas dos setores. Cada chave $k_i$ agrega, dentre os demais dados , as dicas de todos os locais que se localizam dentro da área de um setor, bem como a informação de renda mediana do setor.

A partir disso, usamos o plano de trabalho criado para calcular o sentimento geral sobre todos os locais de cada setor. Para esta análise foi utilizado o programa SentiStrength [Thelwall et al. 2010] para classificar o sentimento expressado nas dicas em positivo ou negativo em uma escala de -4 (fortemente negativo) a 4 (fortemente positivo), 0 indica um sentimento neutro. Esse programa é aplicado a cada dica e depois agregado por lugar e por último por setor. Com isso, calculamos o sentimento médio para todos os locais de um determinado setor. Em seguida, agrupamos os setores por cinco faixas de renda: inferior a US$25.000; entre US$25.000 e US$50.000; entre US$50.000 e US$75.000; entre US$75.000 e US$100.000; e superior a US$100.000. Finalmente, calculamos o valor médio do sentimento para cada um dos cinco grupos de renda, considerando todos os setores que pertencem a cada um desses grupos. A tabela 2 apresenta esse resultado, e para cada grupo de renda, a percentagem do valor médio do sentimento dos setores em cinco faixas de sentimento: (+3,+4); (+1,+2); (0); (-1,-2); e (-3,-4). Conforme é possível observar, o resultado sugere que setores pobres tendem a ter o pior sentimento expressado pelos usuários. Isto pode estar associado a serviços de baixa qualidade nesses setores. Com o aumento da renda do setor as opiniões tendem a ser melhores. Apesar do sentimento médio do grupo de setores mais ricos (superior a US$100000) ser um pouco menor do que o segundo mais rico (entre US$75000 e US$100000), esse ainda tem um maior número de dicas positivas do que todos os outros grupos, e ainda não possui dicas negativas. Note o potencial desta análise para estudos sociais, por exemplo para o estudo de desigualdades na qualidade de serviços em cidades.

Para a segunda análise utilizamos o dataset CHECKINS-NY, que é usado para representar uma camada primitiva chamada check-ins ($r_2$). Nós agregamos as camadas $r_2$ e $c_1$ por localidade no plano de trabalho $M_2$. A partir de $M_2$ criamos um grafo $G_2 = \textbf{uGRAPH}(M_2)$. Nós usamos $G_2$ para gerar um grafo de fluxo $G_2^{flow} = \textbf{dEDGE}(G_2)$, onde as arestas são as

**Tabela 2. Sentimento geral por grupos de setores**

| Grupo | Sent. Médio (std) | (+3,+4)% | (+1,+2)% | (0)% | (-1,-2)% | (-3,-4)% |
|---|---|---|---|---|---|---|
| $<$25000 | 0,46 (0,67) | 0 | 73,08 | 21,15 | 5,77 | 0 |
| $\geqslant$25000 e $<$50000 | 0,73 (0,63) | 1,23 | 84,31 | 12,92 | 1,23 | 0,31 |
| $\geqslant$50000 e $<$75000 | 0,81 (0,46) | 0,40 | 93,28 | 5,93 | 0,39 | 0 |
| $\geqslant$75000 e $<$100000 | 0,9 (0,36) | 0 | 96,97 | 3,03 | 0 | 0 |
| $\geqslant$100000 | 0,87 (0,28) | 0,96 | 98,08 | 0,96 | 0 | 0 |

transições (como explicado na Seção 5), realizadas pelo mesmo usuário em diferentes setores (nós no grafo). Excluímos laços, ou seja, visitas do mesmo usuário no mesmo setor. Com isso, estudamos a assortatividade relacionada com a renda mediana por setor em $G_2^{flow}$. Isso é uma maneira de tentar observar a existência de segregação.

A assortatividade mede a similaridade de conexões na rede em relação a um determinado atributo, e varia de -1 a +1 [Newman 2002]. Em uma *rede assortativa* (com assortatividade positiva), vértices com valores similares por um dado atributo (por exemplo, a mesma faixa de renda), tendem estarem conectados (serem semelhantes) entre si, enquanto que em uma *rede disassortativa* (com assortatividade negativa), acontece o contrário. Todos os setores foram associados a uma classe com base no rendimento mediano do setor. A classe A para a renda mediana até \$75.000 e a classe B para uma renda mediana superior. A assortatividade considerando essas duas classes como atributos de $G_2^{flow}$ é de 0,14 . Assim, a rede para este atributo é assortativa, o que indica um traço de segregação, ou seja, usuários tendem a compartilhar conteúdo em (ou frequentar) setores que possuem a mesma classe de renda.

Depois disso, criamos dez grafos aleatórios[11] $G_{Ri}(V, E_{Ri})$, onde $i = 1, \ldots, 10$. Cada grafo aleatório é construído usando o mesmo número de transições individuais de $G_2^{flow}$. No entanto, em vez de considerar a transição real $v_i \rightarrow v_j$ realizada por um indivíduo, escolhemos aleatoriamente dois setores para substituir $v_i$ e $v_j$, simulando transições aleatórias realizadas pelos usuários. Para cada grafo $G_{Ri}$ também é associado aleatoriamente a classe de um nó, A ou B. O número de nós da classe A e B também estão de acordo com o observado em $G_2^{flow}$. Depois disso, calculamos a assortatividade para todos os grafos aleatórios $G_{R1..10}$. A assortatividade para todos os grafos, com 95% de confiança, está no intervalo: $[-0,0084, -0,0014]$. Como podemos ver, essas redes aleatórias são disassortativas, não indicando segregação. Obviamente, a fim de tirar qualquer conclusão nesse sentido é necessário uma investigação mais detalhada, no entanto, isso serve para demonstrar o potencial de análise conjunta de múltiplas camadas.

Observe ainda o potencial de se considerar as mesmas camadas para gerar um plano de trabalho $M_3$ combinado por usuários. Além de identificar as preferências dos usuários, também podemos tentar inferir sua classe social estudando a renda dos setores que o usuário frequenta. Isto pode ser útil para estudos sociais, bem como para uma publicidade mais eficaz.

## 7. Conclusão e Trabalhos Futuros

Neste estudo apresentamos o que é, no melhor de nosso conhecimento, o primeiro trabalho que define e usa o conceito emergente de camadas de sensoriamento. Para isso, propomos um arcabouço para integração de múltiplas camadas de sensoriamento, que foi ilustrado na construção de duas aplicações que utilizam múltiplas camadas de sensoriamento.

Como trabalhos futuros, pretendemos desenvolver outras aplicações que exploram o arcabouço proposto para integrar outras camadas de sensoriamento. Por exemplo, a camada alerta

[11]Note que poderíamos ter uma nova operação no arcabouço para esse propósito.

de tráfego derivada do Waze e a camada condição do clima derivada do Weddar.

## Agradecimentos

Este trabalho é parcialmente financiado pelo INCT-Web (MCT/CNPq 57.3871/2008-6), e pelas bolsas e projetos individuais dos autores financiados pelo CNPq, CAPES (bolsa 7356/12-9), e FAPEMIG.

## Referências

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., and Srivastava, M. B. (2006). Participatory sensing. In *Workshop WSW'06*, pages 117–134.

Cheng, Z., Caverlee, J., Lee, K., and Sui, D. Z. (2011). Exploring Millions of Footprints in Location Sharing Services. In *Proc. of ICWSM'11*, Barcelona, Spain.

Cho, E., Myers, S. A., and Leskovec, J. (2011). Friendship and mobility: user movement in location-based social networks. In *Proc. of ACM KDD'11*, pages 1082–1090, San Diego, California, USA.

Crandall, D. J., Backstrom, L., Huttenlocher, D., and Kleinberg, J. (2009). Mapping the world's photos. In *Proc. of the WWW '09*, pages 761–770, Madrid, Spain. ACM.

Cranshaw, J., Schwartz, R., Hong, J. I., and Sadeh, N. (2012). The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City. In *Proc. of the ICWSM'12*, Dublin, Ireland.

Dey, A. K. and Abowd, G. D. (2000). Towards a Better Understanding of Context and Context-Awareness. In *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*.

Frias-Martinez, V., Soto, V., Hohwald, H., and Frias-Martinez, E. (2012). Characterizing urban landscapes using geolocated tweets. In *Proc. of PASSAT-SocialCom'12*, pages 239–248, Amsterdam, The Netherlands. IEEE.

Long, X., Jin, L., and Joshi, J. (2012). Exploring trajectory-driven local geographic topics in foursquare. In *Proc. of the UbiComp'12*, pages 927–934, Pittsburgh, Pennsylvania. ACM.

Newman, M. E. (2002). Assortative mixing in networks. *Phy. rev. let.*, 89(20):208701.

Noulas, A., Scellato, S., Mascolo, C., and Pontil, M. (2011a). An Empirical Study of Geographic User Activity Patterns in Foursquare. In *Proc. of ICWSM'11*, Barcelona, Spain.

Noulas, A., Scellato, S., Mascolo, C., and Pontil, M. (2011b). Exploiting Semantic Annotations for Clustering Geographic Areas and Users in Location-based Social Networks. In *Proc. of ICWSM'11*, Barcelona, Spain.

Quercia, D., Capra, L., and Crowcroft, J. (2012). The social world of twitter: Topics, geography, and emotions. In *Proc. of the ICWSM'12*, Dublin, Ireland.

Sagl, G., Blaschke, T., Beinat, E., and Resch, B. (2012). Ubiquitous geo-sensing for context-aware analysis: Exploring relationships between environmental and human dynamics. *Sensors*, 12(7):9800–9822.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: real-time event detection by social sensors. In *Proc. of the WWW'10*, pages 851–860, New York, NY, USA. ACM.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., and Loureiro, A. A. F. (2013a). Challenges and opportunities on the large scale study of city dynamics using participatory sensing. In *Proc. of the IEEE ISCC'13*, Split, Croatia.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., and Loureiro, A. A. F. (2013b). Uma Fotografia do Instagram: Caracterização e Aplicação. In *Proc. of XXXII SBRC'13*, Brasília, DF.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., Salles, J., and Loureiro, A. A. F. (2012). Visualizing the invisible image of cities . In *Proc. of IEEE CPScom'12*, Besancon, France.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., Salles, J., and Loureiro, A. A. F. (2013c). A comparison of foursquare and instagram to the study of city dynamics and urban social behavior. In *Proc. ACM SIGKDD UrbComp'13*, Chicago, USA.

Silva, T. H., Vaz de Melo, P. O. S., Viana, A., Almeida, J. M., Salles, J., and Loureiro, A. A. F. (2013d). Traffic Condition is more than Colored Lines on a Map: Characterization of Waze Alerts. In *Proc. of the SocInfo'13*, Kyoto, Japan.

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. (2010). Sentiment in short strength detection informal text. *J. Am. Soc. Inf. Sci. Technol.*, 61(12):2544–2558.

Vasconcelos, M., Ricci, S., Almeida, J., Benevenuto, F., and Almeida, V. (2012). Caracterização e Influência do Uso de Tips e Dones no Foursquare. In *Proc. of XXX SBRC'12*, Ouro Preto, MG.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 9*

## *Redes de Sensores Sem Fio*

# Reduzindo o Erro de Algoritmos de Localização Baseados em Alcance Durante o Rastreamento de Alvos em Redes de Sensores Sem Fio

**Éfren L. Souza**[1], **Richard W. Pazzi**[3], **Eduardo F. Nakamura**[1,2]

[1] Universidade Federal do Amazonas – UFAM

[2]Fundação Centro de Análise, Pesquisa e Inovação Tecnológica – FUCAPI

[3]University of Ontario Institute of Technology – UOIT

efren@icomp.ufam.edu.br, richard.pazzi@uoit.ca, eduardo.nakamura@fucapi.br

***Abstract.*** *In target tracking of Wireless Sensor Networks (WSNs), the nodes cooperate to estimate the target position based on geographical informations previously provided by localization algorithms. However, the errors introduced by localization algorithms may affect the performance of target tracking. Range-based localization algorithms use the distance between nodes to estimate node locations. In this case, the errors are generated by noises in these distances estimated. In this paper, we demonstrate and evaluate an simultaneos localization and tracking approach that uses multiple distance estimations to filter the noises. In this process there is no extra communication cost because it uses the messages sent by tracking application. We use Recursive Position Estimation (RPE) or Directed Position Estimation (DPE) – two range-based localization algorithms – to pre-localize the nodes, then their erros are reduced during tracking. We also evaluate the performance of predictions methods (Kalman, Particle, and Alpha-Beta filters) during the localization refinement. The results show that localization errors can be reduced more than 70%, depending of scenario.*

***Resumo.*** *Em rastreamento de alvos em Redes de Sensores Sem Fio (RSSFs), os nós cooperam para estimar a posição do alvo com base em informações geográficas previamente fornecidas por algoritmos de localização. Entretanto, a localização desses algoritmos possuem erros que afetam o desempenho do rastreamento. Algoritmos de localização baseados em alcance estimam a distância entre os nós para localizá-los, nesse caso os erros são consequência de ruídos presentes nessas distâncias estimadas. Neste artigo, demonstramos e avaliamos uma abordagem de localização e rastreamento simultâneos em que os ruídos da distância estimada entre um par de nós são filtrados usando múltiplas estimativas de distância. Esse processo não possui custo de comunicação adicional, pois aproveita as mensagens do rastreamento. Usamos Recursive Position Estimation (RPE) ou Directed Position Estimation (DPE) – dois algoritmos de localização baseados em alcance – para pré-localizar os nós, então reduzimos os seus erros durante o rastreamento. Também avaliamos o desempenho das previsões dos filtros de Kalman, Partículas e Alfa-Beta enquanto o erro de localização é reduzido. Os resultados mostram que os erros de localização podem ser reduzidos em mais de 70%, dependendo do cenário.*

## 1. Introdução

Uma Rede de Sensores Sem Fio (RSSF) [Akyildiz et al., 2002] é um tipo especial de rede *ad-hoc* composta por dispositivos com recursos limitados, chamados nós sensores. Esses sensores são capazes de monitorar um ambiente, coletar dados, realizar processamento localmente e disseminar dados. O maior objetivo dessas redes é monitorar o ambiente para detectar e avaliar eventos de interesse.

Rastreamento de alvos é uma importante aplicação que vem sendo bastante estudada nos últimos anos [Sharma et al., 2011; Teng et al., 2012; Campos et al., 2012]. O propósito dessa aplicação é coletar dados sobre um ou mais alvos (objetos móveis de interesse) e então estimar suas posições no campo de sensores. Em algumas situações, prever a próxima posição do alvo também é interessante [Nakamura et al., 2007; Sharma et al., 2011]. A previsão pode ser usada pelo usuário da aplicação, por exemplo, para interceptar um animal que se dirige para uma área de risco, como estradas em uma reserva ecológica.

Aplicações de rastreamento usam informações geográficas para estimar a posição do alvo, dessa forma os nós precisam conhecer as suas localizações. O sistema de posicionamento global (GPS) [Goswami, 2013] pode ser usado para obter a localização dos nós, mas equipar todos os nós com um receptor de GPS representa um alto custo que torna sua utilização inviável em redes de sensores [Fang et al., 2007]. Além disso, as redes de sensores podem ser implantadas em grande escala e em locais de difícil acesso, impossibilitando que a localização dos nós seja pré-determinada. Esses problemas criaram a necessidade de se desenvolver algoritmos que utilizem a capacidade de cooperação dos nós para obter a suas localizações de forma distribuída. Esses algoritmos são chamados de algoritmos de localização [Boukerche et al., 2007].

Os algoritmos de localização geralmente usam um conjunto de nós que possuem a capacidade de auto-localização, chamados nós *beacons*, utilizando GPS ou configuração manual. Os outros nós da rede calculam as suas localizações com base nos dados de localização fornecidos pelos nós *beacons* [Oliveira et al., 2009; Yifeng & Lamont, 2011].

Os algoritmos de localização podem ser classificados como não baseados em alcance (*range-free*) [Niculescu & Nath, 2003; Fang et al., 2007] ou como baseados em alcance (*range-based*) [Albowicz et al., 2001; Oliveira et al., 2009], esses se diferem nos dados usados para calcular as posições dos nós. Os algoritmos não baseados em alcance assumem que os nós são distribuídos uniformemente pelo campo de sensores, de forma que a quantidade de saltos entre dois nós represente a distância entre eles. Já os algoritmos baseados em alcance assumem que os nós são equipados com algum dispositivo que possibilita medir a distância entre os nós. A potência de sinal recebida (RSSI) é a técnica mais comum para estimar a distância entre os nós, pois não precisa que o nó sensor seja equipado com hardware adicional [Oliveira et al., 2009].

A posição estimada pelos algoritmos de localização possuem erros que podem prejudicar o funcionamento do rastreamento [Souza et al., 2013; Huang et al., 2013]. Os erros dos algoritmos de localização baseados em alcance são gerados por ruídos presentes nas distâncias estimadas entre os nós [Oliveira et al., 2009; Yifeng & Lamont, 2011].

Neste artigo desenvolvemos uma extensão do trabalho de Souza et al. [2013]. O objetivo é reduzir o erro dos dos algoritmos de localização baseados em alcance durante o rastreamento. Para isso, demonstramos e avaliamos uma abordagem de localização e

rastreamento simultâneos em que os ruídos da distância estimada entre um par de nós são filtrados usando múltiplas estimativas de distância. Cada referência de localização em um dado nó possui um Filtro de Kalman (KF) [Nakamura et al., 2007] responsável por filtrar esse ruídos iterativamente. Com isso, a localização dos nós e, consequentemente, o rastreamento ficam mais precisos com o passar do tempo. O processo de filtragem e a aplicação de rastreamento são executados simultaneamente.

Antes do rastreamento iniciar, os nós se localizam usando com localização baseada em alcance, e.g., *Recursive Position Estimation* (RPE) [Albowicz et al., 2001] e *Directed Position Estimation* (DPE) [Oliveira et al., 2009]. Durante o rastreamento, os nós que detectam o alvo trocam mensagens das suas observações sobre o evento. Essa cooperação é necessária para calcular a posição do alvo. Em nossa abordagem, antes de calcular a posição do alvo, os nós aproveitam essas mensagens para filtrar as distâncias estimadas das suas referências e melhorar as suas localizações. A posição do alvo é calculada usando as novas posições dos nós como referências. Dessa forma, os erros de localização dos nós são reduzidos por onde o alvo passa. Enquanto o erro de localização é refinado, avaliamos o desempenho das previsões dos filtros de Kalman, Partículas e Alfa-Beta [Nakamura et al., 2007; Sharma et al., 2011].

O restante deste artigo está organizado como segue. A Seção 2 apresenta e discute os trabalhos relacionados. A Seção 3 define alguns conceitos usados neste artigo. A Seção 4 apresenta nossa proposta que reduz o erro de localização durante o rastreamento. A Seção 5 apresenta a metodologia experimental e as avaliações quantitativas. Por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

## 2. Trabalhos Relacionados

Niculescu & Nath [2003] apresentam a solução pioneira para localização não baseada em distância. Todos os nós da rede obtêm a quantidade de saltos até os nós *beacons*. Com isso, a distância média de um salto é estimada para calcular a localização dos nós. Esse algoritmo é melhorado por Chen et al. [2008] que usam um esquema de correção de erro diferencial para reduzir o erro acumulado durante os múltiplos saltos. Fang et al. [2007] apresenta um algoritmo de localização não baseado em distância que dispensa a utilização de nós *beacons*. Esse considera que os nós são implantados em grupos, sendo que os nós de um mesmo grupo são implantados ao redor de uma posição conhecida e com uma probabilidade de distribuição Gaussiana. Com isso, a localização dos nós é calculada usando estimação estatística, observando os grupos dos vizinhos de cada nó.

Albowicz et al. [2001] propõem a solução pioneira para localização baseada em alcance. Nesse algoritmo, os nós *beacons* divulgam suas localizações para ajudar outros nós a se localizarem. Os nós que se localizam se tornam referências para outros nós. Com base nessa solução, Oliveira et al. [2009] propõem um algoritmo que usa estruturas de *beacons* para fazer a recursão iniciar de um único ponto e seguir uma direção conhecida. Com isso, um nó pode estimar sua localização usando apenas duas referências.

Alguns trabalhos da literatura avaliam o impacto dos erros de localização nos algoritmos geográficos. Oliveira et al. [2009] avaliam o efeito desses erros no roteamento geográfico. Souza et al. [2009] e Campos et al. [2012] avaliam o desempenho do rastreamento na presença desses erros. Os erros de localização desses trabalhos são resultantes de algoritmos de localização baseados em alcance.

Alguns trabalhos reduzem o erro dos algoritmos de localização. Taylor et al. [2006] e Teng et al. [2012] reduzem o erro de localização enquanto o rastreamento é executado. Os nós são pré-localizados por um algoritmo não baseado em alcance, similar ao apresentado por Fang et al. [2007]. O erro é reduzido observando as inconsistências geométricas entre as localizações dos nós e do alvo. Souza et al. [2013] reduzem o erro de localização de algoritmos baseados em alcance. Para isso, esse realiza múltiplas estimativas de distância durante a localização para obter uma estimativa de distância mais precisa. Nesse caso, a aplicação deve definir o *trade-off* entre custo e precisão.

## 3. Considerações e Definições

Os nós sensores são distribuídos aleatoriamente pelo campo de sensores. Todos os nós possuem os mesmos raios de comunicação e sensoriamento. Os nós estão sincronizados para detectar o alvo nos mesmos intervalos de tempo.

Um algoritmo de localização baseado em alcance é executado para pré-localizar os nós. Nesses algoritmos os nós podem ser classificados como: (1) beacons – nós que conhecem precisamente e antecipadamente as suas localizações, pois são equipados com GPS ou configurados manualmente; (2) estabelecidos – nós que calculam as suas posições durante a execução do algoritmo de localização; (3) livres – nós que não conhecem a sua localização. Durante a localização, os nós estimam as distâncias entre eles e suas referências. Essas distâncias estimadas possuem ruídos que são responsáveis pelo erro de localização que se acumula a cada iteração do algoritmo de localização.

A rede de sensores é aplicada para rastrear um alvo que se move aleatoriamente pelo campo de sensores. Os nós periodicamente consultam seus sensores para verificar se podem detectar o alvo. O modelo de detecção binário é usado para detectar o alvo, i.e., um nó sempre detecta o alvo se ele está dentro do seu raio de sensoriamento.

Para reduzir o tráfego de comunicação, os nós que detectam o alvo formam um agrupamento. Consideramos que o raio de comunicação é duas vezes o raio de sensoriamento, com isso apenas um agrupamento é formado em cada amostragem e todos os membros do agrupamentos podem se comunicar com apenas um salto. Cada nó que detecta o alvo estima a distância entre ele e o alvo, depois disso envia uma mensagem contendo a sua localização e a distância estimada para o alvo. Essa distância estimada também está sujeita à ruídos que interferem no cálculo de posição do alvo. Com isso, cada nó passa a conhecer a localização de todos os membros do agrupamento e a que distância cada um deles está do alvo. Todos os nós calculam a posição do alvo com multilateração [Oliveira et al., 2009] e verificam qual nó está mais próximo do alvo.

Somente o nó mais próximo do alvo reporta a posição do alvo para o nó *sink*, que por sua vez faz previsões da posição do alvo usando os filtros de Kalman, Partículas e Alfa-Beta. Esses filtros são considerados as soluções canônicas de previsão em rastreamento de alvos. As previsões também são afetadas pelos erros de localização.

## 4. Abordagem Proposta

Neste trabalho, aproveitamos as mensagens enviadas durante o rastreamento para melhorar a localização dos nós e, consequentemente, melhorar o rastreamento. A localização dos nós é refinada ao longo do tempo, a medida que o alvo é rastreado, sem utilizar

comunicação adicional. Além disso, os nós que não puderam ser localizados durante a execução do algoritmo de localização podem ser localizados durante o rastreamento.

Nos algoritmos de localização baseados em distância, os ruídos presentes nas distâncias estimadas entre os nós são responsáveis pelos erros de localização. Dessa forma, neste trabalho, refinamos a localização dos nós filtrando esses ruídos. Souza et al. [2009] também filtra os ruídos das distâncias estimadas entre os nós para reduzir o erro de localização. Entretanto o custo de localização aumenta, pois precisa que cada nó envie múltiplas mensagens durante a execução do algoritmo de localização.

A Figura 1 mostra que cada nó sensor possui dois módulos: o módulo de localização e o módulo de rastreamento. O módulo de localização possui as rotinas que são executadas pelos nós durante a localização. Esse módulo armazena a localização do nó e os dados sobre as referências usadas para calcular essa localização. O módulo de rastreamento possui as rotinas necessárias para calcular a posição do alvo. O módulo de rastreamento precisa da localização do nó para calcular a posição do alvo, ele deve consultar o módulo de localização para obter essa informação

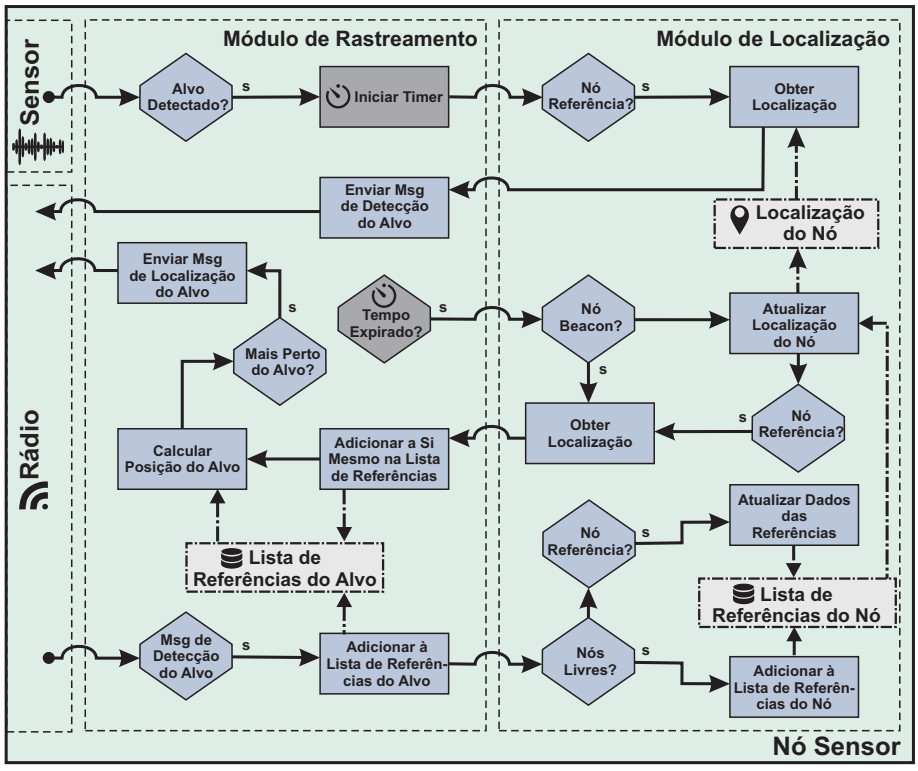



**Figura 1. Comunicação entre os módulos de rastreamento e localização.**

Cada nó que detecta o alvo inicia um *timer* e divulga uma mensagem do tipo *TargetDetection*, essa contém a sua localização e a distância entre ele e o alvo. Enquanto o *timer* não expirar, o nó fica esperando por mensagens de outros nós que também detectaram o alvo. Essa troca de informações é necessária para calcular a posição do alvo. Nossa abordagem tira proveito dessa troca de mensagens para filtrar os ruídos presentes nas distâncias estimadas entre os nós.

O nó que recebe uma mensagem *TargetDetection* adiciona os dados dessa mensa-

gem como referência de posição do alvo. Depois disso, o nó verifica no seu módulo de localização se o nó que originou a mensagem é uma referência de localização. Caso positivo, a distância estimada dessa referência é filtrada e atualizada na lista de referências.

Para filtrar os ruídos das distâncias estimadas das referências, cada referência de localização de um nó possui uma instância de KF configurada como

$$\begin{cases} x_{k+1} = d_{k+1} = \left[1\right] \times \left[d_k\right] + w_k \\ y_k = \left[1\right] \times \left[d_k\right] + v_k, \end{cases} \quad (1)$$

em que $x = d$ representa o estado (distância nesse caso) de um evento $k$; $y$ é o valor de medição; $w$ e $v$ são os ruídos do processo e da medição, respectivamente.

Sempre que um nó estabelecido recebe uma mensagem de uma referência de localização, ele usa a potência de sinal para estimar uma nova distância entre ele e a referência. Essa distância é passada para o KF correspondente, que por sua vez filtra os ruídos e retorna uma estimativa de distância mais precisa. O KF calcula iterativamente a média das distâncias estimadas, dessa forma o nó não precisa armazenar as informações de distância e refazer o cálculo sempre que uma mensagem nova for recebida.

Em algumas situações, um nó não consegue ser localizado pelo algoritmo de localização. Isso é inevitável quando o nó está isolado de forma que não obtém as referências necessárias. Colisão de pacotes durante a localização também pode fazer com que o nó não complete as referências para se localizar. Isso também pode acontecer devido ao erro das estimativas de distâncias que tornam inviável o calculo de posição. Nossa abordagem, além de reduzir o erro de localização, possibilita localizar os nós livres durante o rastreamento. As mensagens do tipo *TargetDetection* possuem todas as informações necessárias para a localização. Então um nó livre adiciona os nós que originam essas mensagens como referências e depois calculam a sua localização.

Quando as referências são atualizadas, as melhores referências são escolhidas de acordo com os critérios do algoritmo de localização. No caso do RPE as melhores referências são aquelas que possuem menor valor residual [Albowicz et al., 2001]. No caso do DPE são escolhidas as duas referências mais distantes entre si e mais próximas da origem da recursão [Oliveira et al., 2009].

A abordagem apresentada reduz os erros de localização gerados por ruídos presentes nas estimativas de distância. Para isso, tira proveito das mensagens enviadas durante o rastreamento, então não há custo de comunicação adicional. Nossa abordagem foi adotada em uma aplicação de rastreamento, mas pode ser adaptada para qualquer aplicação geográfica em que os nós precisam divulgar as suas localizações.

## 5. Avaliações

### 5.1. Metodologia

As avaliações são realizadas por meio de simulações com ns-2. A configuração padrão da rede é composta de 289 nós sensores monitorando uma região de $200 \times 200 \text{m}^2$. Cada simulação tem duração de $10^4$s, sendo que os primeiros 10s são reservados para o algoritmo de localização. Os raios de comunicação dos nós da rede são de 30m. Isso

possibilita que todos os nós se comuniquem através de múltiplos saltos e garante a cobertura necessária para localizar a maioria dos nós. Já o raio de sensoriamento é de 15m. Isso garante, na maioria dos casos, que existe pelos menos três nós detectando o alvo independente da sua posição no campo de sensores.

As localizações dos nós são estimadas com RPE ou DPE. O RPE usa 10% dos nós como *beacons* e o DPE usa quatro estruturas de *beacons* para localizar a maior parte dos nós. Para simular os erros de RSSI nas distâncias estimadas entre os nós, cada amostra de distância é perturbada por uma variável Gaussiana de média zero e desvio padrão de 8% da distância. As distâncias estimadas entre os nós e o alvo também são perturbadas por uma variável Gaussiana de média zero e desvio padrão de 6% da distância. A posição inicial do alvo é aleatória. A velocidade do alvo é de 1m/s e o seu modelo de mobilidade é a caminhada aleatória correlacionada (CRW) [Wu et al., 2000] com grau de correlação de 0,70, com isso o alvo percorre todo o campo de sensores.

O *sink* prevê a próxima posição do alvo usando KF, PF, ou ABF. No Filtro Alfa-Beta $\alpha = \beta = 0,99$. O filtro de Kalman tem seu sistema de equações lineares configurado como a Equação 2, em que $x$ representa o estado de um tempo discreto $k$ formado pela posição ($px$, $py$) e velocidade ($vx$, $vy$); $y$ é o valor de medição; $w$ e $v$ são os ruídos do processo e da medição, respectivamente. O filtro de Partículas foi fixado em 1000 partículas, valor obtido em avaliações anteriores que mostraram que mais de 1000 partículas não reduz significativamente o erro do rastreamento. O filtro de Partículas usado nos experimentos é representado pelo Algoritmo 1.

$$\begin{cases} x_{k+1} = \begin{bmatrix} px_{k+1} \\ py_{k+1} \\ vx_{k+1} \\ vy_{k+1} \end{bmatrix} = \\ \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} px_k \\ py_k \\ vx_k \\ vy_k \end{bmatrix} + w_k \\ y_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} px_k \\ py_k \\ vx_k \\ vy_k \end{bmatrix} + z_k, \end{cases} \quad (2)$$

**input** : medição $y_k$
**output**: previsão $x_{k+1}$

1 **for** $i = 1 : N$ **do** $x_0^i \leftarrow random();$ ;
2 $totalWeight \leftarrow 0;$
3 **for** $i = 1 : N$ **do**
4 $\quad x_k^i \leftarrow x_{k-1}^i + v_{k-1}^i + gaussian();$
5 $\quad v_k^i \leftarrow v_{k-1}^i + gaussian() * 0.05;$
6 $\quad w_k^i \leftarrow \frac{1}{distance(y_k, x_k^i)};$
7 $\quad totalWeight \leftarrow$
$\quad totalWeight + w_k^i;$
8 **end**
9 **for** $i = 1 : N$ **do** $w_k^i \leftarrow \frac{w_k^i}{totalWeight};$ ;
10 $slice_k^0 \leftarrow w_k^0;$
11 **for** $i = 2 : N$ **do**
$slice_k^i \leftarrow slice_k^{i-1} + w_k^i$ ;
12 **for** $i = 1 : N$ **do**
13 $\quad c \leftarrow random();$
14 $\quad j \leftarrow 0;$
15 $\quad$ **while** $j < N - 1$ *and* $slice_k^j < c$
$\quad$ **do** $j \leftarrow j + 1;$ ;
16 $\quad resampling_k^i \leftarrow particle_k^j;$
17 **end**
18 **for** $i = 1 : N$ **do**
$x_{k+1} \leftarrow x_{k+1} + x_k^i \times w_k^i;$ ;
19 $return\ x_{k+1};$

**Algorithm 1:** PF para rastreamento.

Para efeito de ilustração, o Algoritmo 1 considera apenas uma dimensão, sendo $x$ a posição, $v$ a velocidade e $w$ o peso de cada uma das $N$ partículas de um tempo discreto $k$; $y$ é o valor de medição passado como entrada. O algoritmo distribui as partículas

aleatoriamente (linha 1). A propagação das partículas e o cálculo de suas importâncias consideram a distância da posição de cada partícula para a posição da medição (linhas 3–8). O processo de normalização (linha 9) prepara o peso das partículas para o processo de reamostragem (linhas 12–17). Por fim, a previsão é calculada. (linha 18).

Cada mensagem enviada pelos nós tem tamanho 32*bytes*. Modelamos o consumo de energia de acordo com os diferentes modos de operação do rádio CC2420 [Wang et al., 2007]. A Tabela 1 mostra os parâmetros de energia usados nas simulações.

**Tabela 1. Parâmetros de energia.**

| Parâmetro | Valor |
|---|---|
| Energia inicial | 18720 joules |
| Sensoriamento | $2\times10^{-5}$ watts |
| Transmissão | $3132\times10^{-5}$ watts |
| Recepção | $3546\times10^{-5}$ watts |
| Escutando | $76{,}68\times10^{-5}$ watts |

Os resultados dos experimentos são obtidos da média de 35 execuções diferentes. As barras de erro representam um intervalo de confiança de 99%. Consideramos três métricas: erro de localização, erro de estimação, erro de previsão, mensagens e energia. O erro de localização é a distância Euclidiana entre a posição real do nó e a posição estimada pelo algoritmo de localização. O erro de estimação é a distância Euclidiana entre a posição real do alvo e a posição estimada pelo algoritmo de rastreamento. O erro de previsão é a distância Euclidiana entre a posição real do alvo e a previsão. A métrica de mensagens contabiliza todas as mensagens enviadas para localizar os nós e para rastrear o alvo. A métrica de energia é a energia consumida pelos nós até o final da simulação.

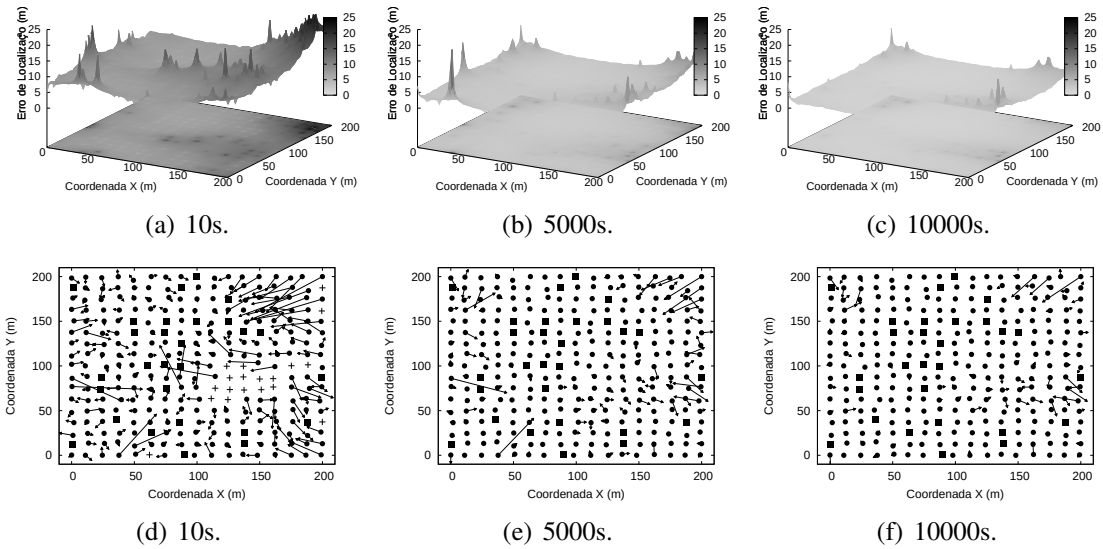
## 5.2. Resultados das Simulações

### 5.2.1. Avaliação Geral – Localização, Rastreamento e Tempo

A abordagem apresentada reduz o erro de localização filtrando as estimativas de distância usando as mensagens enviadas durante o rastreamento. As Figuras 2 e 3 mostram a distribuição dos erros de localização ao longo do tempo em intervalos de 5000s. Nessas figuras, os quadrados são os nós *beacons*, as cruzes são os nós livres, e os círculos representam a localização real dos nós estabelecidos, sendo que as setas apontam para a localização estimada.

Na Figura 2 os nós são pré-localizados com RPE, enquanto na Figura 3 os nós são pré-localizados com DPE. As figuras que marcam 10s de simulação mostram a localização inicial dos nós, i.e., a localização gerada apenas pelos algoritmos de localização. Nos demais tempos, a localização dos nós é refinada durante o rastreamento. O DPE apresenta erros de localização menores que o RPE, pois a forma como a posição do nó é calculada sofre menos interferência dos ruídos presentes nas estimativas de distância Oliveira et al. [2009]. Os nós mais distantes dos *beacons*, geralmente, possuem erros de localização maiores, devido ao erro acumulado nos nós estabelecidos.

Os nós que estão próximos do alvo tiram proveito da troca de mensagens, necessária para calcular a posição do alvo, para filtrar os ruídos das estimativas de distância




(a) 10s. (b) 5000s. (c) 10000s.

(d) 10s. (e) 5000s. (f) 10000s.

**Figura 2. Distribuição dos erros de localização do RPE ao longo do tempo.**

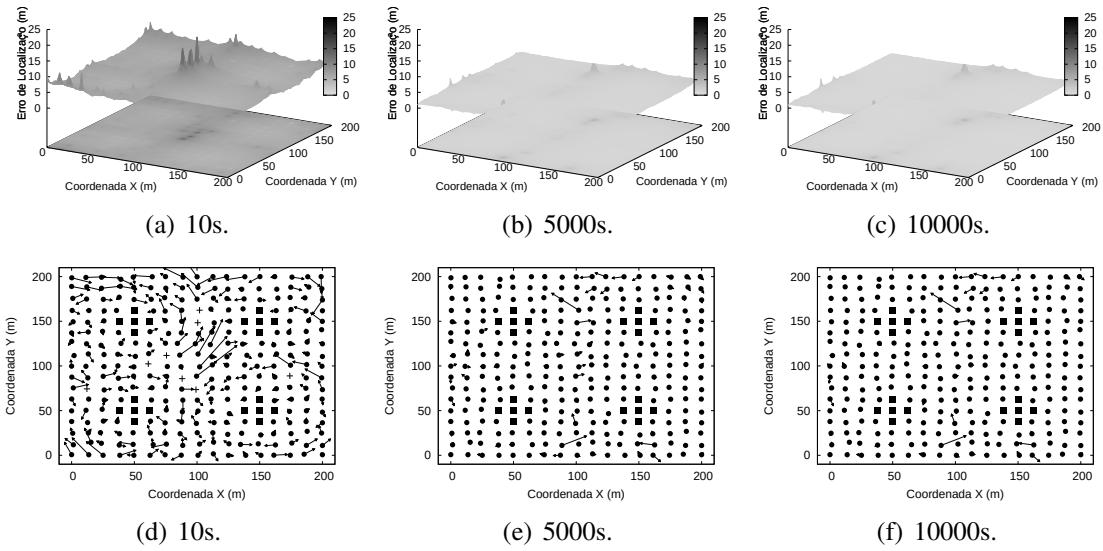
e melhorar as suas localizações estimadas. O erro de localização é reduzido gradualmente à medida que o tempo passa, como pode ser observado nas figuras e na Tabela 2. Quando a rede é pré-localizada com RPE, a solução converge com 7500s de rastreamento. Com DPE, a solução converge em 5000s, pois o erro de localização inicial é menor, como as referências são melhores, o trabalho de reduzir o erro de localização é facilitado.

**Tabela 2. Erro de localização e nós livres reduzidos durante o rastreamento.**

| Tempo de | RPE | | DPE | |
|---|---|---|---|---|
| Rastreamento | Erro Médio | Nós Livres | Erro Médio | Nós Livres |
| 10s | 9.60m | 19 | 6.45m | 8 |
| 2500s | 7.07m | 4 | 3.64m | 0 |
| 5000s | 3.60m | 0 | 1.52m | 0 |
| 7500s | 2.57m | 0 | 1.32m | 0 |
| 10000s | 2.35m | 0 | 1.23m | 0 |

Alguns nós não conseguem se localizar porque o erro das estimativas de distância não possibilitam o cálculo de posição. Além disso, os nós podem não conseguir as referências para isso, pois os dados são perdidos em colisões. Aplicando nossa abordagem, os nós livres podem se localizar mesmo depois do algoritmo de localização. Isso ocorre porque as mensagens de rastreamento melhoram as estimativas de distância entre os nós, possibilitando que a posição seja calculada. As mensagens também são usadas para adicionar referências aos nós que precisam. Na localização inicial do RPE, há um grupo de nós próximo do ponto (140, 70) que não foram localizados. Em 2500s de rastreamento a maior parte desses nós é localizada, e em 5000s de rastreamento todos eles são localizados. No caso do DPE, em 2500s de rastreamento, todos os nós livres são localizados.

Mesmo com 10000s de simulação, uma fração dos nós ainda apresenta erro de localização elevado. Esses nós são, geralmente, os mais afastados dos *beacons*, então os erros acumulados em suas referências durante os saltos de comunicação impossibilitam




(a) 10s. (b) 5000s. (c) 10000s.

(d) 10s. (e) 5000s. (f) 10000s.

**Figura 3. Distribuição dos erros de localização do DPE ao longo do tempo.**

um cálculo de posição mais preciso. Futuramente, pretendemos resolver esse problema aplicando seleção de referências [Xiao, 2011].

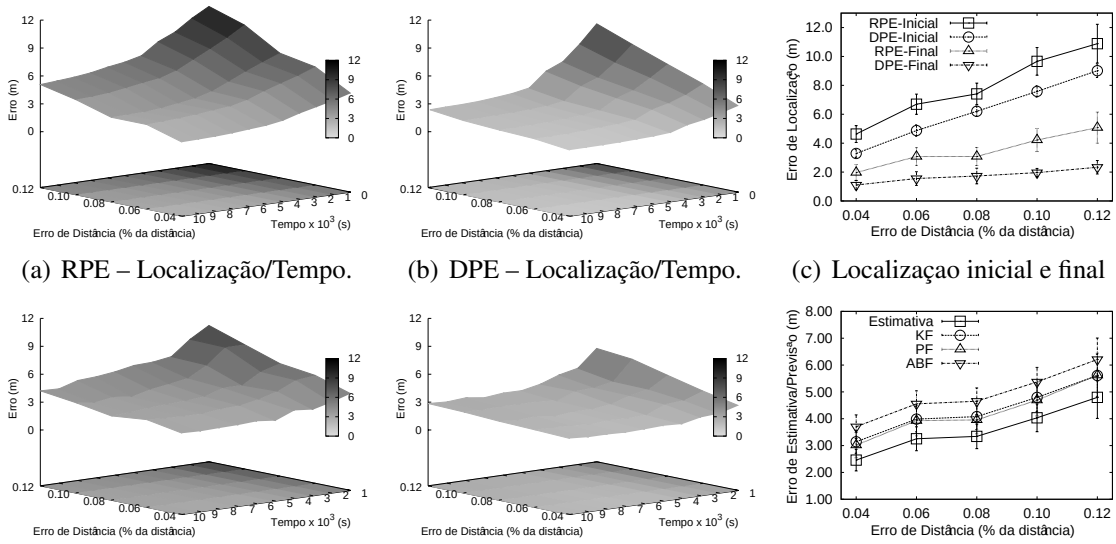
### 5.2.2. Imprecisão das Estimativas de Distância do Rádio

A distância estimada pelos nós sensores não é perfeita. Dependendo do ambiente monitorado, os erros associados são altos, afetando o desempenho da localização e do rastreamento. Em geral, esses erros dependem da distância e podem ser modelados como uma variável Gaussiana de média zero, em que o desvio padrão é um percentual da distância atual [Boukerche et al., 2007]. Desse modo, para avaliarmos diferentes situações, variamos o desvio padrão de 4% a 12% da distância para RPE e DPE.

As Figuras 4(a) e 4(b) mostram como a localização dos nós é afetada pela imprecisão das estimativas de distância e como ela é refinada durante o rastreamento. Enquanto a aplicação de rastreamento é executada, os nós filtram os ruídos presentes nas distâncias estimadas para suas referências e com isso melhoram a localização dos nós. Entretanto, chega um ponto em que a qualidade da localização converge e não apresenta mais melhoras significativas. O RPE possui erros de localização iniciais maiores que o DPE, por isso os ajustes nos erros de localização do RPE precisam de cerca de 7000s para convergir, enquanto o DPE converge em aproximadamente 5000s.

A Figura 4(c) compara os erros de localização antes e depois do refinamento. No pior caso, quando a imprecisão das estimativas de distância é de 12%, o RPE tem seus erros reduzidos em aproximadamente 54% e o DPE em 78%. O DPE é ajustado de forma mais eficiente porque direciona a recursão durante a localização, dessa forma o cálculo de posição dos nós é menos influenciado pelos erros das referências.

As Figuras 4(d) e 4(e) mostram como as previsões do KF são afetadas pela imprecisão das estimativas de distância e como elas são ajustadas ao longo do tempo. As previsões são afetadas pelo erro das estimativas de posição do alvo, que por sua vez são




(a) RPE – Localização/Tempo. (b) DPE – Localização/Tempo. (c) Localizaçao inicial e final

(d) RPE-KF – Previsão/Tempo. (e) DPE-KF – Previsão/Tempo. (f) RPE e previsões.

**Figura 4. Impacto das imprecisões das estimativas de distância na localização.**

afetadas pelos erros de localização. Dessa forma, o desempenho das previsões melhora com o passar do tempo porque a localização dos nós usados como referências para calcular a posição do alvo também melhora.

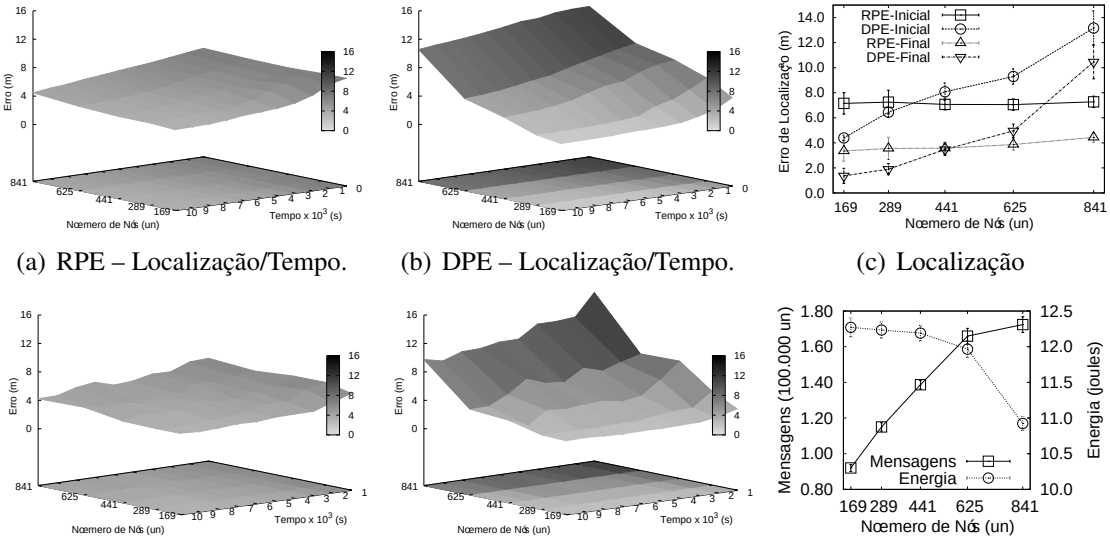
A Figura 4(f) mostra as estimativas de posição do alvo com RPE e também comparam o desempenho das previsões dos métodos KF, PF e ABF. A localização dos nós são usadas como referências para estimar a posição do alvo, logo o erro das estimativas de posição do alvo crescem juntamente a imprecisão das estimativas de distância. Essas estimativas são usadas como medidas pelos filtros KF, PF e ABF para fazerem previsões, por isso a qualidade das previsões refletem a qualidade das estimativas de posição do alvo.

O ABF é uma versão simplificada do KF, ele tem desempenho inferior, porém é menos complexo computacionalmente. O KF é a solução ótima para problemas lineares e com ruídos Gaussianos, entretanto os ruídos introduzidos pelos erros de localização são não-Gaussianos, por isso o PF possui um desempenho melhor. O PF, devido à sua natureza não-linear e não-Gaussiana, reduz uma fração do ruído não-Gaussiano introduzido pelos algoritmos de localização.

### 5.2.3. Escalabilidade

Avaliamos como a escala da rede afeta a combinação localização-rastreamento e a redução dos erros ao longo do tempo. Essa análise é importante porque as redes de sensores sem fio preveem situações em que precisam ser implantadas em larga escala. Sendo assim, variamos a quantidade de nós de 169 até 841, mantendo constante a densidade de 0,007 nós/m$^2$. Portanto, a área monitorada é redimensionada de acordo com a quantidade de nós sensores. Como o percentual de *beacons* usados no RPE é de 10%, o número de *beacons* aumenta de acordo com a quantidade de nós. O DPE mantém o uso de 4 estruturas de *beacons*, e passa a usar 5 estruturas quando a rede possui 441 nós ou mais.

As Figuras 5(a) e 5(b) mostram o impacto da escala da rede na localização. No RPE, a quantidade de nós *beacons* sempre é 10% da quantidade total de nós, por isso o erro de localização se mantém constante em qualquer escala usada. Já no DPE, a escala aumenta, mas a quantidade de estrutura de *beacons* utilizada não acompanha o crescimento da rede. Com isso, mais saltos de comunicação são necessários para alcançar toda a rede e mais erros são acumulados. Dessa forma, aumentar a escala da rede implica em aumentar o erro de localização. Por outro lado, a qualidade da localização do DPE seria mantida se a quantidade de estruturas de *beacons* utilizadas acompanhasse a escala da rede. Esses dois algoritmos de localização têm seus erros reduzidos ao longo do tempo, isso pode ser visto de forma mais clara na Figura 5(c).




(a) RPE – Localização/Tempo. (b) DPE – Localização/Tempo. (c) Localização

(d) RPE-KF – Previsão/Tempo. (e) DPE-KF – Previsão/Tempo. (f) Mensagens e energia

**Figura 5. Impacto da escala da rede na localização.**

As Figuras 5(d) e 5(e) mostram que o rastreamento reflete o mesmo comportamento da localização na mudança de escala, pois a localização dos nós são usadas como referências para localizar o alvo. No RPE o erro de rastreamento permanece o mesmo em qualquer escala, pois a quantidade de *beacons* cresce na mesma proporção que o número de nós. Já com o DPE, no cenário que avaliamos, a quantidade de estruturas de *beacons* não foi suficiente para para conservar a qualidade do rastreamento. Em ambos os algoritmos, o erro de localização é reduzido com o tempo, por isso o rastreamento é beneficiado e tem seus erros reduzidos com o passar do tempo.

A Figura 5(f) mostra o impacto da escala na quantidade de mensagens enviadas e energia consumida pelos nós. A quantidade de mensagens enviadas cresce com a escala da rede porque mais mensagens são necessárias para localizar os nós e para reportar o resultado do rastreamento para o nó *sink*. Por outro lado, a energia consumida pela rede diminui com a escala. Isso acontece porque a quantidade de nós que detecta o evento é a mesma em qualquer escala, mas o consumo médio por nó é menor em escalas maiores, uma vez que a carga de trabalho é distribuída entre mais nós.

## 6. Conclusões e Trabalhos Futuros

Neste artigo propomos e avaliamos uma abordagem de fusão de dados que reduz os erros de localização durante o rastreamento. Essa é focada em localização baseados em alcance, em que os erros são resultado da imprecisão das estimativas de distância. Dessa forma, filtramos os erros dessas estimativas para reduzir o erro de localização. Esse processo não possui custo de comunicação adicional, pois é realizado durante o rastreamento, aproveitando as mensagens enviadas pela aplicação. Essa abordagem também possibilita localizar nós que não puderam ser localizados durante a etapa de localização.

Em nossas avaliações utilizamos dois algoritmos de localização baseados em alcance: RPE e DPE. Nossa abordagem reduz com sucesso o erro desses dois algoritmos de localização. Dependendo do cenário, os erros podem ser reduzidos em mais de 70%. Entretanto, se a localização precisa de muitos saltos de comunicação para ser completada, os erros acumulados nas referências geram erros que são difíceis de serem removidos.

Também comparamos três métodos de previsão em rastreamento de alvos: KF, ABF e PF. Na presença de erros de localização, o PF apresenta um desempenho superior quando comparado com as outras técnicas. Diferente do KF e do ABF, o PF pode ser aplicado para filtrar ruídos não-Gaussianos, por isso é menos influenciado pelo cálculo de posição do alvo estimado com referências que possuem erros de localização.

Futuramente pretendemos avaliar as inconsistências geométricas nas posições nós e nas distâncias estimadas entre eles para reduzir o erro de localização acumulado.

## Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422.

Albowicz, J., Chen, A., and Zhang, L. (2001). Recursive position estimation in sensor networks. In *Proceedings of the 9th International Conference on Network Protocols (ICNP'01)*, pages 35–41, Riverside, USA.

Boukerche, A., de Oliveira, H. A. B. F., Nakamura, E. F., and Loureiro, A. A. F. (2007). Localization systems for wireless sensor networks. *IEEE Wireless Communications*, 14(6):6–12.

Campos, A. N., Souza, E. L., Nakamura, F. G., Nakamura, E. F., and Rodrigues, J. J. P. C. (2012). On the impact of localization and density control algorithms in target tracking applications for wireless sensor networks. *Sensors*, 12(6):6930–6952.

Chen, H., Sezaki, K., Deng, P., and So, H. C. (2008). An improved dv-hop localization algorithm with reduced node location error for wireless sensor networks. *Trans. on Fundamentals of Electronics, Comm. and Computer Sciences*, E91-A(8):2232–2236.

Fang, L., Du, W., and Ning, P. (2007). A beacon-less location discovery scheme for wireless sensor networks. In *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, volume 30, pages 33–55. Springer.

Goswami, S. (2013). Global positioning system. In *Indoor Location Technologies*, pages 51–63. Springer.

Huang, B., Yu, C., and Anderson, B. D. O. (2013). Understanding error propagation in

multihop sensor network localization. *IEEE Transactions on Industrial Electronics*, 60:5811–5819.

Nakamura, E. F., Loureiro, A. A. F., and Orgambide, A. C. F. (2007). Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys*, 39(3):55. Article 9.

Niculescu, D. and Nath, B. (2003). Dv based positioning in ad hoc networks. *Telecommunication Systems*, 22(1-4):267–280.

Oliveira, H. A. B. F., Boukerche, A., Nakamura, E. F., and Loureiro, A. A. F. (2009). An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Transactions Computing*, 58(5):677–691.

Sharma, S., Deshpande, S., and Sivalingam, K. (2011). On guided navigation in target tracking sensor networks using alpha-beta filters. In *Proceedings of the 31st Conference on Distributed Computing Systems Workshops (ICDCSW'11)*, pages 294 –303, Minneapolis, Minnesota, USA.

Souza, E. L., Nakamura, E. F., de Oliveira, H. A. B. F., and Figueiredo, C. M. S. (2013). Reducing the impact of location errors for target tracking in wireless sensor networks. *Journal of the Brazilian Computer Society*, 19(1):89–104.

Souza, E. L., Nakamura, E. F., and Oliveira, H. A. B. F. (2009). On the performance of target tracking algorithms using actual localization systems for wireless sensor networks. In *Proceedings of the 12th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'09)*, pages 418–423, Tenerife, Canary Islands, Spain.

Taylor, C., Rahimi, A., Shrobe, H., Bachrach, J., and Grue, A. (2006). Simultaneous localization, calibration, and tracking in an ad hoc sensor network. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN'06)*, pages 27–33, Nashville, Tennessee, USA.

Teng, J., Snoussi, H., Richard, C., and Zhou, R. (2012). Distributed variational filtering for simultaneous sensor localization and target tracking in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 61(5):2305–2318.

Wang, H., Zhang, X., Naït-Abdesselam, F., and Khokhar, A. (2007). DPS-MAC: an asynchronous MAC protocol for wireless sensor networks. In *Proceedings of the 14th international conference on High performance computing (HiPC'07)*, pages 393–404, Goa, India.

Wu, H., Li, B.-L., Springer, T. A., and Neill, W. H. (2000). Modelling animal movement as a persistent random walk in two dimensions: expected magnitude of net displacement. *Ecological Modelling*, 132(2):115–124.

Xiao, Q. (2011). *Range-Free and Range-Based Localization of Wireless Sensor Networks*. PhD thesis, Hong Kong Polytechnic University.

Yifeng, Z. and Lamont, L. (2011). A map registration localization approach based on mobile beacons for wireless sensor networks. In *Proceedings of the Global Telecommunications Conference (GLOBECOM'11)*, pages 1–5, Houston, Texas, USA.

# Um Algoritmo de Rastreamento para Interceptação de Alvo em Redes de Sensores Estruturadas em Faces

**Éfren L. Souza**[1], **Richard W. Pazzi**[3], **Eduardo F. Nakamura**[1,2]

[1] Universidade Federal do Amazonas – UFAM
Instituto de Computação – IComp

[2]Fundação Centro de Análise, Pesquisa e Inovação Tecnológica – FUCAPI

[3]University of Ontario Institute of Technology – UOIT

efren@icomp.ufam.edu.br, richard.pazzi@uoit.ca, eduardo.nakamura@fucapi.br

***Abstract.*** *Target tracking is a popular and well studied application of Wireless Sensor Networks (WSN). An interesting addition to this application is the presence of a tracker that aims to intercept the target. The WSN provides the position of the target to the tracker, so that it can move towards the target. The objective is to reach the target in the shortest possible time, using the fewest number of nodes dedicated for the task to save energy. In this paper, we propose and evaluate a Tracking Algorithm for Target Interception (TATI) in face-structured sensor Networks to reduce energy consumption and to shorten the route of the tracker. In order to facilitate node cooperation, the network is structured in faces, which are obtained by a planarization technique. The proposed algorithm reduces the number of active nodes based on displacement of the target, activating only the faces that are feasible to be achieved by the target until the next sampling. Furthermore, it uses the nominal communication range of the nodes to shorten the route of the tracker. Simulation results show that the proposed algorithm can save up to 15% in energy consumption, and it can keep the tracker around 10 meters closer to the target compared to the related scheme.*

***Resumo.*** *Rastreamento de alvos é uma popular e bem estudada aplicação de Redes de Sensores Sem Fio (RSSFs). Um interessante adicional para esse problema é a presença de um objeto guiado que visa interceptar o alvo. A RSSF deve fornecer a posição do alvo para o objeto guiado que se dirige para essa posição. O objetivo é alcançar o alvo no menor tempo possível e reduzir a quantidade de nós ativos para economizar energia. Neste artigo, propomos e avaliamos um algoritmo de rastreamento para interceptação de alvo (TATI) em redes de sensores estruturadas em faces que reduz o consumo de energia e a rota do objeto guiado. Para facilitar a cooperação entre os nós, a rede é estruturada em faces, que são obtidas por técnicas de planarização. O algoritmo proposto reduz o número de nós ativos baseado no deslocamento do alvo, ativando somente as faces que o alvo pode alcançar na próxima amostragem. Além disso, esse usa o raio de comunicação nominal dos nós para reduzir a rota do objeto guiado. Os resultados das simulações mostram que o algoritmos proposto pode reduzir o consumo de energia em 15% e manter o objeto guiado 10 metros mais próximo do alvo se comparado com a abordagem relacionada.*

## 1. Introdução

Uma Rede de Sensores Sem Fio (RSSF) [Akyildiz et al., 2002] é um tipo especial de rede *ad-hoc* composta por dispositivos com recursos limitados, chamados nós sensores. Esses sensores são capazes de monitorar um ambiente, coletar dados, realizar processamento localmente e disseminar dados. O maior objetivo dessas redes é monitorar o ambiente para detectar e avaliar eventos de interesse. As redes de sensores podem ser usadas para monitorar um ambiente para verificar a ocorrência de eventos de interesse, e.g. presença do alvo em aplicações de rastreamento [Nakamura et al., 2007]. Eficiência energética é uma das mais importantes questões em RSSF, uma vez que ela determina o tempo de vida da rede.

Rastreamento de alvos é uma importante aplicação que vem sendo bastante estudada nos últimos anos [Sharma et al., 2011; Teng et al., 2012; Campos et al., 2012]. Aplicações de rastreamento tradicionais visam coletar dados sobre um ou mais alvos para estimar as suas posições no campo de sensores, e então reportar o resultado para um nó *sink* [Wang et al., 2010; Xu et al., 2004].

Por outro lado, o problema de rastreamento pode ser formulado para incluir a presença de um objeto guiado [Tsai et al., 2007; Bhuiyan et al., 2010; Hsu et al., 2012]. O objeto guiado deve seguir o alvo, mas ele não pode detectá-lo, então a rede de sensores ajuda o objeto guidado a se aproximar do alvo. O método trivial para tratar esse problema é baseado em *floodings* sucessivos. O objeto guiado inicia um *flooding* na rede solicitando a posição do alvo. Ele recebe a posição do alvo e se dirige até ela. Chegando ao destino, ele faz um novo *flooding* para saber seu próximo destino. Esse processo tem um custo muito alto, pois toda a rede é consultada a cada amostragem.

Métodos mais elaborados usam somente parte dos nós para rastrear o alvo. Esse grupo de nós muda de acordo com a trajetória do alvo, evitando que toda a rede seja consultada. Esses métodos geralmente organizam a rede em faces [Karp & K, 2000; Huang et al., 2005] para facilitar a escolha do grupo de nós que deve ser ativado a cada amostragem. A rede seleciona um conjunto de nós *beacons* que formam a rota que o objeto guiado deve seguir.

Neste trabalho propomos e avaliamos o TATI (*Tracking Algorithm for Target Interception*), trata-se de um algoritmo de rastreamento para redes de sensores com objeto guiado baseado no DOT (*Dynamic Object Tracking*). Esse algoritmo reduz a quantidade de faces ativas e o percurso do objeto guiado, dessa forma a rede economiza mais energia e o alvo é interceptado mais rápido. Para isso, a rede é organizada em faces, a posição atual do alvo e a velocidade máxima registrada durante o rastreamento são usados como parâmetros para definir as faces que devem ser ativadas, isso reduz o número de faces ativas e garante que há nós para detectar o alvo independente da direção que ele seguir. Além disso, o TATI calcula pontos de consulta com base no alcance de comunicação para reduzir o percurso do objeto guiado.

O restante deste artigo está organizado como segue. A Seção 2 apresenta e discute os trabalhos relacionados. A Seção 3 define alguns conceitos usados neste artigo. A Seção 4 descreve o algoritmo de rastreamento TATI. A Seção 5 apresenta a metodologia experimental e as avaliações quantitativas. Por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

## 2. Trabalhos Relacionados

O método trivial de rastrear o alvo em rede de sensores com objeto guiado é baseado em *floodings* sucessivos. O objeto guiado inicia um *flooding* na rede solicitando a posição do alvo. Ele recebe a posição do alvo e se dirige até ela. Chegando ao destino, ele faz um novo *flooding* para saber seu próximo destino. Esse processo tem um custo muito alto, pois toda a rede é consultada a cada amostragem [Tsai et al., 2007].

Métodos mais elaborados usam parte dos nós para rastrear o alvo, esse grupo muda de acordo com a trajetória do alvo, evitando que toda a rede seja consultada. Esses métodos geralmente organizam a rede em faces [Karp & K, 2000; Huang et al., 2005] para facilitar a escolha do grupo de nós que deve ser ativado. Tsai et al. [2007] propõem o DOT, esse algoritmo determina quem é o nó mais próximo do alvo, que por sua vez ativa todas as suas faces adjacentes, enquanto os nós das demais faces da rede permanecem desativados para economizar energia. Assim, o DOT garante que na maioria dos casos haverá nós ativos para rastrear o alvo independente da direção que ele siga.

Alguns métodos usam previsão para reduzir a quantidade de faces ativas e economizar mais energia. Bhuiyan et al. [2010] usam um método de previsão baseado em regressão para calcular a velocidade e direção do alvo. A previsão é usada para escolher a quantidade apropriada de sensores que estão próximos do alvo, se a previsão for precisa, ela seleciona um grupo de sensores que pode detectar o alvo sem perda de desempenho.

Hsu et al. [2012] usam o mesmo método de previsão baseado em regressão para prever a face em que o alvo estará na próxima amostragem. Durante o rastreamento, o nó sensor mais próximo do alvo ativa os outros nós da mesma face para rastrear o alvo. Se a previsão indica que o alvo está deixando o território da face atual, a face prevista é ativada. Previsão baseada em regressão também é usada por Ji et al. [2009], eles combinam um algoritmo de hexágono com as estruturas de faces para limitar a próxima previsão do alvo em $1/6$ da área da face e reduzir as faces ativas.

Em uma abordagem de previsão diferente, Sharma et al. [2011] usa o Filtro Alfa-Beta (ABF) em suas previsões para superar as previsões lineares básicas. O objetivo é minimizar o tempo/distância do objeto guiado. A informação de localização é obtida e processada pelo nó sink em tempo real. Entretanto, para isso, o sink transmite diretamente a localização atualizada para o objeto guiado usando um rádio de alta potência. Métodos de previsão ajudam a reduzir o consumo de energia, mas eles assumem que a mobilidade do alvo possui uma correlação que torna as previsões relevantes.

## 3. Visão Geral

Na formulação do problema de rastreamento tratado neste trabalho, a rede deve ajudar o objeto guiado a se aproximar do alvo. Os algoritmos para esse problema fornecem uma rota que tornem a interceptação rápida Tsai et al. [2007]; Hsu et al. [2012]. Além disso, esses devem reduzir a quantidade de nós que efetivamente rastreiam o alvo para economizar energia. O alvo é o objeto de interesse, ele se move aleatoriamente pelo campo de sensores e pode ser detectado pelos nós sensores. A posição do alvo é calculada com multilateração [Oliveira et al., 2009], então são necessários pelo menos três nós para localizar o alvo. Embora o objeto guiado não possa detectar o alvo, ele possui recursos de comunicação e mobilidade, assim ele pode consultar os sensores da rede sobre a posição do alvo e seguir em direção a ela.

Os nós sensores são sincronizados. Cada nó conhece seu raio de comunicação nominal e sua localização, que pode ser obtida por GPS ou por algum algoritmo de localização distribuído [Boukerche et al., 2007; Oliveira et al., 2009]. Todos os nós possuem os mesmos alcances de comunicação e sensoriamento. Os nós que não estão participando do rastreamento desligam seus rádios para economizar energia. Os nós podem operar em cinco estados: *sleep*, *awake*, *active*, *near*, e *beacon*. Esses estados definem o nível de consumo de energia dos nós e suas funções no rastreamento.

Enquanto um nó está no estado *sleep*, ele não pode detectar o alvo, nem se comunicar com outros nós. Em um período pré-definido o nó muda seu estado para *awake*. Se ele recebe uma mensagem nesse período, muda seu estado para *active*, caso contrário ele volta para o estado *sleep*. Enquanto o nó está em estado *active*, ele pode enviar e receber mensagens e detectar o alvo. O nó permanece em estado *active* até o tempo de atividade expirar. No estado *active*, o nó pode mudar seu estado para *near* ou *beacon*. Quando a posição do alvo é calculada, o nó mais próximo do alvo muda seu estado para *near*. Nesse estado, o nó é responsável por selecionar e ativar outros nós. O nó no estado *near*, muda seu estado para *beacon* para registrar a trajetória do alvo. A Tabela 1 mostra os elementos que atuam no rastreamento.

Reduzir o número de nós ativos e a quantidade de mensagens enviadas são os mais importantes fatores que contribuem na economia de energia. Entretanto, isso pode fazer com que o alvo seja perdido, caso ele se dirija pra uma área que não possui nós ativos.

A Figure 1 é um exemplo de rastreamento de alvos em redes de sensores estruturada em faces. Somente os nós próximos do alvo ficam ativados, enquanto o restante permanece em estado *sleep* para economizar energia. Os nós *beacons* formam um rastro para levar o objeto guiado até o alvo, eles representam a trajetória do alvo. A rede é estruturada em faces de forma distribuída, a construção dessa estrutura é dividida em dois passos: planarização e identificação de faces [Huang et al., 2005; Hsu et al., 2012].

No primeiro passo, a rede é modelada como um grafo, os nós eliminam as arestas

**Tabela 1. Elementos envolvidos no rastreamento.**

| Gráfico | Descrição |
|---|---|
|  | Alvo – Objeto móvel que é rastreado pela rede. |
|  | Objeto guiado – Elemento móvel que objetiva alcançar o alvo. Ele se comunica com a rede para obter informações sobre o alvo. |
|  | Nó em estado *sleep* – Nó em modo de economia de energia, não pode detectar o alvo nem receber mensagens. |
|  | Nó em estado *active* – Nó que efetivamente rastreia o alvo. Ele pode detectar o alvo e enviar e receber mensagens a qualquer momento. |
|  | Nó em estado *beacon* – Nós que representam a trajetória do alvo. O objeto guiado consulta os *beacons* para descobrir a direção que deve seguir. |
|  | Nó em estado *near* – É o nó mais próximo do alvo. Ele calcula a posição do alvo e seleciona os nós que devem ser ativados. |

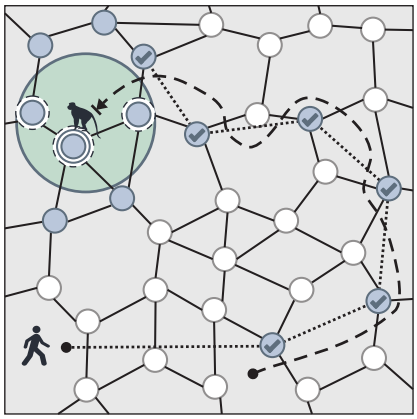


**Figura 1. Visão geral do rastreamento com objeto guiado.**

cruzadas de forma que a rede represente um grafo planar [Huang et al., 2005]. Com isso, o campo de sensores é dividido em regiões chamadas faces. Os vizinhos de face de um nó são os nós que compõem suas faces adjacentes, e.g., na Figura 2(c), o nó $n_1$ tem três faces adjacentes ($F_1$, $F_2$, e $F_3$) e doze vizinhos de faces ($n_2$, $n_3$, $n_4$, $n_5$, $n_6$, $n_7$, $n_8$, $n_9$, $n_{10}$, $n_{11}$, $n_{12}$, e $n_{13}$), somente três deles são vizinhos de face imediatos ($n_2$, $n_6$, e $n_{10}$). Os algoritmos usados para remover as arestas e formar os grafos planares são *Gabriel Graph* (GG) [Tsai et al., 2007] e *Relative Neighborhood Graph* (RNG) [Hsu et al., 2012], como mostram as Figuras 2(a) e 2(b)

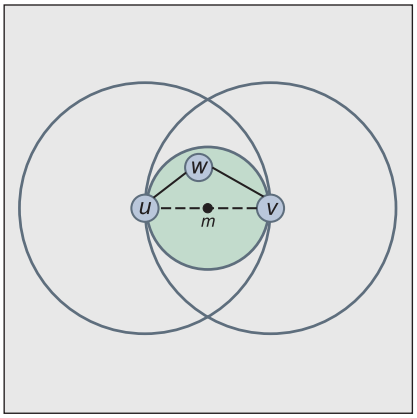
No segundo passo, cada nó deve descobrir suas faces adjacentes. Para isso, um nó de cada face envia uma mensagem que atravessa toda a face até voltar a origem. As posições dos nós pelos quais a mensagem passou são adicionadas à mensagem. Com isso, o nó que originou a mensagem coleta informação de toda face. Depois, outra mensagem atravessa a face novamente para informar os outros nós sobre a face descoberta. Cada nó tem um *buffer*-circular que armazena e ordena seus vizinhos de face imediatos para ajudar as mensagens a seguir o fluxo da face corretamente. No exemplo da Figura 2(c), se a origem da mensagem é $n_1$, o destino é $n_3$, uma vez que $n_3$ vem depois de $n_1$ no *buffer*-circular de $n_2$. Dessa maneira, a mensagem continua sendo encaminhada até chegar em $n_1$, que é o nó que descobre a face e informa os outros nós sobre isso.

## 4. Algoritmo de Rastreamento para Interceptação de Alvo

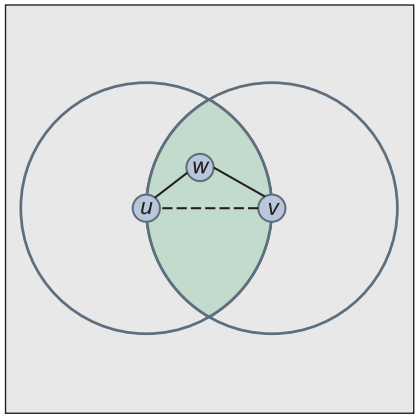
O algoritmo TATI é aplicado para auxiliar um objeto guiado a se aproximar do alvo em uma rede de sensores estruturada em faces. Esse algoritmo economiza energia reduzindo a quantidade de faces ativas e também reduz a rota do objeto guiado usando o raio de comunicação nominal dos nós. A seguir descrevemos o TATI em detalhes.

### 4.1. Configuração

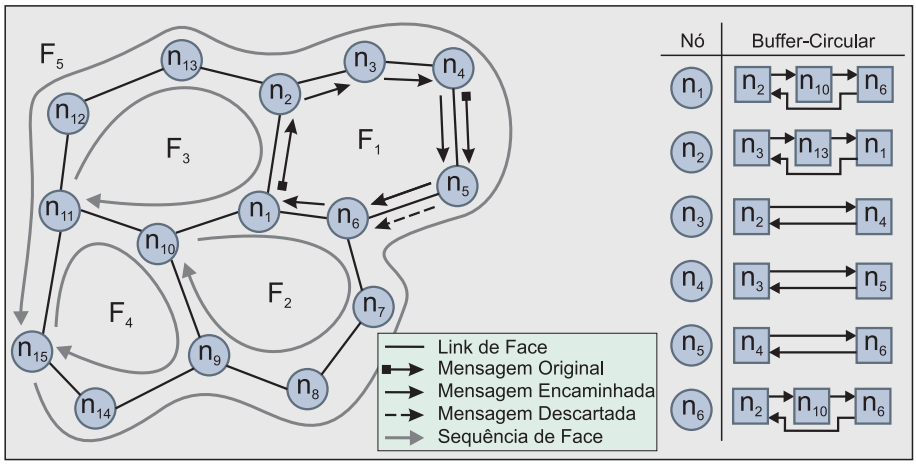
O TATI reserva um tempo inicial para configurar a rede antes do rastreamento começar. Durante a configuração, a rede de sensores é organizada em faces de forma distribuída. Cada nó envia uma mensagem com a sua posição para que todos os nós conheçam seus vizinhos e façam a planarização com GG ou RNG. Depois cada nó inicia o processo de identificação de faces para descobrir suas faces adjacentes. Depois disso, a rede está pronta para começar o rastreamento.




(a) GG – A aresta $(u, v)$ é removida porque a área sombreada possui um outro vértice $w$. Essa área tem centro $m$ e diâmetro $dist(u, v)$.




(b) RNG – A aresta $(u, v)$ é removida porque $w$ está na área sombreada. Essa área é a interseção dos círculos ao redor de $u$ e $v$, cujos raios são $dist(u, v)$.



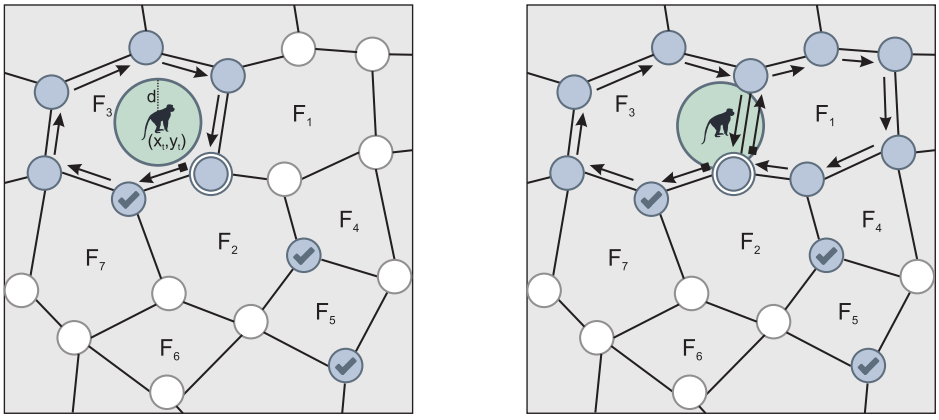

(c) Processo de identificação das faces. As mensagens seguem o sentido horário auxiliadas por um buffer-circular armazenado em cada nó.

**Figura 2. Estruturação da rede em faces de forma distribuída.**

## 4.2. Descoberta do Alvo

A posição do alvo é desconhecida antes do rastreamento. Então, quando os nós estão em estado *awake*, o objeto guiado inicia um *flooding* com a mensagem *TargetRequest* para solicitar a posição do alvo. Os nós que recebem essa mensagem tentam detectar o alvo e passam a mensagem adiante. Uma rota temporária é criada entre os nós e o objeto guiado durante esse processo.

Os nós que detectam o alvo enviam uma mensagem *TargetDetected*, contendo a distância estimada entre o nó e o alvo. Os nós calculam a posição do alvo e determinam qual deles está mais próximo do alvo. Esse muda seu estado para *near*, ativa todos os nós das suas faces adjacentes, e se torna *beacon* da face em que o alvo foi encontrado. Esse mesmo nó envia uma mensagem com a posição do alvo para o objeto guiado. Se alguma falha ocorre e o objeto guiado não receber essa mensagem, um novo *flooding* é realizado. Uma vez que o rastreamento é iniciado, ele é mantido pelos nós ativos.




(a) Somente a face $F_3$ é ativada. (b) As faces $F_1$ e $F_3$ são ativadas.

**Figura 3. Ativação das faces que fazem interseção com o círculo.**

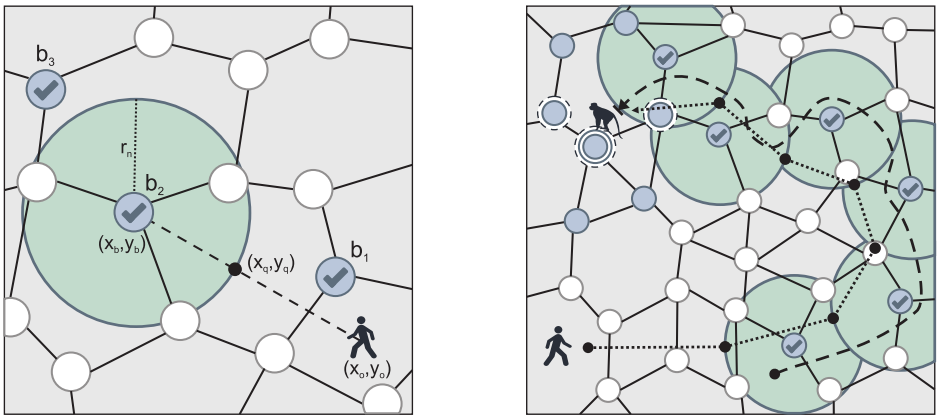
### 4.3. Rastreamento

Somente o conjunto de nós perto do alvo é necessário para rastreá-lo. TATI seleciona os nós que devem ser ativados baseado na estrutura de faces e no deslocamento do alvo. O nó em estado *near* envia uma mensagem *Wakeup* periodicamente para os nós que devem ser ativados. Essa mensagem segue o fluxo da face até alcançar todos os nós da mesma. A mensagem de *Wakeup* possui todas os dados necessários para o rastreamento.

Os nós em estado *sleep* que recebem uma mensagem *Wakeup* mudam seus estados para *active*. Nesse estado, eles não entram no estado *sleep* no período pré-definido, somente voltam pra ele quando o tempo de atividade expira. O tempo de atividade é reiniciado se um nó em estado *active* recebe uma mensagem *Wakeup*. Os nós em estado *active* que detectam o alvo trocam entre si mensagens *TargetDetected*. Eles usam essa mensagem para calcular a posição do alvo e definir qual deles está mais próximo do alvo agora. Dessa forma, a função de nó *near* passa de um nó para outro a medida que o alvo se move pela face e, consequentemente, o conjunto de nós ativos é ajustado.

O nó em estado *near* é responsável por originar as mensagens *Wakeup*, os outros nós apenas encaminham essas mensagens pela face. O nó *near* seleciona quais de suas faces adjacentes devem ser ativas com base na distância máxima $d$ percorrida pelo alvo entre duas amostragens. As faces que devem ser ativadas fazem interseção com o círculo de centro $(x_t, y_t)$ e raio $d$, em que $(x_t, y_t)$ é a posição atual do alvo, como mostra a Figura 3. Qualquer que seja a direção que o alvo siga, ele provavelmente estará dentro do círculo na próxima amostragem, pois raio do círculo é baseado no histórico de mobilidade do alvo que é registrado pela rede. Nesse caso, o consumo de energia é reduzido porque menos nós permanecem ativos e uma quantidade menor de mensagens de *Wakeup* são enviadas. Na Figura 3(a), o círculo representa a área em que o alvo pode estar na próxima amostragem. O círculo faz interseção somente com a face $F_3$, então somente os nós dessa face são ativados. No caso da Figura 3(b), esse círculo faz interseção com as faces $F_1$ e $F_3$, então essas duas faces são ativadas.

O nó em estado *near* muda para o estado *beacon* quando o alvo passa de uma face para outra. Existe um nó *beacon* para cada face por onde o alvo passou. O objetivo disso é registrar a trajetória do alvo.




(a) Calculando ponto consulta

(b) Objeto guiado se aproximando do alvo usando pontos de consulta.

**Figura 4. Pontos de consulta que reduzem a rota do objeto guiado.**

O alvo é perdido quando entra em uma área que não possui nós ativos para detectá-lo. Nessa situação, algoritmos de recuperação podem ser usados para reencontrar o alvo sem consultar toda a rede [Khare & Sivalingam, 2011; Hsu et al., 2012]. Esses algoritmos pode ser facilmente acrescentados ao TATI. Entretanto, nas avaliações deste artigo, o rastreamento é reiniciado sempre que o alvo é perdido.

### 4.4. Rota do Objeto Guiado

Os nós *beacons* registram a trajetória do alvo. Esses nós podem ser vistos como uma lista duplamente encadeada, pois cada um deles conhece o *beacon* anterior e posterior. Dessa forma, o primeiro *beacon* é aquele que não tem referência para o anterior e o último *beacon* é aquele que não tem referência para o próximo.

O objeto guiado se aproxima do primeiro *beacon* para consultar o caminho que deve seguir. A consulta é feita enviando uma mensagem *QueryNext*, essa é respondida com uma mensagem *QueryNextReply*. Se o nó consultado não é o último *beacon*, ele envia a posição do próximo ponto de consulta, caso contrário ele envia a posição do alvo. Então o objeto guiado se move em direção a posição informada. Quando chega ao destino, o objeto guiado faz outra consulta e o processo é repetido enquanto o rastreamento durar.

Depois que um nó *beacon* informa o ponto de consulta, ele retorna para o estado *sleep*. Mas antes ele envia uma mensagem *InfoFirst* para o próximo *beacon*. Essa mensagem informa ao próximo *beacon* que agora ele é o primeiro *beacon*. Da mesma forma que o DOT, TATI remove os laços e cria atalhos no caminho formado pelos *beacons*.

Os trabalhos da bibliografia usam a localização do *beacon* como ponto de consulta. TATI usa o raio de comunicação nominal $r_n$, a posição do objeto guiado $(x_o, y_o)$, e a localização do próximo *beacon* $(x_b, y_b)$ para calcular o ponto de consulta $(x_q, y_q)$ que reduz a rota do objeto guiado. Dessa forma, $(x_q, y_q)$ é a interseção entre a reta que conecta os pontos $(x_o, y_o)$ e $(x_b, y_b)$ com o círculo de centro $(x_b, y_b)$ e raio $r_n$, como mostra a Figura 4(a). Se o objeto guiado já está dentro do raio de comunicação do *beacon*, sua própria posição é o ponto de consulta. O valor de $r_n$ é menor que o raio de comunicação real, essa diferença é uma margem de erro para evitar possível problemas de comunicação.

# 5. Avaliações

## 5.1. Metodologia

As avaliações são realizadas por meio de simulações com ns-2. A configuração padrão da rede é composta de 81 nós sensores monitorando uma região de $200\times200\text{m}^2$. Cada simulação tem duração de 2000s, sendo que os primeiros 200s são reservados para a rede ser estruturada em faces com GG. Depois disso o objeto guiado inicia a fase de descoberta do alvo. O objeto guiado permanece seguindo o alvo até o final da simulação.

Os raios de alcance dos nós da rede são de 40m, isso possibilita que todos os nós se comuniquem através de múltiplos saltos e que as faces sejam criadas apropriadamente durante a configuração da rede. Já o raio de sensoriamento é de 30m para garantir que existe pelos menos três nós cobrindo qualquer ponto do campo de sensores.

O modelo de mobilidade do alvo é o *Correlated Random Walk* (CRW) [Wu et al., 2000] com 0,70 de correlação, com isso o alvo se move por todo o campo de sensores. A posição inicial do alvo é aleatória e sua velocidade é de 10m/s. A posição inicial do objeto guiado é o centro do campo de sensores e sua velocidade é de 10m/s.

Modelamos o consumo de energia de acordo com os diferentes modos de operação do rádio CC2420 [Suh et al., 2006; Wang et al., 2007]. Cada mensagem enviada pelos nós tem tamanho 32*bytes*. A Tabela 2 mostra os parâmetros de energia usados nas simulações.

**Tabela 2. Parâmetros de energia usados nas simulações do TATI.**

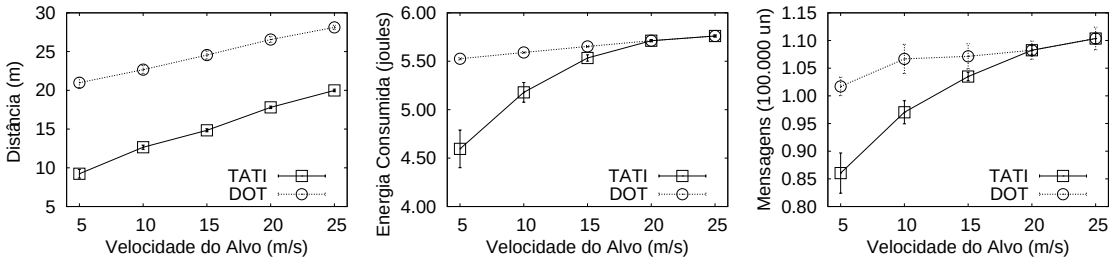
| Parâmetro | Valor |
|---|---|
| Energia inicial | 18720 joules |
| Sensoriamento | $2\times10^{-5}$ watts |
| Transmissão | $3132\times10^{-5}$ watts |
| Recepção | $3546\times10^{-5}$ watts |
| Escutando | $76{,}68\times10^{-5}$ watts |
| Dormindo | $3{,}6\times10^{-5}$ watts |
| Transição de estado | $5\times10^{-5}$ watts |

Os resultados dos experimentos são obtidos da média de 35 execuções diferentes. As barras de erro representam um intervalo de confiança de 99%. Consideramos três métricas: distância, mensagens e energia. A primeira métrica é a distância euclidiana entre o alvo e o objeto guiado no decorrer do rastreamento. A segunda métrica é a quantidade de mensagens enviadas durante o rastreamento, todas as mensagens são contabilizadas, inclusive as mensagens usadas para configurar a rede. A última métrica é a energia consumida pelos nós até o final da simulação.

## 5.2. Resultados das Simulações

### 5.2.1. Velocidade do Alvo

Nesta avaliação variamos a velocidade do alvo de 5m/s a 25m/s, a velocidade do objeto guiado sempre é igual a velocidade do alvo. A Figura 5 mostra os resultados.




(a) Distância. (b) Energia consumida. (c) Mensagens enviadas.

**Figura 5. Avaliações variando a velocidade do alvo.**

A Figura 5(a) mostra a distância entre o alvo e o objeto guiado quando a velocidade do alvo aumenta. A distância aumenta com a velocidade porque há um atraso para detectar o alvo e informar o resultado. Enquanto isso, o alvo continua a se mover e quanto mais rápido é o alvo mais ele se distancia. O TATI consegue reduzir essa distância porque os pontos de consulta minimizam a rota que objeto guiado usa para consultar os *beacons*, enquanto o DOT faz o objeto guiado se dirigir até a posição do *beacon*.

A Figura 5(b) mostra que o TATI consome menos energia que o DOT quando a velocidade do alvo é inferior a 20m/s. Isso ocorre porque o deslocamento do alvo entre duas amostragens é pequeno o suficiente para proporcionar que o TATI não ative todas as faces adjacentes de um nó. Nos casos em que a velocidade é superior a 15m/s, o consumo dos métodos avaliados é aproximado, pois o deslocamento do alvo faz com que o TATI ative todas as faces adjacentes ao nó *near*, da mesma forma que o DOT.

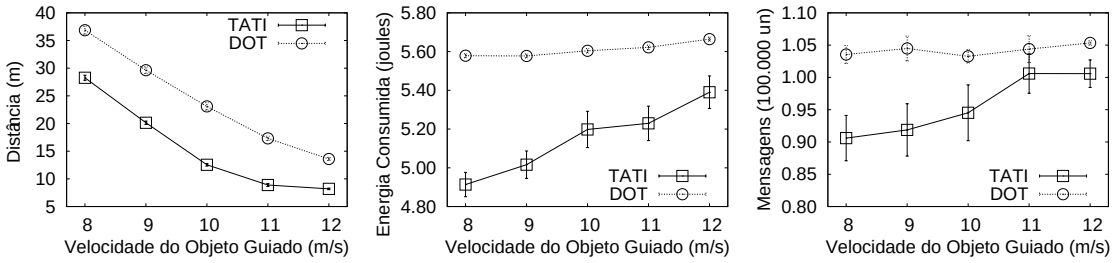
Uma face é ativada usando mensagens que percorrem todos os nós que a compõe. Como o TATI ativa menos faces, menos mensagens são enviadas durante o rastreamento, como mostra a Figura 5(c). Quando a velocidade é superior a 15m/s, a quantidade de mensagens enviadas aumenta nos dois métodos. Isso acontece porque o alvo é perdido mais vezes, sendo assim um *flooding* é feito na rede para recuperar o alvo.

### 5.2.2. Velocidade do Objeto Guiado

Nesta seção avaliação a diferença de velocidade entre o alvo e o objeto guiado. A velocidade do alvo é mantida em 10m/s, enquanto a velocidade do objeto guiado varia de 8m/s a 12m/s (Figura 6), para considerar os casos em que o objeto guiado é mais lento e mais rápido que o alvo.

Na Figura 6(a), o TATI mantém o objeto guiado mais próximo do alvo, pois os pontos de consulta minimizam a rota do objeto guiado. Quando o objeto guiado é mais lento que o alvo, ele tem dificuldade de se aproximar, mesmo que a rota seja minimizada, pois o alvo se move constantemente e se mantém afastado devido a velocidade superior.

Mesmo quando o objeto guiado é mais rápido que o alvo, ele ainda fica afastado do alvo por alguns metros (aproximadamente 10m com TATI e 15m com o DOT). Isso acontece porque o objeto guiado alcança o último *beacon*. Nesse caso ele se aproxima do alvo enquanto estiver dentro do raio de alcance do *beacon*. Depois disso deve esperar até que outro *beacon* seja criado, nesse tempo o alvo se afasta.




(a) Distância. (b) Energia consumida. (c) Mensagens enviadas.

**Figura 6. Avaliações variando a diferença de velocidade entre o objeto guiado e o alvo.**

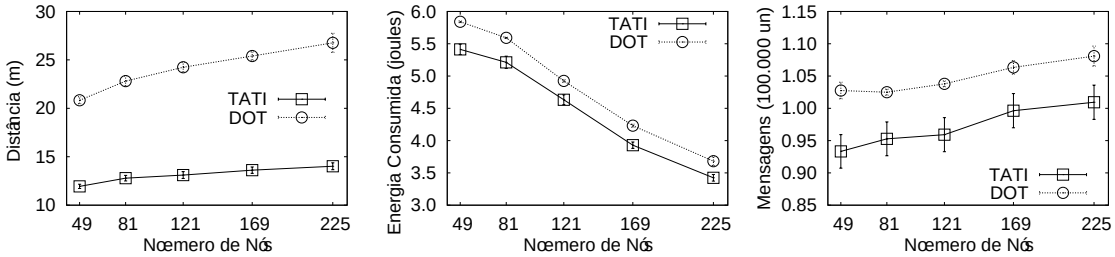
Na Figura 6(b), o TATI consome menos energia que o DOT porque usa menos faces ativas. Ambos os métodos consomem mais energia quando a velocidade do objeto guiado aumenta, principalmente o TATI. A razão disso é que o objeto guiado, quando está próximo do alvo, consulta frequentemente o último *beacon* para continuar a aproximação. Como o DOT tem mais dificuldade de fazer o objeto guiado alcançar o último *beacon*, não há muita diferença do consumo de energia quando a velocidade do objeto guiado é maior que a velocidade do alvo.

O TATI usa menos mensagens que o DOT porque envia menos mensagens para ativar as faces, como mostra a Figura 6(c). O TATI aumenta a quantidade de mensagens quando a velocidade do objeto guiado é superior a velocidade do alvo (11m/s e 12m/s), pois consulta frequentemente o último *beacon* para manter a aproximação.

### 5.2.3. Escalabilidade

A escalabilidade é avaliada variando o tamanho do campo de sensores de $150\times150\text{m}^2$ a $350\times350\text{m}^2$ com uma densidade constante de 0,002 nós/m$^2$. Dessa forma, variamos a quantidade de nós de 49 até 225 nós, como mostra a Figura 7.

Na Figura 7(a) a distância entre o alvo e o objeto guiado aumenta com a escala da rede. Quanto maior é o campo de sensores maior é a distância entre o alvo e o objeto guiado no início do rastreamento. Dessa forma, o objeto guiado leva mais tempo para alcançar o alvo, o que aumenta a distância média. O TATI mantém o objeto guiado mais próximo do alvo porque minimiza a rota usando os pontos de consulta.




(a) Distância. (b) Energia consumida. (c) Mensagens enviadas.

**Figura 7. Avaliação de escalabilidade.**

A Figura 7(b) mostra que a energia consumida pela rede diminui com o aumento da escala. Isso acontece porque a quantidade de nós aumenta com a escala, porém a quantidade de nós que efetivamente rastreia o alvo permanece a mesma. Em outras palavras, aproximadamente a mesma quantidade de energia é consumida durante o rastreamento, mas ela é distribuída entre uma quantidade maior de nós, logo o consumo por nó é menor.

A Figura 7(c) mostra que a quantidade de mensagens aumenta com a escala da rede. Isso acontece porque mais mensagens são usadas no *flooding* para encontrar o alvo no início do rastreamento ou para recuperá-lo caso ele seja perdido. Além disso, mais mensagens são necessárias para gerar as faces na fase de configuração da rede.

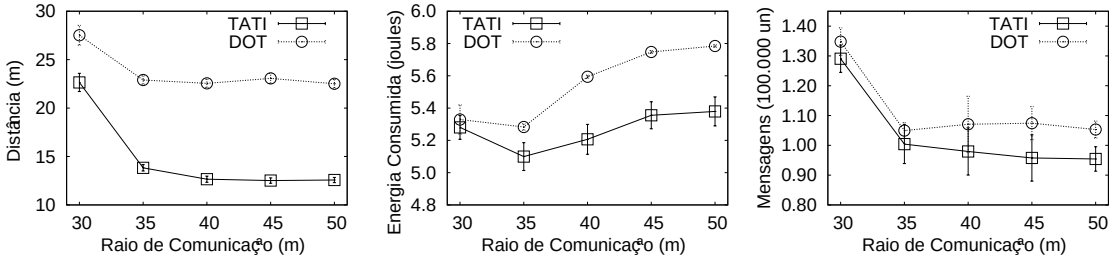
### 5.2.4. Raio de Comunicação

O raio de comunicação dos nós modifica a quantidade de vizinhos dos nós e consequentemente a formação das faces. Nesta avaliação variamos o raio de comunicação dos nós de 30m a 50m. Os resultados são apresentados na Figura 8.

A Figura 8(a) mostra que a distância entre o alvo e o objeto guiado é maior quando o raio de comunicação é de 30m, essa distância diminui e estabiliza nos demais casos. O TATI e o DOT coletam dados de seus vizinhos para calcular a posição do alvo. Por isso, o ideal é que o raio de comunicação seja duas vezes o de sensoriamento. Se o alcance é de 30m, muitas vezes os nós não possuem dados suficientes para calcular a posição do alvo, então ele é perdido. O tempo para recuperar o alvo é suficiente para que ele se afaste.

Na Figura 8(b) o consumo de energia aumenta com o raio de comunicação. Isso acontece porque mais nós irão receber uma mesma mensagem. Uma exceção ocorre quando o raio de comunicação é de 30m, pois supera o consumo de quando o raio de comunicação é de 35m. Isso ocorre porque o alvo é perdido com mais frequência, fazendo com que toda a rede seja consultada para recuperar o alvo. Por essa mesma razão, mais mensagens são enviadas quando o alcance é de 30m, como mostra a Figura 8(c).

Nos demais casos da Figura 8(c), a quantidade de mensagens utilizadas é aproximada, pois as mensagens percorrem a sequência das faces, então aproximadamente a mesma quantidade de mensagens é enviada independente do raio de comunicação. Entretanto, a variação é reduzida quando o raio de comunicação é grande (50m), pois as faces geradas ficam com tamanhos aproximados.




(a) Distância. (b) Energia consumida. (c) Mensagens enviadas.

**Figura 8. Avaliações aumentando o raio de comunicação dos nós.**

## 6. Conclusões e Trabalhos Futuros

Neste artigo propomos e avaliamos TATI, usando o DOT como *baseline*. O TATI é um algoritmo distribuído para rastreamento de alvo em redes de sensores, criando uma rota que é informado ao objeto guiado que objetiva interceptar o alvo.

Esse algoritmo reduz o consumo de energia por ativar menos faces, assim economiza em mensagens e menos nós permanecem em atividade. Além disso, reduz a rota percorrida pelo objeto guiado usando pontos de consulta. Esse pontos são a menor distância entre a posição do objeto guiado e a área de comunicação dos *beacons*.

O TATI registra o deslocamento máximo do alvo entre duas amostragens para escolher as faces que devem ser ativadas. Portanto, ele ativa menos faces que o DOT quando a velocidade do alvo impossibilita que ele possa se deslocar para múltiplas faces diferentes até a próxima amostragem. Entretanto o consumo de energia desses métodos é aproximado quando a velocidade do alvo é alta (20m/s ou mais).

No geral, o TATI mantém o objeto guiado mais próximo do alvo, pois usa a informação do raio de comunicação nominal dos nós para reduzir o percurso do objeto guiado. Ele envia o objeto guiado para o ponto mais próximo em que é possível consultar o *beacon* da vez. Além disso, evita que o objeto guiado se desloque desnecessariamente se ele já está na área de comunicação do *beacon*.

Futuramente pretendemos: (1) incluir um mecanismo de recuperação; (2) avaliar o impacto de erros de localização; (3) comparar o rastreamento em diferentes estrutura de faces (e.g., RNG e GG); e (4) incluir e avaliar métodos de previsões robustos (e.g., filtros de Kalman e Partículas).

## Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422.

Bhuiyan, M., jun Wang, G., Zhang, L., and Peng, Y. (2010). Prediction-based energy-efficient target tracking protocol in wireless sensor networks. *Journal of Central South University of Technology*, 17(2):340–348.

Boukerche, A., de Oliveira, H. A. B. F., Nakamura, E. F., and Loureiro, A. A. F. (2007). Localization systems for wireless sensor networks. *Wireless Comm.*, 14(6):6–12.

Campos, A. N., Souza, E. L., Nakamura, F. G., Nakamura, E. F., and Rodrigues, J. J. P. C. (2012). On the impact of localization and density control algorithms in target tracking applications for wireless sensor networks. *Sensors*, 12(6):6930–6952.

Hsu, J.-M., Chen, C.-C., and Li, C.-C. (2012). POOT: An efficient object tracking strategy based on short-term optimistic predictions for face-structured sensor networks. *Computers & Mathematics with Applications*, 63(2):391–406.

Huang, Q., Bhattacharya, S., Lu, C., and Roman, G.-C. (2005). FAR: Face-Aware Routing for mobicast in large-scale sensor networks. *ACM Transactions on Sensor Networks*, 1(2):240–271.

Ji, X., Zhang, Y.-Y., Hussain, S., Jin, D.-X., Lee, E.-M., and Park, M.-S. (2009). FOTP: Face-based Object Tracking Protocol in wireless sensor network. In *Proceedings of*

*the 4th International Conference on Computer Sciences and Convergence Information Technology (ICCIT'09)*, pages 128–133, Los Alamitos, USA.

Karp, B. and K, H. T. (2000). GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Conference on Mobile Computing and Networking (MobiCom'00)*, pages 243–254, Boston, Massachusetts, USA.

Khare, A. and Sivalingam, K. M. (2011). On recovery of lost targets in a cluster-based wireless sensor network. In *Proceedings of the Pervasive Computing and Communications Workshops (PERCOM'11)*, pages 208–213, Seattle, USA.

Nakamura, E. F., Loureiro, A. A. F., and Orgambide, A. C. F. (2007). Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys*, 39(3):55. Article 9.

Oliveira, H. A. B. F., Boukerche, A., Nakamura, E. F., and Loureiro, A. A. F. (2009). An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Transactions Computing*, 58(5):677–691.

Sharma, S., Deshpande, S., and Sivalingam, K. (2011). On guided navigation in target tracking sensor networks using alpha-beta filters. In *Proceedings of the 31st Conference on Distributed Computing Systems Workshops (ICDCSW'11)*, pages 294 –303, Minneapolis, Minnesota, USA.

Suh, C., Ko, Y.-B., Lee, C.-H., and Kim, H.-J. (2006). Numerical analysis of the idle listening problem in ieee 802.15.4 beacon-enable mode. In *Proceedings of the 1st Conference on Communications and Networking in China (ChinaCom'06)*, pages 1–5, Beijing, China.

Teng, J., Snoussi, H., Richard, C., and Zhou, R. (2012). Distributed variational filtering for simultaneous sensor localization and target tracking in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 61(5):2305–2318.

Tsai, H.-W., Chu, C.-P., and Chen, T.-S. (2007). Mobile object tracking in wireless sensor networks. *Computer Communications*, 30(8):1811–1825.

Wang, H., Zhang, X., Naït-Abdesselam, F., and Khokhar, A. (2007). DPS-MAC: an asynchronous MAC protocol for wireless sensor networks. In *Proceedings of the 14th international conference on High performance computing (HiPC'07)*, pages 393–404, Goa, India.

Wang, Z., Lou, W., Wang, Z., Ma, J., and Chen, H. (2010). A novel mobility management scheme for target tracking in cluster-based sensor networks. In *Proceedings of the 6th IEEE Conference on Distributed Computing in Sensor Systems (DCOSS'10)*, pages 172–186, Santa Barbara, California, USA.

Wu, H., Li, B.-L., Springer, T. A., and Neill, W. H. (2000). Modelling animal movement as a persistent random walk in two dimensions: expected magnitude of net displacement. *Ecological Modelling*, 132(2):115–124.

Xu, Y., Winter, J., and Lee, W.-C. (2004). Prediction-based strategies for energy saving in object tracking sensor networks. In *Proceedings of the 5th International Conference on Mobile Data Management (MDM'04)*, pages 346–357, Berkeley, California, USA.

# Minimização de Retransmissores em Redes de Sensores em fio com limite de alcance de rádio transmissão.

**Diego Giacomelli Cardoso**[1], **Renato Elias Nunes de Moraes**[1]

[1]Departamento de Computação e Eletrônica
Universidade Federal do Espírito Santo
Rodovia BR 101 Norte, Km 60 – Bairro Litorâneo – 29932-540
São Mateus – ES – Brazil

dg.cardoso@yahoo.com.br,renato.moraes@ufes.br

***Abstract.*** *Wireless sensor networks (WSNs) have been applied to many applications since emerging. The design of a WSN should focus on their performance and lifetime due to the high operational costs caused by the lack of infrastructure and its typical use. In this paper the reduction of the operating cost of WSNs is proposed by limiting the transmission range of the radio transceivers of the network nodes and the insertion of relays nodes. The problem to be solved consists in, given an upper limit of transmission range and the sensor nodes positioned in the plane, computing the minimum number of relays, its position and transmission range and the transmission range of the sensor nodes such that the resulting network is connected and no node exceeds the transmission range limit. Since this problem is NP-complete, this work presents the implementation of a constructive heuristic, a local search algorithm and a GRASP metaheuristic to solve it, showing the effectiveness of the algorithms through computational experiments.*

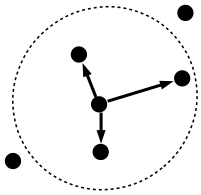
***Resumo.*** *As Redes de Sensores Sem Fio (RSSFs) têm sido usadas em muitas aplicações desde o seu surgimento. O projeto de RSSFs deve privilegiar o desempenho e a vida útil dessas redes devido ao alto custo operacional gerado pela falta de infra-estrutura e a forma como são utilizadas. Neste trabalho é proposta a redução do custo operacional das RSSFs através da limitação do alcance de transmissão dos rádios transceptores dos nós da rede e da inserção de nós retransmissores. O problema a ser solucionado consiste em, dado um limite superior de alcance de transmissão e os nós sensores posicionados no plano, computar o número mínimo de retransmissores, a posição e o alcance de transmissão de cada um deles e o alcance de transmissão dos nós sensores tais que a rede resultante seja conexa e nenhum nó ultrapasse o limite de alcance de transmissão. Por ser um problema NP-completo, este trabalho apresenta a implementação de uma heurística construtiva, um algoritmo de busca local e uma metaheurística GRASP para resolvê-lo, mostrando a eficácia dos algoritmos através de experimentos computacionais.*

## 1. Introdução

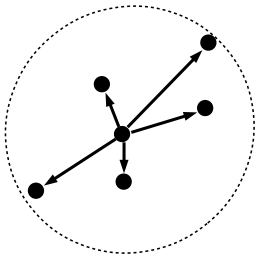
Redes de sensores sem fio (RSSFs) são sistemas que contêm nós sensores conectados por enlaces sem fio [Akyildiz et al. 2002]. Nós sensores, ou simplesmente nós, são dispositivos constituídos de sensores capazes de detectar mudanças em um ambiente, uma

pequena unidade de processamento capaz de gravar e processar as informações captadas pelos sensores e um rádio transceptor utilizado para disseminá-las. Redes de sensores são usadas de maneira eficiente em ambientes sem infra-estrutura pré-estabelecida, pois cada nó sensor é capaz de criar e manter um enlace com outros nós utilizando seu próprio rádio transceptor. As redes de sensores são amplamente utilizadas [Wang and liu 2011] em monitoramento de vulcões, oceanos, em engenharia civil, em aplicações militares, entre outros.

Como, geralmente, não há infra-estrutura pré-estabelecida, os enlaces de uma rede sem fio são determinados pelo alcance de transmissão dos rádios transceptores dos seus nós. Quanto maior o alcance de transmissão, maior o número de enlaces estabelecidos e maior a conexidade da rede (ver Figura 1). RSSFs são utilizadas para produzir efeitos em macroescala a partir de microdispositivos através de atividades coordenadas entre vários sensores, deste modo, conexidade é uma questão importante na construção de uma RSSF [Cheng et al. 2008]. Entretanto, grande alcance de transmissão implica em potência alta que, por sua vez, implica em aumento no consumo de energia e no aumento da interferência. Enquanto que a energia é um fator operacional limitante em redes alimentadas por baterias como as RSSFs [Min et al. 2002], a interferência provoca colisão e retransmissão de mensagens, influenciando negativamente tanto na vida útil da rede como em sua vazão [Cheng et al. 2008].



(a) Menor alcance de transmissão implica em menor número de enlaces estabelecidos.



(b) Maior alcance de transmissão implica em maior número de enlaces estabelecidos.

**Figura 1. O alcance de transmissão do rádio transceptor de um nó determina os enlaces de uma rede sem fio.**

Uma tentativa de reduzir o consumo de energia dos nós consiste na limitação do alcance de transmissão dos seus rádios transceptores que, como consequência, pode gerar redes desconexas e sem utilidade [Cheng et al. 2008]. Para garantir uma rede conexa e o respeito ao limite de transmissão estabelecido, sempre que um enlace exigir alcance maior que o limite, retransmissores terão que ser inseridos na rede. Entretanto, a adição de nós retransmissores de alta capacidade aumenta o custo da rede.

Dessa forma, o projeto de redes de sensores sem fio deve considerar o conflito entre desempenho, vida útil e custo financeiro. Para obter maior desempenho, os nós de uma RSSFs devem aumentar o seu alcance de transmissão (potência dos transceptores) introduzindo mais enlaces e alcançando assim maior vazão. Ao mesmo tempo, grande alcance de transmissão aumenta a interferência, aumenta o consumo de energia e reduz a vida útil da rede. Ao priorizar vida útil, deve-se diminuir o alcance dos nós reduzindo,

consequentemente, a interferência, o número de enlaces e o desempenho. Além disso, menor número de enlaces pode provocar a perda de conexidade da rede, aumentando o custo financeiro ao exigir a inserção de uma grande quantidade de retransmissores para se obter novamente uma rede conexa.

O problema estudado neste documento, chamado de minimização de retransmissores em redes de sensores sem fio com limite de alcance de rádio transmissão, visa uma solução equilibrada entre custo financeiro, desempenho e vida útil das RSSFs através da limitação do alcance de transmissão dos rádios transceptores dos nós da rede e inserção de nós retransmissores. Ou seja, a vida útil da rede é preservada ao limitar o alcance em um valor capaz de poupar a energia dos seus nós e, ao mesmo tempo, o custo é reduzido pela introdução do número mínimo de retransmissores que garante a conexidade da rede.

Formalmente, o problema a ser solucionado consiste em, dado um limite superior de alcance de transmissão e os nós sensores previamente posicionados no plano, computar o número mínimo de nós retransmissores, a posição e o alcance de transmissão de cada um deles e o alcance de transmissão dos nós sensores tais que a rede resultante seja conexa e nenhum nó ultrapasse o limite de transmissão. Esse problema pode ser modelado [Cheng et al. 2008] como o problema NP-completo [Lin and Xue 1999] de Árvore de Steiner com minimização dos nós de Steiner e limite de custo de aresta. O problema estudado se difere do Problema da Árvore de Steiner [Martins et al. 1999] clássico pois o conjunto de nós de Steiner não é conhecido a priori. No problema tratado, a quantidade e o posicionamento dos nós de Steiner devem ser decididos pelo algoritmo de solução.

Neste trabalho são propostos um algoritmo construtivo aleatório, uma busca local e uma metaheurística GRASP para a solução do problema de minimização de retransmissores em redes sensores sem fio com limite de alcance de rádio transmissão, sendo a eficácia dos algoritmos e a qualidade das soluções obtidas analisadas e comparadas com uma heurística gulosa da literatura através de estudos computacionais.

O artigo está dividido da seguinte forma. Na Seção 2 será feito uma formalização do problema tratado. Na Seção 3 serão propostos o algoritmo construtivo aleatório, o algoritmo de busca local e a estratégia GRASP para a solução do problema em questão. Na Seção 4 serão mostrados os resultados computacionais dos algoritmos implementados. A conclusão do trabalho será apresentada na Seção 5.

## 2. Modelo de Sistema e Definição do Problema

Considere um conjunto $V$ de nós sensores, com dimensão $|V|$, fixos em um plano bidimensional e equipados com antenas omnidirecionais responsáveis por enviar e receber sinais.

Em uma rede de sensores sem fio a potência de transmissão $p_u$ atribuída ao rádio transceptor de cada nó $u \in V$ determina o alcance de transmissão $r_u$ do nó. A potência de transmissão de um nó emissor é ajustada de acordo com a distância até o nó receptor e do ruído do ambiente. No modelo mais comum de atenuação de potência [Rappaport 2001, Moraes and Ribeiro 2013] a potência do sinal decai com $1/d^{\varepsilon}$, onde $d$ é a distância entre emissor e receptor e $\varepsilon$ é o expoente de perda de caminho (valores típicos de $\varepsilon$ estão entre 2 e 4). Usando esse modelo, o requisito de potência do nó $u$ para suportar a transmissão

através de um enlace de $u$ para o nó $v \in V$ é dado por

$$p_u \geq d_{uv}^{\varepsilon} q_v, \tag{1}$$

onde $d_{uv}$ é a distância Euclideana entre o transmissor $u$ e o receptor $v$, e $q_v$ é a potência limite de detecção do sinal pelo receptor que é geralmente normalizada para 1.

O alcance de transmissão, ou raio de cobertura, de um nó transmitindo com uma potência $p_u$ é um disco de raio $r_u = (p_u)^{\frac{1}{\varepsilon}}$ centralizado no nó [Santi 2005], ou seja, todo nó $v$ a uma distância $d_{uv} \leq r_u$ receberá a transmissão de $u$ (ver Figura 2). Dessa forma, uma rede de sensores sem fio é estabelecida através da atribuição do alcance de transmissão $r_u$ de cada nó $u \in V$.

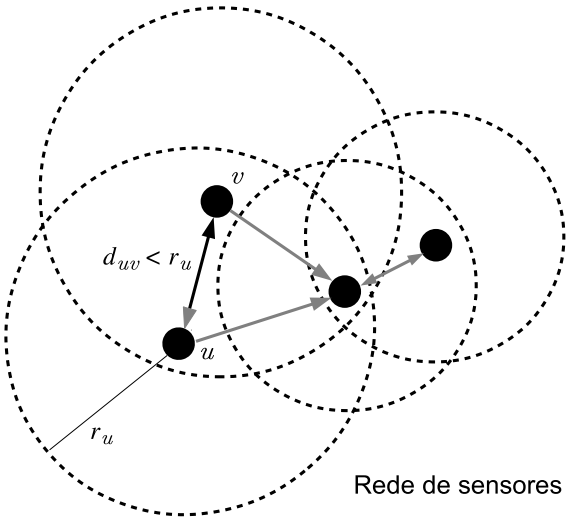



**Figura 2. Rede de sensores estabelecida pelo alcance de transmissão de seus nós.**

Para o problema estudado, um limite de alcance de transmissão $R$ deve ser fixado para todos os nós, ou seja, em uma solução do problema com limite de rádio transmissão, todo nó $u \in V$ deve possuir alcance de transmissão $r_u$ dentro do limite estabelecido, $r_u \leq R$. Quando essa limitação de alcance de transmissão é imposta, o estabelecimento de uma rede conexa só é possível com a construção de um conjunto de nós retransmissores $V_{rt}(\ell)$. Essa construção significa determinar a localização $\ell_w$ para todo nó $w \in V_{rt}(\ell)$ e a quantidade de nós retransmissores necessários para obter uma solução conexa.

Considera-se também no modelo adotado que, para cada par ordenado de nós $(u, v)$, com $u, v \in V \cup V_{rt}(\ell)$, tem-se um custo não negativo $e(u, v)$ associado a cada aresta $(u, v)$ tal que o custo representa a distância Euclidiana do arco $e(u, v) = d_{uv}$. A comunicação do nó $u$ para o nó $v$ será possível sempre que $r_u \geq e(u, v)$ e $r_v \geq e(v, u)$, ou seja, serão considerados para comunicação apenas os enlaces bidirecionais já que a comunicação bidirecional é implicitamente assumida em diversos protocolos de roteamento [Moraes et al. 2009].

Assim, o problema de minimização de retransmissores em redes de sensores sem fio com limite de rádio transmissão (MR-Sensor) pode ser formulado como: dado o conjunto de nós $V$, os custos não negativos dos arcos $e(u, v)$ para qualquer $u, v \in V$ e o parâmetro $R > 0$, determinar uma localização $\ell_w$ e uma atribuição de alcance de

transmissão $r_w$ para todo nó $w \in V_{rt}(\ell)$ e uma atribuição de alcance de transmissão $r_u$ para todo nó $u \in V$ tais que o número de retransmissores $|V_{rt}(\ell)|$ necessários para manter a conexidade global da rede seja minimizado e o grafo de comunicação resultante $G(r, \ell) = (V \cup V_{rt}(\ell), E(r, \ell))$ seja conexo, onde $E(r, \ell) = \{(u, v) : u, v \in V \cup V_{rt}(\ell), r_u \geq e(u, v), r_v \geq e(v, u), r_u \leq R, r_v \leq R\}$.

Esse problema é exatamente [Lloyd and Xue 2007] o problema de árvore de steiner mínima com número mínimo de pontos de steiner e limite de comprimento de aresta (*Steiner Minimum Tree with Minimum number of Steiner Points and bounded edge length - SMT-MSPBEL*) definido por [Lin and Xue 1999]. Eles provaram que o problema SMT-MSPBEL é NP-completo e apresentaram um algoritmo $5$-aproximativo baseado em árvore geradora mínima (AGM).

Em [Chen et al. 2000], foi provado que o algoritmo de [Lin and Xue 1999] na verdade é um algoritmo $4$-aproximativo e foi apresentado um algoritmo $3$-aproximativo para solucionar o mesmo problema. Em [Cheng et al. 2008] é apresentado um algoritmo $3$-aproximativo de menor complexidade que o de [Chen et al. 2000] e um algoritmo randomizado com taxa de aproximação de $2.5$. Mais recentemente, [Senel and Younis 2012] propuseram uma heurística baseada na Triangulação de Delaunay.

O problema de posicionamento de retransmissores também foi estudado para o caso de tolerância a falhas [Kashyap et al. 2006, Han et al. 2010] e para o modelo de redes de dois-níveis (*two-tiered network model*) [Tang et al. 2006, Lloyd and Xue 2007, Zhang et al. 2007] no qual os sensores são agrupados em torno de um nó concentrador que processa as informações recebidas e envia a informação processada para uma estação base. Em [Younis and Akkaya 2008, Wu et al. 2013, Senturk et al. 2014] pode ser encontrado um levantamento bibliográfico abrangente e atualizado para o problema de posicionamento de retransmissores com diferentes objetivos e abordagens.

Na seção seguinte, serão apresentados os algoritmos para solucionar o problema de minimização de retransmissores em redes de sensores sem fio com limite de rádio transmissão.

## 3. GRASP

Um GRASP, procedimento de busca gulosa, aleatória e adaptativa, pode ser visto como uma metaheurística que captura boas qualidades de um algoritmo guloso e de procedimentos de construção aleatórios [Martins et al. 1999].

O GRASP é um método iterativo de múltiplas partidas no qual cada iteração consiste de dois passos: um passo de partida e um passo de melhoramento. No passo de partida, utiliza-se um algoritmo construtivo aleatório (uma discussão de como esse algoritmo funciona é feito na Subseção 3.1) para se obter uma solução viável, não necessariamente a melhor, e a cada iteração do GRASP, o algoritmo construtivo aleatório é capaz de obter uma nova solução diferente da anterior. No passo de melhoramento, utiliza-se uma estratégia de busca local para melhorar a qualidade da solução obtida no passo de partida. A melhor solução encontrada em todas as iterações do GRASP é retornada como resultado.

O Algoritmo 1 mostra o pseudocódigo do GRASP para solucionar o problema MR-Sensor. O laço da linha 1 até a linha 5 implementa o procedimento de busca aleatorizada. O laço é executado enquanto o critério de parada $crit\_parada$ não é alcançado.

Na linha 2, dados o conjunto de nós sensores $V$, os custos $e(u,v) : u,v \in V$ e o limite de alcance de transmissão $R$, computa-se de forma aleatória os alcances de transmissão $r$ e as localizações $\ell$ dos retransmissores tal que o grafo de comunicação estabelecido $G(r,\ell) = (V \cup V_{rt}(\ell), E(r,\ell))$ seja uma solução para o problema MR-Sensor. A função `AGM-R()` é baseada na construção da árvore geradora mínima de um grafo e será descrita na Subseção 3.1.

**Algoritmo 1** Procedimento GRASP

**Entrada:** Parâmetros $R$ e $crit\_parada$, conjunto de nós $V$ e custos $e(u,v), \forall u,v \in V$.
**Saída:** Alcances de transmissão $r^*$, localizações $\ell^*$ tais que o grafo de comunicação $G(r^*,\ell^*) = (V \cup V_{rt}(\ell^*), E(r^*,\ell^*))$ é conexo e $\forall u \in V \cup V_{rt}(\ell^*) : r_u \leq R$.

1: **enquanto not** $crit\_parada$ **faça**
2: $r, \ell \leftarrow$ `AGM-R(`$V, e(u,v), R$`);`
3: $r', \ell' \leftarrow$ `BuscaLocal(`$r, \ell, R$`);`
4: `atualiza(`$r', \ell', r^*, \ell^*$`);`
5: **fim enquanto**
6: **retorna** $r^*, \ell^*$

Na linha 3 é realizada uma busca local (descrita na Subseção 3.2) a partir de $r, \ell$ calculados na linha anterior. A função `BuscaLocal()` retorna $r', \ell'$ tais que a solução $G(r', \ell')$ seja conexa, respeite o limite $R$ e seja um ótimo local na vizinhança de $G(r,\ell)$. Se a solução estabelecida por $r', \ell'$ é melhor do que a solução corrente $r^*, \ell^*$, essa é atualizada na linha 4. Após o término do laço, a melhor solução encontrada é retornada na linha 6.

### 3.1. Construtivo Aleatório

O algoritmo construtivo aleatório do GRASP deve possuir baixa complexidade, garantindo um número maior de iterações no mesmo intervalo de tempo. Quanto maior o número de iterações [Resende and Ribeiro 2003], melhor será a solução encontrada. Sendo assim, o algoritmo construtivo aleatório proposto baseia-se no algoritmo guloso descrito por [Lin and Xue 1999] que, por estar fundamentado na construção da árvore geradora mínima de um grafo, apresenta uma baixa complexidade de tempo, podendo ser implementado em $O(|V|^2 \log |V|)$.

O algoritmo construtivo aleatório é mostrado no Algoritmo 2. Como entrada o algoritmo demanda o conjunto de nós $V$, os custos $e(u,v)$ para qualquer $u,v \in V$ e o parâmetro $R > 0$. Sua saída é um grafo conexo $G(r',\ell) = (V \cup V_{rt}(\ell), E(r',\ell))$ estabelecido pelas localizações $\ell$ dos retransmissores e os alcances de transmissão $r'$ dos nós da rede tais que para todo nó $u \in V \cup V_{rt}(\ell)$ tem-se $r_u \leq R$.

Na linha 1 do Algoritmo 2 computa-se uma árvore geradora $T(r) = (V, E(r))$ de forma aleatória e sem considerar o limite de alcance de transmissão $R$, onde $E(r) = \{(u,v) : u,v \in V, r_u \geq e(u,v), r_v \geq e(v,u)\}$. Para calcular $T(r)$, é utilizada uma variação aleatória do algoritmo de Prim. No algoritmo de Prim puramente guloso, escolhe-se dentre as arestas candidatas $(u,v)$ aquela com o menor custo de aresta $e(u,v)$ para entrar na solução parcial. Uma aresta $(u,v)$ é considerada candidata se o nó $u$ encontra-se fora da solução parcial e o nó $v$ já está inserido na solução parcial. Um algoritmo de Prim com complexidade $O(|V|^2 \log |V|)$ pode ser encontrado em [Cormen et al. 2001]. No

**Algoritmo 2** Algoritmo aleatório baseado em árvore geradora mínima

**Entrada:** Conjuntos de nós $V$, custos $e(u,v), \forall u,v \in V$ e parâmetro $R$.
**Saída:** Alcances de transmissão $r'$, localizações $\ell$ tais que o grafo de comunicação $G(r',\ell) = (V \cup V_{rt}(\ell), E(r',\ell))$ é conexo e $\forall u \in V \cup V_{rt}(\ell) : r_u \leq R$.
1: $T(r) = (V, E(r)) \leftarrow$ `prim-r(`$V, e(u,v)$`);`
2: $r', \ell \leftarrow$ `AplicaLimiteAlcance(`$E(r)$`);`
3: **retorna** $r', \ell$

Prim aleatorizado proposto, acrescenta-se uma lista restrita de candidatos (LRC) formada por todas as arestas candidatas $(u,v)$ tais que

$$e(u,v) \leq e_{min} + \alpha(e_{max} - e_{min}), \tag{2}$$

onde $\alpha \in [0,1]$, $e_{min} = \min\{e(u,v) : u,v \in V$ e $(u,v)$ é uma aresta candidata$\}$ e $e_{max} = \max\{e(u,v) : u,v \in V$ e $(u,v)$ é uma aresta candidata$\}$. A aresta para entrar na solução parcial é escolhida aleatoriamente da LRC com probabilidade uniforme.

Como a árvore geradora $T(r) = (V, E(r))$ contém arestas que não respeitam o limite de alcance de transmissão, a função `AplicaLimiteAlcance()` é utilizada na linha 2 para garantir que o limite $R$ não seja ultrapassado. Essa função recebe o conjunto de arestas $E(r)$ e substitui toda aresta $(u,v) \in E(r) : e(u,v) > R$ por um conjunto mínimo de retransmissores e suas respectivas arestas com custo menor que $R$ de forma que o caminho entre os nós $u$ e $v$ seja mantido (ver Figura 3). A substituição é feita com complexidade $O(|V|)$.


(a) Aresta $(u,v)$ não respeita limite de alcance de transmissão $R$.

(b) Arestas $(u,w)$ e $(w,v)$ respeitam o limite de alcance de transmissão $R$ e mantêm o caminho entre os nós $u$ e $v$.

**Figura 3. Substituição de um enlace $(u,v)$ com custo $e(u,v) > R$, por dois enlaces com um retransmissor $w$ tais que $e(u,w) < R$ e $e(w,v) < R$.**

### 3.2. Busca Local

Dado o espaço de soluções $S$ de um problema, a vizinhança $\mathcal{N}(s)$ de uma solução $s \in S$ é definida como um conjunto de soluções $N(s) \subseteq S$. Uma solução $s' \in N(s)$ pode ser alcançada pela aplicação de modificações elementares, chamadas de movimentos, em $s$. Os procedimentos que exploram sistematicamente uma vizinhança $\mathcal{N}(s)$ são denominados de busca local. Começando de uma solução inicial, a busca local explora a vizinhança

na procura de uma solução melhor do que a corrente. A solução corrente muda sucessivamente, até chegar-se a um ótimo local.

A busca local implementada tem por objetivo melhorar uma solução encontrada pelo algoritmo construtivo aleatório apresentado na Seção 3.1. Para o problema proposto, a busca local pretende reduzir a quantidade de retransmissores da solução construtiva previamente computada. Considere uma solução $G(r, \ell) = (V \cup V_{rt}(\ell), E(r, \ell))$ encontrada pelo Algoritmo 2, o movimento utilizado para exploração da vizinhança consiste na remoção de dois retransmissores $w_i, w_j \in V_{rt}(\ell)$, e inclusão de um único retransmissor $w_k$ que mantenha a conexidade global da rede sem que o limite de alcance de transmissão $R$ seja ultrapassado (ver Figuras 4(a) e 4(c)).

O algoritmo de busca local é mostrado no Algoritmo 3. Nas linhas 1 e 2 são inicializadas as variáveis necessárias. O laço da linha 3 à 14 executa enquanto não for encontrado um ótimo local. O laço da linha 5 à 13 junto com o laço da linha 6 à 12 testa todos os movimentos possíveis.

**Algoritmo 3** Algoritmo de busca local

**Entrada:** Parâmetro $R$, alcances de transmissão $r$ e localizações $\ell$ tais que o grafo de comunicação $G(r, \ell) = (V \cup V_{rt}(\ell), E(r, \ell))$ é conexo e $\forall u \in V \cup V_{rt}(\ell) : r_u \leq R$.

**Saída:** Alcances de transmissão $r'$, localizações $\ell'$ tais que o grafo de comunicação $G(r', \ell') = (V \cup V_{rt}(\ell'), E(r', \ell'))$ é conexo, $\forall u \in V \cup V_{rt}(\ell) : r_u \leq R$ e $|V_{rt}(\ell')| < |V_{rt}(\ell)|$.

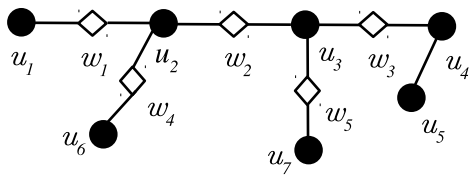
```
1:  r', ℓ' ← r, ℓ
2:  melhoria ← verdadeiro
3:  enquanto melhoria faça
4:      melhoria ← falso;
5:      para todo w_i ∈ V_rt(ℓ') faça
6:          para todo w_j ∈ V_rt(ℓ') faça
7:              r^rs, ℓ^rs ← removeSteiner(r', ℓ', w_i, w_j);
8:              r^rc, ℓ^rc ← restabeleceConexidade(r^rs, ℓ^rs);
9:              se (atualiza(r', ℓ', r^rc, ℓ^rc)) então
10:                 melhoria ← verdadeiro
11:             fim se
12:         fim para
13:     fim para
14: fim enquanto
15: retorna r', ℓ'
```

A cada movimento, o algoritmo retira o par de nós retransmissores sendo testado através da função `removeSteiner()` (linha 7) que possui complexidade $O(|V|)$. Na linha 8, a função `restabeleceConexidade()` encontra a estrutura de componentes conexas na qual se transformou o grafo de comunicação após a remoção dos retransmissores na linha anterior, em seguida, é feita uma busca por uma localização $P$ que esteja entre três nós $r, s, t \in V \cup V_{rt}(\ell)$ de três componentes conexas distintas tal que o alcance de transmissão de um retransmissor colocado em $P$ até os nós $r, s, t$ não ultrapasse o limite $R$. As coordenadas da localização $P$ são encontradas utilizando o cálculo da média das coordenadas de $r, s, t$. Quando os nós $r, s, t$ formam um triângulo o resultado da média é

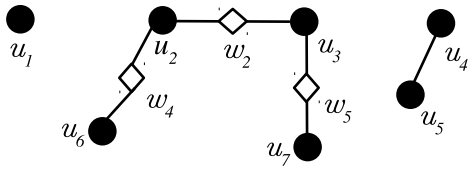
o baricentro desse triângulo. Como o cálculo das componentes conexas de um grafo, que pode ser implementado em $O(|V|^2)$, é executado apenas uma única vez e a busca pela localização $P$ tem complexidade $O(|V|^3)$, a função `restabeleceConexidade()` possui complexidade total de $O(|V|^3)$.

Por exemplo, dado o grafo de comunicação $G(r', \ell')$ conexo da Figura 4(a), a Figura 4(b) mostra um movimento não aprimorante pois, após a remoção dos nós $w_1$ e $w_3$, não é possível encontrar um ponto $P$ de reconexão. Já a Figura 4(c) mostra um exemplo de movimento aprimorante no qual são removidos os nós $w_1$ e $w_4$ e, em seguida, é construída a solução $G(r^{rc}, \ell^{rc})$ pela inserção do retransmissor $w_6$.

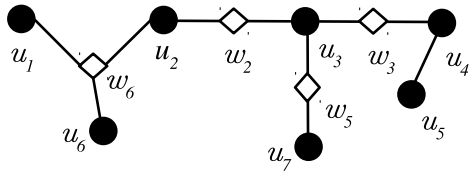
Quando um movimento encontra uma solução vizinha conexa $r^{rc}, \ell^{rc}$ melhor que a solução corrente $r', \ell'$, $|V_{rt}(\ell^{rc})| < |V_{rt}(\ell')|$, esta é atualizada na linha 9 e a variável $melhoria$ é marcada na linha 10, indicando que a busca local deve continuar em busca de um ótimo local. Quando um ótimo local é encontrado, a busca local termina e a solução encontrada é retornada na linha 15.



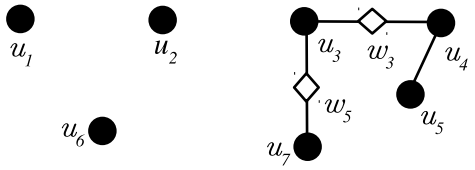

(a) Solução corrente $G(r', \ell')$.




(b) Remoção de nós distantes $w_1$ e $w_3$ de $G(r', \ell')$ não permite reconexão.




(c) Remoção dos nós $w_1$ e $w_4$ de $G(r', \ell')$ permite reconexão com nó $w_6$ e construção da solução vizinha $G(r^{rc}, \ell^{rc})$.




(d) Remoção do nó $w_2$ de $G(r^{rc}, \ell^{rc})$ junto com nó de grau maior que dois $w_6$ não permite reconexão.

**Figura 4. Exemplo de movimentos da busca local.**

Com o intuito de acelerar a busca local, deve-se reduzir a quantidade de vezes que a função de maior complexidade de tempo `restabeleceConexidade()` executa. Para isso, ao invés de testar a etapa de remoção do movimento para todo par de nós da solução corrente $r', \ell'$ somente os pares com maior possibilidade de permitir encontrar soluções conexas de boa qualidade são testados sem que movimentos importantes sejam descartados. Observando a estrutura dos grafos de comunicação, as seguintes regras foram estabelecidas para a retirada dos retransmissores na etapa de remoção de um movimento:

- Dois nós retransmissores $w_i, w_j \in V_{rt}(\ell)$ só podem ser removidos no mesmo

movimento se possuírem um enlace com um mesmo nó sensor $u$, ou seja, existem as arestas $(u, w_i)$ e $(w_j, u)$.
- Os nós retransmissores $w_i$ e $w_j$ devem possuir apenas dois enlaces cada um.

Essas regras se devem ao fato de que, durante a análise dos resultados da busca local, notou-se que a maioria dos movimentos aprimorantes possuía essas características. A primeira regra garante a proximidade entre os nós das diferentes componentes conexas criadas com a remoção, facilitando a existência do ponto $P$ entre três nós com alcance de transmissão dentro do limite $R$. A Figura 4(b) mostra que a remoção de dois retransmissores distantes dificulta a reconexão da rede com apenas um retransmissor.

A segunda regra evita que a rede seja quebrada em muitas componentes conexas. A Figura 4(d) mostra que a criação de várias componentes conexas dificulta a reconexão da rede com apenas um retransmissor. A Figura 4(c) mostra que os movimentos restritos às regras definidas são aqueles que efetivamente reduzem o número de retransmissores.

## 4. Resultados Computacionais

Experimentos computacionais para o GRASP foram feitos em um conjunto de instâncias com $|V| \in [25, 180]$ nós distribuídos aleatoriamente em uma circunferência normalizada. Para cada tamanho de $|V|$, dez instâncias diferentes foram produzidas e os resultados mostram a média das dez execuções independentes de cada algoritmo.

Todas as instâncias geradas são representadas como grafos completos. O valor do parâmetro $R$ foi escolhido de forma que a quantidade de nós retransmissores utilizados estivesse próxima da metade da quantidade de nós sensores. Um cluster computacional com Intel Xeon de 2.40GHz e 8 Gbytes de memória RAM sendo executado no GNU/Linux 2.6.24 foi usado nos experimentos.

A eficácia dos algoritmos será avaliada em termos de tempo de computação e qualidade da solução. O parâmetro $\alpha$ foi definido usando a estratégia reativa descrito em [Prais and Ribeiro 2000] com a distribuição de probabilidade sendo atualizada a cada 20 iterações. A heurística GRASP usa como critério de parada o máximo de 1500 iterações sem melhoria na solução corrente.

Na Tabela 1 são apresentados os resultados médios referentes à quantidade de retransmissores e ao tempo de execução em segundos para os algoritmos GRASP regular (GR) e GRASP acelerado (GR-a). O GRASP regular utiliza o procedimento de busca local descrito no Algoritmo 3. O GRASP acelerado utiliza o procedimento de busca local com as regras de aceleração descritas na Seção 3.2. A coluna $d(\%)$ da Tabela 1 mostra a diferença percentual entre o valor ($X$) do resultado encontrado por GR em relação ao valor ($Y$) do resultado encontrado por GR-a, calculado como $100 \times (Y - X)/X$.

Os resultados da Tabela 1 mostram que o uso do procedimento de busca local acelerado faz o GR-a encontrar soluções tão boas quanto ao GR mas em tempo de execução menor, ou seja, as regras de aceleração permitem que a busca local encontre as soluções aprimorantes na vizinhança com menos movimentos. O algoritmo GR-a encontra soluções com diferença de quantidade de retransmissores em torno de 1.0% em relação às soluções de GR, entretanto GR-a chega a gastar 94.0% menos de tempo que GR para encontrar as suas soluções. A Figura 5 compara a ordem de crescimento dos algoritmos

**Tabela 1. Comparação dos resultado médios de tempo e qualidade das soluções entre os algoritmos GR e GR-a**

| | GR | | GR-a | | | |
|---|---|---|---|---|---|---|
| $\|V\|$ | retrans. | tempo (s) | retrans. | $d(\%)$ | tempo (s) | $d(\%)$ |
| 25 | 12.1 | 27.1 | 12.2 | 0.8 | 5.0 | -81.5 |
| 40 | 19.2 | 209.6 | 19.1 | -0.5 | 24.2 | -88.4 |
| 65 | 29.5 | 1117.2 | 29.6 | 0.3 | 79.0 | -92.9 |
| 80 | 36.4 | 2937.6 | 36.7 | 0.8 | 174.2 | -94.0 |

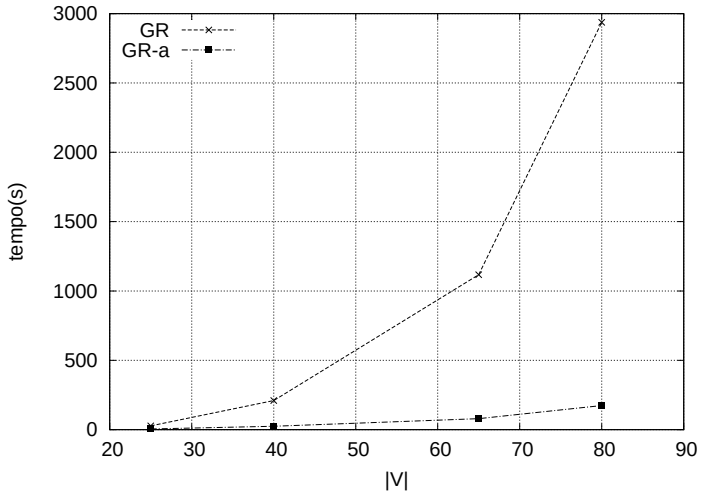



**Figura 5. Comparação entre os tempos médios de execução dos algoritmos GR e GR-a.**

GR e GR-a, mostrando que o tempo de execução de GR torna-se alto a partir de $|V| = 80$. Sendo assim, utilizou-se no próximo experimento somente o algoritmo GR-a.

Na Tabela 2 é apresentada a comparação entre os resultados médios referentes à quantidade de retransmissores da solução encontrada pelo algoritmo puramente guloso proposto por [Lin and Xue 1999] (LX) e pelo algoritmo GRASP acelerado (GR-a). A tabela mostra a diferença percentual $d(\%)$ entre as soluções dos dois algoritmos e o tempo de CPU em segundos que GR-a gastou para encontrar as suas soluções. Os tempos não são comparados pois o algoritmo guloso encontrou todas as suas soluções em menos de um segundo.

A Figura 6 mostra a comparação entre a quantidade média de retransmissores da solução encontrada por LX e GR-a. Os resultados da Tabela 2 e da Figura 6 mostram que algoritmo GR-a é estável pois mantém, independente da quantidade $|V|$ de nós sensores da rede, a mesma taxa de melhoria da solução em relação ao algoritmo puramente guloso LX. O algoritmo GR-a encontra soluções com o número de nós retransmissores em torno de 10% menores em comparação com o LX. Entretanto, para encontrar soluções de boa qualidade em instâncias maiores, GR-a gasta mais tempo de processamento já que a complexidade de percorrer a vizinhança aumenta.

Os resultados mostram que a metaheurística GRASP proposta é capaz de encontrar soluções de boa qualidade para o problema de minimização de retransmissores em

**Tabela 2. Comparação dos resultado médios da qualidade das soluções entre os algoritmos LX e GR-a**

| | LX | GR-a | | |
|---|---|---|---|---|
| $\|V\|$ | retrans. | retrans. | $d(\%)$ | tempo (s) |
| 25 | 13.9 | 12.2 | -12,2 | 5.0 |
| 40 | 21.1 | 19.1 | -9,4 | 24.2 |
| 65 | 33.9 | 29.6 | -12,6 | 79.0 |
| 80 | 42.0 | 36.7 | -12,6 | 174.2 |
| 95 | 47.1 | 41.7 | -11,4 | 297.4 |
| 110 | 51.5 | 45.9 | -10,8 | 412.9 |
| 125 | 61.5 | 54.9 | -10,7 | 674.4 |
| 140 | 71.2 | 64.2 | -9,8 | 958.8 |
| 165 | 75.8 | 67.5 | -10,9 | 1308.1 |
| 180 | 84.7 | 76.1 | -10,1 | 1312.5 |

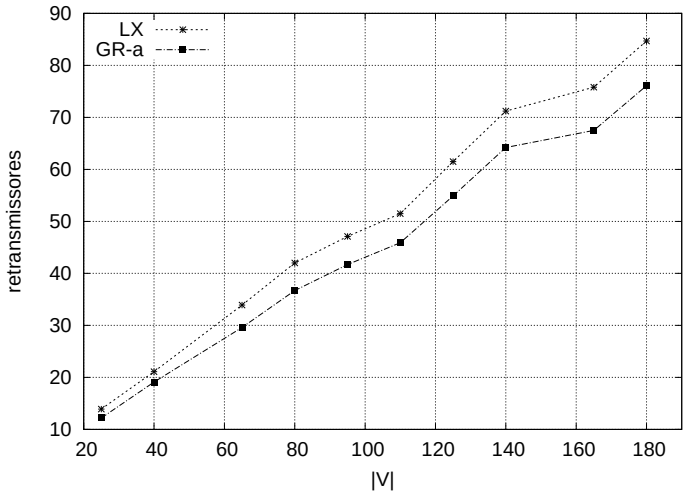



**Figura 6. Comparação entre a quantidade média de retransmissores da solução encontrada por LX e GR-a.**

redes de sensores sem fio com limite de rádio transmissão.

## 5. Conclusão

Neste artigo foi estudado o problema de minimização de retransmissores em redes de sensores sem fio com limite de alcance de rádio transmissão. Esse problema pode ser modelado como o problema NP-completo de Árvore de Steiner com minimização dos nós de Steiner e limite de custo de aresta.

Por se tratar de um problema NP-completo, foram propostos três algoritmos: um algoritmo construtivo aleatório, uma busca local e uma metaheurística GRASP. Para aumentar a eficácia do GRASP, também foi proposto um esquema de aceleração da busca local. Todos os algoritmos propostos, mais um algoritmo guloso aproximativo da literatura, foram implementados testados e comparados computacionalmente.

A diferença do problema estudado em comparação ao problema clássico de árvore de Steiner, no qual o conjunto e a localização dos nós de Steiner são conhecidos a priori,

gera grande flexibilidade na escolha da quantidade e posicionamento dos nós de Steiner ou, no caso da redes de sensores, dos nós retransmissores. Essa flexibilidade tornou a implementação da busca local mais desafiadora pois criou uma infinidade de possibilidades de posicionamento de retransmissores. Além disso, os algoritmos apresentados na literatura são algoritmos construtivos que podem ter sua taxa de aproximação demonstrada, não havendo outros algoritmos de busca local a serem comparados.

Os testes computacionais mostraram que o esquema de aceleração da busca local permitiu ao GRASP ter resultados mais rápidos sem perda de qualidade das soluções. Os testes também compararam o GRASP acelerado com um algoritmo construtivo da literatura com baixa ordem de complexidade. Os resultados dos testes demonstraram que soluções de algoritmos aproximativos podem ser melhoradas através de procedimentos de busca local e que o algoritmo GRASP implementado mantém uma taxa constante de melhoria da solução gulosa independente do tamanho da rede.

Para trabalhos futuros, dada a flexibilidade de localização dos nós retransmissores, pode-se considerar a implementação de buscas locais que façam a remoção de nós utilizando-se da movimentação dos retransmissores já pertencentes à solução.

## Agradecimentos

Este trabalho foi parcialmente financiado pelo Fundação de Amparo à Pesquisa do Espírito Santo (FAPES).

## Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38:393–422.

Chen, D., Du, D.-Z., Hu, X.-D., Lin, G.-H., Wang, L., and Xue, G. (2000). Approximations for steiner trees with minimum number of steiner points. *Journal of Global Optimization*, 18:17–33.

Cheng, X., Du, D.-Z., Wang, L., and Xu, B. (2008). Relay sensor placement in wireless sensor networks. *Wireless Network*, 14:347–355.

Cormen, T., Leiserson, C., Rivest, R. L., and Stein, C. (2001). *Introductions to Algorithms*. McGraw-Hill.

Han, X., Cao, X., Lloyd, E. L., and Shen, C.-C. (2010). Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9:643–656.

Kashyap, A., Khuller, S., and Shayman, M. (2006). Relay placement for higher order connectivity in wireless sensor networks. In *Proc. of the 25th IEEE International Conference on Computer Communications*, volume 6, pages 1–12, Barcelona.

Lin, G.-H. and Xue, G. (1999). Steiner tree problem with minimum number of steiner points and bounded edge-length. *Information Processing Letters*, 69:53–57.

Lloyd, E. L. and Xue, G. (2007). Relay node placement in wireless sensor networks. *IEEE Transactions on Computers*, 56:134–138.

Martins, S., Pardalos, P., Resende, M., and Ribeiro, C. (1999). Greedy randomized adaptive search procedures for the steiner problem in graphs. In *Proc. of the Randomization Methods in Algorithm Design: DIMACS Workshop*, pages 133–145. American Mathematical Soc.

Min, R., Bhardwaj, M., Cho, S.-H., Ickes, N., Shih, E., Sinha, A., Wang, A., and Chandrakasan, A. (2002). Energy-centric enabling technologies for wireless sensor networks. *IEEE Wireless Communications*, 9:28–39.

Moraes, R. E. N. and Ribeiro, C. C. (2013). Power optimization in ad hoc wireless network topology control with biconnectivity requirements. *Computers & Operations Research*, 40:3188–3196.

Moraes, R. E. N., Ribeiro, C. C., and Duhamel, C. (2009). Optimal solutions for fault-tolerant topology control in wireless ad hoc networks. *IEEE Transactions on Wireless Communications*, 8:5970–5981.

Prais, M. and Ribeiro, C. C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176.

Rappaport, T. (2001). *Wireless Communications: Principles and Practice*. Prentice Hall.

Resende, M. G. C. and Ribeiro, C. C. (2003). Greedy randomized adaptive search procedures. In *Handbook of metaheuristics*, pages 219–249. Springer.

Santi, P. (2005). *Topology control in wireless ad hoc and sensor networks*. John Wiley & Sons, Ltd.

Senel, F. and Younis, M. (2012). Optimized relay node placement for establishing connectivity in sensor networks. In *Proc. of the IEEE Global Communications Conference*, pages 512–517, Anaheim, California.

Senturk, I. F., Akkaya, K., and Yilmaz, S. (2014). Relay placement for restoring connectivity in partitioned wireless sensor networks under limited information. *Ad Hoc Networks*, 13, Part B:487 – 503.

Tang, J., Hao, B., and Sen, A. (2006). Relay node placement in large scale wireless sensor networks. *Computer communications*, 29:490–501.

Wang, F. and liu, J. (2011). Networked wireless sensor data collection: Issues, challenges, and approaches. *IEEE Communications Surveys & Tutorials*, 13:673–687.

Wu, D., Youcef-Toumi, K., Mekid, S., and Ben Mansour, R. (2013). Relay node placement in wireless sensor networks for pipeline inspection. In *American Control Conference (ACC), 2013*, pages 5905–5910, Washington, DC.

Younis, M. and Akkaya, K. (2008). Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks*, 6:621–655.

Zhang, W., Xue, G., and Misra, S. (2007). Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms. In *Proc. of the 26th IEEE International Conference on Computer Communications*, pages 1649–1657, Anchorage.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 10*

## *Redes Definidas por Software*

# OpenWiMesh: um Framework para Redes Mesh Sem fio Definidas por Software

**Italo Valcy Brito[1], Sérgio Gramacho[1], Ibirisol Ferreira[1], Marcelo Nazaré[1], Leobino Sampaio[1], Gustavo B. Figueiredo[1]**

[1]Dep. de Ciência da Computação – Universidade Federal da Bahia (UFBA)
Caixa Postal 15.064 – 91.501-970 – Salvador – BA – Brasil

{italo,sergioluis,ibirisol,marcelonaza,leobino,gustavo}@dcc.ufba.br

***Abstract.*** *The adoption of Software-Defined Networks (SDN) in the context of wireless mesh networks opens up room for new functionalities such as traffic engineering, flow based traffic forwarding and, more generally, "programmability". If on one hand the new paradigm brings many perspectives, for another some challenges need to be addressed, such as the contrast between the SDN's centralized model and mesh networks' inherently distributed architecture; or additionally, the demand for a separate control channel apart from the data channel. This work introduces OpenWiMesh, a framework which integrates SDN, more precisely OpenFlow, in wireless mesh networks, using in-band signaling and allows to experiment different traffic engineering techniques to improve network utilization. Sample applications implemented and evaluated with emulation shows the framework's feasibility and potential.*

***Resumo.*** *A adoção do paradigma de Redes Definidas por Software (SDN) no contexto de redes mesh sem fio abre espaço para novas funcionalidades, como engenharia de tráfego, encaminhamento de pacotes baseado em fluxos e, mais genericamente, "programabilidade" da rede. Se por um lado o novo paradigma agrega diversas perspectivas, por outro, alguns desafios precisam ser endereçados, a exemplo do contraste entre o modelo centralizado de SDN e a arquitetura inerentemente distribuída da rede mesh; ou ainda, da necessidade de se fazer o controle de forma integrada à rede de dados, sem demandar recursos adicionais (sinalização in-band). Este trabalho apresenta OpenWiMesh, um framework que integra SDN, mais precisamente Openflow, em redes mesh sem fio, com sinalização in-band, e que permite experimentar diversas técnicas de engenharia de tráfego para melhorar a utilização da rede. Exemplos de aplicações implementadas e avaliadas com emulação demonstram a viabilidade e potencial de uso do framework.*

## 1. Introdução

Uma rede sem fio em malha (WMN, do inglês *Wireless Mesh Network*) é uma rede sem fio de múltiplos saltos em que cada nó é capaz de se comunicar com todos os demais nós da rede, seja diretamente com aqueles ao seu alcance (vizinhos) ou através do encaminhamento por nós intermediários [Akyildiz and Wang 2009]. As WMNs têm sido reiteradamente apontadas como a solução tecnológica de melhor custo benefício para a construção de uma plataforma de comunicação altamente escalável e capaz de promover

conectividade a um custo relativamente baixo. Sua utilização traz benefícios potenciais como alta capacidade de comunicação, elevada disponibilidade e tolerância a falhas, além de propiciar uma rápida implantação de serviços de rede.

Estes fatos motivam a utilização de WMNs em muitos cenários, tais como em redes empresariais e domésticas, redes metropolitanas, suporte a sistemas de gestão de catástrofes, dentre outras [Akyildiz and Wang 2009]. Entretanto, essa heterogeneidade de cenários e, em última instância, de perfis de tráfego, implica a necessidade de alto grau de flexibilidade em estratégias de encaminhamento e de um controle mais eficiente da operação da rede, que propicie melhor utilização dos recursos da mesma, sob pena de perda de desempenho e degradação dos serviços.

Diferentes protocolos de roteamento, como AODV [Lee and Gerla 2001] e OLSR [Jacquet et al. 2001], têm sido utilizados em WMNs. Entretanto, mesmo com todas as extensões disponíveis (o que não implica facilidade de extensão), esses protocolos não facilitam a inovação da rede ou, ao menos, não no mesmo ritmo e com a mesma facilidade que SDN possibilita [Dely et al. 2011]. Além disso, as decisões de roteamento são tomadas de forma distribuída, baseadas no conhecimento local, e, portanto, parcial, que cada nó *mesh* dispõe acerca das condições da rede. Tal fato pode limitar em grande medida as estratégias que podem ser utilizadas para engenharia de tráfego nas WMN.

Uma abordagem alternativa seria realizar o roteamento e encaminhamento baseados em fluxos. Fluxos de diferentes aplicações entre o mesmo par origem e destino podem ser transportados por diferentes caminhos, distribuindo a carga e o consumo energético pelos diversos nós da rede. Além disso, o roteamento por fluxos permite o encaminhamento de fluxos de natureza confidencial somente por nós que façam parte de uma categoria de alta credibilidade, enquanto os demais fluxos poderiam ser encaminhados por quaisquer nós. Neste exemplo, os critérios de seleção de caminhos vão além de aspectos técnicos, explorando informações existentes em outras camadas da rede. Tais características da rede podem ser alcançadas pela adoção do novo paradigma de Redes Definidas por Software (SDN, do inglês *Software Defined Networks*). O uso desse paradigma permite tornar as redes mais inteligentes e programáveis. Para a comunicação entre os componentes SDN foi usado o protocolo *Openflow* [McKeown et al. 2008].

O Openflow é protocolo mais frequente em redes SDN, e tem obtido significativa atenção tanto da academia como da indústria nos últimos anos. No contexto das redes cabeadas, Openflow é conhecido, aceito e tem sido utilizado de forma prática. No contexto das redes sem fio, alguns trabalhos iniciais de pesquisa foram realizados [Yap et al. 2010, Dely et al. 2011], porém utilizando uma abordagem híbrida, com redes paralelas às WMN para separação do tráfego de controle do tráfego de dados. As redes paralelas fazem uso de nós com múltiplas interfaces sem fio ou de duas redes virtuais numa mesma interface sem fio. Além disso, o encaminhamento na rede de controle utiliza protocolos de encaminhamento distribuídos, como OLSR. Estas abordagens são chamadas híbridas, por usarem a estratégia distribuída na rede de controle e a nova estratégia SDN na rede de dados. Elas podem ser consideradas onerosas do ponto de vista de recursos usados (interfaces e espectro de frequências para duas redes sem fio) ou sofrem de problemas de interferência (contenção de transmissão e redução de relação sinal ruído). Outra desvantagem destas abordagens híbridas é que o controle da rede não tira proveito dos benefícios de engenharia de tráfego aplicados na rede pelos algoritmos de encaminhamento

SDN centralizados.

Neste trabalho é apresentado um *framework* para engenharia de tráfego em redes *mesh* sem fio. Através dele é possível a adoção de técnicas e algoritmos que melhorem o funcionamento da rede, além de permitir um ambiente controlado para experimentação em WMNs. Fez-se uso do protocolo Openflow para a comunicação entre os componentes SDN. O controle da rede é centralizado e a sinalização *in-band*. No melhor do nosso conhecimento, nenhum trabalho anterior utilizou sinalização *in-band* em WMNs com muitos nós.

O restante do trabalho está organizado da seguinte maneira: na Seção 2 apresenta-se o *framework* e seus componentes estruturais e funcionais; em seguida, na Seção 3, mostra-se uma prova de conceito do *framework*, demostrando sua viabilidade de uso; já a Seção 4 lista os trabalhos relacionados; e, por fim, a Seção 5 apresenta a conclusão e trabalhos futuros.

## 2. OpenWiMesh: arquitetura do framework

Esta seção apresenta a descrição do *framework* OpenWiMesh, seus principais componentes e estratégias de controle da WMN, que foram adotadas a fim de construir uma rede flexível e eficiente. A flexibilidade da rede visa viabilizar a adoção de diversas técnicas de roteamento e engenharia de tráfego, dando maior versatilidade e programabilidade à rede. De forma complementar, a eficiência da rede torna factível sua implantação em ambientes reais. Em particular, neste trabalho a eficiência da rede diz respeito ao consumo de recursos (redes paralelas para controle e dados) e desempenho de encaminhamento.

### 2.1. Visão geral dos componentes

A Figura 1 sintetiza os principais componentes do OpenWiMesh e seus relacionamentos. Nela é possível observar que a rede *mesh* sem fio é composta por nós *mesh* (MN, do inglês *Mesh Node*, Fig. 1b), que fazem o encaminhamento na rede, e o nó controlador central (CTL, Fig. 1a), um nó especial responsável por determinar como os outros nós encaminham o tráfego. Opcionalmente, o *framework* suporta nós clientes não modificados, que apenas utilizam a rede como meio de acesso. Os principais componentes (estruturais e funcionais) do *framework* são:

- Comutador SDN: software de encaminhamento de pacotes no MN com suporte a Openflow. Consiste da tabela de fluxos, que executa o casamento e encaminhamento de pacotes, e do canal de comunicação com o controlador;
- *GraphClient*: sistema responsável pela coleta de informações do sistema sem fio e envio ao controlador central;
- Regras iniciais de encaminhamento: definem como os MNs encaminham o tráfego inicialmente na rede *mesh in-band*;
- Mecanismo de controle *in-band*: conjunto de estratégias necessárias para viabilizar um canal de sinalização *in-band* entre o nó controlador central e os demais nós da rede;
- Grafo da rede: estrutura de dados que armazena informações sobre os MNs ativos, associações entre eles e informações do sistema sem fio;
- Algoritmo de roteamento: estratégia de escolha de caminhos entre uma origem e um destino na rede, podendo fazer uso das informações do grafo da rede e também de critérios não técnicos;

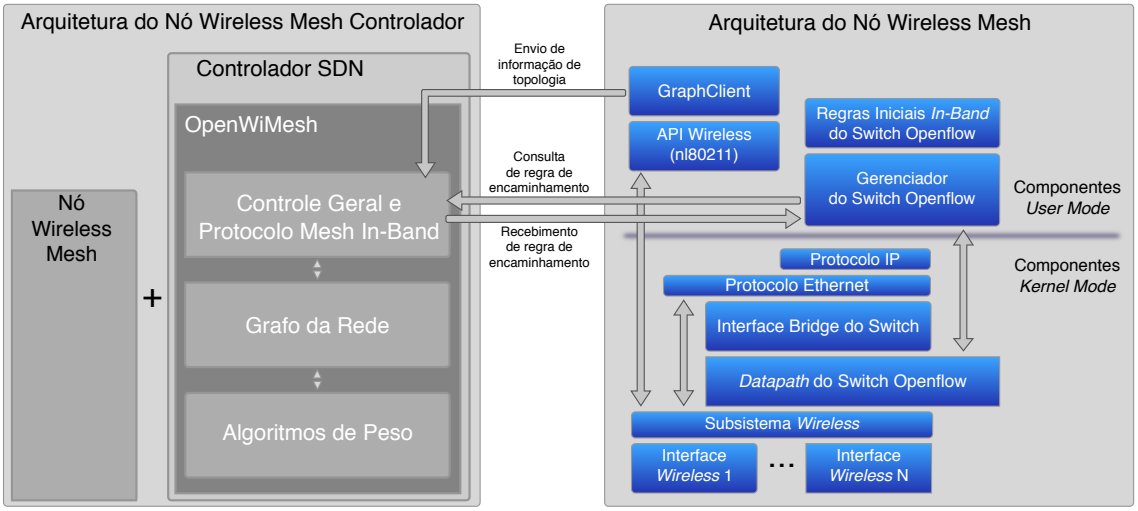



**Figura 1. Arquitetura do Framework OpenWiMesh, apresentando o MN e o controlador SDN**

- Controle geral da rede: responsável pelo recebimento e tratamento das requisições de encaminhamento (*packet-in*), execução do algoritmo de roteamento e instalação da regra de encaminhamento em cada MN do caminho escolhido.

Ao longo desta seção, todos esses componentes serão descritos em detalhes.

Os MNs devem ser equipados com, no mínimo, uma interface sem fio IEEE 802.11 operando no modo *Ad-Hoc*. Devem, ainda, oferecer suporte ao protocolo Openflow no comutador SDN e possuir o componente *GraphClient* do *framework*. Já o CTL deve executar, adicionalmente, a aplicação de controle e monitoramento da rede, matendo o grafo da rede atualizado e permitindo a customização dos algoritmos de roteamento.

A implementação de comutador SDN adotada nos MNs foi o Open vSwitch [Pfaff et al. 2009], que permite modificar regras inicias de controle Openflow e oferece melhor desempenho por executar o encaminhamento de pacotes em espaço de kernel. O software de controle da rede é baseado no POX [NOXRepo 2013], que suporta o desenvolvimento de aplicações SDN em Python e utiliza o protocolo Openflow para controlar remotamente os MNs através de um canal de sinalização. O canal de sinalização é apresentado na Seção 2.2.

No plano de controle da rede *mesh*, o nó controlador define, sob demanda, os caminhos que devem ser usados para cada fluxo. Ao receber um pacote de um fluxo desconhecido, o MN encaminha-o para o nó controlador, realizando uma "consulta de encaminhamento" (*packet-in*). O controlador utiliza um algoritmo de roteamento sobre grafo da da rede para decidir a rota que aquele tráfego deve seguir. Adicionalmente, o algoritmo de roteamento pode ser influenciado por critérios não técnicos, como o grau de credibilidade ou confiabilidade de um MN, ou ainda pode fazer uso de um subconjunto de nós com determinado grau de afinidade. Exemplos do potencial de uso do algoritmo de roteamento para influenciar na escolha de caminhos são apresentados na Seção 2.4. Em seguida, o nó controlador executa uma rotina de instalação do caminho na rede *mesh*, onde enviará uma mensagem de configuração para todos os nós pertencentes ao caminho contendo uma "regra de encaminhamento" (*flow-modify*). Pré-configurar todos os nós no

caminho evita que consultas de encaminhamento adicionais sejam geradas para aquele fluxo.

Já no plano de encaminhamento, cada MN consulta sua tabela de fluxos em busca de uma regra de encaminhamento, que definirá o que deve ser feito com o pacote (ex: encaminhar para o próximo nó, descartar, etc.). Tal consulta leva em consideração valores de cabeçalhos de, possivelmente, múltiplas camadas, além dos endereços da Camada de Enlace (endereços MAC). A cada salto, os endereços MAC de origem e de destino devem ser reescritos. O novo MAC origem será o MAC do MN que faz o encaminhamento e o novo MAC destino será aquele do MN que é o próximo salto ou o fim do caminho. Esse processo permite que a interface sem fio do MN opere em modo não-promíscuo, no qual apenas os pacotes destinados àquele nó (*unicast*, *broadcast* ou *multicast*) são processados pela camada superior deste nó.

O fato do MN não necessitar operar em modo promíscuo melhora sobremaneira a eficiência da rede. As implementações de comutadores SDN com suporte a Openflow comumente configuram as interfaces pertencentes a um *datapath* em modo promíscuo, para que possam receber pacotes destinados a qualquer interface. Não obstante, essa configuração gera alta sobrecarga de processamento na pilha Openflow de cada nó (busca na tabela de fluxos, *packet-in*, instalação de fluxos, etc.) e considerável aumento nos pacotes de controle na rede. No OpenWiMesh, mesmo havendo MNs com múltiplas interfaces sem fio, não é necessário que essas interfaces sejam configuradas em modo promíscuo. Pelo contrário, recomenda-se manter o filtro de quadros na Camada Física.

Alguns outros mecanismos de gerenciamento da rede *mesh*, executados pelo controlador, incluem: manter controle sobre os caminhos instalados na rede, permitindo o gerenciamento pró-ativo e até mudança de rota em caminhos sobrecarregados; a partir das informações enviadas pelo *GraphClient*, verificar a consistência na lista de associações mantida no grafo, removendo arcos cuja associação na rede *mesh* não mais exista, e, dessa forma, evitando que esses caminhos ruins sejam escolhidos no algoritmo de roteamento; apresentação visual da topologia da rede.

### 2.2. Canal de sinalização

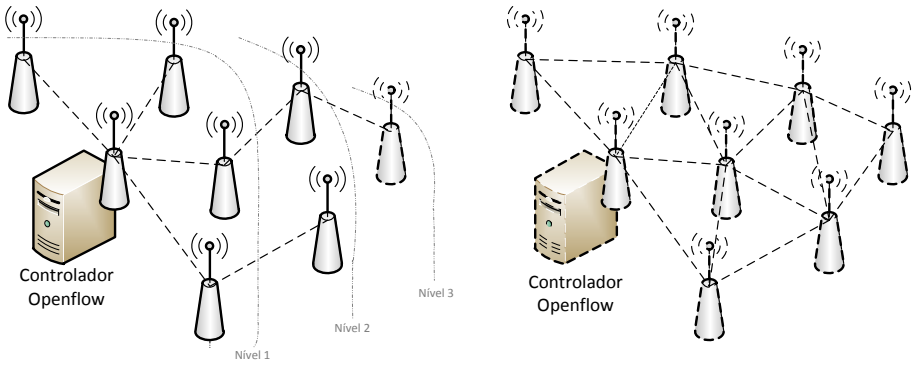
O modelo de funcionamento do Openflow pressupõe um canal de comunicação segura entre o controlador remoto e cada comutador SDN. Também conhecido como canal de sinalização, é através dele que o controlador configura e gerencia os comutadores, além de receber eventos e enviar pacotes [McKeown et al. 2008]. O canal de sinalização pode ser implementado com uma rede fora de banda (do inglês, *out-of-band*) ou valer-se da mesma rede usada para transmitir os dados, sendo então chamado de sinalização em banda (do inglês, *in-band*). A escolha do mecanismo de sinalização pode impactar na viabilidade de implantação da WMN.

Na sinalização *out-of-band*, requer-se uma segunda rede dedicada à transmissão das mensagens de controle. Essa outra WMN pode ser implementada com interfaces físicas (nós sem fio multi-rádio) ou com interfaces virtuais (múltiplos SSIDs). Nesse modelo, a rede de controle é estabelecida utilizando protocolos *mesh* convencionais, como OLSR. Já na sinalização *in-band*, as mensagens de controle fazem uso da mesma rede que transmite os dados, sem necessidade de recursos adicionais. A sinalização *in-band*, dessa forma, mostra-se mais adequada para WMNs, pois diminui os custos de implantação

e reduz a interferência que seria causada pelas redes sem fio sobrepostas. O principal desafio na implementação dessa abordagem é justamente a necessidade de conectividade IP entre os nós ao passo que essa conectividade será provida após o registro dos nós no controlador. Ademais, as implementações atuais de comutadores SDN com suporte a Openflow não estão preparadas para funcionar corretamente com abordagem *in-band* em WMNs, podendo resultar em *loops* na rede. Devido à complexidade supracitada, no melhor do nosso conhecimento, este é o primeiro trabalho que utiliza sinalização *in-band* em WMNs com muitos nós.

O OpenWiMesh utiliza canal de sinalização *in-band* e, para endereçar os desafios dessa abordagem, foi necessário uma adequação nas regras iniciais do comutador Openflow e a criação de uma estratégia de estabelecimento da rede *mesh*.

A modificação das regras Openflow iniciais é indispensável para correto estabelecimento da rede em modo *in-band*. As implementações de comutadores Openflow atuais comumente pré-configuram um conjunto de regras para repassar mensagens do protocolo OpenFlow adiante, na expectativa de que elas, em determinado momento, alcancem o controlador Openflow. Esse repasse das mensagens deve ocorrer em modo inundação (*flooding*), característica que leva à criação de *loops* na rede e inviabiliza seu uso. Para contornar esse problema, o conjunto de regras iniciais do Open vSwitch foi alterado para que o comutador encaminhe apenas o tráfego gerado localmente.




(a) Inicialização (b) Após convergência

**Figura 2. Exemplo do estabelecimento da WMN no OpenWiMesh: inicialmente (a) uma rede com topologia em árvore de dispersão que, então, converge para formar (b) um grafo conexo dirigido.**

A estratégia de estabelecimento da rede *mesh* é ilustrada na Figura 2. Como inicialmente os MNs encaminham apenas o tráfego local, os primeiros nós a estabelecer conexão são os nós vizinhos ao controlador, aqueles que estão a um salto (Fig. 2a, nível 1). Após estabelecer o canal de sinalização, o MN é capaz de encaminhar tráfego em favor de outros nós, através das "consultas de encaminhamento" ao controlador. Dessa forma, os nós que estão a dois saltos do controlador (Fig. 2a, nível 2) agora podem estabelecer o canal de sinalização, e assim sucessivamente. Em suma, durante a inicialização, a topologia da rede assemelha-se a uma árvore de dispersão.

À medida que os MNs são ativados na rede, o componente *GraphClient* envia

a lista de associações e seus atributos de cada MN para o controlador, que atualiza a topologia da rede, transformando-a em um grafo conexo dirigido (Fig. 2b).

A entrada de novo MN na WMN ocorre conforme descrito a seguir. Todos os MNs são configurados com IP estático, logo, ao ser ativado, sua primeira ação será uma tentativa de conexão com o controlador. O MN gera uma requisição ARP pelo MAC do controlador. Essa requisição é recebida por um dos MNs previamente estabelecidos. O nó que recebeu a requisição (MNad) envia uma consulta de encaminhamento desta ao CTL, que prepara todo o caminho para o protocolo Openflow e orienta ao MNad que responda à requisição ARP, assumindo para si a comunicação com o IP do CTL. Em seguida, quando o controlador for responder à requisição Openflow, o processo é repetido, no sentido CTL para novo MN. A configuração de IP estático, bem como uso do ARP para disparar a instalação do caminho de sinalização, foram usados para simplificar o estabelecimento da rede, porém é possível também utilizar mensagens de requisição DHCP para o processo de identificação de novos MNs e sua adição à WMN.

### 2.3. Grafo da rede

A rede *mesh* no *framework* foi modelada como um grafo dirigido $D = (V, A)$, no qual o conjunto de vértices $V$ é composto pelos MNs e o conjunto de arcos $A$ pelos pares ordenados de associações entre os MNs. Uma associação entre dois MNs $(N_1, N_2)$ ocorre quando o sinal de transmissão da interface sem fio de $N_1$ alcança $N_2$ em nível tal que o permita realizar a demodulação do sinal transmitido. Em uma rede *mesh* onde os MNs possuem interfaces sem fio homogêneas, a associação será na prática bidirecional. Ao pressupor associações unidirecionais, atende-se a ambos os cenários. Utilizou-se a biblioteca Python NetworkX [Hagberg et al. 2008] para armazenar o grafo da rede. O grafo da rede é mantido no controlador central, em consonância com o modelo centralizado de SDN, embora pudesse estar disponível em um outro servidor para fins de desempenho.

Cada aresta contém uma série de informações sobre a associação sem fio, tais como nível de sinal, nível de ruído, ocupação do enlace, dentre outras. O peso da aresta, usado pelos algoritmos de determinação de caminhos no grafo, é derivado dessas informações. O experimentador pode facilmente customizar a função de cálculo do peso das arcos, em vias de alcançar seus objetivos de engenharia de tráfego.

Um cuidado especial precisou ser tomado quanto à informação sobre a velocidade da associação sem fio. É comum, nas interfaces sem fio IEEE 802.11, que a velocidade de transmissão seja reduzida para valores mínimos quando em baixa demanda, como estratégia de minimização de temperatura e consumo de energia. Esta velocidade reduzida, portanto, não reflete o potencial máximo de velocidade de uma associação sem fio e pode induzir a escolhas ruins de caminho. Para minimizar este efeito, o componente *GraphClient* tem um recurso de projeção da velocidade potencial em função dos atributos da associação. Esta velocidade é chamada de $P - SPEED$, sendo expressa em Mbps. Para sua determinação, curvas de [Villegas et al. 2007] (OFDM modulation for 802.11g) e de [Mo and Bostian 2005] (DSSS modulation from 802.11b) foram usadas. Estas curvas correlacionam o nível SINR[1] com a BER[2] para cada MCS[3].

[1]Signal to Interference plus Noise
[2]Bit Error Rate
[3]Modulation and Coding Scheme

Os vértices do grafo mantêm informações dos MNs, como seus endereços IP e MAC, portas controladas pelo comutador SDN, lista de nós alcançáveis em cada porta, etc. O grafo é alimentado e atualizado, a partir das informações enviadas pelo *GraphClient* que executa em cada MN. O tempo de atualização deve ser tal que não sobrecarregue a rede *mesh* com tráfego de controle, mas, por outro lado, reflita em tempo aceitável as mudanças na rede. Foi adotado um tempo de atualização de 5 segundos.

O OpenWiMesh prove uma função de visualização do grafo da rede, inclusive contendo os pesos de cada arco. A visualização do grafo é importante para que checar a efetividade do algoritmo de roteamento e também para resolução de problemas. Exemplos de visualização do grafo da rede podem ser encontrados no site do projeto[4].

### 2.4. Algoritmos de roteamento

A possibilidade de customização do algoritmo de roteamento é um dos grandes diferenciais do OpenWiMesh. Através de extensões ou modificações no algoritmo de roteamento, é possível implementar Engenharia de Tráfego (TE) na rede ou fazer cumprir políticas administrativas na escolha dos caminhos. A técnica de TE pode ser usada para alcançar maior largura de banda na comunicação, ou maior balanceamento de carga no encaminhamento entre os MNs entre outros objetivos.

O utilizador do *framework* pode valer-se das informações contidas no grafo da rede para definir seu algoritmo de roteamento. Lista de associações, largura de banda projetada, relação sinal/ruído e banda residual são algumas das informações disponíveis no grafo para apoiar o mecanismo de escolha do melhor caminho. Todas essas informações podem ser combinadas, por exemplo, em uma função de cálculo de peso dos arcos, deixando o algoritmo de Dijkstra convencional escolher as melhores rotas com base no caminho mais curto (menor soma dos pesos dos arcos). Pode-se ainda definir um outro algoritmo de roteamento, que faça, por exemplo, a busca apenas em um subconjunto de vértices (excluíndo-se nós cuja localização não é confiável) ou que determine um caminho disjunto de outros caminhos congestionados.

Enfim, a flexibilidade na definição do algoritmo de roteamento em conjunto com a riqueza de informações disponíveis para o experimentador, potencializam os casos de uso do *framework*. Na Seção 3 são apresentados alguns exemplos de funções de determinação de caminhos.

## 3. Prova de conceito

Nesta seção são apresentados três diferentes algoritmos de determinação de caminhos, que se diferenciam quanto ao grau de espalhamento dos caminhos selecionados para diferentes fluxos de dados. Os algoritmos serão aplicados em experimentos detalhados a seguir. Estes algoritmos exemplo foram denominados:

i) HC (do inglês *Hop-Count*);
ii) HLRB (do inglês *Highest Link Residual Bandwidth*);
iii) HLRB-SHC (do inglês *Highest Link Residual Bandwidth in Same Hop Count*);

Usando as informações disponíveis no grafo da rede, diferentes funções podem ser criadas para definir o peso dos arcos e, portanto, influenciando diretamente na seleção

[4]Site do OpenWiMesh: `http://grade.dcc.ufba.br/OpenWiMesh`

de caminhos. Se um mesmo valor positivo constante for utilizado para o peso dos arcos, define-se uma estratégia que considera apenas a quantidade de saltos entre início e fim do caminho. Essa estratégia é conhecida como *Hop-Count* (algoritmo HC).

Nos algoritmos HLRB e HLRB-SHC os pesos dos arcos são definidos por funções que consideram a banda residual do enlace sem fio ($rB$, expresso em Mbps), que é diferença entre a velocidade do enlace ($S$, expressa em Mbps) e a velocidade consumida ($uS$, expressa em Mbps).

$$rB = S - uS \tag{1}$$

A velocidade consumida ($uS$) é calculada pela divisão do incremento de bytes trafegados no enlace ($incB = B - lastB$) (quantidade de bytes da atualização atual do *GraphClient* ($B$) menos a quantidade de bytes da atualização anterior ($lastB$)) pelo intervalo de tempo desde a última coleta. $uS$ é, portanto, uma velocidade média para o último intervalo de atualização.

$$uS = \frac{(B - lastB) \times 8}{(t - lastt) \times (1024 \times 1024)} \tag{2}$$

A velocidade do enlace ($S$) é obtida diretamente através do *GraphClient* e está armazenada como atributo dos arcos do grafo. Ela está disponível sob duas formas: velocidade da associação informada pela interface sem fio e a velocidade possível projetada pelo *GraphClient* ($P - SPEED$, conforme descrito na Seção 2.3).

### 3.1. Algoritmo HC - Hop-Count

O algoritmo HC escolhe um caminho que ofereça a menor quantidade de nós intermediários. O mesmo caminho é utilizado para todos os fluxos de um dado par de nós fonte/destino, mesmo que o caminho fique congestionado. Para este algoritmo, os pesos ($w$) dos arcos do grafo da rede são definidos como:

$$w = K \tag{3}$$

O valor da constante $K$ deve ser um valor inteiro positivo, assim passível de utilização no algoritmo de busca de menor caminho em grafos de Dijkstra. O valor de $600$ foi assumido para $K$, que representa a máxima velocidade de interfaces sem fio nos padrões IEEE 802.11 b, a, g, n (802.11n, com 40 MHz de largura de banda, 400nS de *Guard Interval* e 4 *Spatial Streams*) [Standards Committee 2012].

#### 3.1.1. Algoritmo HLRB - Highest Link Residual Bandwidth

O algoritmo HC tem a desvantagem de escolher sempre um mesmo caminho, inclusive quando os enlaces deste caminho estão congestionados. O algoritmo HLRB usa a informação de banda residual do enlace ($rB$) para computar o peso $w$ do seu arco. Este algoritmo, portanto, escolhe a cada momento um caminho que ofereça a maior banda residual disponível entre nós fonte e destino.

$$w = \frac{1}{(1 + rB)} \times K \tag{4}$$

$K$ é a mesma constante discutida na Equação 3.

Trabalhando no contexto de WMNs, com nós com apenas uma interface sem fio, está-se limitado aos problemas de interferência já conhecidos na literatura [Ramachandran et al. 2006], como problemas de contenção de transmissão. Contudo, a intenção com este algoritmo é de maximizar a distribuição da carga de comutação de pacotes pelos nós da WMN.

#### 3.1.2. Algoritmo HLRB-SHC - Highest Link Residual Bandwidth in Same Hop Count

O algoritmo HLRB-SHC combina os elementos dos dois anteriores, buscando selecionar os caminhos de maior banda residual, dentre as alternativas que sejam de menor Hop-Count. O cálculo do peso $w$ dos arcos é:

$$w = K + \frac{1}{(1 + rB)} \times K \tag{5}$$

Com este algoritmo reduz-se os efeitos de contenção anteriormente citados. $K$ continua a mesma constante discutida na Equação 3.

#### 3.1.3. Experimento 1 - Tempo de Convergência da WMN

O objetivo do Experimento 1 (Exp1) é verificar o tempo necessário para a convergência da WMN e a diferença de eficiência neste processo para os modos promíscuo e não-promíscuo nas interfaces de rede. Neste experimento são ativados MNs, simultaneamente, em um ambiente WMN emulado. É observado tanto o tempo necessário para que todos os MNs estejam conectados ao CTL (tempo de *Connection UP*) quanto o tempo necessário para que as informações de topologia sejam enviadas ao CTL e atualizadas no grafo da rede (tempo de *Topology UP*). A eficiência no *in-band* é avaliada com duas métricas: o tráfego *in-band* na interface do controlador e a quantidade de pacotes de consulta de encaminhamento desnecessários que foram descartados, ambos avaliados nos modos promíscuo e não-promíscuo. Para avaliar o comportamento destas métricas, alguns parâmetros são variados:

i) quantidade de nós na topologia (10, 20, 30, 40 e 50 MNs);
ii) formato da topologia (5 diferentes topologias para cada quantidade).

Há, portanto, um total de vinte e cinco diferentes topologias utilizadas.

O Emulador utilizado em ambos experimentos foi originalmente desenvolvido para a avaliação do OpenWiMesh. Ele utiliza recursos de virtualização do Sistema Operacional Linux, através da utilização do emulador CORE [Ahrenholz 2010], para a definição dos MN e suas interfaces de rede. O componente Core Daemon do CORE foi utilizado

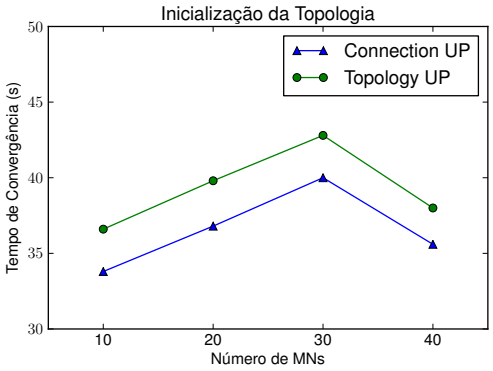



**Figura 3. Tempo de convergência de topologia**

para a definição dos NM em Linux Namespaces. A implementação da WMN (definição de topologia e seus atributos) é feita por meio de um componente especialmente desenvolvido (Aplicativo de Definição de Topologia - ADT). O estabelecimento das restrições de associação entre os MN na WMN foi implementado por meio de um comutador Openflow central e um conjunto de regras de encaminhamento para cada MN, fornecidas pelo ADT. Este Ambiente de Emulação será objeto de divulgação própria.

O funcionamento deste ambiente de emulação pode ser verificado no site do projeto OpenWiMesh[5].

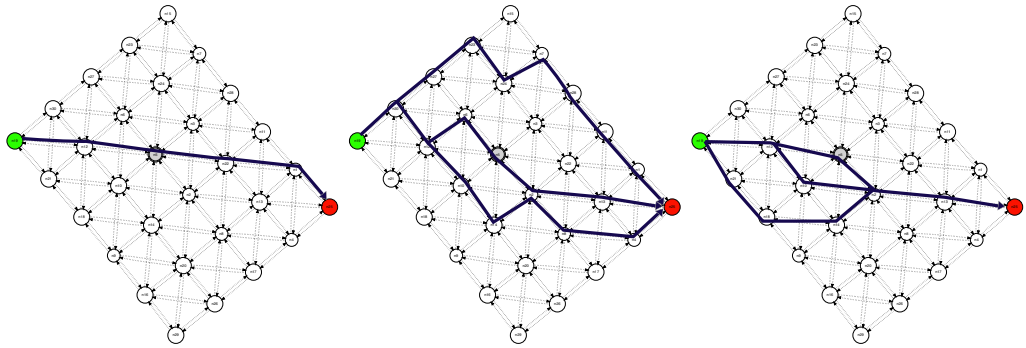
Os resultados do Exp1 podem ser observados na Figura 3. Verifica-se que houve convergência da rede para todas as topologias utilizadas. Além disso, não há um crescimento acentuado entre a quantidade de nós e o tempo de convergência. É importante também salientar que todas as topologias foram geradas de forma aleatória mas para um mesmo espaço geográfico. Isso faz com que o grau de conectividade nas topologias de maior densidade de nós seja maior, o que justifica o decrescimento do tempo de convergência nas topologias de 40 nós.

### 3.1.4. Experimento 2 - Dispersão dos fluxos na WMN

O Experimento 2 (Exp2) visa verificar como diferentes fluxos de dados podem percorrer diferentes caminhos na WMN, em função do uso dos diferentes algoritmos de definição de peso dos arcos. São usados os algoritmos HC, HLRB e HLRB-SHC citados anteriormente. O mesmo ambiente de emulação do Exp1 foi utilizado. Uma única topologia com 30 MNs foi utilizada. Para cada algoritmo de peso, são disparados de um mesmo nó fonte (n19) cinco tráfegos que representam fluxos de dados distintos, pela variação da porta UDP fonte. Todos os fluxos de dados são endereçados a um mesmo nó destino (n25).

A Figura 4 apresenta a dispersão dos fluxos através da rede. O primeiro e mais importante ponto a destacar é que com o uso do *framework* é possível encaminhar os fluxos baseado em diferentes métricas implementadas no módulo algoritmos de roteamento. É possível ainda perceber que, assim como esperado, o algoritmo HC concentrou todos os fluxos no caminho com o menor número de saltos, provocando assim, saturação nos

[5]Site do OpenWiMesh: `http://grade.dcc.ufba.br/OpenWiMesh`



(a) Dispersão usando o HC (b) Dispersão usando o HLRB (c) Dispersão usando o HLRB-SHC

**Figura 4. Dispersão dos fluxos**

arcos utilizados. O algoritmo HLRB-SHC, por sua vez, tentou dispersar o tráfego pelos menores caminhos. Com isso, apesar da dispersão inicial nos primeiros enlaces, há uma aglutinação do tráfego nos arcos próximos ao destino, o que também levou a saturação dos mesmos. Já o algoritmo HLRB, por utilizar uma métrica baseada apenas na banda residual, conseguiu dispersar o tráfego por um número maior de caminhos, diminuindo a saturação dos enlaces, porém com um custo adicional de encaminhar o tráfego por caminhos mais longos.

## 4. Trabalhos relacionados

A proposta de estender o modelo SDN para o ambiente de redes sem fio teve origem no trabalho de [Yap et al. 2010], que propôs o *OpenRoads*, um *framework* que permite a pesquisadores implementarem diferentes algoritmos e avaliá-los em uma rede sem fio com suporte a virtualização. O *OpenRoads* fez uso da tecnologia SDN Openflow. Para demonstrar o potencial da proposta, foram criadas e comparadas diferentes estratégias de gerenciamento de mobilidade, uma para cada fatia da rede. Não está contemplado no *OpenRoads* o suporte a redes sem fio de múltiplos saltos, redes *mesh*.

O trabalho de [Dely et al. 2011] foi o primeiro a integrar Openflow e WMNs. A arquitetura proposta considera duas redes sem fio virtuais, implementadas através da funcionalidade de múltiplos SSIDs, sendo a primeira rede usada exclusivamente para a sinalização de controle do Openflow (sinalização *out-of-band*) e a segunda usada para os dados em si. A WMN usada para sinalização Openflow foi baseada no padrão IEEE 802.11 e no protocolo *mesh* OSLR. A arquitetura foi validada com a implementação e avaliação de uma solução para mobilidade em um *testbed* real chamado KAUMesh.

Já o trabalho de [Moraes et al. 2013] aplica, de forma prática e simplificada, os princípios de SDN em redes sem fio de múltiplos saltos e compara seu desempenho com uma rede *mesh* convencional, implementada com protocolo *mesh* HWMP (IEEE 802.11s). Neste trabalho, é adotada a sinalização Openflow *out-of-band* e a configuração da tabela de fluxos é manual, inviabilizando o emprego de técnicas de engenharia de tráfego. Em verdade, devido à configuração estática das regras Openflow, o canal de sinalização na prática não é utilizado.

Por fim, alguns desafios na integração de Openflow e WMNs são relatados no

trabalho de [Chung et al. 2012]. Um dos desafios dessa integração é justamente a forma de implementação do canal de controle (*in-band* versus *out-of-band*). Além do ambiente de avaliação reduzido, uma das limitações daquele trabalho é o uso de uma topologia em árvore para conexão dos nós, sem fazer uso dos múltiplos caminhos providos pela WMN.

## 5. Conclusões e trabalhos futuros

As redes *mesh* sem fio (WMN) são conhecidas por sua larga aplicabilidade o que demanda destas alto grau de flexibilidade no seu controle. Entretanto, os protocolos de roteamento atualmente empregados são conhecidamente pouco flexíveis e de dificil extensão no que tange a suas estratégias de roteamento e encaminhamento.

A integração de SDN e WMN é um campo altamente promissor mas ainda apresenta diversos desafios. Alguns destes, bastante complexos e sequer explorados na literatura. Este é o caso da sinalização *in-band*, aspecto fundamental para a implementação eficiente de WMN no paradigma SDN.

Este artigo propôs um *framework* para controle e engenharia de tráfego em redes *mesh* sem fio, denominado OpenWiMesh, que adiciona flexibilidade e programabilidade nestas redes através do paradigma de redes definidas por software. Esta adição foi feita de forma eficiente com o estabelecimento de canais de controle das redes SDN através de sinalização *in-band* sem necessidade de duplicação de recursos.

Adicionalmente, foram implementadas ferramentas para determinação e visualização automática de topologia e representação numa estrutura de grafo centralizado no controlador da rede. Esta estrutura permite que um utilizador do *framework* possa definir novos algoritmos ou estratégias de roteamento com base no grafo da rede ou por meio de métricas não técnicas. Uma prova de conceito foi apresentada demonstrando a viabilidade da proposta. Resultados mostram que expectativas foram plenamente atingidas, através do controle efetivo dos fluxos na rede. Além disso, é importante mencionar que o referido *framework* tem sido utilizado como base para o desenvolvimento de outros estudos envolvendo controle de redes WMN.

Como trabalhos futuros pretende-se implementar novas técnicas de adição dos nós, através de novos métodos de configuração IP. Além disso, almeja-se a realização de outros experimentos comparando o desempenho do *framework* ora proposto com o de algoritmos distribuidos convencionais avaliando-se a sobrecarga de sinalização das duas propostas.

## Referências

[Ahrenholz 2010] Ahrenholz, J. (2010). Comparison of core network emulation platforms. In *MILITARY COMMUNICATIONS CONFERENCE, 2010-MILCOM 2010*, pages 166–171. IEEE.

[Akyildiz and Wang 2009] Akyildiz, I. and Wang, X. (2009). *Wireless mesh networks*, volume 3. John Wiley & Sons.

[Chung et al. 2012] Chung, J., Gonzalez, G., Armuelles, I., Robles, T., Alcarria, R., and Morales, A. (2012). Experiences and challenges in deploying openflow over a real wireless mesh network. In *Communications (LATINCOM), 2012 IEEE Latin-America Conference on*, pages 1–5. IEEE.

[Dely et al. 2011] Dely, P., Kassler, A., and Bayer, N. (2011). Openflow for wireless mesh networks. In *20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6. IEEE.

[Hagberg et al. 2008] Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*.

[Jacquet et al. 2001] Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., and Viennot, L. (2001). Optimized link state routing protocol for ad hoc networks. In *IEEE International Multi Topic Conference (IEEE INMIC). Technology for the 21st Century. Proceedings*.

[Lee and Gerla 2001] Lee, S.-J. and Gerla, M. (2001). Split multipath routing with maximally disjoint paths in ad hoc networks. In *IEEE International Conference on Communications (ICC)*.

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. In *ACM SIGCOMM Computer Communication Review*.

[Mo and Bostian 2005] Mo, T. and Bostian, C. (2005). A throughput optimization and transmitter power saving algorithm for IEEE 802.11 b links. *IEEE Wireless Communications and Networking Conference*, 1:57–62.

[Moraes et al. 2013] Moraes, M., Pinheiro, B., Nascimento, V., and Abelém, A. (2013). Redes sem fio de multiplos saltos definidas por software. In *IV Workshop de Pesquisa Experimental da Internet do Futuro*.

[NOXRepo 2013] NOXRepo (2013). About POX. `http://www.noxrepo.org/pox/about-pox/`. Accessed: 2013-03-26.

[Pfaff et al. 2009] Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., and Shenker, S. (2009). Extending networking into the virtualization layer. In *Hotnets*. URL: `http://openvswitch.org/`. Accessed: 2013-03-26.

[Ramachandran et al. 2006] Ramachandran, K. N., Belding, E. M., Almeroth, K. C., and Buddhikot, M. M. (2006). Interference-aware channel assignment in multi-radio wireless mesh networks. In *IEEE International Conference on Computer Communications (INFOCOM)*.

[Standards Committee 2012] Standards Committee, I. L. (2012). IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE-SA Standards Board*.

[Villegas et al. 2007] Villegas, E. G., Lopez-Aguilera, E., Vidal, R., and Paradells, J. (2007). Effect of adjacent-channel interference in IEEE 802.11 WLANs. *2nd International Conference on Cognitive Radio Oriented Wireless Networks and Communications - CrownCom 2007*, pages 118–125.

[Yap et al. 2010] Yap, K.-K., Kobayashi, M., Sherwood, R., Huang, T.-Y., Chan, M., Handigol, N., and McKeown, N. (2010). Openroads: Empowering research in mobile networks. *ACM SIGCOMM Computer Communication Review*, 40(1):125–126.

# Uma Arquitetura Elástica para Prevenção de Intrusão em Redes Virtuais usando Redes Definidas por Software*

**Antonio Gonzalez Pastana Lobato,**
**Ulisses da Rocha Figueiredo,**
**Martin Andreoni Lopez,**
**Otto Carlos Muniz Bandeira Duarte**

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{antonio,ulisses,martin,otto}@gta.ufrj.br

***Resumo.*** *A técnica de virtualização de redes permite a coexistência de diversas redes em um mesmo substrato físico, atendendo a diferentes requisitos. Contudo, a virtualização de redes ainda apresenta grandes desafios de segurança e gerenciamento de recursos. Este artigo propõe uma arquitetura elástica para prevenção de intrusão em redes virtuais. A proposta combina um Sistema de Detecção de Intrusão, que identifica e alerta eventos anômalos, com as Redes Definidas por Software, que oferecem uma maior programabilidade para bloquear ataques. A elasticidade é obtida ao se criar ou destruir dinamicamente instâncias de detecção de intrusão. As propriedades das Redes Definidas por Software são também usadas para distribuir ou agrupar o tráfego entre as instâncias de detecção de intrusão.*

***Abstract.*** *The technique of network virtualization allows multiple networks to coexist on the same physical substrate, catering to different requirements. However, network virtualization still presents major challenges in security and resources management. This paper proposes an elastic architecture for intrusion prevention in virtual networks. The proposal combines an Intrusion Detection System, which identifies and alerts anomalous events, with Software Defined Networks, which offer greater programmability to block attacks. Elasticity is achieved by dynamically creating or destroying instances of intrusion detection. The Software Defined Networks are also used to distribute or cluster traffic between the instances of intrusion detection.*

## 1. Introdução

Os ambientes virtualizados permitem compartilhar os recursos das máquinas físicas em diversas máquinas virtuais. Porém, esses ambientes apresentam diversos desafios como garantir o isolamento entre as máquinas virtuais e prover a segurança no cenário de redes virtuais [Pearce et al. 2013, Ray and Schultz 2009]. O isolamento nas redes virtuais significa que uma rede não afeta as outras que executam na mesma infraestrutura [Wu et al. 2010, Mattos and Duarte 2012]. Além desses novos desafios, problemas tradicionais relacionados com a segurança da Internet continuam existindo em ambientes virtualizados.

*Este trabalho foi realizado com recursos da CNPq, CAPES, FINEP, FUNTTEL e FAPERJ.

Mais de 60% dos ataques à segurança da rede é de origem interna [Lynch 2006]. Uma das formas de se proteger desses ataques internos é monitorar o tráfego em busca de atividades maliciosas ou violação de políticas. Para realizar o monitoramento de pacotes da rede, a ferramenta mais utilizada é um Sistema de Detecção de Intrusão (*Intrusion Detection System*-IDS) que realiza o monitoramento passivo dos pacotes na rede. Porém, esse tipo de análise não permite que sejam tomadas ações para prevenir que atividades maliciosas ocorram, portanto, se torna necessário um mecanismo que seja não só capaz de monitorar o fluxo de pacotes, mas também possa agir ativamente na rede de forma a bloquear o ataque, logo, um Sistema de Prevenção de Intrusão (*Intrusion Prevention System*-IPS) [Ruohomaa and Kutvonen 2005]. Os Sistemas de Detecção e Prevenção de Intrusão (IDPS) utilizam a técnica de Inspeção Profunda de Pacotes (*Deep Packet Inspection*-DPI) para realizar a análise de pacotes, consumindo recursos nesse processo.

Atualmente, para que um Sistema de Prevenção de Intrusão possa atuar sobre o tráfego da rede, é necessário que ele seja colocado diretamente na rota do pacote. Esta abordagem acaba introduzindo problemas, como, por exemplo: latência, já que cada pacote deverá ser processado individualmente antes de ser encaminhado para seu destino; acurácia, pois o IPS só consegue ver os pacotes do enlace onde ele está localizado, então, por desconhecer o comportamento dos demais tráfegos da rede, ele pode classificar uma atividade como sendo maliciosa de forma equivocada; flexibilidade, visto que não são instaladas regras nos comutadores de forma dinâmica em relação ao bloqueio do tráfego, o que pode afetar fluxos legítimos da rede.

Este artigo apresenta uma arquitetura elástica de detecção e prevenção de intrusão que realiza a detecção fora da rota do pacote. Para isso, as características de um ambiente virtualizado são utilizadas, o que permite uma rede lógica independente da rede física. A abordagem proposta no artigo é usar Redes Definidas por *Software* (*Software Defined Networks*-SDN), cujo protocolo mais utilizado é o OpenFlow [McKeown et al. 2008]. No protocolo OpenFlow, o controlador é centralizado e pode se comunicar com os comutadores da rede, portanto é capaz de analisar todo tráfego que passa por estes nós virtuais, podendo tomar ações sobre todos os fluxos de pacotes de forma dinâmica, conforme os alarmes gerados pelo Sistema de Detecção de Intrusão.

A arquitetura proposta utiliza o protocolo OpenFlow para monitorar o tráfego da rede, espelhando os pacotes para uma máquina virtual que inspeciona estes pacotes individualmente. A ferramenta de análise de tráfego utilizada é o Bro [Paxson 1999], que gera um alarme caso julgue o fluxo como malicioso. Para que este fluxo afete o mínimo do desempenho da rede, o controlador OpenFlow toma medidas para bloquear este fluxo o mais perto possível de sua origem. Desta forma, os pacotes são encaminhados tanto para seu destino quanto para a máquina que inspeciona os pacotes, que não está localizada na rota do fluxo.

Se existir um cenário de sobrecarga de uma máquina de detecção, parte do tráfego pode deixar de ser analisado o que acarretaria em um risco para a rede. Portanto, o artigo propõe o uso de uma ferramenta de monitoramento das máquinas que realizam DPI, para fornecer recursos de acordo com a demanda de forma elástica. Se houver sobrecarga, uma mensagem é enviada ao sistema de gerência da rede, que cria uma réplica da máquina de DPI. O controlador OpenFlow é então informado para que seja instaurada uma nova regra para a distribuição dos fluxos espelhados. Caso após algum tempo o tráfego diminua e não

sejam mais necessárias tantas máquinas de DPI, um outro alerta é enviado para que estas máquinas sejam desativadas, garantindo assim um melhor aproveitamento de recursos.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados na literatura; a Seção 3 apresenta os modelos de virtualização usados na proposta; na Seção 4 são apresentados os modos de operação para Sistemas de Detecção de Intrusão na rede; a Seção 5 apresenta a arquitetura proposta e a Seção 6 trata dos resultados do protótipo implementado; por último, a Seção 7 conclui o artigo e apresenta os trabalhos futuros.

## 2. Trabalhos Relacionados

Outras propostas que visam o uso de Redes Definidas por *Software* para a implementação de aplicações de segurança em ambientes virtualizados são apresentadas nesta seção. Estas propostas se servem da flexibilidade provida pelo paradigma da separação de planos para realizar a interação entre os alarmes e os comutadores. Shin *et al.* propõem uma nova linguagem de alto nível baseada em OpenFlow específica para criação de aplicações para segurança de forma modular [Shin et al. 2013]. Kim *et al* apresentam uma abordagem baseada nos estados da rede, que usa a alta programabilidade oferecida por SDN para decidir que aplicações executar dependendo do evento [Kim et al. 2013], nele a linguagem Pyretic, que modulariza funções de rede, é utilizada.

OpenSAFE [Ballard et al. 2010] propõe um sistema para redirecionamento de fluxos para aplicações de monitoramento e segurança. OpenSAFE também especifica uma linguagem para um gerenciamento simplificado de fluxos em aplicações de segurança. O OpenSAFE se baseia no protocolo OpenFlow para implementar funções básicas para manipulação de fluxos, como por exemplo filtros, que selecionam fluxos específicos e também desvios, para encaminhar o tráfego para aplicações de monitoramento e segurança. Há ainda um componente OpenFlow que interpreta as políticas definidas pela linguagem de gerenciamento de fluxo para que as ações sejam realizadas.

Especificamente para um Sistema de Prevenção de Intrusão (*Intrusion Prevention System*–IPS), Nagahama *et al.* realizam um estudo de caso para implementação de um IPS baseado em OpenFlow com uma captura seletiva de fluxos para o módulo de detecção de intrusão [Nagahama et al. 2012]. Porém, não foram consideradas possíveis sobrecargas neste módulo, o que poderia acarretar em não analisar parte do tráfego. Além disso, como a captura é seletiva, nem todos os fluxos passam por este módulo de inspeção, que caso mal configurado pode gerar uma falha de segurança.

O SnortFlow [Xing et al. 2013] é outra ferramenta de detecção de intrusão, onde é utilizado o analisador de tráfego Snort [Roesch et al. 1999], localizado no Domínio 0 do hipervisor Xen [Egi et al. 2008], para realizar a inspeção de pacotes da rede. Mehra compara o desempenho das ferramentas de detecção de intrusão Bro e Snort, sendo que a primeira apresenta um melhor desempenho na inspeção de pacotes [Mehra 2012]. Ao se instalar o Snort no Domínio 0, conectado a um comutador ligado às máquinas virtuais de uma máquina física, o Snort só consegue analisar o tráfego dessas máquinas virtuais. Para uma visão global da rede seria necessário alguma forma de comunicação e sincronismo, que não é apresentada no artigo do SnortFlow. Na arquitetura proposta, a máquina com o Bro consegue ver todos os pacotes da rede, visto que o controlador desvia todo o tráfego da rede para a máquina de análise de pacotes. Caso haja réplicas da máquina de DPI, o

controlador distribui os fluxos considerando sua origem para facilitar a detecção de um possível atacante.

## 3. Modelos de Virtualização

A proposta consiste em um ambiente híbrido de virtualização. Para criar esse ambiente virtual foram utilizadas a virtualização de máquinas provida pelo Xen e a comutação de fluxos programável provida pelo OpenFlow. A vantagem da utilização do Xen é a criação de máquinas virtuais isoladas entre si e com um controle distribuído. Essa vantagem possibilita que sejam desenvolvidas novas aplicações para o gerenciamento da infraestrutura virtual. A vantagem da utilização do protocolo OpenFlow em um comutador programável é que a migração de fluxos é feita de forma simplificada e, consequentemente, também é possível a migração de enlaces virtuais, o que permite o remapeamento de topologias lógicas sobre a topologia física. Na proposta, também são coletadas estatísticas de consumo de recursos das máquinas geradas pelo Xen para o monitoramento do ambiente virtual.

### 3.1. A Plataforma de Virtualização de Máquinas Xen

O Xen [Egi et al. 2008] é uma plataforma de virtualização de computadores pessoais. Essa plataforma de virtualização é usada em servidores de centro de dados (*datacenters*) em empresas como a *Amazon*, onde a virtualização do Xen teve seu desempenho avaliado [Wang and Ng 2010]. A arquitetura do Xen se baseia em uma camada hipervisora de virtualização logo acima do *hardware*, que monitora o comportamento das máquinas virtuais, como mostrada na Figura 1. As máquinas virtuais são executadas sobre o hipervisor, que garante um acesso independente a recursos como CPU, memória, rede e disco.

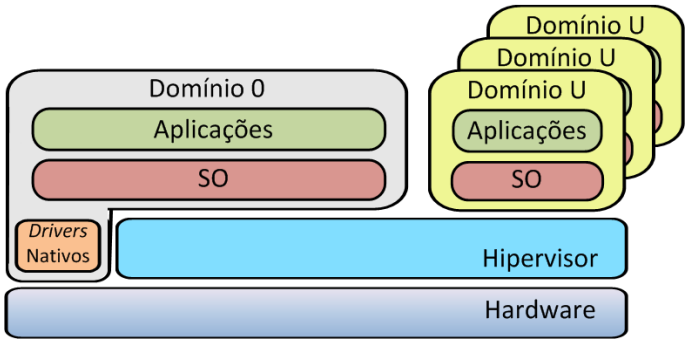



**Figura 1. Arquitetura Xen com a camada hipervisora e a separação entre o domínio privilegiado e os demais domínios.**

No Xen, cada ambiente virtual está isolado dos demais, ou seja, o funcionamento de uma máquina virtual não afeta o de outras. Além disso, permite que essas máquinas virtuais possuam sistemas operacionais distintos. O Xen possui um ambiente virtual privilegiado, o Domínio 0, que executa as operações de gerência do hipervisor e detém a exclusividade no acesso aos dispositivos físicos, como os dispositivos de Entrada/Saída (E/S). Desta forma, as máquinas virtuais só conseguem acesso aos dispositivos físicos através de *drivers* virtuais que se comunicam com o Domínio 0 ou através de uma porta da máquina física dedicada à máquina virtual. Outra solução seria utilizar um hardware específico, porém de alto custo.

### 3.2. Monitoramento de Recursos

Em ambientes virtuais é possível gerenciar a alocação de recursos de acordo com a demanda ou de acordo com políticas pré-estabelecidas. Essa gerência é obtida através do remapeamento das instâncias virtuais nas diversas máquinas físicas, de acordo com a quantidade de recursos disponíveis e as políticas estabelecidas [Carvalho and Duarte 2012]. Portanto, monitorar o consumo de recursos de máquinas virtuais e físicas permite a criação de métricas para alocação das máquinas virtuais de acordo com um objetivo estabelecido. Um exemplo disso é a computação em nuvem, na qual os provedores de serviço gerenciam de forma dinâmica os recursos fornecidos aos clientes de forma a entregar os recursos contratados com o menor custo possível.

Na proposta, o monitoramento foi feito a partir da coleta de dados do Xen através da Interface de Programação de Aplicação *Libvirt* [Bolte et al. 2010]. A *Libvirt* tem acesso às informações do hipervisor através de uma conexão segura e permite que essas informações sobre o consumo das máquinas físicas e virtuais possam ser usadas em diversas linguagens de programação.

### 3.3. O Comutador de Fluxos OpenFlow

A abordagem das Redes Definidas por Software (*Software Defined Networking* – SDN) permite o gerenciamento do tráfego de redes. Elas permitem a definição de fluxos de pacotes de forma dinâmica. O paradigma de SDN é o da separação de planos, em que a rede é dividida em plano de dados e plano de controle. O plano de controle é responsável pela definição da próxima rota de um fluxo e, para isso, executa algoritmos de controle da rede. Enquanto o plano de dados faz o encaminhamento dos pacotes de acordo com políticas definidas pelo controlador. Essa separação fornece a facilidade de que vários comutadores podem compartilhar o mesmo plano de controle, que detém a visão global da rede. Desta forma, é possível redefinir as rotas de fluxos de múltiplos comutadores apenas mudando a aplicação no controlador.

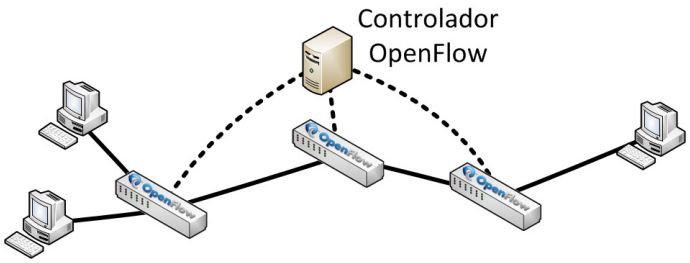


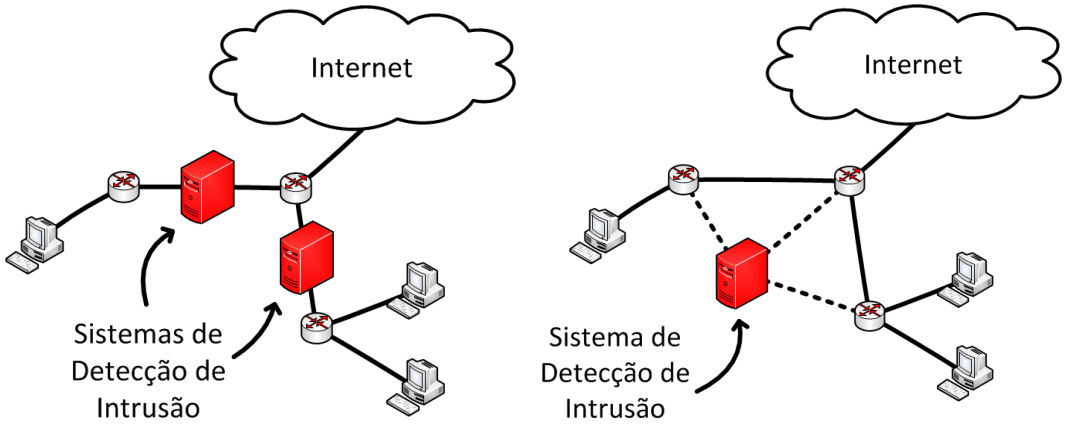

**Figura 2. Arquitetura OpenFlow na qual diversos comutadores se comunicam com o controlador para definição de fluxos.**

O protocolo OpenFlow [McKeown et al. 2008] define regras e padrões para SDN. Na arquitetura OpenFlow, existe uma tabela de fluxos nos comutadores que é atualizada pelo plano de controle. Quando chega um pacote de um fluxo que já está definido nessa tabela, ele é simplesmente comutado. Caso chegue um pacote de um fluxo não definido, ele é enviado para o controlador, onde é definido qual será o próximo destino desse pacote. O destino é escolhido de acordo com as aplicações sendo executadas e também de acordo com o cabeçalho do pacote. O controlador então adiciona essa regra na tabela de fluxo

do elemento encaminhador. É importante ressaltar que só o primeiro pacote de um fluxo não definido é encaminhado ao controlador, sendo a partir dele criada a regra na tabela de fluxo, o que faz que os demais pacotes desse fluxo sejam somente comutados, afetando pouco o desempenho da rede. A arquitetura OpenFlow está ilustrada na Figura 2.

## 4. A Análise de Tráfego e a Detecção de Intrusão na Rede

A análise de tráfego e a detecção de intrusão na rede são essenciais para a segurança. Deve ser ressaltado que a detecção de intrusão é mandatória para a detecção de ameaças provocadas por usuários já autenticados ou que conseguiram passar pelo *firewall*. O sistema de detecção de intrusão analisa pacotes para monitorar eventos que ocorrem na rede em busca de um comportamento suspeito que indique sinais de intrusão, que podem ser detectados por assinatura, quando o ataque é conhecido, ou por anomalia, quando a rede se comporta de uma forma diferente da habitual.




(a) Captura e análise na rota do pacote. (b) Captura e análise do pacote por espelhamento.

**Figura 3. A captura e análise dos pacotes para detecção de intrusão. A interceptação na rota só analisa os pacotes no enlace no qual a captura é realizada. Com a duplicação dos pacotes e o envio de mensagens, um único sistema de detecção de intrusão é capaz de analisar os pacotes de toda a rede.**

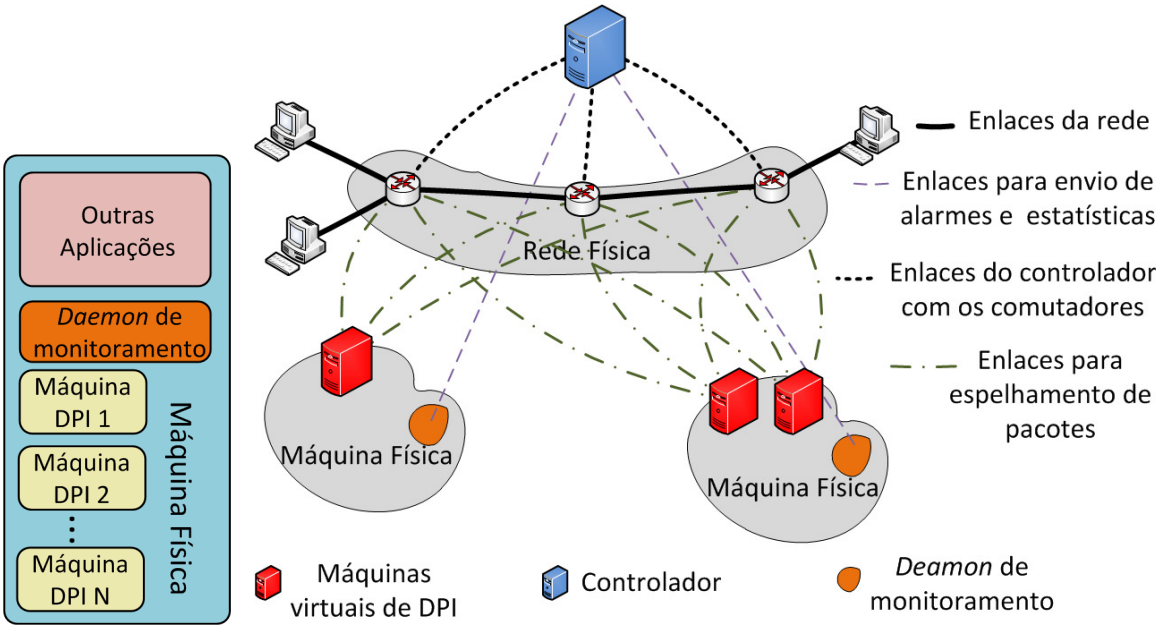
Um Sistema de Detecção de Intrusão (*Intrusion Detection System*-IDS) em rede pode atuar de dois modos distintos: na rota dos pacotes, no qual o IDS captura e analisa os pacotes e depois os encaminha para o próximo salto de sua rota; e fora da rota, na qual os pacotes são espelhados em um comutador, sendo encaminhados tanto para um Sistema de Detecção de Intrusão, quanto para o seu próximo destino na rota. Esses modos de operação estão ilustrados na Figura 3. Na arquitetura proposta, o modo utilizado foi o fora da rota do pacote, pois desta forma evita-se os problemas de latência devido ao processamento antes de encaminhar o pacote para seu destino, como acontece no modo na rota do pacote. No modo fora da rota, uma maior acurácia também é obtida, pois um único IDS fora da rota pode receber pacotes de diversos enlaces da rede. Enquanto na rota, o IDS estaria restrito aos pacotes no enlace em que ele está localizado.

## 5. Arquitetura Proposta

O artigo propõe uma arquitetura elástica para detecção de intrusão em redes virtuais, com alocação dinâmica de máquinas virtuais de análise de pacotes de acordo com a

demanda. Além da elasticidade de criação e destruição de máquinas para IDSs, a arquitetura provê um mecanismo de redefinição dos fluxos para balancear o tráfego direcionado às máquinas que realizam a detecção de intrusão e a tomada de ações em caso de alguma atividade maliciosa. A arquitetura de detecção de intrusão proposta consiste em três módulos principais, interligados, para controlar a rede: o módulo de detecção de intrusão, o módulo de gerenciamento de fluxo e o módulo de gerenciamento de recursos.

Nesta arquitetura, as máquinas físicas podem alocar máquinas virtuais de DPI e executar aplicações em paralelo, como mostrada na Figura 4(a). Além disso, apresentam um *daemon* para monitoramento dos recursos consumidos tanto pela máquina física quanto pelas máquinas de DPI.




(a) Máquina física com máquinas virtuais de DPI.

(b) Exemplo da arquitetura proposta e a comunicação entre seus elementos.

**Figura 4. Na arquitetura proposta, uma máquina física pode conter diversas máquinas de DPI, Figura 4(a). Em 4(b), pode ser visto o controlador ligado aos comutadores para a redefinição de fluxos levando em conta as estatísticas e alarmes das máquinas de DPI.**

Na Figura 4(b), está ilustrado um exemplo de uma rede que utiliza a arquitetura proposta para a detecção de intrusão. O tráfego da rede é espelhado nos comutadores para as máquinas de detecção de intrusão. Existe uma comunicação do controlador com as máquinas físicas para o envio de estatísticas sobre o consumo de recursos e alarmes em caso de atividades maliciosas.

### 5.1. Módulo de Detecção de Intrusão

Este módulo é composto por máquinas virtuais que executam uma ferramenta de detecção de intrusão por Inspeção Profunda de Pacotes (DPI). Essas máquinas analisam todo o tráfego da rede e geram alarmes para o controlador OpenFlow. Na proposta, a

ferramenta utilizada é o Bro. Como o Bro não pode executar em *multi-threading*, ele só usa um único núcleo da CPU, portanto, cada uma dessas máquinas virtuais foi configurada de forma a ter acesso a somente um núcleo, para evitar desperdício de recursos.

A comunicação deste módulo com o Módulo de Gerenciamento de Fluxo em caso de alarme é feita através de um canal seguro. Quando uma das máquinas de DPI detecta um ataque, um alarme é disparado contendo as informações sobre o fluxo malicioso, como sua origem, que são utilizadas para a posterior tomada de ações.

O Módulo de Detecção de Intrusão se adapta à demanda. O número de máquinas virtuais com Bro pode variar de acordo com a quantidade de fluxos da rede, sendo instanciadas dinamicamente. Desta forma, a arquitetura apresenta um comportamento elástico.

### 5.2. Módulo de Gerenciamento de Fluxo

Este módulo é responsável pela distribuição dos fluxos entre as máquinas com o Bro e também por tomar ações contra possíveis ataques alertados. O módulo é composto por uma aplicação do controlador OpenFlow POX, que se comunica tanto com o Módulo de Detecção de Intrusão, quanto com o Módulo de Gerenciamento de Recursos. O espelhamento de pacotes da rede para as máquinas de DPI é feita através de um túnel GRE *Generic Routing Encapsulation* e a análise dos pacotes é feita após o desencapsulamento, para garantir que estes pacotes não tenham sido alterados. Essa é uma das formas de se fazer o espelhamento e apresenta a vantagem que as máquinas virtuais podem estar em redes diferentes.

Caso o Módulo de Detecção de Intrusão possua mais de uma máquina DPI, a distribuição dos fluxos é feita levando em conta dois aspectos: a quantidade de recursos para análise de pacotes disponíveis em cada máquina virtual de DPI; e a origem dos pacotes. Um fluxo de uma origem nova é alocado na máquina de DPI que possuir o menor processamento no momento. Já os fluxos de mesma origem têm prioridade para serem alocados na mesma máquina de DPI, para evitar que um ataque desta origem passe despercebido. Ao distribuir os fluxos entre diversos IDSs, um ataque Negação de Serviço Distribuída (*Distributed Denial of Service* - DDoS) pode ter sua detecção dificultada. Entretanto, como o controlador OpenFlow determina todos os fluxos da rede, ele pode auxiliar na detecção desse ataque, como mostrado em [Braga et al. 2010].

A tomada de ações de bloqueio de fluxo é feita analisando a origem do fluxo malicioso. A origem é informada no alarme gerado pelo Módulo de Detecção de Intrusão. Assim, todos os fluxos provenientes desta origem já instalados nos comutadores OpenFlow são então listados e bloqueados. Além disso, uma outra regra é adicionada aos comutadores para que os futuros fluxos desta origem sejam bloqueados automaticamente. A regra para o bloqueio é instalada com um tempo de duração previamente determinado, deixando o atacante em quarentena. É importante listar os fluxos já instalados, pois no protocolo OpenFlow regras com correspondência mais completa tem prioridade sobre as demais. Caso contrário, a regra de apenas bloqueio da origem não surtiria efeito sobre os fluxos já existentes.

### 5.3. Módulo de Gerenciamento de Recursos

O modulo de gerenciamento dos recursos se encontra no domínio privilegiado, Domínio 0, de todas as máquinas físicas da rede. Nesse módulo são monitorados os con-

sumos de banda, de processamento e de memória da máquina física e também o quanto desses recursos é consumido por cada máquina virtual. Este monitoramento é feito através da coleta de dados do Xen realizadas pela *Libvirt*. Cada máquina física executa um *daemon* para coleta de dados e das estatísticas de todas as máquinas físicas da rede que são agregadas no controlador. Desta forma, o controlador tem a informação da quantidade de máquinas de DPI da rede e o consumo de recursos de cada uma.

Para evitar sobrecarga ou a presença de recursos ociosos, o sistema analisa principalmente as máquinas do Módulo de Detecção de Intrusão. Em caso de sobrecarga, este módulo analisa os recursos disponíveis nas máquinas físicas e decide então onde instanciar uma nova máquina de DPI. De maneira análoga, também é feito o monitoramento conjunto de todas as máquinas de DPI, para detectar quando for possível uma redistribuição de fluxos que permita a desativação de uma dessas máquinas, garantindo a elasticidade da proposta.

## 6. Resultados

Para validar a proposta da arquitetura elástica de detecção de intrusão, foi desenvolvido um protótipo na plataforma FITS[1] (*Future Internet Testbed with Security*) [Mattos et al. 2012] que é uma rede de testes interuniversitária para experimentação de propostas para a Internet do Futuro. O FITS é baseado nos mecanismos Xen e OpenFlow para prover uma arquitetura pluralista, permitindo a coexistência de múltiplas redes em paralelo executando aplicações diferentes. Na plataforma FITS o plano de controle executa nas máquinas virtuais Xen e o encaminhamento de pacotes é desempenhado pelo OpenFlow. A arquitetura está ilustrada na Figura 5, na qual os nós das redes são compostos por máquinas virtuais Xen isoladas agindo como comutadores OpenFlow.

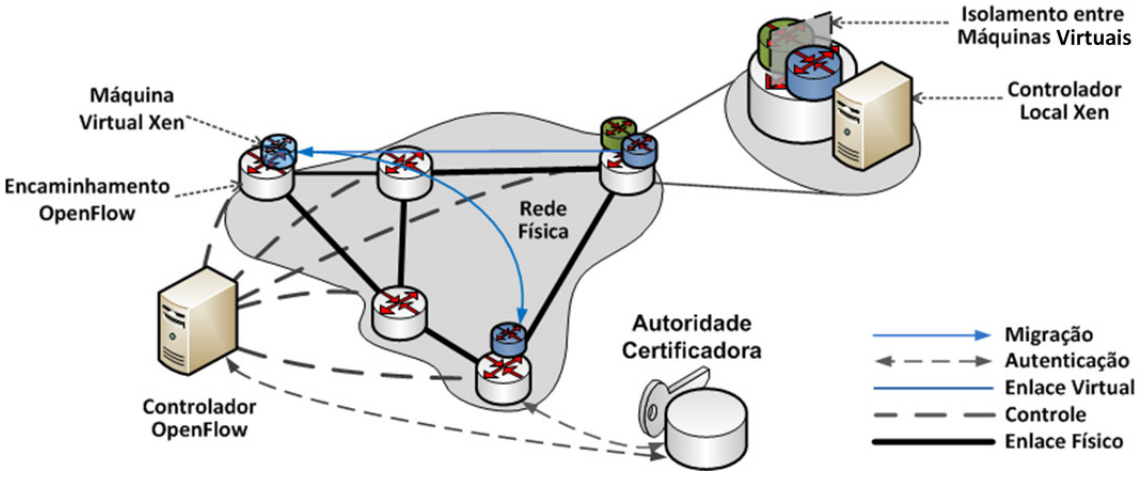



**Figura 5. A arquitetura da plataforma FITS: roteadores virtuais com isolamento executam nos roteadores físicos usando a plataforma de virtualização Xen e encaminhamento de dados usando OpenFlow.**

Foram realizados os seguintes experimentos para avaliar o correto funcionamento e o bom desempenho da arquitetura elástica de detecção de intrusão proposta: bloqueio

[1] www.gta.ufrj.br/fits

de um fluxo malicioso; avaliação dos recursos consumidos para análise de pacotes; e elasticidade de sobrecarga e descarga no Módulo de Detecção de Intrusão.

Os experimentos foram realizados em um servidor com processador Intel Xeon X5690 com 24 núcleos, com frequência de 3.47GHz de *clock* e com 48 GB de memória RAM. Para os testes foram geradas taxas de pacotes do tipo SYN entre máquinas clientes da rede para simular um ataque de negação de serviço por inundação. Ao passarem pelo comutador esses fluxos são espelhados para o Módulo de Detecção de Intrusão para análise.

### 6.1. Experimento de Bloqueio de Fluxo Malicioso

Este experimento analisa o efeito do bloqueio de um fluxo considerado malicioso sobre os demais fluxos da rede. O resultado das medidas está ilustrado na Figura 6 e mostra que quando o bloqueio do ataque acontece, o fluxo considerado legítimo não é afetado. Quando uma intrusão é detectada, o Módulo de Detecção de Intrusão manda um alerta para o controlador que atualiza a tabela de fluxos, bloqueando o fluxo malicioso em todos os comutadores OpenFlow da rede. Esta visão global da rede, característica das redes definidas por software, resulta em que o ataque seja bloqueado o mais perto possível de sua origem.

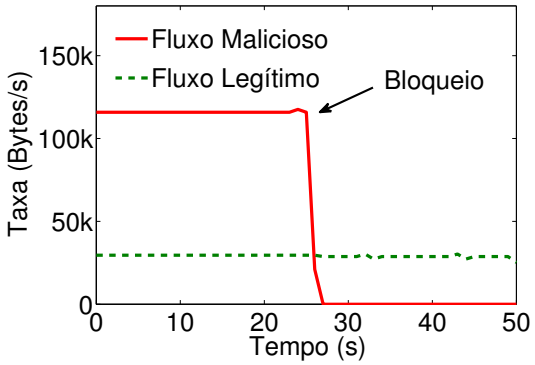


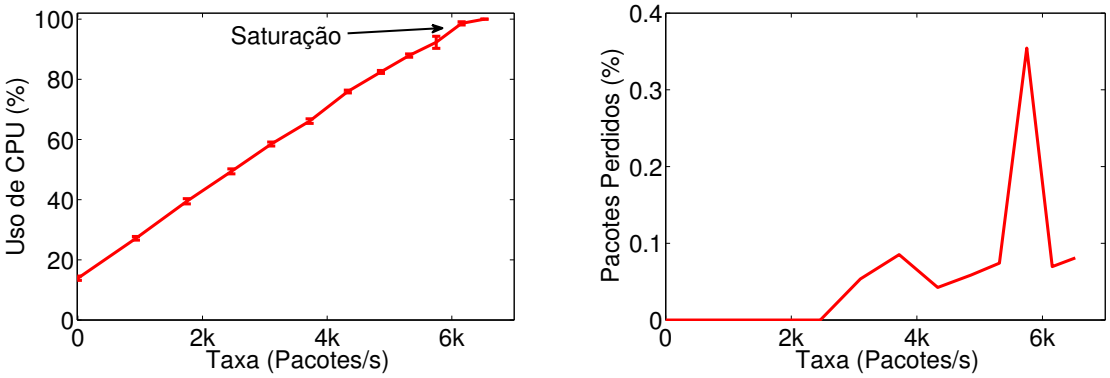

**Figura 6. Ação eficaz de bloqueio de um fluxo malicioso de negação de serviço por inundação sem afetar o tráfego legítimo.**

### 6.2. Avaliação dos Recursos Consumidos pelo Bro em uma Máquina Virtual

O Sistema de Detecção de Intrusão utilizado captura e analisa em tempo real os pacotes espelhados e, portanto, há uma sobrecarga associada a esta função. Assim, avaliou-se o consumo de recursos da análise de pacotes pelo Sistema de Detecção de Intrusão para se determinar qual dos aspectos, banda ou processamento, é o mais crítico para a detecção deste ataque de negação de serviço por inundação. É importante ressaltar que o processamento exigido pela análise dos pacotes depende da política de segurança e de que tipos de ameaças são analisadas. A técnica de Inspeção Profunda de Pacotes (*Deep Packet Inspection*–DPI) requer uma quantidade considerável de processamento. Neste experimento foram geradas taxas crescentes de pacotes e foram analisados tanto o processamento gasto pela máquina de DPI, quanto o percentual de pacotes perdidos pela máquina.

Conforme ilustrado na Figura 7, a máquina de DPI saturou o processamento antes de ter uma perda significativa de pacotes. No ponto crítico de processamento a perda

de pacotes foi menor que 0.5%. Portanto, para os demais experimentos, é possível definir o critério para saturação de uma máquina do Módulo de Detecção de Intrusão como consumo de CPU.




(a) Consumo de CPU da máquina de DPI para análise de pacotes.

(b) Percentual de pacotes perdidos pela máquina de DPI.

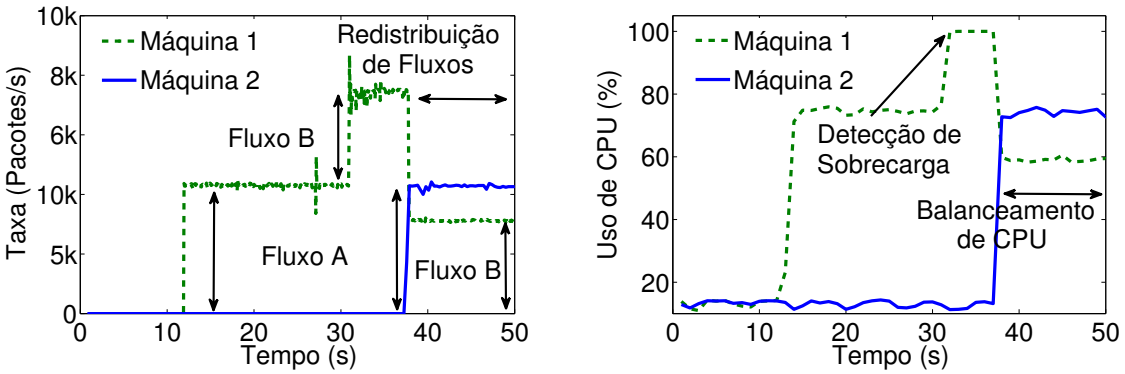
**Figura 7. Processamento e percentual de perdas de pacotes em uma máquina de DPI. Comparando as Figuras 7(a) e 7(b), percebe-se que não há perda significativa de pacotes para a taxa em que o processamento é saturado.**

### 6.3. Sobrecarga na Detecção de Intrusão

Este experimento visa avaliar a performance da arquitetura em caso de sobrecarga no Módulo de Detecção de Intrusão, que ocorre caso uma taxa elevada de pacotes precise ser analisada. Para este experimento, foram geradas taxas de pacotes constantes que eram inspecionados pela única máquina de DPI ativa no momento devido à baixa demanda. Em seguida, um novo fluxo foi criado, de forma a sobrecarregar esta máquina. Com isso, a máquina física onde a máquina de DPI estava localizada enviou uma mensagem de sobrecarga para o controlador. Neste experimento, a sobrecarga foi avaliada pelo processamento da máquina com o Bro, pois, como visto na Seção 6.2, o processamento satura antes da banda.

Quando o controlador recebe a mensagem de sobrecarga, ele executa o algoritmo de balanceamento para decidir em qual máquina física deve ser instanciada a nova máquina de detecção de intrusão. Após a ativação da nova máquina, os fluxos são redistribuídos levando em conta sua origem e os recursos de cada máquina de análise de pacotes.

Os resultados ilustrados na Figura 8 mostram que a partir do segundo fluxo ocorre uma sobrecarga na máquina de DPI. Quando a sobrecarga é detectada, existe um intervalo de tempo, devido à instanciação da nova máquina, até que os fluxos sejam redistribuídos. Após o balanceamento dos fluxos, todos os pacotes passam a ser analisados sem saturar o Módulo de Detecção de Intrusão.

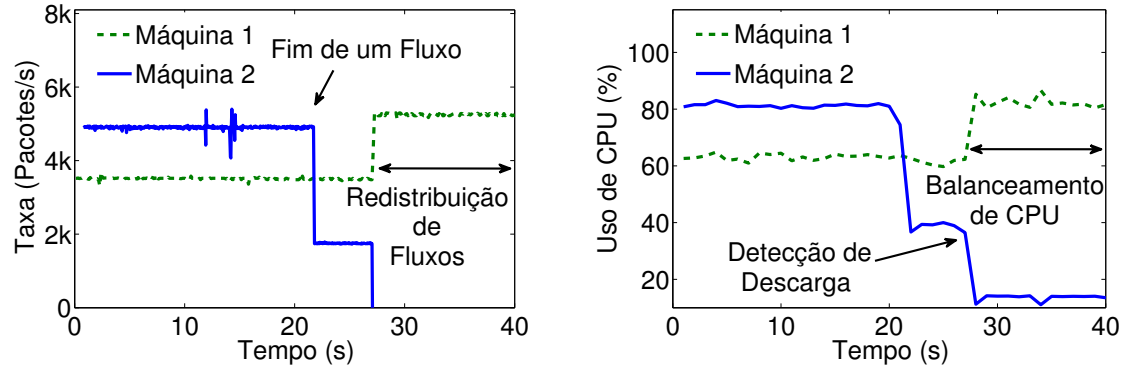



(a) Taxa de Pacotes recebida pelas máquinas de DPI. (b) Consumo de CPU de cada máquina de DPI.

**Figura 8. Análise temporal do Módulo de Detecção de Intrusão em caso de sobrecarga. São iniciados dois fluxos na máquina 1, Figura 8(a), causando sobrecarga de CPU, Figura 8(b). Para evitar a saturação da máquina 1, a máquina 2 é ativada e os fluxos são redistribuídos, balanceando o consumo de CPU.**

### 6.4. Descarga na Detecção de Intrusão

Este experimento foi feito para medir a elasticidade da arquitetura. Caso haja recursos ociosos, a proposta visa redistribuir os fluxos de forma que uma máquina possa ser desativada. Os recursos consumidos são constantemente analisados e informados ao controlador para que ele possa detectar quando um remanejamento de fluxos irá permitir a desativação de uma das máquinas do Módulo de Detecção de Intrusão.




(a) Taxa de Pacotes recebida pelas máquinas de DPI. (b) Consumo de CPU de cada máquina de DPI.

**Figura 9. Análise temporal do Módulo de Detecção de Intrusão em caso de descarga. Duas máquinas de DPI estão operando quando um dos fluxos termina, Figura 9(a). Então, o controlador analisa o consumo de CPU das duas máquinas, Figura 9(b), e redistribui os fluxos de forma a poder desativar uma das máquinas.**

O teste começa com duas máquinas virtuais de DPI recebendo taxas constantes de pacotes, como mostrada na Figura 9. Após um período de tempo, um desses fluxos foi cancelado, causando uma queda perceptível no processamento de uma das máquinas. Ao detectar esta descarga, o controlador redistribui os fluxos de forma que a máquina com menor processamento passe a receber nenhum fluxo e, portanto, possa ser desativada.

## 7. Conclusão e Trabalhos Futuros

O artigo propõe uma arquitetura elástica para Prevenção de Intrusão. Os resultados mostram que o processamento de CPU durante a análise de pacotes na detecção de intrusão é um ponto crítico conforme o aumento do tráfego. Desta forma, a arquitetura proposta analisa esse aspecto para que os recursos sejam fornecidos de maneira dinâmica, evitando que pacotes deixem de ser inspecionados. A proposta permite a instanciação e a desativação dinâmica de máquinas de DPI de forma que a inspeção de todos os pacotes da rede seja feita evitando recursos ociosos.

O comportamento da arquitetura proposta foi analisado para os casos de sobrecarga e descarga do Módulo de Detecção de Intrusão. No caso da sobrecarga, a distribuição dos fluxos entre duas máquinas de DPI, permitiu que todos os pacotes fossem analisados, já que ouve um balanceamento de CPU. No caso de descarga, o fluxo foi redistribuído de forma que uma máquina parasse de receber fluxos, o que permite a desativação dessa máquina de DPI. Portanto, a arquitetura fornece recursos de acordo com a demanda, ou seja, realiza a prevenção de intrusão de forma elástica.

A análise mais detalhada do comportamento da arquitetura proposta para o ataque de negação de serviço distribuída será realizada em um trabalho futuro. Além disso, pretende-se integrar a arquitetura ao FITS, para fornecer uma ferramenta de prevenção de intrusão elástica como serviço para novas propostas de experimentação para a Internet do Futuro.

## 8. Referências

[Ballard et al. 2010] Ballard, J. R., Rae, I., and Akella, A. (2010). Extensible and scalable network monitoring using opensafe. *Proc. INM/WREN*.

[Bolte et al. 2010] Bolte, M., Sievers, M., Birkenheuer, G., Niehörster, O., and Brinkmann, A. (2010). Non-intrusive virtualization management using libvirt. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 574–579. European Design and Automation Association.

[Braga et al. 2010] Braga, R., Mota, E., and Passito, A. (2010). Lightweight DDoS flooding attack detection using NOX/OpenFlow. In *IEEE 35th Conference on Local Computer Networks (LCN)*, pages 408–415.

[Carvalho and Duarte 2012] Carvalho, H. E. and Duarte, O. C. (2012). Voltaic: volume optimization layer to assign cloud resources. In *Proceedings of the 3rd International Conference on Information and Communication Systems*, page 3. ACM.

[Egi et al. 2008] Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Huici, F., and Mathy, L. (2008). Towards high performance virtual routers on commodity hardware. In *Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–12. ACM.

[Kim et al. 2013] Kim, H., Gupta, A., Shahbaz, M., Reich, J., Feamster, N., Clark, R., and Tech, G. (2013). Simpler network configuration with state-based network policies. *Georgia Institute of Technology, Atlanta, USA*.

[Lynch 2006] Lynch, D. M. (2006). Securing against insider attacks. *Information Systems Security*, 15(5):39–47.

[Mattos and Duarte 2012] Mattos, D. M. F. and Duarte, O. C. M. B. (2012). QFlow: Um sistema com garantia de isolamento e oferta de qualidade de serviço para redes virtualizadas. In *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2012*.

[Mattos et al. 2012] Mattos, D. M. F., Mauricio, L. H., Cardoso, L. P., Alvarenga, I. D., Ferraz, L. H. G., and Duarte, O. C. M. B. (2012). Uma rede de testes interuniversitária a com técnicas de virtualizacao híbridas. In *Salão de Ferramentas do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'12*.

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: in campus networksenabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*.

[Mehra 2012] Mehra, P. (2012). A brief study and comparison of snort and bro open source network intrusion detection systems.

[Nagahama et al. 2012] Nagahama, F. Y., Farias, F., Aguiar, E., Gaspary, L., Granville, L., Cerqueira, E., and Abelém, A. (2012). Ipsflow–uma proposta de sistema de prevenção de intrusão baseado no framework openflow.

[Paxson 1999] Paxson, V. (1999). Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463.

[Pearce et al. 2013] Pearce, M., Zeadally, S., and Hunt, R. (2013). Virtualization: Issues, security threats, and solutions. *ACM Computing Surveys (CSUR)*, 45(2):17.

[Ray and Schultz 2009] Ray, E. and Schultz, E. (2009). Virtualization security. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, page 42. ACM.

[Roesch et al. 1999] Roesch, M. et al. (1999). Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238.

[Ruohomaa and Kutvonen 2005] Ruohomaa, S. and Kutvonen, L. (2005). Trust management survey. In *Trust Management*, pages 77–92. Springer.

[Shin et al. 2013] Shin, S., Porras, P., Yegneswaran, V., Fong, M., Gu, G., and Tyson, M. (2013). Fresco: Modular composable security services for software-defined networks. In *Proceedings of Network and Distributed Security Symposium*.

[Wang and Ng 2010] Wang, G. and Ng, T. E. (2010). The impact of virtualization on network performance of amazon ec2 data center. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

[Wu et al. 2010] Wu, H., Ding, Y., Winer, C., and Yao, L. (2010). Network security for virtual machine in cloud computing. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pages 18–21. IEEE.

[Xing et al. 2013] Xing, T., Huang, D., Xu, L., Chung, C.-J., and Khatkar, P. (2013). Snortflow: A openflow-based intrusion prevention system in cloud environment. In *Research and Educational Experiment Workshop (GREE), 2013 Second GENI*, pages 89–92. IEEE.

# Aplicação de IA-RWA em Redes Ópticas Virtuais através de PCE implementado como aplicação SDN

**Giovanni C. Santos[1], Alberto T. Hirata[1], Marcos B. Trindade[1], Juliano R. F. Oliveira[1], Fabian N. C. van't Hooft[1], Marcos A. Siqueira[2], Christian E. Rothenberg[2], Júlio C. R. Oliveira[1]**

[1]Fundação CPqD – Campinas, SP – Brasil

[2]Faculdade de Engenharia Elétrica e Computação, Unicamp – Campinas, SP – Brasil

{gcuriel, ahirata, trindade, jrfo, fabian}@cpqd.com.br,
marcos.siqueira@gmail.com chesteve@dca.fee.unicamp.br,
julioc@cpqd.com.br

***Abstract.*** *As ROADMs manufacturing process evolved, it became possible to build optical switching nodes which are colorless, directionless and contentionless. These types of switching nodes increase significantly the possibilities of setting up new optical paths, allowing new connectivity services to be offered, such as transport network topology virtualization. One solution to deal with the higher complexity in control operations in such scenario is the SDN architecture, which allows the inclusion of new control features by means of "network applications". This paper presents one use-case of such applications in a SDN scenario by using a PCE capable of doing physical layer impairment analysis, such as total OSNR estimation.*

***Resumo.*** *Com o avanço da construção de ROADMs foi possível a montagem de nós de comutação WDM colorless, directionless e contentionless. Estes nós aumentam significativamente o número de possibilidades de configuração de caminhos, o que permite a oferta de novas classes de serviços de conectividade, como virtualização da topologia da rede de transporte. Uma solução para lidar com o aumento da complexidade no controle da rede é a arquitetura SDN, a qual permite a inclusão de novas funcionalidades de controle através de 'aplicações'. Este trabalho se propõe a apresentar uma forma de uso de um PCE como aplicação em um cenário SDN, sendo capaz de executar análises de restrições físicas, como estimativa de OSNR total.*

## 1. Introdução

A evolução das redes ópticas tem trazido vários desafios para os sistemas de controle e gerência, como suporte a evolução de topologias ponto-a-ponto para redes reconfiguráveis com ROADMs (*Reconfigurable Optical Add-Drop Multiplexer*) com suporte a arquiteturas CDC (*Colorless, Directionless, Contentionless*) e grade flexível, consideração de *transponders* coerentes com modulação adaptável, necessidade de novos serviços dinâmicos como virtualização de redes, etc. Tais requisitos promoveram o desenvolvimento de soluções como mecanismos para RSA (*Routing and Spectrum Assignment*) com suporte a restrições (largura de espectro, restrições de camada física, equalização óptica), desenvolvimento de plano de controle com suporte a unificação de camadas de tecnologia de transporte, mecanismos de otimização entre camadas, entre outros.

Uma das possibilidades que estas novas funcionalidades trazem é a virtualização da infraestrutura de rede. Com ela, é possível fornecer a um determinado elemento de controle da rede um subconjunto da infraestrutura completa de transporte já instalada, permitindo que este elemento possa configurá-la de forma isolada. Uma solução para tornar mais simples e mais robusta a virtualização de uma rede é a arquitetura SDN (*Software Defined Networking*), a qual pode ser definida como uma abordagem para a

separação dos planos de controle e de encaminhamento da rede através de uma interface simples e padronizada, como *OpenFlow* [Rothenberg 2011]. Esta arquitetura permite que o controle da rede seja simplificado, possuindo apenas um ponto para configuração, além de possibilitar a automação de operações a serem realizadas na rede através de interfaces programáveis, facilitando a implantação de novos serviços com base em funcionalidades de controle implementadas em software.

Este artigo apresenta uma evolução de um controlador SDN para redes ópticas projetado para permitir implementação de funções de automação e virtualização destas redes. A evolução mencionada e principal contribuição deste artigo consiste na implementação de um PCE (*Path Computation Element*) com suporte a IA-RWA (*Impairment-Aware Routing and Wavelength Assignment*) que é executado como uma aplicação SDN no controlador e na construção automática de redes ópticas virtuais com base na especificação de uma demanda. Este controlador implementa uma camada de abstração que atua como um sistema operacional de redes de transporte, ou T-NOS (*Transport Network Operating System*) permitindo aplicações SDN terem fácil acesso a informações topológicas, de configuração e desempenho, de forma que possam ter visões global da rede e local dos elementos de rede, permitindo que estas realizem variadas funções. Adicionalmente, a camada de abstração implementa o conceito de particionamento de rede (*network slicing*), permitindo a implementação de serviços de redes ópticas virtuais, ou VONs (*Virtual Optical Networks*). Com o objetivo de permitir interoperabilidade com redes legadas, foi proposta e validada a virtualização da operação de outro software de controle de redes baseado em GMPLS [Mannie 2004] (*Generalized Multi-Protocol Label Switching*) rodando como aplicação do controlador SDN.

Este artigo está organizado da seguinte forma: a seção 2 apresenta a arquitetura SDN proposta, a seção 3 apresenta o projeto do PCE como aplicação SDN, incluindo o detalhamento dos algoritmos IA-RWA, e a seção 4 apresenta o *testbed* experimental e a demonstração dos resultados obtidos. Na seção 5 são feitas as conclusões e na seção 6 são feitos os agradecimentos.

## 2. Arquitetura SDN

A arquitetura do sistema, ilustrada na Figura 1, é composta de elementos de rede O-NE (*Optical Network Elements*), os quais representam equipamentos físicos instalados na rede de transporte, e pelo controlador SDN, o qual implementa funcionalidades como particionamento de redes (*slicing*), visão abstrata e unificada da rede. Este controlador fornece uma interface padronizada para que aplicações SDN possam implementar funções de controle de rede e de serviços. Esta arquitetura é definida e detalhada no trabalho feito em [Siqueira 2013].

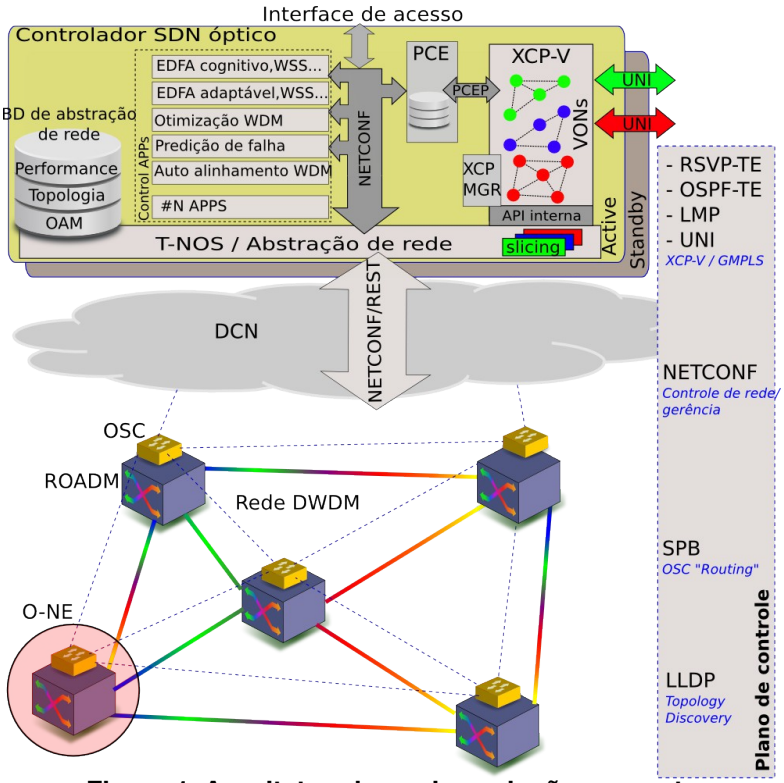



**Figura 1. Arquitetura base da evolução proposta**

Embora vários componentes tenham sido definidos nesta arquitetura, apenas alguns são relevantes a este trabalho. São eles:

- T-NOS / Abstração da rede: esta camada é responsável pelo fornecimento de APIs locais de controle dos equipamentos do plano de transporte. Uma vez que apenas esta interface possui acesso direto aos nós da rede, ela é capaz de controlar quais recursos um determinado elemento existente no controlador pode utilizar. Além disso, esta camada pode oferecer também diferentes visões (conjunto de recursos disponíveis) a elementos distintos, possibilitando a criação de VONs;
- XCP-V (*Extensible Control Plane – Virtualized*): neste componente estão presentes instâncias virtuais do software de plano de controle ASON/GMPLS, aqui denominado XCP, a fim de se ter suporte a redes que possuem esta tecnologia;
- PCE: este elemento é responsável pelo mecanismo de cálculo de rota para criação de caminhos na rede de transporte.

Mais detalhes sobre estes componentes no contexto deste trabalho serão dados nas próximas seções.

## 2.1. Sistema Operacional de Rede – T-NOS

O conceito de sistema operacional de rede da arquitetura mostrada na Figura 1 foi criado assumindo que os elementos de rede possuam um modelo baseado na linguagem YANG [Bjorklund 2010]. Este requisito foi definido pois, ao se modelar equipamentos de transmissão nesta linguagem e uma vez que eles ofereçam uma interface NETCONF [Enns 2011], é possível definir estruturas, relações e restrições aos componentes dos elementos de rede, como WSS (*Wavelength Selective Switch*), OCM (*Optical Channel Monitor*), amplificadores ópticos, comutadores *Multicast* (*MCS* –

*Multicast switch*), entre outros, de forma independente da interface de acesso definida pelo equipamento físico. Tal modelo pode ser exportado e/ou transformado em estruturas que representam tais conexões e restrições como, por exemplo, grafos de propriedades.

No escopo deste trabalho, são considerados ROADMs modelados em YANG de forma nativa. Estes equipamentos expõem interfaces REST/NETCONF ao T-NOS através de controladores, os quais fazem papel dos O-NEs. Estes elementos são mostrados na Figura 2.

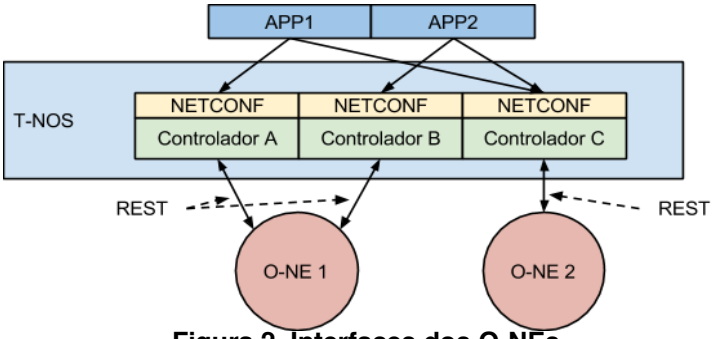



**Figura 2. Interfaces dos O-NEs**

Nesta figura são representados três controladores que representam dois equipamentos de rede. Estes controladores se comunicam com os equipamentos físicos através do protocolo REST e apresentam uma interface de configuração às aplicações através de NETCONF. É interessante notar que podem existir vários destes controladores acessando um mesmo equipamento, cada um deles configurado para permitir comandos referentes a um subconjunto dos recursos existentes no equipamento físico (tais como comprimentos de onda disponíveis ou portas de acesso). Desta forma, o T-NOS elabora um ponto de vista global da rede, concatenando um subconjunto das informações dos controladores em um modelo unificado.

Uma consequência importante do modelamento do T-NOS é a possibilidade de se escrever uma aplicação de controle que permita a consideração de toda a rede (sendo representada por um conjunto de controladores) como apenas um único equipamento. Neste equipamento abstrato, as portas de acesso seriam representações de O-NEs, enquanto a tarefa de comutação de tráfego e que permite comutar tráfego de um ponto de entrada a um outro ponto de saída.

## 2.2. Plano de Controle ASON/GMPLS

Com o objetivo de manter a interoperabilidade com redes legadas baseadas em plano de controle GMPLS e possibilitar a utilização de suas funcionalidades ainda não existentes para cenários SDN, a arquitetura do controlador possibilita a execução de instâncias do XCP em um ambiente virtual. Este ambiente permite que a interação do XCP com o T-NOS seja feita através da mesma interface que é fornecida pelos elementos de rede para comunicação com o XCP em um ambiente real. Assim, uma vez possuindo uma implementação de plano de controle GMPLS, não são necessárias grandes alterações na codificação deste software para que seja executado neste ambiente. Outras possibilidades que surgem com esta abordagem são: implementação de VONs com diferentes instâncias de XCP, diminuição da complexidade e de máquinas de estados nos elementos de rede, possibilidade de atualização de software do XCP sem alterações nos elementos de rede.

Um exemplo de uso do XCP no cenário proposto neste trabalho é mostrado na Figura 3.

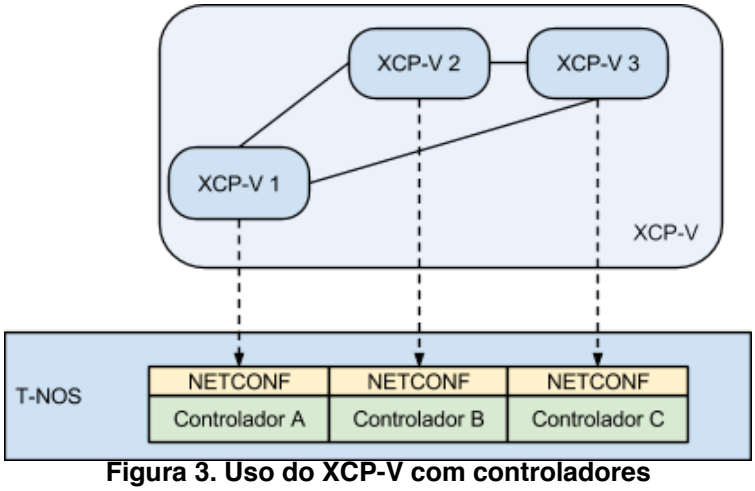



**Figura 3. Uso do XCP-V com controladores**

Nesta figura, são apresentadas três instâncias do XCP que configuram três controladores distintos oferecidos pelo T-NOS. Estes controladores podem representar o equipamento físico em sua totalidade ou não, dependendo da decisão de configuração da rede virtual empregada. Desta forma, as instâncias do plano de controle são capazes de operar em uma rede virtual sem a necessidade de modificações no código do plano de controle.

### 2.3. *Path Computation Element* - PCE

O PCE [Farrel 2006] apresenta uma arquitetura que permite que o mecanismo de cálculo de rota seja oferecido como um serviço prestado a outros elementos da rede. O protocolo empregado nesta arquitetura é o PCEP [Vasseur 2009], o qual considera todas as possibilidades descritas no documento de arquitetura do PCE para a codificação de suas mensagens. No entanto, vários itens importantes para a consideração de redes ópticas que devem estar presentes nestas mensagens ainda estão em processo de especificação [Lee 2013][Bernstein 2013][Bernstein 2013-2].

Um conceito envolvendo esta arquitetura que tem atraído atenção é o *Stateful PCE* [Crabbe 2013], o qual pode ser entendido como extensões tanto na arquitetura do PCE quanto do protocolo PCEP para que se possa ter conhecimento sobre a situação atual dos caminhos criados na rede (tanto para que se possa tomar melhores decisões durante o cálculo de rota quanto para possíveis melhorias em rotas que já estejam configuradas na rede de transporte). Este conceito pode fazer uso extenso das funcionalidades fornecidas pelo controlador utilizado neste trabalho já que todo o acesso às informações dos equipamentos na rede de transporte é feito localmente.

Na próxima seção serão apresentados detalhes sobre o funcionamento do PCE desenvolvido neste trabalho.

## 3. PCE como aplicação SDN

O PCE foi integrado ao cenário SDN apresentado da forma mostrada na Figura 4.

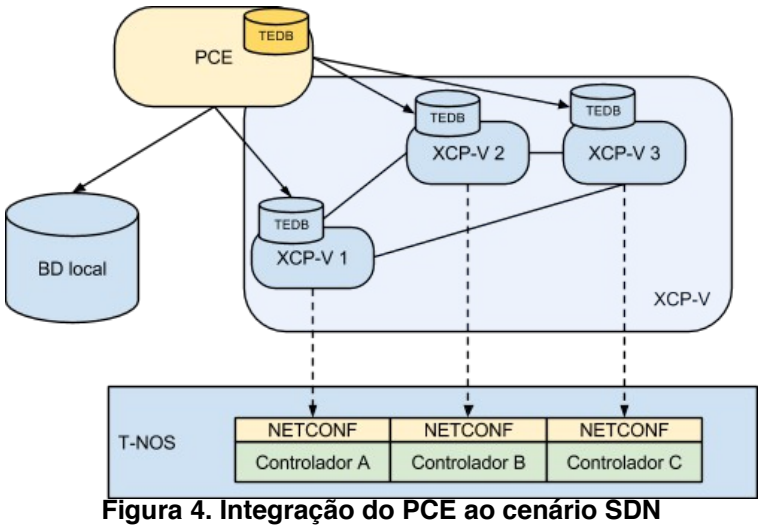



**Figura 4. Integração do PCE ao cenário SDN**

Nesta figura, são apresentadas as instâncias do plano de controle virtualizado, já apresentado anteriormente. Cada uma das instâncias possui sua própria base de dados de engenharia de tráfego, a qual é populada através da troca de mensagens do protocolo OSPF-TE, próprio da tecnologia GMPLS. Esta base de dados contém informações sobre nós da rede e seus enlaces, incluindo banda total disponível em cada enlace, lista de comprimentos de onda disponíveis, restrições de conectividade entre duas portas de um nó, etc. Existe ainda um banco de dados externo ao plano de controle virtualizado, também representado na figura, que contém informações mais detalhadas sobre características dos elementos da rede, não limitados apenas a nós e enlaces. Para este trabalho, estas informações se constituem em:

- Informações sobre elementos de amplificação instalados, como modelo, parâmetros de operação, etc.;
- Características dos elementos de amplificação, como figura de ruído e inclinação de canais para várias potências de entrada e ganhos configurados. O trabalho de extração e organização destes dados é feito em [Moura 2013];
- Associação entre estas informações e enlaces de engenharia de tráfego configuradas em instâncias do plano de controle.

Estas informações são importantes para o cálculo da estimativa de OSNR (*Optical Signal-Noise Ratio*), que será apresentado na próxima seção.

Foi decidido que o armazenamento destes dados seria em um banco de dados local por causa das características destas informações: são volumosas (principalmente no que se refere aos valores de figura de ruído e inclinação) e relativamente estáticas. Caso estas informações fossem colocadas no banco de dados de engenharia de tráfego presente em cada instância do XCP virtualizado, cada nó deveria disseminar e armazenar uma cópia destes dados, o que poderia trazer problemas como consumo excessivo e desnecessário de memória.

O PCE (enquanto elemento de software) também possui um banco de dados próprio que é preenchido por uma instância do OSPF-TE que é capaz de receber as mensagens geradas pelas instâncias virtualizadas do XCP. Quando uma requisição de cálculo de rota é recebida, o processo do PCE lê esta base de dados e procura na base de dados local informações extras sobre a topologia, adicionando-as às informações de cada enlace.

O procedimento de cálculo de rota considerado foi baseado no algoritmo apresentado em [Santos 2012]. O mecanismo inclui, além da execução do algoritmo de

cálculo de rota, analisadores que modificam a topologia a fim de modelar as restrições de conectividade apresentadas pelos nós da rede, possibilitando a consideração de nós baseados em ROADMs com arquiteturas direcionadas e coloridas [Nascimento 2012]. Neste trabalho, foi adicionado a estes analisadores a obtenção de valores que auxiliam a estimativa da OSNR de um caminho. Este mecanismo é apresentado na próxima seção.

### 3.1. Estimativa de OSNR

A análise de OSNR feita neste trabalho é baseada nas considerações feitas na recomendação [ITU G680], que analisa o impacto da OSNR devido a elementos ópticos instalados em cascata. A equação para o cálculo da OSNR, definida em termos logarítmicos (dB) na equação (1), é dada por:

$$OSNR_{out} = -10\log\left(10^{-\left(\frac{OSNR_{\text{in}}}{10}\right)} + 10^{-\left(\frac{P_{in1} - NF_1 - 10\times\log(h\,\nu\,\nu_r)}{10}\right)} + \ldots + 10^{-\left(\frac{P_{inN} - NF_N - 10\times\log(h\,\nu\,\nu_r)}{10}\right)}\right) \quad (1)$$

onde:

- $OSNR_{out}$ é o logaritmo, na base 10, da OSNR (dB) na porta de saída do elemento óptico;
- $OSNR_{\text{in}}$ é o logaritmo, na base 10, da OSNR (dB) na porta de entrada do elemento óptico;
- $P_{in1}, P_{inN}$ são as potências do canal (dBm) nas entradas dos amplificadores, ou de qualquer outro elemento óptico da rota em questão.
- $NF_{1,} NF_N$ são as figuras de ruído (dB) dos amplificadores, ou de qualquer outro elemento óptico da rota em questão.
- NF é a Figura de Ruído (dB) do elemento óptico;
- h é a constante de Planck;
- $\nu$ é a frequência óptica em Hz;
- $\nu_r$ é a largura de banda de referência em Hz (normalmente utiliza-se 0.1 nm).

A consideração da OSNR feita neste trabalho tem o intuito de utilizar o próprio mecanismo de cálculo de rota para que se maximize este parâmetro para um determinado caminho. Assim, o objetivo é transformar cada parcela do cálculo da OSNR em valores que podem ser somados linearmente, tornando-os custos de enlaces, de forma que a minimização desta soma signifique a maximização da OSNR. Neste sentido, ao se dividir ambos os termos da equação (1) por (-10) obtém-se:

$$\frac{OSNR_{out}}{-10} = \log\left(10^{-\left(\frac{OSNR_{\text{in}}}{10}\right)} + 10^{-\left(\frac{P_{in1} - NF_1 - 10\times\log(h\,\nu\,\nu_r)}{10}\right)} + \ldots + 10^{-\left(\frac{P_{inN} - NF_N - 10\times\log(h\,\nu\,\nu_r)}{10}\right)}\right) \quad (2)$$

A linearização desta equação pode ser obtida através do uso da seguinte transformação:

$$\log_{10} Y = X \Leftrightarrow 10^X = Y \quad (3)$$

Desta forma, o cálculo da OSNR total se torna:

$$10^{\left(\frac{-OSNR_{out}}{10}\right)} = 10^{-\left(\frac{OSNR_{\text{in}}}{10}\right)} + 10^{-\left(\frac{P_{in1} - NF_1 - 10\times\log(h\,\nu\,\nu_r)}{10}\right)} + \ldots + 10^{-\left(\frac{P_{inN} - NF_N - 10\times\log(h\,\nu\,\nu_r)}{10}\right)} \quad (4)$$

Onde:

- $10^{\left(\frac{-OSNR_{out}}{10}\right)}$ é o termo de referência em função da OSNR da rota óptica.

Ao tornar este valor o custo de um enlace, pode-se procurar a rota de menor custo na rede, o que significa encontrar a rota com maior estimativa de OSNR.

Para que esta estimativa possa ser calculada, ainda são necessárias outras informações referentes aos equipamentos fotônicos instalados na rede, como a figura de ruído e a potência de entrada em cada um dos elementos. Estas informações são adicionadas à topologia levantada pelo PCE através de arquivos de configuração contendo a especificação destes parâmetros para cada enlace. Antes de se iniciar o procedimento de cálculo de rota, o PCE identifica para cada um dos enlaces que são reconhecidos na base de dados e adiciona a ele as informações extras. Após o término deste procedimento, o valor final da estimativa da OSNR é calculado.

## 4. Cenário de rede e testes de validação

A rede experimental, mostrada na Figura 5 é composta por cinco O-NEs e o controlador SDN. Cada O-NE inclui um ROADM multi-direcional, amplificadores ópticos, canais de supervisão e *transponders* ópticos. Os experimentos foram realizados com canais na grade DWDM com espaçamento de 50GHz, padronizada pelo ITU. O controlador SDN que executa o T-NOS e as instâncias virtualizadas do XCP é equipado com um processador Intel Core i7, 2.8GHz, com 8GB de RAM.

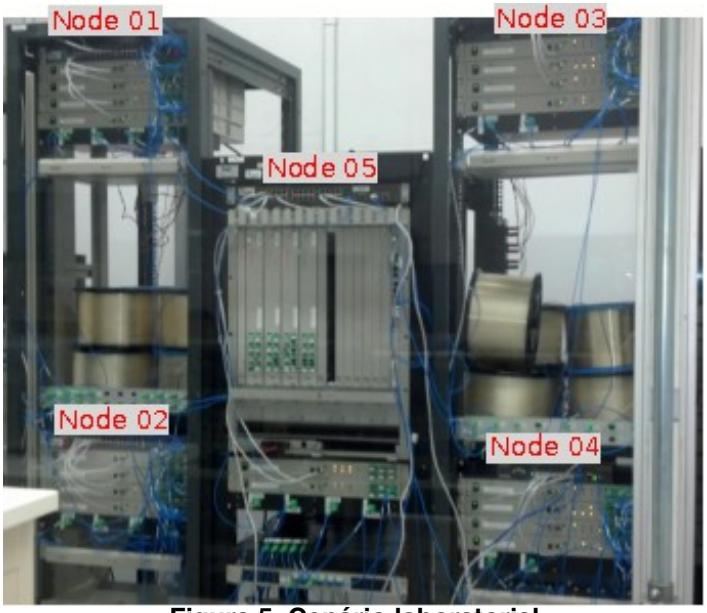



**Figura 5. Cenário laboratorial**

Na imagem não é mostrado o nó central, onde as instâncias do XCP, os controladores dos nós, etc., são executados. A conectividade lógica entre estes nós (incluindo o nó central) é fornecida através da presença de canais de serviço e *switches* Ethernet instalados nos nós. É importante notar que o nó centralizado não possui conexão direta com os equipamentos, apenas com o elemento de supervisão de cada nó, o qual é o único que possui acesso direto às chaves e aos amplificadores.

Para conseguir executar várias instâncias do plano de controle no nó central, foi empregado LXCs (*Linux Containers*) para o isolamento dos processos entre si. Esta abordagem permite que as instâncias do XCP possam ser criadas, configuradas e acessadas de forma programática e sem aumentar excessivamente o consumo de recursos do nó central.

A topologia física da rede experimental é mostrada na Figura 6.

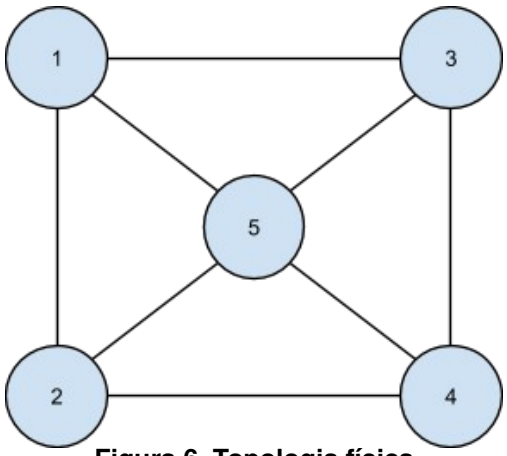



**Figura 6. Topologia física**

A topologia lógica segue as conexões físicas da rede mostrada na Figura 6. Ela será modificada para a configuração de uma VON utilizada para iniciar e configurar as instâncias do plano de controle.

Os testes de validação da implementação constituem na criação de rotas utilizando toda a infraestrutura apresentada. Nesta tarefa, todas as funcionalidades dos elementos apresentados até aqui serão utilizados. As etapas destes testes são:

- A topologia da rede óptica virtual será gerada automaticamente através da descrição de uma demanda. Esta topologia será utilizada para a configuração das instâncias do plano de controle;
- Várias rotas serão configuradas na rede de teste usando o plano de controle virtualizado. A rota será escolhida pelo PCE utilizando os bancos de dados obtidos através do OSPF-TE e do banco de dados local;
- Durante a criação da rota, as instâncias do plano de controle utilizarão o controlador dos nós da rede para efetuar as configurações necessárias.

Além de validar o funcionamento do PCE enquanto aplicação SDN, estes testes também comprovam a expansibilidade da arquitetura SDN proposta para a adição futura de novas aplicações de controle e gerência de rede.

### 4.1. Descoberta de Topologia e Instanciação de VONs

O procedimento de descoberta da topologia está descrito em [Oliveira 2013]. Ele utiliza como base o LLDP (*Link Layer Discovery Protocol*) [LLDP], protocolo da camada de enlace que envia informações locais para seus vizinhos diretos e armazena os dados recebidos em uma base local. O protocolo foi modificado para incluir e enviar o identificador do nó e a interface local. O T-NOS extrai essas informações dos nós e, em conjunto dos modelos dos equipamentos, cria a topologia completa da rede e gera um arquivo XML contendo todos os dados coletados.

O processo de geração automática das VONs está ilustrada na Figura 7.

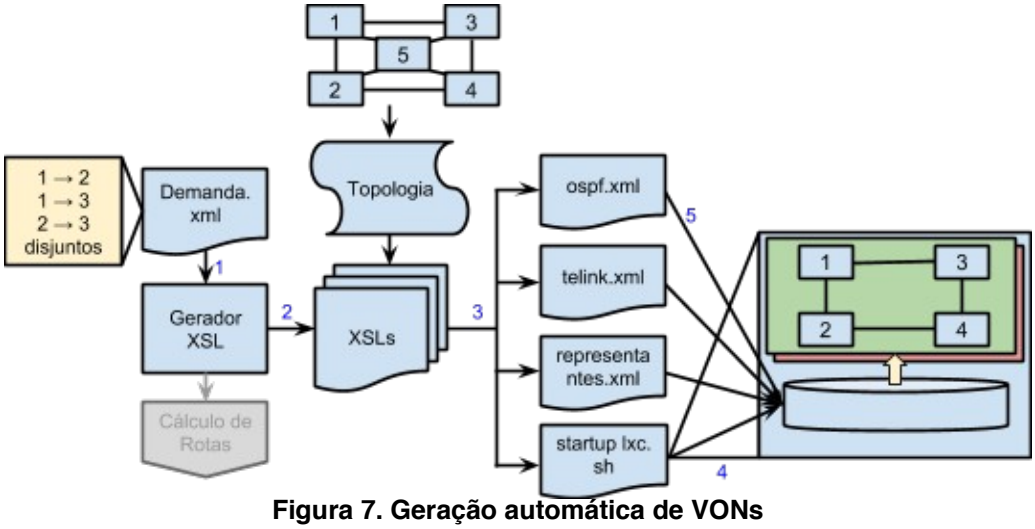



**Figura 7. Geração automática de VONs**

Inicia-se o procedimento a partir de dois arquivos XML: o primeiro contendo as conexões e os comprimentos de onda desejados e o segundo que descreve a topologia da rede descrito anteriormente. Uma aplicação realiza a leitura do arquivo de requisição de nova VON (1) e cria os arquivos de transformação XSL (2). Eles utilizam os dados da topologia para criar os arquivos para a configuração da VON no T-NOS e o "*script*" responsável pela inicialização e configuração dos LXCs (3). Os *containers* são iniciados (4) e as configurações são carregadas para o controlador (5).

É possível utilizar alguma aplicação responsável pelo cálculo de rotas para fornecer caminhos disponíveis. Isto permite que o arquivo da demanda contenha apenas as conexões fim a fim, deixando a cargo do Cálculo de Rotas devolver os nós intermediários.

Para os testes feitos neste trabalho, a topologia mostrada na Figura 6 foi modificada de forma a eliminar o nó 5, assim como mostrado na Figura 8.

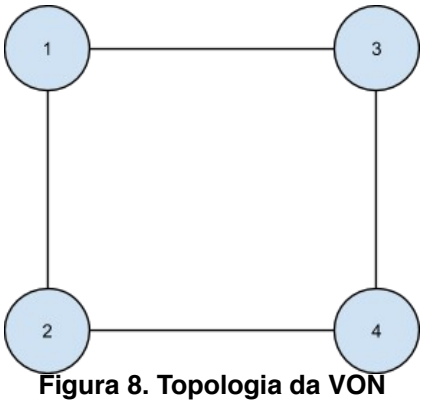



**Figura 8. Topologia da VON**

Desta forma, as instâncias do plano de controle não terão conhecimento das conexões entre os nós 1, 2, 3 e 4 e o nó 5, não sendo possível configurar rotas ou obter outras informações utilizando este nó ou os enlaces conectados a ele.

## 4.2. Criação de rotas no plano de transporte

Após a correta criação e inicialização das instâncias do plano de controle

virtualizado, a disseminação das informações topológicas começa a ser executada pelo protocolo OSPF-TE. Neste momento, inicia-se também o processo do OSPF-TE responsável pelo preenchimento do banco de dados de topologia do PCE.

Através da interface de configuração fornecida pelo T-NOS, são criadas várias requisições entre nós distintos da rede. Um exemplo da captura de pacotes durante a requisição e sinalização de uma rota é mostrada na Figura 9.

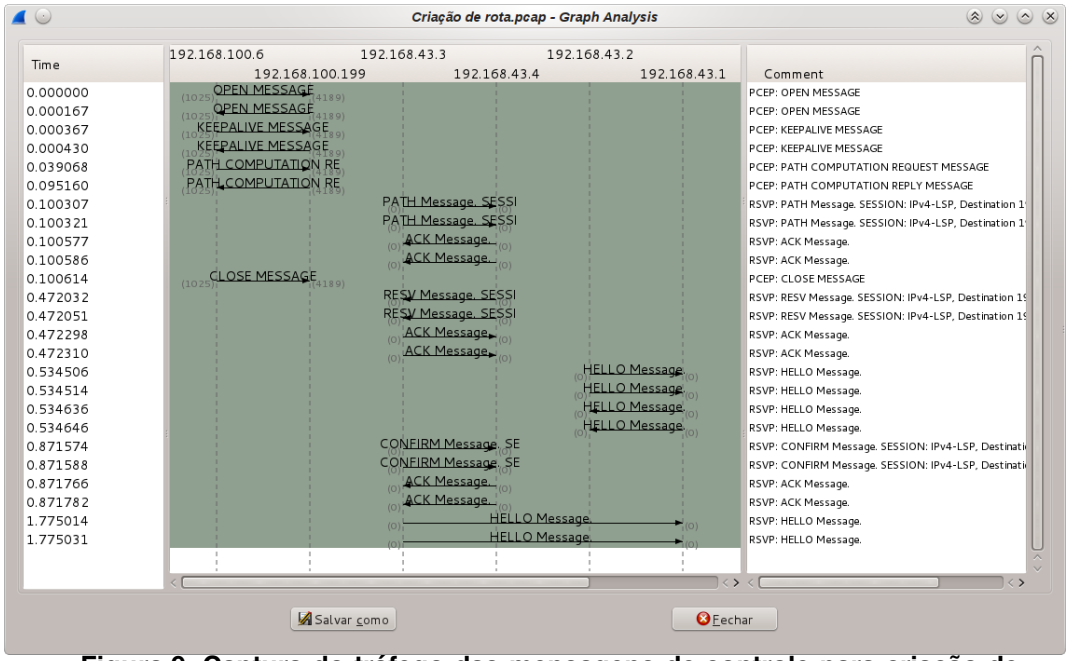



**Figura 9. Captura do tráfego das mensagens de controle para criação de um caminho**

Na figura são mostrados todos os pacotes trocados entre uma instância do plano de controle. Inicialmente, ao se iniciar a configuração de um novo caminho, uma instância do plano de controle se comunica com o PCE (configurado estaticamente no XCP). Ao receber a resposta, o nó inicia o procedimento de sinalização do caminho especificado pelo PCE.

Durante o processo de cálculo de rota, os custos de cada um dos enlaces são alterados para valores relacionados ao cálculo referente à OSNR, como mostrado anteriormente. A base de dados local do PCE inclui estruturas que associam identificadores de enlaces de engenharia de tráfego, configurados nas instâncias do plano de controle, e uma lista de amplificadores ópticos. Aliado a isto, são incluídos os valores obtidos da caracterização destes amplificadores, os quais incluem uma lista extensa de medidas de figura de ruído e inclinação dos canais para cada par { potência de entrada, ganho }.

A Figura 10 mostra a topologia considerada pelo PCE, que é constituída pela topologia divulgada pelo OSPF-TE e pelas informações presentes no banco de dados local.

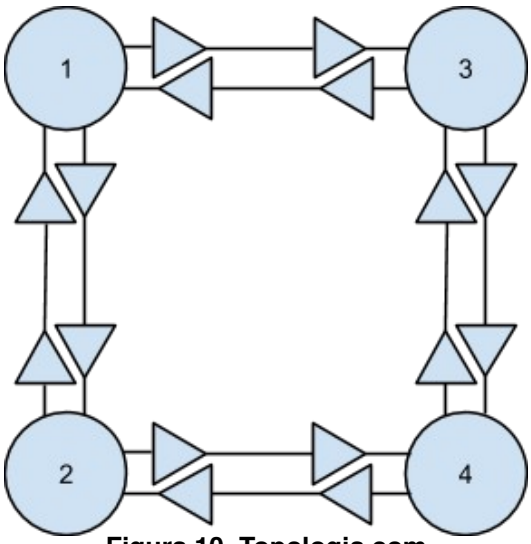



**Figura 10. Topologia com informações extras**

Foi considerado que a potência de entrada de cada amplificador presente na topologia mostrada na figura anterior é fixa em 0dBm e o ganho de cada um é configurado como 20dB (ambos os valores são mantidos por mecanismos de controle automático de ganho do próprio amplificador).

Para verificar se os valores de OSNR calculados pelo PCE condizem com a realidade, um caminho foi criado um caminho percorrendo os nós 3, 1, 2 e 4 cujo sinal corresponde ao canal C38 (193,8THz) é injetado na rede com OSNR inicial de aproximadamente 35dB. A fim de simular um ambiente mais próximo do real, foram configurados caminhos para outros canais da banda C, todos eles percorrendo a mesma sequência de nós do caminho original (encaminhando o canal C38). Neste cenário, o valor estimado pelo PCE para a OSNR do caminho testado foi de 25,2dB, enquanto o medido foi de aproximadamente 29,8dB. Uma das principais causas da diferença destes dois valores pode ser atribuída ao ganho não homogêneo que os amplificadores aplicam em cada um dos sinais ópticos passantes, podendo alterar de forma significativa o valor final da OSNR. Desta forma, faz-se necessária uma análise mais profunda tanto do mecanismo quanto das fórmulas que modelam o comportamento da OSNR em um cenário multicanal.

## 5. Conclusões

Neste trabalho foi apresentado um cenário de aplicação de um PCE capaz de considerar restrições físicas durante o cálculo de rota. Para tal, ele faz uso de instâncias virtualizadas do plano de controle e de um banco de dados local que detalha as informações obtidas das instâncias para conseguir estimar um parâmetro físico, no caso a OSNR. Neste cenário, tanto a topologia fornecida pelo plano de controle quanto as informações do banco de dados local podem ser construídas através de mecanismos de virtualização da infraestrutura de rede, a qual gera uma visão para aplicações, neste caso o PCE, para operação dos equipamentos.

Este trabalho pode ser estendido a fim de considerar novas restrições físicas, utilizar novos bancos de dados que detalham e/ou modelam outros parâmetros físicos da rede, executem análises destes parâmetros de formas mais precisas, etc. Além disso, a infraestrutura empregada neste trabalho pode ser reutilizada para o desenvolvimento de outras aplicações de rede (assim como o PCE) para diversas outras funcionalidades, com análise e previsão de falhas, melhoria da condição de operação de determinados

equipamentos de rede (amplificadores, chaves ópticas, etc.), equalização de canais baseando em informações globais, etc. Uma outra sugestão de extensão deste trabalho é o suporte a outros protocolos e modelos para a comunicação com os equipamentos do plano de transporte ao invés do NETCONF/YANG, como o *OpenFlow*.

## 6. Agradecimentos

O presente trabalho foi desenvolvido em conjunto com a Anatel e suporte financeiro do Funttel – Fundo para o Desenvolvimento Tecnológico das Telecomunicações e da FINEP (Agência Brasileira da Inovação).

## 7. Referências

Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., Magalhães, M. F. (2011). OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. *Caderno CPqD de Tecnologia*

Mannie, E. (2004). Generalized Multi-Protocol Label Switching (GMPLS) Architecture. RFC 3954 *(Proposed Standard)*

Siqueira, M., Oliveira, J. R. F., Santos, G. C., Hirata, A., Hooft, F. N. C., Nascimento, M. R., Rothenberg, C. E., Oliveira, J. C. F. (2013). An Optical SDN Controller for Transport Network Virtualization and Autonomic Operation. *Globecom Communications Conference (GLOBECOM)*

Bjorklund, M. (2010). YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020 *(Proposed Standard)*

Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A. (2011). Network Configuration Protocol (NETCONF). RFC 6241 *(Proposed Standard)*

Farrel, A., Vasseur, J. P., Ash, J. (2006). A Path Computation Element (PCE)-Based Architecture. RFC 4655 *(Proposed Standard)*

Vasseur, J. P., Le Roux, J. L. (2009). Path Computation Element (PCE) Communication Protocol (PCEP). RFC 5440 *(Proposed Standard)*

Lee, Y., Bernsein, G., Li, D., Imajuku, W. (2013). Routing and Wavelength Assignment Information Model for Wavelength Switched Optical Network. draft-ietf-ccamp-rwa-info-19. *(Work in progress)*

Bernstein, G., Lee, Y., Li, D., Imajuku, W. (2013). General Network Element Constraint Encoding for GMPLS Controlled Networks. draft-ietf-ccamp-general-constraint-encode-13. *(Work in progress)*

Bernstein, G., Lee, Y., Li, D., Imajuku, W. (2013). Routing and Wavelength Assignment Information Encoding for Wavelength Switched Optical Networks. draft-ietf-ccamp-rwa-wson-encode-23. *(Work in progress)*

Crabbe, E., Medved, J., Minei, I., Varga, R. (2013). PCEP Extensions for Stateful PCE. draft-ietf-pce-stateful-pce-07. *(Work in progress)*

Moura, U. C., Oliveira, J. R. F., Oliveira, J. C. R. F., César, A. C. (2013). EDFA Adaptive Gain Control Effect Analysis over an Amplifier Cascade in a DWDM Optical System. In *Microwave & Optoelectronics Conference (IMOC), 2013 SBMO/IEEE MTT-S International*

Santos, G. C., Negreiros, R., Costallat, R. F. G., Scaraficci, R. (2012). Método para consideração de restrições em ROADMs na definição de caminhos Ópticos utilizando algoritmos de RWA genéricos. *XXX Simpósio Brasileiro de Telecomunicações*

Nascimento, M., Magalhães, E. C., Oliveira, V. G. (2012). Estudo Comparativo de

Arquiteturas de ROADMs para Redes Ópticas de Próxima Geração. *XXX Simpósio Brasileiro de Telecomunicações*

ITU-T (2007). Physical transfer functions of optical network elements. G.680 *(Proposed Standard)*

Oliveira, J. R. F. et al. (2013). Experimental Testbed of Reconfigurable Flexgrid Optical Network with Virtualized GMPLS Control Plane and Autonomic Controls Towards SDN.

LLDP: , Station and Media Access Control Connectivity Discovery (LLDP), 2009


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 11*

## *Middleware e Serviços*

# A Service Selection Mechanism using Fault-Tolerance Techniques

**Higor Amario de Souza**[1], **Felipe Pontes Guimaraes**[1], **Fabio Kon**[1], **Daniel Macedo Batista**[1]

[1]Department of Computer Science – IME
University of Sao Paulo (USP) – Sao Paulo, SP, Brazil

{hamario, felipepg, kon, batista}@ime.usp.br

***Abstract.*** *Service selection in Service-oriented computing (SOC) is an important step towards obtaining the required level of quality. Fault-Tolerance Techniques (FTT) can be used to enhance the reliability of compositions considering Quality of Service (QoS) requirements. Services that do not perform as expected must be replaced by equivalent services. We propose a mechanism called FTTServiceSelector (FSS) to select services that meet QoS constraint values defined by the user. FSS uses a Constraint-Satisfaction Problems (CSP) approach to generate a global optimal service selection. Experiments show that FSS can be used to generate solutions to form compositions with up to ten roles and ten equivalent services. In most cases, FSS can generate solutions in a feasible time.*

## 1. Introduction

Service-oriented computing (SOC) is the computing paradigm that introduced the concept of software-as-a-service. Services are self-describing, platform-agnostic computational elements that can be used to compose distributed applications [Papazoglou 2003]. Services can also be viewed as functions accessible via well-established protocols. These services can vary from small to complex functions. Services distributed over the Internet are known as Web services. They can be used by organizations in their business processes in a wide range of devices, such as desktops and smartphones [Papazoglou 2003]. Web services are broadly used for several activities, from obtaining information about weather to performing complex stock exchange transactions.

According to Papazoglou [Papazoglou 2003], the main characteristics of services to be offered to organizations over the Internet are: technology neutrality, loose coupling and location transparency. *Technology neutrality* relies on compatibility with almost all computational environments. *Loose coupling* is related to independence of knowledge about internal structures or conventions between clients and servers. *Location transparency* is concerned with the availability of the services.

Service-oriented architecture (SOA) is a way of reorganizing software applications and support infrastructure to provide an interconnected set of services to perform SOC. It is an architecture composed of service providers, service discovery agencies and service requestors [Papazoglou 2003]. In SOA, the communication between clients and services occurs using repositories that keep information about available services. Well-defined protocols are used to exchange requests and responses.

Applications exposed as services can be combined to implement standard business processes. Service composition is a term used to define applications that use

different services to produce high level functionalities. Service compositions can lead to the development of rapid, low-cost and interoperable distributed applications [Papazoglou et al. 2007]. Orchestration and choreography are the main models of service compositions to create business processes. Orchestration refers to an executable business process that interacts with services. Choreography refers to compositions in which the control of the interactions is decentralized – the services involved are only aware of their own roles in the interactions [Peltz 2003]. In a pool of services it is possible that there are equivalent services – if a service is not available it may be replaced by some of its equivalent services so compositions do not suffer from unexpected failures.

Although compositions have several benefits, there are some challenges to improve their delivered QoS. The number of available services has increased over the years [Rao and Su 2005]. As a consequence, there is an increasing number of selectable services to form compositions. Services could be unavailable or updated on the fly. Thus, selecting services that meet certain quality criteria is an important task to obtain reliable compositions.

This paper proposes a mechanism to select services for compositions from combinations of services that are generated using four Fault-Tolerance Techniques (FTT): *Retry*, *Recovery Block*, *N-Version Programming*, and *Active Replication*. We use a Constraint-Satisfaction Problem (CSP) approach to generate these combinations of services. Combinations of services that meet specific QoS constraint values are selected. The QoS requirements used were *Response Time* and *Reliability*. This information can be used by decision models to choose services more suitable to the constraints.

The remainder of this paper is organized as follows. Section 2 shows the key concepts used in this work. Section 3 presents related work. Section 4 presents the proposed FSS mechanism and its related concepts. Section 5 presents our experimental evaluation and the results are shown in Section 6. Then, in Section 7 we discuss the results. Section 8 shows the threats to validity and we present the conclusions and future work in Section 9.

## 2. Key Concepts

The FSS mechanism uses CSP to generate all possible combinations of compositions based on FTT. Then, QoS requirements are used to the service selection process. These key concepts are presented in this section.

### 2.1. Quality of Service (QoS)

In distributed systems, QoS is a combination of qualities or properties of a service [Menasce 2002]. Some qualities measured in distributed systems are availability, cost, response time, and reliability. *Availability* is the percentage of time that a service is operating. *Cost* is the effort spent by the service. *Response time* is the time spent by the service to respond a request. *Reliability* is related to the correctness of the response.

QoS requirements can be used to enhance the quality of applications according to desired criteria. Thus, the reliability of compositions can be improved using QoS requirements to select services so that the formed composition keeps certain quality requirements. These values can be seen as constraints chosen by an interested user.

### 2.2. Fault-Tolerance Techniques (FTT)

Fault-tolerance is the ability to deliver a service that can be trusted, avoiding service failures in the presence of faults [Avizienis et al. 2004]. Fault-tolerance techniques (FTT) are used in different areas of computer systems: embedded systems, microprocessors, or grid applications. Considering that any system is failure prone, fault-tolerance techniques can help to avoid the loss of program executions, loss of data, or even to finish the execution of a function without break a software.

FTT can be used to enhance the reliability of a composition operation in the presence of faults. Once we have equivalent services to choose for compositions, FTTs can be used to generate different combinations of these services. Thus, it is possible to choose combinations of services more suitable to the composition execution according to the expected input constraint values.

Some FTTs used in distributed systems came from other computing areas. These FTTs can be divided as sequential and parallel approaches. Two sequential techniques are Recovery Block and Retry. *Recovery Block* (RB) was first used for error recovery in exception handling of software systems. The RB approach aims at providing fault-tolerant functional components that may be nested within a sequential program [Randell and Xu 1995]. Should some component fail during execution, it is replaced by an alternative component to continue the sequence of the execution. *Retry* is a variation of RB that uses only one component for each functionality in a sequential program. This component is tried repeatedly until it violates a deadline or else it produces an expected output [Ammann and Knight 1988].

Two parallel techniques are N-Version Programming (NVP) and Active Replication (Active). *NVP* is a technique that uses the results from different program versions to choose a consensual result [Avizienis 1995]. The idea is that different versions can reduce the probability of obtaining an erroneous result. *Active replication* invokes software component equivalent services on distinct host computers concurrently [Chereque et al. 1992].

In the distributed systems domain, FTTs can be used to invoke equivalent services in compositions. These techniques can inhibit failures of a service in a composition, replacing it by other and keep the application working correctly. However, it is necessary to choose which FTT to use and what are the best services to meet the expected composition behavior. Given a set of services, the feasibility of combining these services to form compositions should be evaluated.

### 2.3. Constraint-Satisfaction Problems (CSP)

A large number of problems in different computer science areas can be viewed as Constraint-Satisfaction Problems (CSP) such as diagnostic reasoning, scheduling, or decision process [Kumar 1992]. A CSP can be defined as a problem with a given set of variables, a finite and discrete domain for each variable, and a set of constraints. The goal is to find possible solutions that satisfy these constraints [Kumar 1992].

There are different approaches to solve CSP. Algorithms to solve CSP usually fall into two main families: systematic and local search algorithms [Jussien and Lhomme 2002]. Two examples of systematic and local search algorithms

are, respectively, Backtracking and Constraint Propagation. *Backtracking* is a systematic algorithm that uses all possible combinations between the variables to test which of them satisfy the constraints [Kumar 1992]. *Constraint Propagation* is a local search algorithm. It considers that information is deduced from a local group of constraints and nodes, and is recorded as a change in the network. Further deductions make use of these changes to make further changes [Davis 1987].

Backtracking could be used to generate all potential solutions for CSPs. However, the backtracking runtime complexity is exponential [Kumar 1992]. Conversely, local search algorithms such as Constraint Propagation can reduce the complexity, but they may not find a solution [Jussien and Lhomme 2002].

The problem of selecting services for compositions that meet QoS constraints can be viewed as a CSP. In this work, we use the backtracking approach in the FTTServiceSelector mechanism to do this task, since we are interested in a method that returns a solution. The QoS requirements are the constraints of our problem. Each FTT can generate a different number of combinations of equivalent services that could lead to possible solutions to these constraints. Thus, it is possible to use CSP to generate solutions in a feasible time using these QoS constraints for compositions with up to 10 roles and 10 equivalent services per role.

## 3. Related Work

There are other techniques that use QoS requirements to the service selection task. Katterpur et al. [Kattepur et al. 2013] investigate the use of QoS in choreographies. In this work, they measured QoS requirements such as *Deadlock Freeness*, *Conformance* and *Realizability*. All these parameters are related to the messages passed among the services. Yu and Lin [Yu and Lin 2005] propose a technique to use QoS to select services for compositions. They use the concept of QoS broker to maximize the utility function value.

Ai and Tang [Ai and Tang 2008] present a genetic algorithm that uses a penalty-based concept to the service selection. The results show that the technique is scalable for compositions with up to 100 roles. However, the technique finds local optimal solutions. Alrifai et al. [Alrifai et al. 2012] propose a technique that combines local and global optimal selection. They used mixed integer programming to find the global optimization of QoS requirements as local constraints and local optimization to select the services that matches these local constraints. The evaluation carried out using simulated datasets show that the technique can reduce the time spent to find solutions.

Other previous works combine the use FTTs and QoS to select services for compositions dynamically. Zheng and Lyu [Zheng and Lyu 2010] propose a middleware that uses QoS requirements with different FTTs. The QoS constraints are informed by users. The authors combine the four FTTs summarized in Section 1 to propose a FTT called *Dynamic*. The experiments show that Dynamic outperforms the other FTTs used separately.

Dobson [Dobson 2006] uses WS-BPEL to model fault-tolerance strategies for processes. These models include FTTs such as the four mentioned in the previous section. A tool for mapping FTTs in WS-BPEL processes is presented. Preliminary experiments are carried out to show the feasibility of the approach. Salatge and Fabre

[Salatge and Fabre 2007] propose the idea of connectors to deal with unreliable services. These connectors are modeled using WSDL to insert detection actions such as replication strategies in the presence of failures. Thus, equivalent services can be defined to replace unreliable services for compositions. The authors carried a case study simulating equivalent services to show the feasibility of their approach.

Differently from the previous works, our mechanism can be used to construct combinations of services for compositions before their execution. Moreover, FSS can collect QoS values from previous executions to the service selection process that can be used in future executions. We use the CSP approach to evaluate all possible compositions that meet the expected QoS constraints. These constraints can be defined to keep the desired QoS values for compositions. The results can be used by other techniques to take decisions about what are the best solutions to use when enacting a composition.

## 4. FTTServiceSelector (FSS)

FTTServiceSelector (FSS) is based on the CSP backtracking approach. The problem solved by FSS is related to generating all possible combinations between roles and equivalent services that could be combined according to the FTT strategies. These combinations are evaluated to select services that meet the given QoS constraints. Thus, these services can be used in compositions. We use two QoS requirements in FSS: *response time* and *reliability* for the complete composition as the requirements to select suitable combinations of services.

Algorithm 1 presents the FSS mechanism. It generates all possible combinations of services according to the characteristics of the FTTs. This algorithm runs once for each FTT separately. The four FTTs are also considered as variables of our problem: *Retry* requests the same Web service a certain number of times; *Recovery Block* (RB) requests equivalent services if a primary Web service fails; *N-Version Programming* invokes equivalent services at the same time and chooses the consensual result; *Active Replication* invokes equivalent services and takes the first returned result. Each of these techniques leads to different ways to calculate response time and reliability (QoS constraints) of compositions using the response time and the reliability of each service involved. The combinations that meet the QoS constraints are stored as solutions.

FSS receives input files in yaml[1] format. These files represent the resources available: a list of roles and the equivalent services with their response time and reliability values. FSS also receives an input file with the expected QoS constraint values for the whole composition. Then, FSS generates the combinations of services according to the FTT selected using the backtracking strategy (lines 1-3). Next, FSS uses the formulas presented in the Subsection 4.1 to calculate the constraints for each combination (lines 4-5) and store the combinations that meet the QoS constraints (lines 6-9) as solutions. The output is a yaml file that contains all solutions for each FTT used (line 13). To reduce the memory space used, FSS calculates each combination and records only those that satisfy the QoS values.

Figure 1 shows the FSS process. The mechanism receives files containing roles, available services, and QoS constraint values. Then, FSS generates all possible combinations of services based on FTTs. If the values of a combination meet the QoS constraint

[1]http://www.yaml.org/spec/1.2/spec.html

values, this combination is recorded as a solution. After obtaining all solutions, FSS creates an output file with these results. Thus, this file can be used by decision models to choose services for compositions.

```
   Data: Roles, equivalent services, QoS constraints, FTT
   Result: Solutions
 1 for each role do
 2     for each equivalent service do
 3         generate all possible combinations for equivalent service using FTT;
 4         for each combination do
 5             calculate constraints;
 6             if combination meets QoS constraints then
 7                 create a new solution using the combination;
 8                 store solution;
 9             end
10         end
11     end
12 end
13 generate output file;
```

**Algorithm 1**: FSS algorithm.

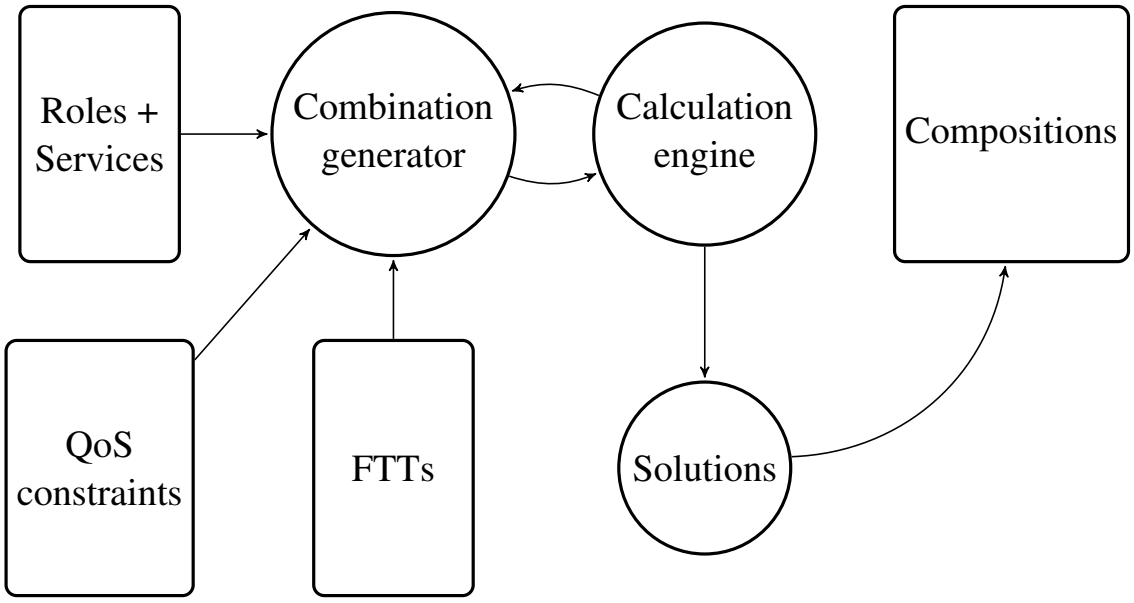



**Figure 1. FSS process.**

### 4.1. Calculation Engine

The formulas to calculate the expected QoS constraints for response time and reliability are presented in this subsection. We use these formulas to select combinations of services that reach these constraints according to the different characteristics of each FTT. These formulas are based in the work of Zheng and Lyu [Zheng and Lyu 2010]. Response time

must be less or equal than the expected value for the composition. Reliability must be greater or equal than the expected value for the composition. If both values are met the combination is selected.

The formulas represent the QoS values of each service used in a composition. Each FTT has a different strategy to form combinations of services. For the response time formulas, shown in Equations 1 to 4, the response time of each service used from all roles is summed to obtain the response time of a composition. The formulas presented in Equations 5 to 8 calculate the reliability of a whole composition considering each individual reliability value.

Equation 1 serves to evaluate the response time for Retry. For each service of a role $i$, its response time $rt_i$ is multiplied by the number of tries ($tries_i$) that a service in $i$ is requested. This value is summed for all services used in the $n$ roles of the composition. The response time for RB is calculated using Equation 2. First, the response times $rt_{im}$ of the requested equivalent services of a same role are summed. Next, the response times of all $n$ roles are summed to obtain the response time of the composition.

The response time for NVP is shown in Equation 3. The maximum response time value among all equivalent services $rt_{im}$ of a same role $i$ is chosen. These values are summed for all $n$ roles to obtain the total response time of a composition. As all services of each role are called in NVP, only the higher response time value of each role needs to be summed. Equation 4 is used for calculating response time for Active. The minimum response time value among all equivalent services $rt_{im}$ of a same role $i$ is chosen. This is because the fastest service of each role is used in Active. The response time values for all $n$ roles are summed to obtain the total response time of a composition.

Equation 5 is used for calculating the reliability of a composition for Retry. First, the reliability of each service $rel_i$ of a role $i$ is raised to the power of the number of tries ($tries_i$). Then, the reliability of the entire composition is the product of the obtained values for all $n$ roles. For RB, the Equation 6 calculates the composition reliability as the product of the requested equivalent services ($rel_{im}$) of a role $i$ by the product of all $n$ roles. For Retry and RB, only the services used of each role are considered.

Equation 7 represents the reliability for NVP. For each role, we consider that $\frac{m}{2}+1$ services return an expected result. The formula is the product of these $\frac{m}{2}+1$ equivalent services multiplied for all $n$ roles. This estimation is different from that used by Zheng and Lyu [Zheng and Lyu 2010], which consider that the failure-rate of $m$ services is $\frac{m}{2}+1$. Finally, Equation 8 is used to calculate the reliability for Active. The product of all $m$ equivalent services is multiplied for $n$ roles to obtain the reliability of a composition. Thus, all services of each role are considered to obtain the reliability of a composition.

$$RT_{retry} = \sum_{i=1}^{n} (tries_i * rt_i) \quad (1)$$

$$RT_{rb} = \sum_{i=1}^{n} (rt_{i1} + rt_{i2} + ... + rt_{im}) \quad (2)$$

$$RT_{nvp} = \sum_{i=1}^{n} max(rt_{i1}, rt_{i2}, ..., rt_{im}) \quad (3)$$

$$RT_{active} = \sum_{i=1}^{n} min(rt_{i1}, rt_{i2}, ..., rt_{im}) \quad (4)$$

$$Rel_{retry} = \prod_{i=1}^{n} (1 - (1 - rel_i)^{tries_i}) \quad (5)$$

$$Rel_{rb} = 1 - \prod_{i=1}^{n} ((1 - rel_{i1}) * (1 - rel_{i2}) * ... * (1 - rel_{im})) \quad (6)$$

$$Rel_{nvp} = 1 - \prod_{i=1}^{n} ((1 - rel_{i1}) * (1 - rel_{i2}) * ... * (1 - rel_{i\lfloor \frac{m}{2} \rfloor + 1})) \quad (7)$$

$$Rel_{active} = 1 - \prod_{i=1}^{n} ((1 - rel_{i1}) * (1 - rel_{i2}) * ... * (1 - rel_{im})) \quad (8)$$

### 4.2. Growth of Combinations

The different strategies implemented by each FTT determine the way to find possible solutions to the QoS constraints. Retry uses one service at a time for different roles. This fact leads to a certain number of combinations of services. Each combination is equivalent to a solution that represents a composition. The number of tries influences the parameter values and whether a combination reaches the expected values, but does not impact the growth of combinations. The growth of combinations for Retry is shown in Equation 9. The number of combinations is the product of the number of equivalent services ($size_i$) for all $n$ roles.

$$Combinations_{retry} = \prod_{i=1}^{n} size_i \quad (9)$$

For RB, NVP, and Active, FSS makes all possible combinations of services. Compared with Retry, the number of combinations is exponentially greater. Equation 10 shows the growth of combinations for these FTTs. The number of equivalent services per role impacts the number of combinations more than the number of roles, as shown in Table 1.

$$Combinations_{rb,nvp,active} = \prod_{i=1}^{n} 2^{size_i} - 1 \quad (10)$$

## 5. Experimental Evaluation

Our experiments aimed at measuring the effectiveness of FTTServiceSelector to find solutions for the QoS constraints using the four FTTs. The effectiveness was measured by time spent by FSS for generating and calculating the combinations, and storing solutions. The goal is to evaluate the scalability of the technique according to the growth of combinations. The factors that affect the scalability of our service selection technique are: the number of roles, the number of equivalent services, the four FTTs, and the QoS constraints. FSS is a multidimensional multiple-choice knapsack problem (MMKP), which is known as a NP-hard problem [Kumar 1992].

We used yaml files as data input to simulate compositions. These files contain the available services for each role. Each service is composed of fixed QoS values for

response time and reliability. We chose different sizes of compositions to the experiments. These sizes varied from three to ten roles and from three to ten equivalent services for each role. We used the same number of equivalent services for the roles in a same file. For Retry, we used a number of three tries for each service. We fixed an arbitrary value for the number of tries because it does not influences the growth of combinations.

The expected constraint values for response time and reliability for the composition were fixed in 5,000 seconds and 0.50, respectively. We used these values to store all combinations as solutions (i.e., all compositions meet the expected QoS constraints) . Thus, we avoid that variations in the number of stored solutions influenced the execution time of FSS.

The times to create and evaluate all combinations were measured as well as the number of combinations generated. We consider that each role is called sequentially from the first to the last role.

We implemented FSS in Java using JDK 1.7.0_45. The experiments were performed using a 64-bit PC with processor Intel Core i7 model 2620M and 8GB of RAM. The OS used was an Ubuntu 12.10. The experiments were repeated ten times for each composition size and the averages of time spent between these ten executions were recorded in the results. Since the dispersion of the results was low, this information is not presented in the graphs to improve the visualization.

All the code, data, and documentation used in the experiments is publicly available on GitHub[2]. This information is provided to facilitate the reproducibility of all the experiments reported in this paper.

## 6. Results

The size of combinations generated for different roles and equivalent services is shown in Table 1 (columns Retry and RB, NVP, Active). The values were separated for Retry and for the other FTTs, since the other FTTs have the same number of combinations. The column ES represents the number of equivalent services per role.

In Table 1, it is possible to see the exponential growth of the combinations generated by FTTs. The larger the number of roles, the larger the number of combinations. In the same way, the number of combinations grows when the number of equivalent services per role grows.

For RB, NVP, and Active, it can be noticed that the number of equivalent services per role has a great impact in the growth of the combinations than the number of roles. For three roles and ten equivalent services, more than 1 billion combinations were generated. Using ten roles and three equivalent services per role, the number of combinations was around 280 million. For Retry, the growth of combinations was more influenced by the number of tries and the number of roles, since only one service was used per role.

Figure 2 shows the relation between the time spent and the number of combinations obtained by FSS for Retry. The time includes generation, calculation, and recording of the solutions. The $x$ axis is the number of combinations. The $y$ axis represents the time spent in seconds. As the number of combinations for Retry is smaller compared

[2]`https://github.com/higoramario/ServiceSelector`

**Table 1. Number of combinations generated.**

| Roles | ES | Retry | RB, NVP, Active | Roles | ES | Retry | RB, NVP, Active |
|---|---|---|---|---|---|---|---|
| 3 | 3 | 27 | 343 | 5 | 3 | 243 | 16,807 |
| 3 | 4 | 64 | 3,375 | 5 | 4 | 1,024 | 759,375 |
| 3 | 5 | 125 | 29,791 | 5 | 5 | 3125 | 28,629,151 |
| 3 | 6 | 216 | 250,047 | 5 | 6 | 7,776 | 992,436,543 |
| 3 | 7 | 343 | 2,048,383 | 6 | 3 | 729 | 117,649 |
| 3 | 8 | 512 | 16,581,375 | 6 | 4 | 4,096 | 11,390,625 |
| 3 | 9 | 729 | 133,432,831 | 6 | 5 | 15,625 | 887,503,681 |
| 3 | 10 | 1,000 | 1,070,599,167 | 7 | 3 | 2,187 | 823,543 |
| 4 | 3 | 81 | 2,401 | 7 | 4 | 16,384 | 170,859,375 |
| 4 | 4 | 256 | 50,625 | 8 | 3 | 6,561 | 5,764,801 |
| 4 | 5 | 625 | 923,521 | 9 | 3 | 19,683 | 40,353,607 |
| 4 | 6 | 1,296 | 15,752,961 | 10 | 3 | 59,049 | 282,475,249 |
| 4 | 7 | 2,401 | 260,144,641 | – | – | – | – |

to the other FTTs, the time spent by FSS is less than 0.8 seconds, except for one case. For ten roles and three equivalent services per role, which generated around 59 thousand combinations, FSS spent around 1.5 seconds.

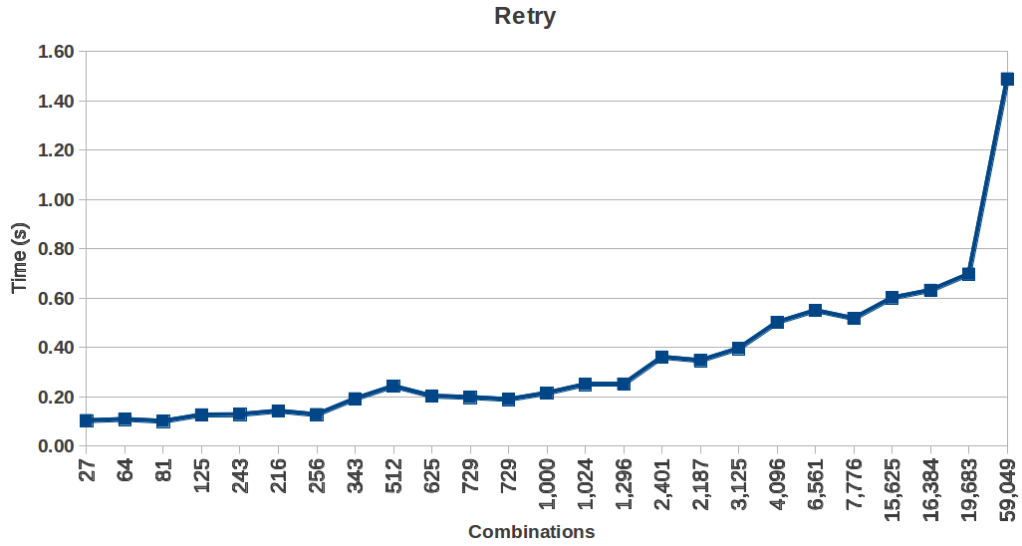



**Figure 2. Time spent x Combinations for Retry.**

The results of Retry, RB, NVP, and Active were separated in two figures to highlight the differences obtained by the FTTs for the number of combinations. Figure 3 shows the time spent in the *y* axis. The *x* axis is the number of combinations (in thousands) up to two million combinations. Figure 4 shows the number of combinations to values greater than two million combinations.

For the number of combinations in Figure 3, the time spent by RB, NVP, and Active were similar. NVP spent less time to be executed by FSS than RB and Active with a small difference. In Figure 4, as the number of combinations grew, the difference

in time spent between NVP related to RB and Active also grew. For up to 40 million combinations, FSS generated solutions for all FTTs spending no more than 16 minutes. This is equivalent of nine roles and three equivalent services. For Retry the time spent was smaller than RB, NVP, and Active due to the number of combinations.

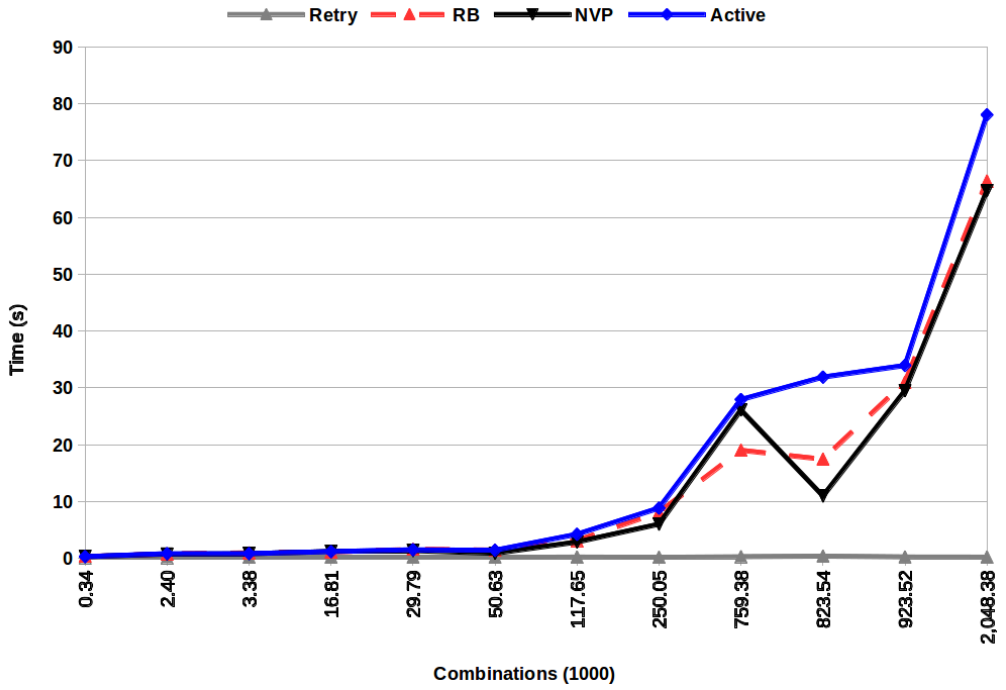



**Figure 3. Time spent x Combinations for Retry, RB, NVP and Active (Part 1).**

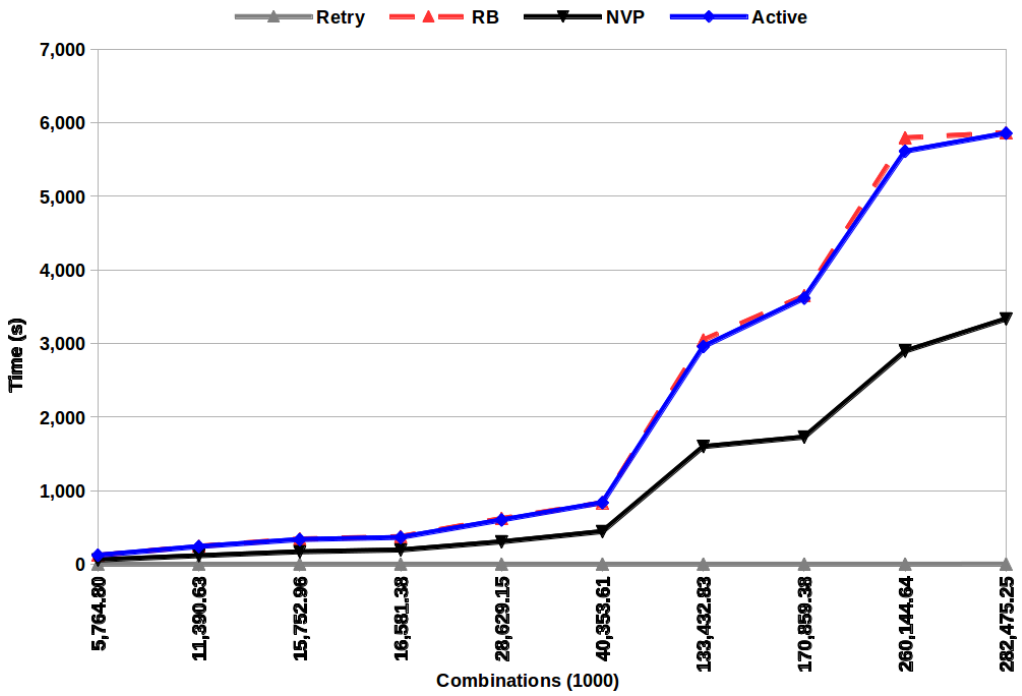



**Figure 4. Time spent x Combinations for Retry, RB, NVP and Active (Part 2).**

## 7. Discussion

The results show that FSS is scalable for compositions up to ten roles and up to ten equivalent services. The time spent by FSS to obtain solutions increases exponentially with the number of equivalent services per role. For most of the cases, FSS spent less than 60 minutes to generate the solutions. In the worst case for the values used in the experiments, FSS generated solutions in less than 140 minutes. Although these times can be viewed as excessive, they are irrelevant if FSS is used in a system where the QoS requirements of the services are monitored and stored in a historical way. In this scenario the mechanism represents a valuable contribution, in which FSS uses these previous information to the service selection process. Besides, if some services are frequently used in conjunction, due for example to delay or other network limitations, they can be viewed as only one service, which can reduce the number of services considered to each composition.

Retry can generate solutions with a complexity much smaller than the other FTTs, but the strategy considers only one equivalent service at a time per role. As the size of combinations increases, the execution time also increases, specially for RB, NVP, and Active. The time obtained for RB and Active are similar for most of the cases. These two FTTs spent more time than NVP to generate the solutions. As the number of combinations is the same for RB, NVP, and Active, we realize that the difference occurs due to the calculation of reliability for the FTTs. For NVP, FSS calculates the reliability for half of the equivalent services when compared to RB and Active.

The time spent by FSS is directly related to the FTT strategies. If the FTT needs to generate a large number of combinations, the time spent could grow proportionally. In the same way, the more the number of roles and equivalent services, the more the time spent to execute FSS. The number of solutions found by the techniques depends on the QoS constraints values passed as parameters and the formulas of each FTT. FSS could find the solutions that met these constraints even for the highest number of roles and services evaluated.

One problem faced during the experiments was the size of data generated to record the solutions calculated by FSS. The larger the number of combinations that meet the QoS constraints, the higher the execution time to run FSS. It occurs due to problems with memory space used by the JVM and the management of the Java Garbage Collector. When we use QoS values that were met by a reduced number of solutions (with up to five millions solutions), the execution of FSS did not present these problems.

The use of the CSP backtracking approach to service selection seems to be suitable when we are interested in finding solutions for generating all possible combinations between equivalent services through roles of compositions. Thus, considering that variations in the QoS values of services in a composition may occur, it is possible to select different compositions to maintain the expected QoS constraints.

## 8. Threats to Validity

Our experiments were performed using files with simulated QoS constraint values for the services and compositions. This approach is executed using static information from services. The use of values obtained from the execution of real compositions could lead to different results. The use of files with fixed values does not consider dynamic variations

that could occur during executions of compositions such as network connections or execution time of services providers. Other issues that can influence the results are the use of FSS in different domains and the use of other QoS requirements.

During the experiments, we did not vary the number of equivalent services per role. Thus, experiments using more scenarios could lead to different results. Another threat is the number of solutions. Depending on the constraint values, the number of solutions in memory could affect the time spent to generate all solutions. We avoid this threat using QoS constraints so that all combinations generated were accepted as solutions.

## 9. Conclusions and Future Work

FTTServiceSelector (FSS) is a technique that uses a CSP backtracking approach to select services for compositions. FSS uses four Fault-Tolerance Techniques to compose different strategies for the selection process: Retry, Recovery Block, N-Version Programming, and Active Replication. The selection is made using two QoS constraint values for compositions: response time and reliability.

Experiments with different composition sizes were performed to evaluate the effectiveness of FSS to find solutions using QoS constraints. The efficiency of FSS was measured evaluating the execution time to calculate the solutions. The results showed that FSS could be used to select services that meet the QoS constraints with times between 2 seconds to 140 minutes for compositions varying from three roles and three equivalent services per role to ten roles and three equivalent services per role. For most cases FSS generated solutions spending no more than 17 minutes. The information generated by FSS could be used by decision models responsible for selecting services for compositions in automated processes.

As future work, we intend to make improvements to reduce the time spent to generate the solutions. One possible way to improve the technique could be the use of filters to exclude individual services that can degrade the composition. The use of other QoS requirements and FTTs are other issues to be addressed in the future. Moreover, the use of historical data about QoS values obtained from previous executions of compositions can be used to make experiments in the future.

We also intend to use other CSP strategies, such as local search algorithms, to measure the feasibility to generate solutions reducing the time spent. For instance, if we are interested in finding only one solution to a composition, other CSP strategies, such as Constraint Propagation, could be evaluated in the future to obtain more knowledge about the use of CSP for selecting services. Experiments in runtime environments using more roles and more equivalent services are also among our future work.

## References

Ai, L. and Tang, M. (2008). A penalty-based genetic algorithm for qos-aware web service composition with inter-service dependencies and conflicts. In *Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control Automation*, pages 738–743.

Alrifai, M., Risse, T., and Nejdl, W. (2012). A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web*, 6(2):1–31.

Ammann, P. and Knight, J. (1988). Data diversity: an approach to software fault tolerance. *IEEE Transactions on Computers*, 37(4):418–425.

Avizienis, A. (1995). *The Methodology of N-Version Programming*, chapter 2, pages 23–42. John Wiley & Sons Ltd. Software Fault Tolerance.

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.

Chereque, M., Powell, D., Reynier, P., Richier, J.-L., and Voiron, J. (1992). Active replication in delta-4. In *22th International Symposium on Fault-Tolerant Computing. FTCS-22. Digest of Papers.*, pages 28–37.

Davis, E. (1987). Constraint propagation with interval labels. *Artificial Intelligence*, 32(2):281–331.

Dobson, G. (2006). Using ws-bpel to implement software fault tolerance for web services. In *32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 126–133.

Jussien, N. and Lhomme, O. (2002). Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence*, 139(1):21–45.

Kattepur, A., Georgantas, N., and Issarny, V. (2013). Qos composition and analysis in reconfigurable web services choreographies. In *International Conference on Web Services*, Santa Clara, California, USA.

Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A survey. *AI magazine*, 13(1):32.

Menasce, D. (2002). Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75.

Papazoglou, M. (2003). Service-oriented computing: concepts, characteristics and directions. In *Fourth International Conference on Web Information Systems Engineering*, pages 3–12.

Papazoglou, M., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45.

Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10):46–52.

Randell, B. and Xu, J. (1995). *The Evolution of Recovery Block Concept*, chapter 1, pages 1–17. John Wiley & Sons Ltd. Software Fault Tolerance.

Rao, J. and Su, X. (2005). A survey of automated web service composition methods. In Cardoso, J. and Sheth, A., editors, *LNCS on Semantic Web Services and Web Process Composition*, volume 3387, pages 43–54. Springer Berlin Heidelberg.

Salatge, N. and Fabre, J. C. (2007). Fault tolerance connectors for unreliable web services. In *International Conference on Dependable Systems and Networks*, pages 51–60.

Yu, T. and Lin, K.-J. (2005). Service selection algorithms for web services with end-to-end qos constraints. *Information Systems and e-Business Management*, 3(2):103–126.

Zheng, Z. and Lyu, M. (2010). An adaptive qos-aware fault tolerance strategy for web services. *Empirical Software Engineering*, 15(4):323–345.

# Reflection-based heterogeneous migration of computations *

**Anolan Milanés[1], Noemi Rodriguez[2], Roberto Ierusalimschy[2]**

[1]Department of Computer Science
Federal Center for Technological Education of Minas Gerais ( CEFET-MG)
Av. Amazonas, 7675 - Nova Gameleira – Belo Horizonte - Minas Gerais – Brazil

[2]Department of Computer Science - Pontifical Catholic University of Rio de Janeiro
Rua Marquês de São Vicente, 225, Gávea – 22453-900 – Rio de Janeiro – Brazil

anolan@decom.cefetmg.br, {noemi,roberto}@inf.puc-rio.br

***Abstract.*** *Implementing heterogeneous migration of computations is hard: it demands knowledge of the type of the data in order to be able to capture and restore the computational state. Support for those operations has traditionally been offered through ready-made solutions for specific applications, which are difficult to tailor or adapt to different needs. A more promising approach would be to build specific solutions as needed, over a more general capture and restoration framework. That flexibility can be achieved through computational reflection. This work extends the Lua programming language with a reflective API that provides the programmer fine control over the capture and restoration mechanisms.*

## 1. Introduction

Different migration applications may have different requirements. Migration can be applied in *Opportunistic computing*, when a process that was created in a machine may need to be transferred to another host if the local user reclaims the machine. This may require the computation to be suspended, probably from outside the program, captured with all its dependencies and open files and transmitted and restored to another host, where part of the required libraries could be already in place. Similar requirements may be posed in load balancing, when the computation is moved to another, less loaded host, in order to improve its performance. On the other hand, a mobile agent may require to take along only part of its state, since at destination it will be bound to the new local context.

Systems providing support for migration, persistence, monitoring, or debugging, all have in common the need of dealing with the non-trivial problem of capturing and restoring the execution state of a computation. It would thus be natural to expect a fair amount of reuse or sharing among the developers of such systems. However, decisions regarding issues such as what should be captured, at what moment, and what strategy to choose for exception handling, vary a lot depending on the intended application. Traditionally, decisions about these issues are ingrained in systems with support for migration or persistence instead of providing common support for the general case [Milanés et al. 2008], hampering the possibilities of reusing them for purposes even slightly different from the original ones.

*This work has been partially supported by CNPq Brasil

Factoring out the support for capture and restoration from other system decisions in the form of general mechanisms avoids the need for building new systems from scratch every time a new requirement for state capture and restoration arises. Those mechanisms can then be used to build high-level libraries that implement specific policies for different application areas.

In our search for basic mechanisms for state capture and restoration, it is natural to turn to reflection: the ability of a programming system to observe and change its own behavior through reification and installation. That requires implementing mechanisms for the reification and installation of the execution state. *Reification* mechanisms allow the execution state information to be made available to the programmer as first-class values. *Installation* (also called *Reflection* [Friedman and Wand 1984]) allows the programmer to modify the program state by incorporating data from the program into the execution state. Using reflection, migrating a running computation would be reduced to reifying the continuation of a suspended execution, and then installing this continuation where desired. Similarly, persisting a running computation would involve reifying the continuation of the suspended execution, and installing it on the same host sometime in the future.

However, not any reification and installation will suffice for our needs. The programmer needs control over what parts of the computation will be captured, restored, and possibly bound with other, existing values. Simply starting a capture and letting it go without control may result in retrieving huge amounts of information, which may not be necessary if available at destination, for instance in load balancing. Furthermore, in order to make migration or persistence feasible when the value contains non-portable data, the programmer may need to manipulate the reified representation before it is re-installed. Reified objects must thus expose a structure that the programmer can manipulate in the form of fine-grained first-class values. If the programmer can handle such fine-grained values, (s)he should be able to easily de-compose and inspect them to discover nested references to other values and decide whether these should or not be reified. Navigating down the reified values, (s)he can create arbitrary data structures and use all available language features to code decisions about granularity, error-handling, and other issues.

Note that this approach is different from treating computations as first-class values, as in higher order languages [Clinger et al. 1999]. Those languages can treat computations as black-boxes. Our motivation for manipulating continuations is not their use as control structures, but their importance for applications such as migration and persistence. In our case, the program must be capable to inspect the internals of computations, which must thus be "reifiable" data structures.

In this paper, we experiment the proposed approach with *LuaNua*, an extension to the Lua programming language [Ierusalimschy et al. 2007] with mechanisms for the fine-grained *reification* and *installation* of Lua values. Heterogeneity in that case refers to the possibility of transferring the state of a running Lua program across different architectures and/or operating systems.

The rest of this paper is organized as follows. Section 2 provides an overview of the state of the art on flexible support for heterogeneous capture and restoration of the execution state of running computations. Section 3 describes our approach for the design of state manipulation mechanisms, proposes an API for reifying and installing Lua

values and explores its functionality through some examples. Section 4 describes two benchmark applications we have developed to analyze the impact of the proposed API on performance. Conclusions are given in section 5.

## 2. Related Work

As regards the state capture and restoration in Lua, the Pluto library [Sunshine-Hill 2008] already offers that support. However, in Pluto the captured values are not reified into the language, but directly translated into bytestrings. Among other consequences, this means that it is not possible to restore sharing correctly if values are captured independently.

The most distinguishing feature in LuaNua is the ability to capture and manipulate, as program data, parts of the execution stack. This ability relies heavily on Lua *coroutines* [Moura and Ierusalimschy 2009]. It is not common for languages to provide coroutines, but many provide support for continuations, which are in many ways similar to coroutines. Languages that provide support for capturing continuations rely basically on three methods: (i) the continuations can be captured transparently and dumped, (ii) the programmer may provide custom capture and restoration procedures, and (iii) the programmer may insert marks to indicate limits for marshaling.

An example of the first approach is Smalltalk. Smalltalk's stack inspection provides first-class access to the current execution stack via `thisContext` as a chain of linked stack frames called contexts [Rivard 1996]. The contexts can be captured as complete snapshots and persisted as part of the image, in the form of dumps. The idea is to persist the complete working environment, allowing that exact point of execution to be reinstated, and not, as in our case, to provide generic support for persistence and mobility.

Indeed, capturing a continuation may be ineffective, unnecessary and even unfeasible, when it comes to data that are not portable, such as file descriptors or channels. Stackless Python (version 2.6) provides the ability to capture and restore tasklets (lightweight threads) in a platform-independent manner. However, it is possible that a tasklet can not be restored, for instance, if its dump contains calls to C functions [Stackless 2011]. In this direction, some systems/languages, such as Gambit-C and Pluto, allow the programmer to specify custom capture and restoration procedures in an instance or type basis [Germain et al. 2006, Sunshine-Hill 2008]. In contrast, the stepwise character of our proposal allows for better control over the extension and content of the capture.

Perhaps the approach that is most closely related to ours is that of *delimited continuations*, which allow the programmer to specify, in the code, what part of the remaining execution should be captured in a continuation. Scala continuations [Rompf et al. 2009] are created by means of a program transformation into CPS of the code annotated with `shift-reset` keywords. Those marks allow the programmer to limit the extent of the capture, and enhance the portability of Scala continuations. This approach has been followed also in Swarm [Clarke 2013] for the migration of computations in a distributed system.

## 3. Proposal

This section presents the main design features of the current proposal. We describe our approach for the fine-grained support for capture and restoration, present a brief description of Lua, describe the proposed API, and finally discuss some examples of use.

### 3.1. A fine-grained reflective approach for capture and restoration

We have argued for handling the representation of the execution state as fine-grained first-class values. To that end, we need:

- *reification* mechanisms to allow the programmer to obtain these first-class representations of the program state in the form of data structures that can be freely navigated and manipulated.
- symmetric *installation* mechanisms for installing these program-level representations as part of the program state.

Those are the main operations provided by the API we propose. A reification following

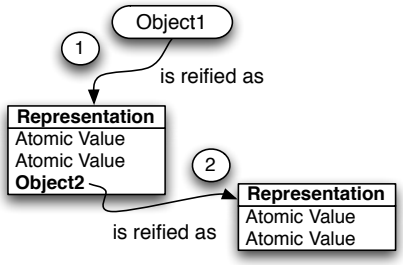



**Figura 1. Reification step by step**

this approach is executed top-down and step-by-step. Capturing an object requires the recursive reification of all the references composing its transitive closure, but the programmer can decide at every step whether to continue inspecting them all. The representation may consist of atomic values, such as numbers, and of complex values, such as functions or activation records. Figure 1 illustrates a step-by-step reification. The reification of *Object1* retrieves its internal representation, which includes two atomic values (that cannot be further reified) and the complex value *Object2*. To inspect *Object2*, the programmer must now reify it. In (2) we see the result of this reification: a representation that contains only atomic values. Here reification can continue no longer. A representation can be arbitrarily manipulated in order to satisfy application requirements (for instance, reducing cost, or replacing non portable data). Installation follows a bottom-up approach, in the reverse direction we executed the reification. That is, in Figure 1 we begin from (2) until we get to (1). Thus, more internal components are installed first, then composed into the representation of the next value to be installed.

Basically, a reified representation is a data structure that contains a copy of the internal representation of the reified entity. Changes to either the reified representation or to the execution state itself do not affect each other until an explicit install operation is issued. This is because we expect these operations to be seldom executed. However, at the time the reification or the installation is issued, the correspondence holds.

We extended the Lua programming language [Ierusalimschy et al. 2007] with support for fine-grained reification and installation. We implemented primitives for *reification* of values (capture) and for their *installation* (restoration) as an extension of Lua 5.1 that we have called *LuaNua* (Portuguese for "Naked Lua"). Our API is clearly separable from the underlying base language, allowing applications that do not use the reflection/meta-programming API to be deployed independently of it.

### 3.2. A brief intro to Lua

Lua is an interpreted, procedural and dynamically-typed language, featuring garbage collection and lexical scoping. Lua values can be of type *nil*, *boolean*, *number*, *string*, *table*, *function*, *thread*, and *userdata*. Tables are the language's single data structuring mechanism and implement associative arrays, indexed by any value of the language except *nil*. *Metatables* allow programmers to change the behavior of tables, providing, for instance, a standard way to implement object orientation, through prototyping.

The type *thread* is used for Lua coroutines. Coroutines are lines of execution with their own stack and instruction pointer, sharing global data with other coroutines [Moura and Ierusalimschy 2009]. Unlike traditional threads (for instance, Posix threads), coroutines are collaborative: a running coroutine suspends execution only when it explicitly requests to do so. Closures and coroutines are first-class values in Lua.

Internally, Lua manipulates two additional types hidden from the language level: *proto* and *upval*. Values of type *proto* are function prototypes. Upvalues are non-local values, that is, values from enclosing scopes available to nested functions.

Lua has a number of reflective facilities which already provide partial support for reification and installation. In Lua, source code and bytecode chunks can be loaded and executed dynamically. Other reflective features include access to the environment and to the names of local variables. The simplicity of concurrency in Lua avoids the need to address problems of synchronization. Lua coroutines are *stackful*, meaning they can suspend (and restart) execution at an arbitrary level of nested function calls. Having coroutines as first class values allows a homogeneous treatment of data and computations.

Lua supports the reification of functions as strings of bytes. However, this representation is not easily handled, and may require translation in case the architectures are different. The language does not allow for the serialization of coroutines.

### 3.3. LuaNua API

The most important functions added by LuaNua to Lua are *reify* and *install*:

- *reify(value, [level])* receives a value as parameter, and returns the representation of its structure. This primitive accepts an optional second parameter, which makes sense only in the case of reification of coroutines and represents the level of the desired activation record [LuaNua 2013].
- *install(representation, type | value,[level])* receives two parameters: the representation and the type or the value to be rebuilt. If successful, an invocation of this function should return a value of the specified type. Like *reify*, *install* receives an additional parameter "level" for the installation of coroutines.

Functions `reify` and `install` are available from the debug namespace. For instance, to reify a closure func, we call

```
local t = debug.reify(func)
```

This maps the internal representation of the closure *func* to a Lua table and saves it in local variable *t*. Lua tables can be inspected in all their extension, thus this solution fulfills our requirement of navigability.

On the other hand, to install a closure we call:

```
local foo = debug.install(t, "function")
```

Representations returned by *reify* may contain atomic values (numbers, booleans, strings) and/or references to complex abstractions (tables, functions, upvalues, prototypes, threads, userdata). The reification of an atomic value returns the value itself. Complex values are reified as a Lua table with references to the internal components of the value, or the components itself if atomic. That representation can be navigated by the programmer who can then decide whether or not to reify inner components. In the specific case of coroutines, the grain of reification and installation is an activation record.

Installation requires a representation that contains all the components of the value to be installed. Thus, the internal components must be installed first, then the values they are components of, and so on, following a bottom-up approach in the reverse direction we followed on reification.

Reification of arbitrary extents of program state with LuaNua thus consists of the progressive construction of a representation of the execution, by traversing the tables returned by successive calls to *reify* and filling them in, as needed, with the results of new invocations. The resulting tables can then be serialized using standard language mechanisms when implementing migration or persistence.

In addition to creating new computations, the programmer should be able to modify existing ones. It may be the case that one wishes to replace a coroutine within the program instead of creating a new coroutine upon restoration. To support this, *install* receives, as an optional parameter, the coroutine where the installation will be made.

### 3.4. Exploring the API

In what follows, we present a series of examples to illustrate the flexibility afforded by LuaNua. The examples were executed saving the resulting state in a file and restoring the stored representation from this file in another instance of the Lua interpreter, thus emulating necessary steps for either migration or persistence.

#### 3.4.1. Basics

First, we show how to reify and install a function using LuaNua. We choose a simple function (*inc*) that receives a single parameter and returns its value incremented by one.

```
local function inc(counter)
  return counter + 1
end
```

Using the LuaNua API, this function can be reified as follows (Comments in Lua begin with "--"; we are using the symbol --> for output.):

```
-- reify function inc
local tinc = debug.reify(inc)
print(tinc) --> {p = 0x532920}
```

The call to *debug.reify* returns a table containing the representation of function *inc*, that is, a reference to the function prototype and a reference to the lexical environment (for simplicity we do not discuss its reification here, as well as that of debug data contained in the representation). The reference to the function prototype is saved in field *p* of table *tinc* (*tinc.p*). Next, we proceed further into the representation by reifying the function prototype, so we make a further call to *debug.reify*:

```
local proto = debug.reify(tinc.p)
```

Now table *proto* contains the bytecode of the function prototype. Because all the values composing the representation are already atomic, reification ends here.

At this point, tables *proto* and *tinc* could be serialized and written to a file or transferred in a message. At some later point in time the tables could be reconstructed from the serialized representation and we would be ready to reinstall function *inc*.

For installation, we must follow from the bottom up. First, we install the internal, non-atomic values in memory space. In this case, the only such value is the function prototype, which was saved in table *proto*. Thus, we install the function prototype (and save it as tinc.p) and then re-create (install) the function represented by table tinc.

```
local tinc = {p = debug.install(proto, 'proto')}
local newinc = debug.install(tinc, 'function')
```

Now that newinc contains the installed function, we can execute it:

```
print(newinc(1)) --> 2
```

We can see in this example that the reification/installation manual procedure allows the programmer to control the composition of the representation. On the other hand, we see that types that were previously hidden (such as prototypes) are now visible. Because the language design does not assume that these values will be manipulated by the programmer, this visibility may lead to unanticipated execution errors (for instance, illegal operations on these values will not be handled elegantly).

### 3.4.2. Reification/installation of executions

A more interesting example is capturing and restoring executing computations. Lua provides asymmetric coroutines, which are controlled through calls to the *coroutine* module. A coroutine is defined through an invocation of *create* with an initial function as a parameter. The created coroutine can be (re)initiated by invoking *resume*, and executes until it invokes *yield*. For instance, function *count* is an iterator that, for each number from 1 to 5, prints the number and yields its value (In Lua, functions may return multiple values. *coroutine.resume* returns a value of *true* or *false* indicating whether the coroutine was resumed successfully and, optionally, values passed to *yield*):

```
-- definição da função
local function count()
   for i = 1, 5 do
      print("Number",i)
      -- send this number back to activator
      coroutine.yield(i)
   end
end
```

Suppose we want to execute *count* until it produces the number 3, and then capture the suspended coroutine. We can write the following code.

```
local coro = coroutine.create(count)
local status, i
repeat
   -- resume returns a status and
   -- the yielded values
   status, i = coroutine.resume(coro)
until (i == 3)
local rep = capture(coro)
```

When executed, this code prints:

```
--> Number  1
--> Number  2
--> Number  3
```

and stops.

Let's turn our attention to the implementation of *capture*. Internally, a Lua coroutine has a stack organized in activation records, each of them corresponding to an active function. We can reify a Lua coroutine by composing its reified frames.

We must iterate over the frames composing the stack, invoking *reify* for each level and unwinding every structured (that is, non atomic) value. This procedure creates a table with the representation of the requested entity and its components.

We begin by obtaining a reference *thr* to a new table:

```
local thr = {}
```

Then we save the coroutine status:

```
thr.status = coroutine.status(coro)
```

Now we iterate over the valid levels of the stack, from the top level:

```
repeat
  level = level + 1
  thr[level] = debug.reify(coro, level)
```

To complete the loop, we increase *level*, to move to the next activation record:

```
function capture(coro)
        local thr = {}
        repeat
          level = level + 1
          thr[level] = debug.reify(coro, level)
        until (thr[level]==nil)
        thr.status = coroutine.status(coro)
   return thr
end
```

Because a coroutine representation contains non-atomic data, its contents must in turn be reified. To that end, we inspect every activation record, searching for non atomic components. We then reify those values recursively until our representation is serializable. The activation record at level 0 corresponds to the call to `yield`. Because `yield` is a C function, it cannot be reified and serialized. The data that are neither portable nor available at the destination can be replaced by the corresponding pair during restoration.

The activation record at level 1 corresponds to the invocation of function `count`. Since that function is not part of the global environment, we need to take it along:

```
local func = thr[1].func --> count is at level 1
local t = debug.reify(func) --> t contains the representation of count
t.env = getfenv(func) --> capture the function environment
if t.env==_G then t.env = nil end --> we do not take along the global environment
t.isC = 0 --> not a C function
```

Function `count` has a prototype, that must also be reified:

```
local proto = debug.reify(t.p)
```

Now our representation is complete. At destination, in order to restore our computation, we need to install every value in the stack, and then install every activation record. We follow the reverse order: first we install the prototype, and then function count. After that, we are ready to install the execution stack. We create a new, empty coroutine, and then reconstruct the coroutine from the representation saved in *thr*.

```
local ncoro = debug.newthread()
for i = #thr, 0, -1 do
  ncoro = debug.install(thr[i], ncoro, 0)
end
debug.setstatus(ncoro,thr.status)
print(coroutine.status(ncoro)) --> suspended
```

Now the coroutine is ready for execution.

```
coroutine.resume(ncoro) —> Number 4
```

If only a partial execution is desired, it can be constructed from scratch with a subset of the reified activation records.

Although flexible, this approach is not scalable, and becomes infeasible for more complex computations. The next section describes the extension of an existing object-oriented library with higher-level functions based on the support provided by LuaNua.

### 3.5. Pickling library

LOOP (Lua Object-Oriented Programming) [Maia 2008] is a set of packages that allows the implementation of different object-oriented programming models in Lua and offers limited support for serialization. We have built migration and persistence libraries by extending the LOOP library with support for capturing and restoring Lua computations. Capture and restoration stages basically behave as follows.

**State Capture** We call the method `put` of the instance `stream` of the LOOP serializer with the suspended coroutine `co` as argument. As a result of this procedure a chunk of code, that allows re-creating the execution along with the environment, is stored and can then be serialized.

```
stream:put(co)
local msg = string.format(''%s'',table.concat(stream))
```

After this, the returned string can be easily persisted.

Capturing files introduces the problem of how to capture non-portable values. To solve it, we have redefined the `open` method of the IO library in order to register file access modes and the filepath when a file is opened. Only registered files will be captured, and their data must be stored at the serializer instance before serialization. At restoration, a function opens the file at the stored path with the access mode and position it had when captured, and returns the file handle.

**State Restoration** State can be restored from the serialized value (*buffer*) produced by the capture step. Similar to the capture, after loading the serializer libraries, we instantiate a serializer. Then we store the buffer in the serializer instance and call the deserialization method. The method returns the captured coroutine.

```
stream.data=buffer
local restoredCo = stream:get()
print(coroutine.status(restoredCo)) —> suspended
```

Now the platform can resume execution:

```
local status, error = coroutine.resume(restoredCo)
```

## 4. Experiments

To evaluate the cost of reification and installation procedures, we implemented capture and restoration of a program that calculates the factorial of a number, and migration of the $n^{th}$ Fibonacci number and a k-NN (k-Nearest Neighbor) application. Those implementations use the pickling library described in Subsection 3.5. The experiments on capture and restoration were conducted to verify the behaviour of the checkpointing latency and the storage/load and capture/load delay ratios. Capture time is defined as the time of reflecting the computation into a string (since the extraction of the representation

of the computation as tables and its conversion to strings are performed together in the serialization functions, we did not separate those times). We measured the cost of migration (migration latency) compared to the execution time of the computation, to verify if this migration procedure is reasonable. The migration cost here is defined as the time between the beginning of the capture and the end of the load stage (after the computation is installed back). The execution time of computations refers to the time elapsed between the start of the execution and the return of the function (with the final result).

We also evaluated our proposal with a real application: a program executing the k-NN algorithm for the classification of a documents database.

The experiments were executed on two CPUs Intel Core 2 Duo 2.16 GHz machines with 1 GB de RAM connected to a 100Mb switch inside a local network. Both machines run GNU/Linux Fedora Core 8 kernel 2.6.25.4-10. Time is measured using the getrusage() function, and thus refers to the CPU time effectively used by the process. All time units are milliseconds. The code for all examples is available at [LuaNua 2013].

### 4.1. Capturing and restoring the Factorial recursive algorithm

Since capture and restoration times depend on the amount of transferred data, our first experiment measures the capture/storage and load/restoration times of a computation as the stack size increases. We execute the following implementation of the Factorial algorithm:

```
local function factorial(n)
    if n==0 then
        coroutine.yield()
        return 1
    else
        return n*factorial(n-1)
    end
end
```

and capture the coroutine in which the function is executing when a call to *yield* is issued.

We ran this code for values from 0 to 50 (with measurements at multiples of 5). Figure 2 plots the delay of capture and restore operations we obtained for different frame sizes. Because storage and load times are very similar and close to zero, we used a logarithmic scale on the **y** axis of Figure 3 to plot the time for capturing, saving, loading, and restoring operations for each frame size.

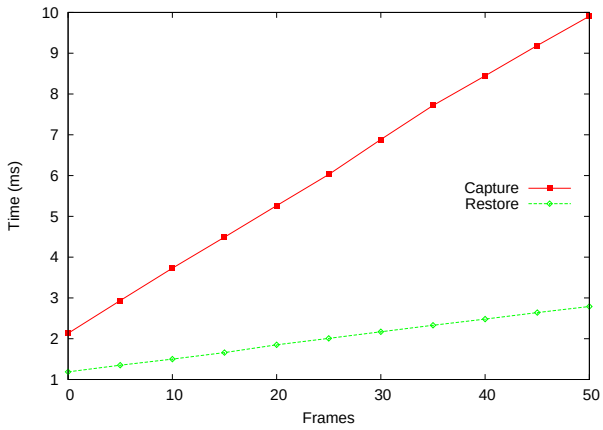



**Figura 2. Capture/Restoration**

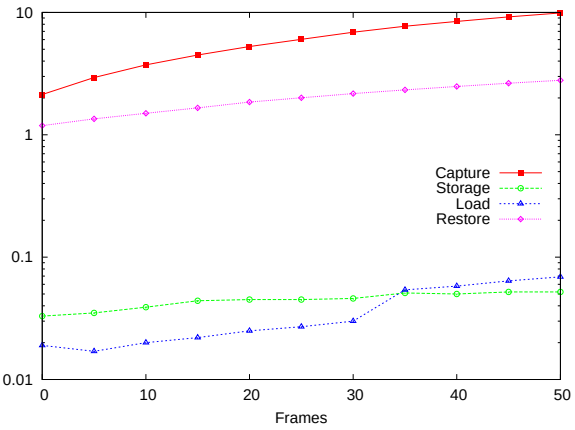



**Figura 3. Capture/Store/Restoration/Load**

The results show the higher complexity of capture when compared to restoration, which is due to the process of constructing the installation code that will be executed at restoration. It is also apparent that both capture and restoration times grow almost linearly with the number of frames on the stack, as expected, when the size of the data increases, while the delay for saving and loading keeps steadily negligible. That is, capture and restoration weigh far more than file-related procedures. Those results are rather different to other observations found in the literature( [Bouchenak et al. 2004]). This is partly due to the verboseness of our library and also to the nature of the reification/installation procedure, which emphasizes flexibility over performance. Unlike Bouchenak's approach, we are mapping execution structures to language values and only then serializing the results.

### 4.2. Migration

In this section, we discuss an experiment with migration. We developed a simple migration platform, implemented two applications, and measured migration times. The first application is the Fibonacci recursive algorithm. For the second application, we chose the k-NN algorithm because it is a real application, is easy to implement in Lua, has a regular behaviour and forces us to consider aspects not usually present in simpler experiments, such as dealing with open files.

#### 4.2.1. Migration Platform

Our migration platform is based on LuaNua, LOOP (see Section 3.5) and ALua [Ururahy et al. 2002], an event-based system for distributed programming. The programmer must explicitly insert suspension points across his code to allow migration to take place. When the application yields and the platform regains control, the application is captured and transferred. The captured computation is saved to a file that is copied to the destination, as are any open files. At the new host, the migration platform executes the received code to recreate the computation, which is then restarted.

To guarantee that all open files are transferred, we redefined function *io.open* to store information about open files in a table for later use.

The experiments were performed using two machines but, to avoid dealing with clock differences, the first machine, A, executes a computation, suspends it, captures its state in a file, sends it through the network to B (along with the open files, it also sends a message for B to initiate the operation of sending them back), from where the files are resent to A and then restored. *Migration time* was computed as the sum of capture, store, restoration and load time, plus the transmission time divided 2. The *Total execution time* is the time elapsed between the initiation of the coroutine and its return, including the time for migrating the coroutine from A to B.

#### 4.2.2. Migration of a Fibonacci execution

We executed the following implementation of the Fibonacci recursive algorithm:

```
local stop = true
local function fibonacci(n)
   if n==0 then
       if stop then coroutine.yield(); stop = false; end
```

```
        return 0
    elseif n==1 then
        return 1
    else
        return fibonacci(n-1)+fibonacci(n-2)
    end
end
```

We included a suspension call that makes the program stop when the value of the parameter is 0. After the first suspension, a flag is set in order to allow the program to continue execution until the end. We have done this in order to measure the total execution time in presence of migration. (We could have also replaced function *yield* for a dummy function on the representation before reinstalling it, with a similar effect). The function was executed varying the Fibonacci's parameter value from 0 to 50 at intervals of 5 units. The results are shown in Figure 4. As the figure shows, the cost of the migration in relation to

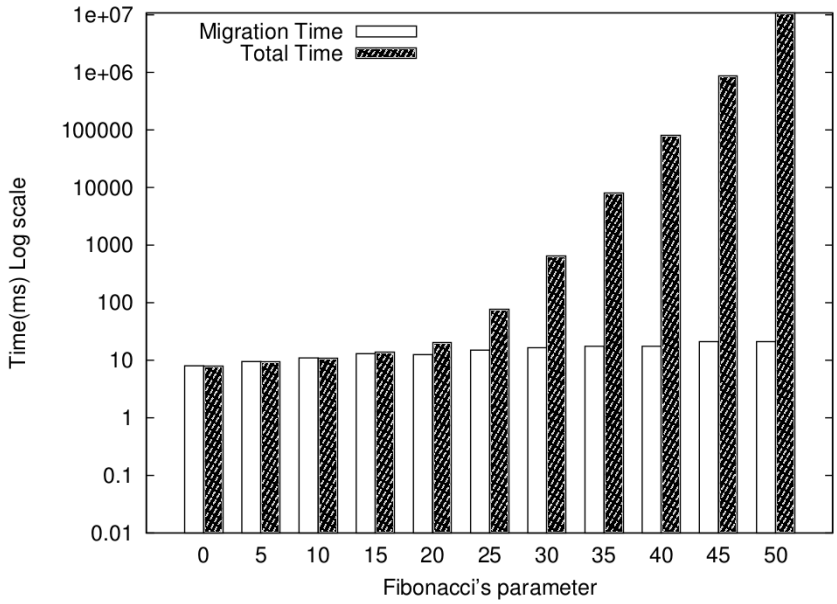



**Figura 4. Migration of a Fibonacci execution**

total execution time rapidly decreases with the increase of computation until it is almost negligible. This indicates that functions involving a certain amount of computation can benefit from migration for *Opportunistic Computing* or load balancing purposes, since the migration time is compensated by the computation time after a certain point. We can also see that migration time grows steadily but very slowly. This is because every activation record has a reference to the same function (the Fibonacci function), which is reified only once, and thus the amount of new information with every new activation record remains constant and small.

### 4.2.3. K-nearest neighbors algorithm (k-NN)

The k-nearest neighbors algorithm (k-NN) is a method for classifying objects based on the closest examples in a training set, frequently used in data mining applications. An

object is included in the most common class amongst its k-nearest neighbors, according to some measure of distance. In our implementation, we chose the cosine distance.

We implemented capturing and restoration of a coroutine executing the k-NN algorithm for classifying a relatively small database of 7 MB (but the same procedure can be used for larger databases).

In k-NN, the training table is traversed for every record of the test table, to compute the distance with the training records. We measured the total execution times of the application with and without migration. Without migration, the program simply runs to completion. With migration, it executes in machine A until it reaches half the number of records in the test base, then yields. At this point the platform initiates the migration of the execution to a machine B, where the state file is transmitted back to A:

1. At A, the current execution state is captured, saved to a file and transmitted to B, along with the working files.
2. At B, the file(s) is received and transmitted back to A.
3. At A, the file(s) is received, loaded and the computation is restored.

Finally, the restored coroutine is executed until its end. The transfer involved a buffer of 14582 kB.

We repeated each experiment seven times to guarantee an error of $0.9\%$ with a confidence interval of $95\%$. Mean time for execution without migration was $446955ms$, while execution time with migration reached $462506ms$. Total migration time (time for capturing, saving, transmitting, loading and restoring the computation) amounted to $15051ms$. The program was stopped after $225945ms$.

## 5. Final remarks

This paper argued that programming languages should offer mechanisms for fine-grained capture and restoration of the execution state in order to allow the implementation of different policies for migration and persistence. To illustrate this idea, we presented an API for reifying and installing computations. Our proposal is different from others in allowing the reification of the execution state of running computations in the form of fine-grained data structures that can be freely manipulated by the programmer. With this API, it is possible to control factors that are typically predefined in black-box serialization frameworks, such as granularity, the amount of execution state to be transferred or persisted, and the way the computation will be rebound to the new local context.

We have shown that this approach allows implementing various and powerful functionalities. A drawback is that the programming burden augments, as does the chance of dealing with representation inconsistencies. The idea is that an API such as LuaNua be used in the development of libraries which implement different policies, while still allowing direct access to the API for more specific applications. The extended version of the LOOP library that we described is an example of such a policy-implementing layer.

Future work includes exploring the flexibility we advocated, building libraries with different policies for distributed systems based on Lua. Besides their role in further evaluating our proposal, these will also be used as support in other research, for instance in investigating the management of concurrency levels in concurrent servers and in opportunistic computing.

## Referências

[Bouchenak et al. 2004] Bouchenak, S., Hagimont, D., Krakowiak, S., Palma, N. D., and Boyer, F. (2004). Experiences implementing efficient Java thread serialization, mobility and persistence. *Software: Practice & Experience*, 34(4):355–393.

[Clarke 2013] Clarke, I. (2013). Swarm-dpl: A transparently scalable distributed programming language. http://swarmframework.org. Last visited on 13/09/2013.

[Clinger et al. 1999] Clinger, W. D., Hartheimer, A. H., and Ost, E. M. (1999). Implementation strategies for first-class continuations. *Higher Order Symbol. Comput.*, 12:7–45.

[Friedman and Wand 1984] Friedman, D. P. and Wand, M. (1984). Reification: Reflection without metaphysics. In *LFP '84: Proceedings of the 1984 ACM Symposium on LISP and functional programming*, pages 348–355, New York, NY, USA. ACM.

[Germain et al. 2006] Germain, G., Feeley, M., and Monnier, S. (2006). Concurrency oriented programming in Termite Scheme. In *Scheme and Functional Programming Workshop (SFPW'06)*, pages 125–135, New York, NY, USA. ACM.

[Ierusalimschy et al. 2007] Ierusalimschy, R., de Figueiredo, L. H., and Celes, W. (2007). The evolution of Lua. In *HOPL III: Proceedings of the third ACM SIGPLAN conference on History of programming languages*, New York, NY, USA. ACM.

[LuaNua 2013] LuaNua (2013). LuaNua. http://homepages.dcc.ufmg.br/˜anolan/ research/-luanua:start. Last visited on 13/09/2013.

[Maia 2008] Maia, R. (2008). LOOP: Lua Object-Oriented Programming. http://loop.luaforge.net/. Last visited on 13/09/2013.

[Milanés et al. 2008] Milanés, A., Rodriguez, N., and Schulze, B. (2008). State of the art in heterogeneous strong computation migration. *Concurrency and Computation: Practice & Experience*, 20:1485 – 1508.

[Moura and Ierusalimschy 2009] Moura, A. L. D. and Ierusalimschy, R. (2009). Revisiting coroutines. *ACM Transactions on Programming Languages and Systems*, 31:1–31.

[Rivard 1996] Rivard (1996). Smalltalk: a Reflective Language. *In Proceedings of Reflection'96*.

[Rompf et al. 2009] Rompf, T., Maier, I., and Odersky, M. (2009). Implementing first-class polymorphic delimited continuations by a type-directed selective cps-transform. *SIGPLAN Notices*, 44:317–328.

[Stackless 2011] Stackless (2011). Stackless Python. http://www.disinterest.org/resource/ stackless/2.6-docs-html/library/stackless/pickling.html. Last visited on 13/09/2013.

[Sunshine-Hill 2008] Sunshine-Hill, B. (2008). *Lua Programming Gems*, chapter Serialization with Pluto. Lua.org, Rio de Janeiro.

[Ururahy et al. 2002] Ururahy, C., Rodriguez, N., and Ierusalimschy, R. (2002). ALua: Flexibility for Parallel Programming. *Computer Languages, Systems & Structures*, 28(2):155 – 180.

# Enabling Efficient Communications with Session Multipathing

**Israel Luiz Borges Ribeiro**[1], **and Bruno Yuji Lino Kimura**[1]

[1]Instituto de Matemática e Computação – IMC
Universidade Federal de Itajubá – UNIFEI
Av. BPS, 1303, Pinheirinho, Itajubá-MG, CEP 37500-903, Brasil

{israelborges, kimura}@unifei.edu.br

***Abstract.*** *In this paper we report an investigation on potential session multipathing strategies that leverage the easy deployment of user space-based protocols. We devised an experimental session layer architecture to deal with multipathing and we implemented such architecture with 4 different libraries of asynchronous I/O processing for Linux systems: Epoll, Posix Threads, Posix AIO, and Libev. We discuss the evaluation of these implementations (a) by comparing them with the general kernel space solution, Multipath TCP (MPTCP), in an emulated network; and (b) by determining both the performance gain factor and the cost of resource consumption as function of the number of paths in a session. We found that Libev API, a full-featured and high-performance event loop, applied to the session multipathing enables an average goodput gain factor of 1.62 faster per path added in a session, while its counterpart is of 2.23% of CPU utilization per path and it requires no more than 4 MB of RAM regardless the number of paths. We also observed that Libev-based multipathing allows overall efficiency slightly higher than MPTCP.*

## 1. Introduction

*Multihoming*, *network striping*, and *multiple paths* are resource aggregation techniques that take the advantage of multiple communication possibilities of a node, divide the application data into partitions, and stripe them over the several communication mediums, e.g. over multiple network interfaces and/or end-to-end sessions.

Several solutions have been proposed to deal with this concern. Carns *et. al* [Carns et al. 2005] proposed a Network Abstraction Layer for Parallel I/O. Yildirim *et. al* [Yildirim et al. 2009] analysed the results of different techniques to balance TCP buffer and parallel streams and present the initial steps to a balanced modelling of throughput based on optimized parameters. Parallel Sockets (PSockets) proposed by Sivakumar *et al* [Sivakumar et al. 2000] uses network striping to allow applications to achieve near optimal utilization of the network bandwidth without the necessity of tuning the TCP window size. Parallel Sockets proposed by Altman *et. al* [Altman et al. 2006] is generic hack to improve throughput attained by TCP for bulk data transfers by opening several TCP connections and striping the data file over them.

Although the use of multiple TCP paths in a session is not a novel idea [Hsieh and Sivakumar 2002] [Magalhaes and Kravets 2001], there is a lack of standardization that contemplates multipathing for today's network resources. Today's most of mobile devices support at least IEEE 802.11 and 3G/4G communication technologies

[Barré et al. 2010], while data centres can offer higher aggregate bandwidth and robustness by creating multiple paths in the core of the network [Raiciu et al. 2011a]. This increases the interest in using several access mediums in the same connection, so that it is possible to transparently change from one medium to another in case of failure, and simultaneously improve end-to-end throughput [Barré et al. 2010].

In this sense, MPTCP (Multipath TCP) [Ford et al. 2011] has become a standard protocol of eminent importance. It assumes that an end-system of an Internet node is able to establish multiple end-to-end reliable data streams (called *paths*) over a single or multiple network interfaces. The simultaneous use of multiple paths for a TCP/IP session improves resource usage within the network and, thus, (a) improve user experience through higher throughput, and (b) improve resilience to network failure [Ford et al. 2013]. The failure resilience intrinsically provides support for mobility, when the node migrates across networks on the Internet. It maintains more than one active link for as long as is feasible, if connectivity is lost for one path, remaining one can continue without interruption [Raiciu et al. 2011c].

MPTCP is essentially implemented in the OS' kernel. On one hand, kernel space-based protocols prevent context switching and memory copy between user and kernel spaces; on the other hand, they are difficult to deploy and maintain when requiring to unify the protocol implementation to the desired kernel code. This constraints are avoided with solutions implemented in the user space, which are software components made independently of kernel restrictions.

From our previous experience on development of session-based mobile communication solutions [Kimura et al. 2013] [Kimura et al. 2012], we take the hypothesis that strategies of asynchronous I/O processing can be used in support for multipath session management in the user space, so that such strategies could allow performance similar to the MPTCP. Session layer striping allows applications to take advantage of multihoming, while avoiding most of the deployability issues traditionally linked with modifying Application Layer code [Habib et al. 2006].

We propose a general multipath session layer architecture and we implemented it by using different techniques of I/O asynchronous processing for Linux system, such as Epoll[1], Posix Threads[2], Posix AIO[3], and Libev[4]. With an emulated network environment, we then compared these multipath techniques with MPTCP. To the best of our knowledge, unlike previous works, this paper provides (a) the cost-benefit analysis of multipathing by evaluating both the performance gain (goodput) and the resource consumption (CPU and memory) as function of the number of paths in a session; (b) the goodput gain factor for adding up a path in a session. In this paper we concerned with determining the boundary of multipathing gain and its resource utilization overhead, as well as with determining the most suitable technique for session multipathing, i.e. the one that provides the best cost-benefit. Provisioning of failure tolerance and mobility support are beyond the scope of this paper.

[1] `http://man7.org/linux/man-pages/man7/epoll.7.html`
[2] `http://man7.org/linux/man-pages/man7/pthreads.7.html`
[3] `http://man7.org/linux/man-pages/man7/aio.7.html`
[4] `http://linux.die.net/man/3/ev`

We observed that user space-based multipathing enables similar or better performance than MPTCP. Libev-based session multipathing has allowed high performance, with 1.62 goodput gain factor, in the meantime low resource consumption, with 2.23% of average CPU utilization per path and no more than 4MB of RAM. While MPTCP, in the same evaluated scenario, has provided 1.53 goodput gain factor, 4.85% of average CPU utilization, and 4-6MB of RAM.

This papers is organized as follows: next section presents details of the proposed multipath session architecture, with its main components: Connection Management, Message Scheduling, and Interface of Interpaths; Section 3 presents details of the implementation with thread-based and event-driven approaches; Section 4 discuss the results of performance gain and resource consumption of our multipath session implementations and MPTCP; and Section 5 brings our conclusions and future directions.

## 2. User space multipath sessions

Our general architecture to deal with session multipathing is illustrated in Figure 1. As in MPTCP, the components were designed to provide basic functionalities, such as:

- **Connection Management**: it recognizes the active network interfaces of the destination node and establishes TCP connections according to each interface IP address. We assume that an active network interface is the one that is associated with the local link, addressed by an IP, and has access routes to the Internet.
- **Message Scheduling**: it splits a message in chunks that are transmitted concurrently over the multiple paths.
- **Interface of Interpaths**: it manages the sending and receiving of message chunks by multiplexing and demultiplexing such chunks in orderly fashion, respectively.

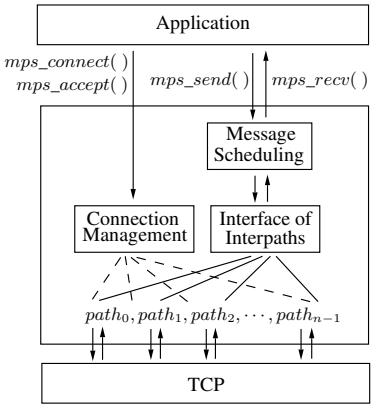



**Figure 1. A general user space architecture for session multipathing.**

The multipath session works as a middleware, which is an interface to the Application Layer implemented by means of a general purpose API. Such middleware provides for particular primitives that encapsulate the handling of multipathing, e.g. $mps\_connect(\ )$, $mps\_accept(\ )$, $mps\_send(\ )$, $mps\_recv(\ )$, etc. These primitives are based on the Socket API semantic in order to preserve the logic of the application code. We bring more details in next subsections.

### 2.1. Connection Management

We assume that the client is aware of at least one server's IP-port pair. Then it establishes a first connection with the server. Once established, the server returns to the client the IP addresses of all its active interfaces. For each server's active interface, the client then establishes a respective TCP connection. Thus, each path is consisted of a single TCP connection, so that the TCP Congestion Control works separately. Once established the TCP connections for all the paths, session establishment ends up.

An essential security mechanism is authenticating the peers involved in a session. Mutual challenge-response authentication between peers could prevent session hijacking, man-in-the-middle, and replay attacks [Kimura et al. 2013]. In addition, encryption mechanisms using symmetric and/or asymmetric keys, from the SSL API, could provide confidentiality between the involved nodes. Our implementations do not deal with security issues for while, and we leave them to be addressed in the future work.

### 2.2. Message Scheduling

Splitting the message in chunks according to the number of available paths is a task of the Message Scheduling component. The chunks are passed to the Interface of Interpaths component, which controls the sequence of the chunks transmitted over each single path. The Message Scheduling applies scatter-gather operations to handle the message chunks. Two abstract functions are used for that, respectively: $scatter(s, m, n)$, and $gather(s)$.

The abstract function $scatter(s, m, n)$ splits a message $m$ of $n$ bytes in length into a number of chunks as the number of paths available in the session $s$. The scatter function works according to Algorithm 1, which is used for both sending and receiving procedures.

**Algorithm 1:** $scatter(s, m, n)$: disjoining the message in chunks.

**Input**: A session $s$; a buffer $m$ that stores the message; the total bytes $n$ of the message to be sent or received.
**Output**: The setup of the session $s$.

```
1  begin
2      chunksize ← n / s.npaths;
3      for i ← 0 to s.npaths − 1 do
4          s.path[i].buf ← m + i × chunksize;
5          s.path[i].cbytes ← 0;
6          if i < s.npaths − 1 then
7              s.path[i].nbytes ← chunksize;
8          else
9              s.path[i].nbytes ← chunksize + (n −
               s.npaths × chunksize);
10         end
11     end
12     return s;
13 end
```

**Algorithm 2:** $gather(s)$: joining the message from chunks.

**Input**: A session $s$ under asynchronous transmission.
**Output**: the total successfully $nc$ bytes sent or received asynchronously.

```
1  begin
2      nc ← 0;
3      wait_async_return(s);
4      for i ← 0 to s.npaths − 1 do
5          nc ← nc + s.path[i].cbytes;
6          s.path[i].nbytes ←
           s.path[i].nbytes − s.path[i].cbytes;
7          s.path[i].buf ←
           s.path[i].buf + s.path[i].cbytes;
8          s.path[i].cbytes ← 0;
9      end
10     return nc;
11 end
```

For the sake of transmission consistency, the scatter algorithm handles the situation when the message splitting results in non-integer chunks. We consider that $chunksize$ is an integer variable will get always the integer part of the division of the message length ($n$) by the number of paths available ($s.npaths$). In the case of non-integer chunks, remaining fractional parts are computed for the last available path, which handles a chunk with one byte more.

The abstract function of $gather(s)$, described in Algorithm 2, attempts to synchronize the asynchronous transmissions. To do so, another abstract function, called

$wait\_async\_return()$, works as a logical barrier by waiting for the return of the initiated asynchronous operations that eventually are running.

After returning, the gather function scans through all the paths and updates the transmission context of each one. For each path, it adds up the number of the current bytes $nc$ successfully transmitted; updates the value that will be transmitted $nbytes$ according to the amount already transmitted $cbyte$; then it updates the reference pointer in the buffer $buf$ according to the amount of already transmitted bytes.

### 2.3. Interface of Interpaths

Two abstract functions are used for sending and receiving messages over the available paths in a session: $mps\_send(s, m, n)$ e $mps\_recev(s, m, n)$.

The function $mps\_send(s, m, n)$ is described in Algorithm 3. It applies the scatter algorithm to assign the portion of the message to be handled in each path. The function attempts to send chunks of the message while the amount of bytes $ns$ is less than the amount of bytes $n$ that should be sent. The asynchronous sending is conducted by the function $async\_send()$, which is a non-blocking function that uses the socket descriptor $fd$ of the path $i$ to accomplish the asynchronous sending. On error, a negative value is returned from the send function.

**Algorithm 3:** $mps\_send(s, m, n)$: multiplexer of message chunks over the multipaths.

**Input**: A session $s$; a buffer $m$ that stores the message to be sent; the total $n$ bytes of the message.
**Output**: Total bytes $ns = n$ asynchronously and successfully sent, or a negative value, on error.

```
1  begin
2      ns ← 0;
3      scatter(s, m, n);
4      while ns < n do
5          for i ← 0 to s.npaths − 1 do
6              r ← async_send(s.path[i].fd, m);
7              if r is failure then
8                  return −1;
9              end
10         end
11         ns ← ns + gather(s);
12     end
13     return ns;
14 end
```

**Algorithm 4:** $mps\_recv(s, m, n)$: demultiplexer of message chunks from the multipaths.

**Input**: A session $s$; a *buffer* $m$ that will store the incoming message; $n$ bytes of the of the message to b received.
**Output**: On success, the total bytes $nr \leq n$ asynchronously received, or a negative um valuer, on error.

```
1  begin
2      nr ← 0;
3      scatter(s, m, n);
4      while nr < n do
5          for i ← 0 to s.npaths − 1 do
6              r ← async_recv(s.path[i].fd, m);
7              if r is failure then
8                  return −1;
9              end
10         end
11         nr ← nr + gather(s);
12     end
13     return nr;
14 end
```

The non-blocking operation of $async\_send()$ allows for scanning all the paths without holding the subsequent processing. After the first scanning, then the current value of $ns$ bytes is updated by adding up the bytes successfully sent. This amount is returned from the function $gather()$, described in Algorithm 2, which works as logical barrier to synchronize the sending. If one or more chunks remain to be sent, this procedure is repeated until the $ns$ bytes reach the expected $n$ ones, or upon an error during the sending.

The asynchronous receiving works according to Algorithm 4. Message receiving is also based on the steps of scatter-gather. The abstract function $async\_receive()$ provides for the use of the socket descriptor $fd$ ready for receiving asynchronously. Upon receiving the bytes on the paths, the function $gather()$ updates the total $nr$ received bytes and synchronizes the receiving. This procedure is repeated until the receipt of the message

is finished, or discontinued upon an error.

## 3. Implementation with asynchronous I/O techniques

The algorithms presented in the previous section are abstract functions of general purpose. They were implemented in Linux system by using asynchronous I/O processing according two strategies: *thread-based* and *event-driven*.

### 3.1. Thread-based session multipathing

We implemented the thread-based multipathing according to the dispatcher-worker model, using Posix Threads API. The dispatcher threads are represented by the abstract functions of $mps\_send()$ and $mps\_recv()$. The multiple worker threads, each one handling its path $i$, are instantiated during the session establishment. Figure 2 illustrates the life cycle of dispatchers and workers for the proposed architecture.

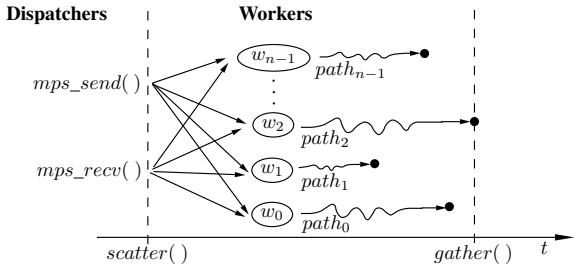



**Figure 2. Life cycle of dispatcher-worker threads for session multipathing.**

The proposed architecture provides for different buffers of sending and receiving, and there are no direct sharing of these resources. A worker thread $i$, works exclusively on a path $i$, so that there is no need for mutual exclusion during the concurrent operations of the threads. When adding a path in the session, a respective worker thread is created to handle the new path.

In $scatter$, dispatchers $mps\_send()$ and $mps\_recv()$ define which workers will handle the transmission of each chunk of the message $m$. All worker threads send and receive message chunks until the synchronization point in the $gather$. We implemented the barrier with functions from the **Posix Threads** API. The dispatchers initialize a barrier object with the function `pthread_barrier_init()`, which defines the number of worker threads will be blocked when reaching a synchronization point. The synchronization point is particularly represented by the abstract function $wait\_async\_return()$, line 3 of Algorithm 2. This function is provided by `pthread_barrier_wait()`, from POSIX Threads API.

**Posix AIO** is another thread-based API that provides asynchronous I/O. The I/O operations are governed by control blocks, called `aiocb`, which are a data structure used to store relevant information about the asynchronous transmission, e.g. the socket descriptor, a buffer and its offset, the size of the data to be transmitted, and the opcode for the asynchronous operation to be performed (`LIO_WRITE` or `LIO_READ`). After the step of $scatter$, the function `lio_listio()` is called to start executing the asynchronous transmission. With `LIO_WAIT` parameter, this function allows to configure the blocking operation, in which the calling thread is blocked until the end of all the started asynchronous

operations. Then, it provides a synchronization barrier. To obtain the return of asynchronous operations during the $gather$, the API provides the function `aio_return()`.

### 3.2. Event-driven session multipathing

Handling asynchronously I/O events is a technique to boost fast server's responses to many client requests. We used this technique in support for session multipathing in both client and server, as Figure 3 illustrates.

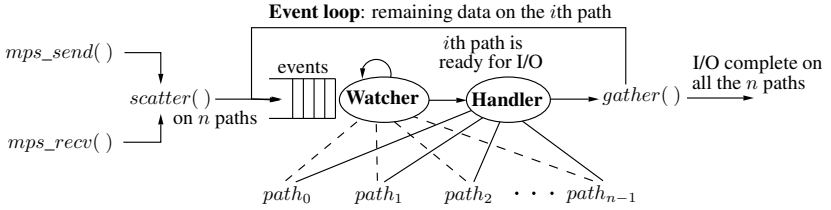



**Figure 3. Life cycle of event-driven session multipathing.**

The event-driven technique allows for a **watcher** that observers a queue of desired events over a poll of socket descriptors. In the proposed architecture, a socket descriptor belongs to a particular session. As soon as events occur on one or more descriptors, e.g. ready for reading or ready for writing, the watcher notifies an **event handler** about the state of an observed descriptor. Then, the event handler attempts to accomplish the input or output operations on the ready descriptor. The event handler is a single thread that makes use of non-blocking I/O operations, hence, allowing asynchronous handling and concurrency. In the proposed architecture, event handlers are represented by the abstract functions $async\_send()$ e $async\_recv()$.

If an I/O operation has not fully accomplished, depending on the asynchronous I/O technique, the desired event can persist or even be reinserted in the event queue. This allows an **event loop**, while there exist remaining data to be transmitted. According to Figure 3, we implemented the event-driven strategies for the proposed architecture using Epoll and Libev.

**Epoll** is a scalable API for monitoring multiple non-blocking socket descriptors to check if I/O is possible on any of them. It provides computational complexity of $O(1)$ as the number of file descriptors increase. Due to its better scalability, Epoll replaces the older ones, Posix Select and Posix Poll. In the proposed architecture, events are observed to deal with operations of sending (`EPOLLOUT`) and receiving (`EPOLLIN`). I/O operations on particular file descriptors are registered with the function `epoll_ctl()`, which is performed once during the $scatter$. The API provides the function `epoll_wait()` to wait for I/O events, which blocks the calling thread if no events are currently available. When events are available, the poll of socket descriptors are scanned to find the ones ready to accomplish the desired I/O operations. The event loop in Figure 3 is performed by combining: the event notification from `epoll_wait()`, the asynchronous I/O operation, and the $gather$.

**Libev** is an API for high performance full-featured event loop. The API provides the function `ev_io_init()` to initialize an watcher to observe a descriptor and to set the respective event handler. The event handler is implemented through an arbitrary call-back function, e.g. $async\_send()$ or $async\_recv()$, which is activated when occurring the

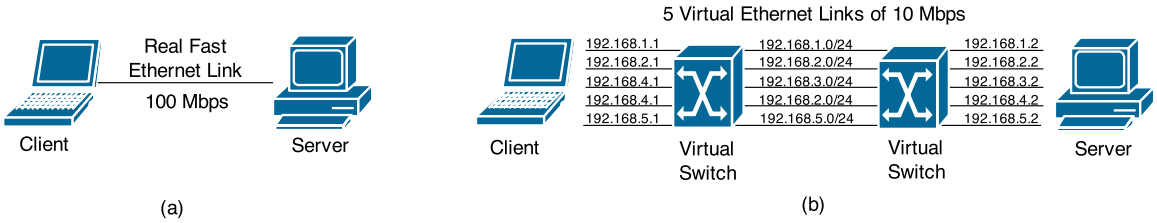



**Figure 4. Network topology of tests: (a) real network topology, and (b) virtualized network topology.**

| VM | Processor | RAM | Disk | Host OS | Virtual OS |
|---|---|---|---|---|---|
| Server | Core i5 | 6 GB | 1TB | Windows 7 | Ubuntu 12.04 LTS |
| Client | Dual Core | 1GB | 320GB | Debian Squeeze | Debian Squeeze |

**Table 1. Hardware-software configuration of the emulated scenario with VMware.**

desired event. The watcher can be stopped by the function `ev_io_stop()`. The synchronization occurs similarly to a barrier for threads, which is implemented by the function `ev_loop()`. This function would be equivalent to the abstract one $wait\_async\_return()$, in $gather$. During the event loop, all the initialized watchers check the pending events for their descriptors and activate the callbacks to handle them.

## 4. Obtained Results

We conducted experiments in emulated network to evaluate multipathing solutions as function of the number of paths in a session. We considered two aspects:

i. **Performance Gain** (goodput) with resource aggregation.
ii. **Cost of resource consumption** of memory and CPU.

### 4.1. Emulated Scenario

The choice of an emulated scenario rather than a real one was due to the easy implementation of a fully controlled test environment, free from noise and interference, besides the low cost of deployment.

The experiments were executed with the emulator VMware[5]. The emulated network topology is illustrated in Figure 4(b), which was virtualized on the real topology illustrated in Figure 4(a). The nodes Client and Server are virtual machines (VM) emulated in two different hosts. Table 1 presents the hardware-software configuration details.

We connected the machines point-to-point with a Fast Ethernet link, which is limited in 100 Mbps. Then, over this real topology, we created a virtual one. VMware allows to virtualize multiple collision domains with bridge configurations. Then, we created virtual switches so that the real Fast Ethernet link was divided into 5 virtual Ethernet links, each one limited in 10 Mbps. In both client and server, for each virtual link, the respective virtual network interface controller was connected in the virtual switch. There was an IP network for each virtual collision domain, as shows Figure 4(b).

Although the network topology utilized to evaluate multipathing is simple, it consists of disjoint and isolated links which allow each path to flow on a particular link

[5]`http://www.vmware.com/`

without compete to each other. This is a suitable scenario to obtain the top performance that multpathing protocols can get, since they work better over alternative paths like these. Once our focus in this paper is to determine the boundary of performance gain, we did not concern about using links of different capacities, parametrizing errors and latencies, or path selection.

### 4.2. Methodology

To evaluate MPTCP we load a VM with its Linux Kernel implementation (Kernel 3.5 MPTCP v0.88[6]). We implemented 5 versions of the proposed architecture, each one is named according to the used technology: Epoll, Epoll64, Threads, Posix AIO, and Libev. Epoll64 is implemented with Epoll technique, however, it is based on a variant $scatter$. When dividing the message by the number of available paths, we give the chunk of the message to be transmitted on the path. In Epoll64, the message chunk is fragmented in small sub-chunks of 64 KB in length regardless the number of paths. We intended to investigate the influence of transmitting small chunks in a multipath session.

To evaluate the implementations, an ordinary client-server application was created to use the access diversity of 5 virtual links. In this application, the client sends messages of varying length to the server by using a multipath session. The message size varies from 2KB and increases exponentially up to 1MB. The multipath session was established between the nodes, where each path is a TCP connection over the respective physical and logical network. In this scenario, experiments were executed to evaluate the performance gain and the cost of resource consumption. For each message size transmitted between the client and server, we collected at least 140 measurements of transmission latency, CPU utilization, and memory consumption.

### 4.3. The gain of resource aggregation

We aim to obtain the goodput of different techniques for multipathing by varying the number of paths in a session. To do so, we collected on the client side the latency for sending messages to the server and the respective message sizes.

From the results in Figure 5, we did not observe significant variation on the goodput of different message sizes, except for Epoll-based implementations that did provide apparent variable behaviour when changing the message size. Epoll is based on non-blocking I/O, so that every single I/O operation requires the application to poll descriptors to check the ones ready for reading and/or writing. Although we provide for a barrier synchronization, the readiness of the polled descriptors varies according to the demand for I/O operations and the respective OS management. We also observed that small message chunks do not resonate on the goodput, since Epoll and Epoll64 provide similar performance, as shows Figure 5.

The average goodput grows linearly as the number of paths, as show the dashed dot-lines over the boxplots in Figure 6(a). Figure 6(b) presents the gain factor when comparing a session of $n$ paths with respect to the gain of a session of $n-1$ paths. The gain factor $g_f$ is calculated as the ratio:

$$g_f = \frac{\bar{g_n}}{\bar{g_{n-1}}}, \tag{1}$$

[6]Available in: `http://multipath-tcp.org/`

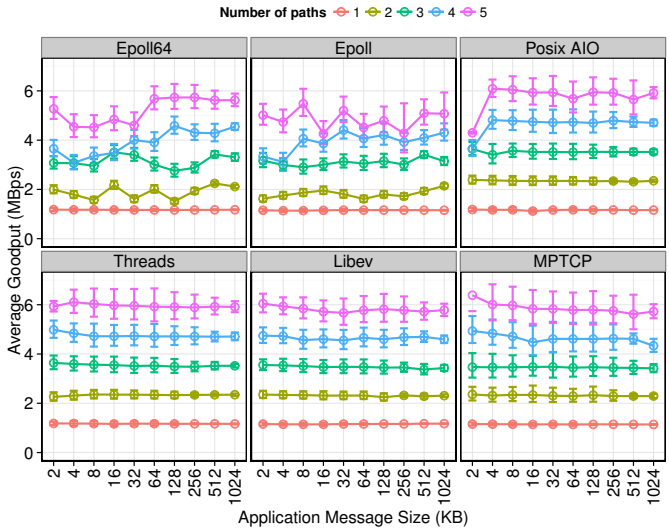



**Figure 5. Average Goodput for I/O technique and number of paths, both as function of the message size.**

where $\bar{g_n}$ and $\bar{g_{n-1}}$ are the average goodputs of sessions that hold $n$ and $n-1$ paths, respectively.

Except for Epoll-based implementations, which have variable behaviour due to the polling operation accomplished by the application, we observed that the gain factor is higher when the number of paths increases from one to two, and the factor decreases from two to three paths, and so on. The gain factor decreases when the number of paths increases because the overhead to manage multiple paths also increases as the number of paths, so that such overhead impacts on the performance.

From the boxplots in Figure 6(a), we observed that MPTCP provides similar performance than the user space-based solutions, such as Posix AIO, Posix Threads, and Libev. Although MPTCP works at the kernel space and prevents context switching during I/O to the Application Layer, the transmission capacity of all paths is combined and managed together by the Coupled Congestion Control [Raiciu et al. 2011b]. On the other hand, user space-based multipath solutions have multiple TCP connections, so that each path capacity is managed separately by the ordinary TCP Congestion Control, and the goodput is then the sum of the capacity of each path. However, MPTCP congestion control improves throughput in a such way that the multipath flow should perform at least as well as a single path flow would on the best of the paths available [Raiciu et al. 2011b]. In addition, each path flows separately in a virtual ethernet link in our emulated network topology, so that there is no competition among TCP flows.

The most efficient technique was Libev, enabling a gain factor of $1.62$ on average per additional path in a session. Epoll enables the lowest gain, increasing the goodput by a gain factor of $1.45$ on average. Second lowest one was Epoll64, with $1.46$ of gain factor on average. Worker-dispatcher thread-based multipathing and Posix AIO, both ones use the support of Posix Threads, provide very similar performance, with $1.53$ and $1.52$ of gain factor on average, respectively. Finally, MPTCP provides a gain of a factor of $1.53$ per path.

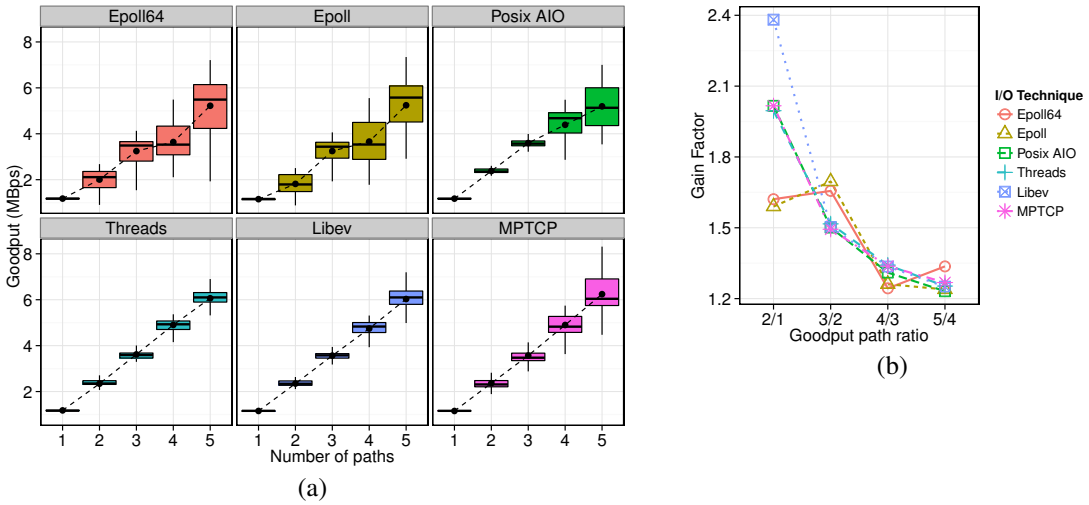



**Figure 6. Analysis of Goodput: (a) boxplot for each I/O technique as function of the number of paths; (b) goodput gain factor $g_f$ as function of the path ratio.**

### 4.4. Resource consumption

Given the gain of resource aggregation, we assessed the cost of the server's session multipathing in terms of memory consumption and CPU utilization. We collected the measurements at the server side, since the server is the entity that provides the service, hence, the one that has its resource consumed by the demand of client requests.

Although CPU and RAM are resources cheap today, mobile devices (from smartphones to mobile robots) are still of limited resource capacity, mostly energy, and the occupations of CPU and RAM impact directly on the node's energy consumption.

#### 4.4.1. Memory Consumption

The current RAM consumption information of the server process was obtained from the file `/proc/[pid]/status`[7] - given by the variable `VmSize`. Figure 7 illustrates the amplitude of server memory consumption as function of the number of paths. The largest memory consumers were the multipath techniques based on Posix Threads, such as our Threads and Posix AIO. The results show the greater the number of paths in a session, the greater the number of workers managed by a dispatcher, hence, the greater the expansion of the *memory heap* to handle multiples stack for different thread control blocks. Maintaining multiple threads to perform I/O is expensive and scales poorly[8] and other alternative models of I/O, such as event-driven, are often preferable [Kerrisk 2010].

Event-driven multipath techniques are quite lightweight with respect to memory consumption, since their event handlers are based on single-thread. This is confirmed by the results in Figure 7. The amplitudes of memory consumption of event-driven multipath ranged from 2MB to 5MB for Epoll and Libev. Epoll64, with its 64KB fragments of message, has required 10 to 12 MB of memory.

[7] `http://linux.die.net/man/5/proc`
[8] `http://man7.org/linux/man-pages/man7/aio.7.html`

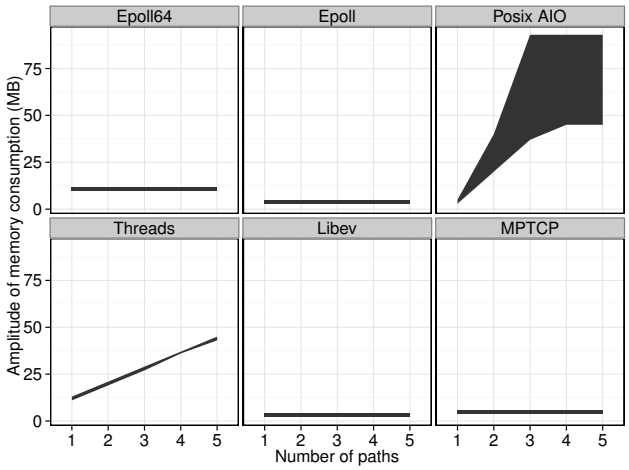



**Figure 7. Amplitude of memory consumption as function of the number of paths.**

### 4.4.2. CPU Utilization

The CPU usage reported by the VM's OS is relative to the CPU resources allocated to it by the host that virtualizes it. Therefore, for individual processes running on the VM's OS, the CPU utilization of such processes should still accurately reflect the relative proportion of the CPU time on the VM's OS.

We used `pidstat`[9] to obtain CPU utilization. This tool allows for monitoring individual tasks managed by the Linux Kernel. Similarly to the memory consumption test, we collected the server's CPU utilization while it remained running during the experiments.

We observed that Epoll and Epoll64 have required significant use of CPU, obtaining CPU utilization greater than 50% on average. Both techniques have consumed more kernel level CPU than any other. This is because of Epoll's event management uses mechanisms and facilities of the kernel for event notification. Epoll operates as a configurable kernel object in the user space, so that migrating from user to kernel mode is costly. Epoll64 often deliveries to the Transport Layer buffers (placed in the kernel space) application data containing the sub-blocks of 64KB. This implies more system calls, context switching, kernel event notifications, and, therefore, more CPU utilization.

The CPU utilization increases as the number of paths, as shows Figure 8. Except for Epoll-based techniques, there was little variation in consumption among techniques, so that Posix AIO, Posix Threads, Libev and MPCTP require similar CPU utilization. The average cost of adding a path in a session is 3.61%, 4.40%, 2.23% and 4.85% of CPU utilization for Posix AIO, Threads, Libev, MPTCP, respectively.

## 5. Conclusions and future directions

Finding out ways by which resources available in computer networks are fully utilized, with minimal or no impact on the existing logical and physical network infrastructure, is a big challenge when users and services on the Internet are constantly growing. Changes

[9] `http://linux.die.net/man/1/pidstat`

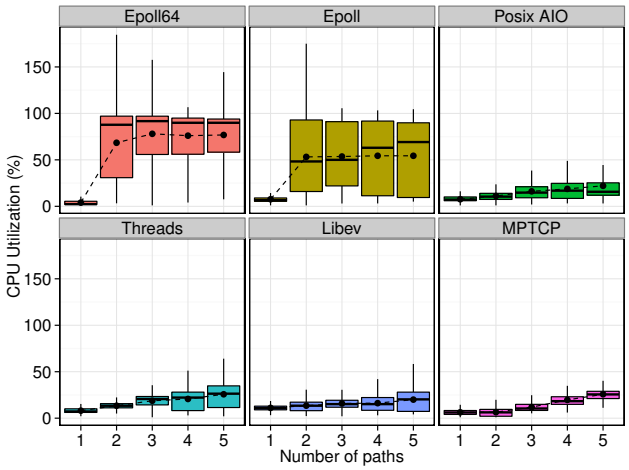



**Figure 8. CPU utilization as function of the number of paths.**

in already deployed infrastructures result in different costs, mostly in financial and operational spectrum.

When looking at the problematic of protocol implementation under the perspective of the general assumptions about OS' memory address spaces, a dilemma arises: we have to renounce ease of deployment and maintenance to obtain better performance, or vice-versa. In this paper, we propose multipath sessions that attempt to leverage the benefits of these two worlds. We provide (a) an investigation on different techniques to implement user space session multipathing in Linux systems, (b) an analysis on per path gain, (c) its counterpart measured through resource consumption, and (d) we point out the most suitable user space asynchronous IO technique that provides the best cost/benefit. We validate our research hypothesis when comparing experimental results of session multipathing with the ones of the general kernel space solution, MPTCP. We found that user space multipathing was able to provide performance slightly better than the kernel space one.

We consider the suitable session multipathing technique the one that provides lower resource consumption and, in the meantime, performance gain with higher goodput. From the results obtained in experiments in emulated environments, we point out Libev as the best cost-benefit, with higher performance and lower resource consumption. We also observed that Epoll and Threads were prohibitive techniques for session multipathing. Epoll-based multipathing requires a lot of CPU utilization, while Thread-based multipathing requires excessive memory.

Security issues involved in the session multipathing, particularly the session establishment, require further study. We also address in future works the analysis of energy consumption as function of the number of paths in session.

## References

Altman, E., Barman, D., Tuffin, B., and Vojnovic, M. (2006). Parallel TCP Sockets: Simple Model, Throughput and Validation. In *INFOCOM 2006: 25th IEEE International Conference on Computer Communications*, pages 1–12.

Barré, S., Bonaventure, O., Raiciu, C., and Handley, M. (2010). Experimenting with multipath TCP. *ACM SIGCOMM Computer Communication Review*, 40(4):443–444.

Carns, P., Ligon III, W., Ross, R., and Wyckoff, P. (2005). BMI: A network abstraction layer for parallel I/O. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 8–pp.

Ford, A., Raiciu, C., Handley, M., Barre, S., and Iyengar, J. (2011). Architectural Guidelines for Multipath TCP Development. *RFC 6182 - Internet Engineering Task Force*.

Ford, A., Raiciu, C., Handley, M., and Bonaventure, O. (2013). TCP Extensions for Multipath Operation with Multiple Addresses. *RFC 6824 - Internet Engineering Task Force*.

Habib, A., Christin, N., and Chuang, J. (2006). Taking Advantage of Multihoming with Session Layer Striping. In *INFOCOM 2006: 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6.

Hsieh, H.-Y. and Sivakumar, R. (2002). pTCP: An end-to-end transport layer protocol for striped connections. In *10th IEEE International Conference on Network Protocols*, pages 24–33.

Kerrisk, M. (2010). *The Linux programming interface*, chapter 29 - Threads: Introduction. No Starch Press.

Kimura, B. Y. L., Guardia, H. C., and Moreira, E. S. (2012). Disruption-tolerant sessions for seamless mobility. In *IEEE Wireless Communications and Networking Conference*, pages 2412–2417.

Kimura, B. Y. L., Guardia, H. C., and Moreira, E. S. (2013). A session-based mobile socket layer for disruption tolerance on the internet. *Mobile Computing, IEEE Transactions on*, PP(99):1–14.

Magalhaes, L. and Kravets, R. (2001). Transport level mechanisms for bandwidth aggregation on mobile hosts. In *Network Protocols, 2001. Ninth International Conference on*, pages 165–171.

Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. (2011a). Improving datacenter performance and robustness with multipath TCP. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 266–277.

Raiciu, C., Handly, M., and Wischik, D. (2011b). Coupled Congestion Control for Multipath Transport Protocols. *RFC 6356 - Internet Engineering Task Force*.

Raiciu, C., Niculescu, D., Bagnulo, M., and Handley, M. J. (2011c). Opportunistic Mobility with Multipath TCP. In *Proceedings of the Sixth ACM International Workshop on MobiArch*, pages 7–12.

Sivakumar, H., Bailey, S., and Grossman, R. L. (2000). PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *The 2000 ACM/IEEE conference on Supercomputing*, page 37.

Yildirim, E., Yin, D., and Kosar, T. (2009). Balancing TCP Buffer vs Parallel Streams in Application Level Throughput Optimization. In *The Second International Workshop on Data-aware Distributed Computing*, DADC '09, pages 21–30.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 12*

## *Sistemas P2P e Redes Sociais Online*

# Você Estava *Online*? Reconstruindo Sessões de Usuários em Sistemas de Larga Escala na Internet

**Weverton Luis da Costa Cordeiro[1,2], Rodrigo Brandão Mansilha[1], Flávio Roberto Santos[1], Luciano Paschoal Gaspary[1], Marinho Pilla Barcellos[1]**

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brasil
{wlccordeiro, rbmansilha, frsantos, paschoal, marinho}@inf.ufrgs.br

[2]Instituto Federal do Pará em Itaituba (IFPA)
Itaituba – PA – Brasil
weverton.cordeiro@ifpa.edu.br

***Abstract.*** *The comprehension of users' behavior is paramount for evaluating improvements to networked, distributed systems. To this end, several strategies have been proposed to obtain traces, based on the capture of usage information, which can then serve for evaluation purposes. One main strategy consists of taking snapshots of online users, using instrumented clients. In spite of its popularity, related proposals have fallen short in ensuring accuracy of obtained data. For a variety of reasons, users may fail to appear in some snapshots, although online. In this paper, we propose a methodology to correct ill-collected snapshots and build more accurate traces from them. In summary, we estimate the probability that a given snapshot is missing some users. The snapshot is corrected if the probability exceeds a given threshold. We use ground-truth data to assess the effectiveness of our methodology. The impact of our proposal is evidenced by means of an analysis of traces from a BitTorrent swarm.*

***Resumo.*** *A compreensão do comportamento de usuários é fundamental para avaliar melhorias a sistemas em rede e distribuídos. Para este fim, várias estratégias tem sido propostas para obter traços, baseadas na captura de informações de uso, os quais podem então servir para fins de avaliação. Uma estratégia básica consiste em obter fotos (*snapshots*) dos usuários* online*, usando agentes instrumentados. Apesar de popular, as propostas relacionadas tem sido incapazes de garantir a acurácia dos dados obtidos. Por diversos motivos, os usuários podem não aparecer em alguns* snapshots*, embora* online*. Neste artigo é proposta uma metodologia para corrigir imprecisões em* snapshots *e construir traços precisos a partir deles. Em resumo, é estimada a probabilidade de que um* snapshot *falhou em "capturar" um usuário* online*. O* snapshot *é corrigido caso a probabilidade exceda um determinado limiar. A eficácia da metodologia é avaliada com base em um conjunto de dados de referência. O impacto da proposta é demonstrado em uma análise de traços de enxames BitTorrent.*

## 1. Introdução

Há um entendimento (explícito ou implícito) na comunidade de que é fundamental compreender o comportamento dos usuários em um determinado sistema antes de se propor melhorias ao mesmo. No âmbito de redes e sistemas distribuídos, esse entendimento tem motivado a investigação de modelos analíticos que permitam analisar o comportamento dos usuários em diversas subáreas; exemplos incluem recuperação de conteúdo em redes orientadas a conteúdos [Rosensweig et al. 2013] e mobilidade em redes celulares [Sridharan and Bolot 2013]. Nesse contexto, um dos maiores desafios científicos é capturar informações precisas sobre

sessões *online*[*] de usuários em sistemas de larga escala na Internet. Esse desafio torna-se mais complexo devido à variedade de paradigmas de sistemas existentes, por exemplo redes par-a-par [Barcellos and Gaspary 2006], redes sociais [Teles et al. 2013], redes de sensores [Ruiz et al. 2006] e Internet das Coisas [França et al. 2011].

Há diversos esforços de pesquisa na direção de novas abordagens para obter traços com base na captura de informações sobre sessões *online* de usuários [Yoshida and Nakao 2011a, Hoãÿfeld et al. 2011]. Além de ajudar na formulação de modelos analíticos, tais traços podem ser úteis como entrada para simulações. Em resumo, essas abordagens podem ser mapeadas em três estratégias básicas: 1) coleta de informações diretamente do banco de dados do sistema alvo [Guo et al. 2007]; 2) instalação de *plug-ins* nos dispositivos dos usuários [Alexa Internet, Inc. 2012]; e 3) usar monitores para capturar periodicamente fotos (*snapshots*) dos usuários, e usá-las para sintetizar traços [Hoãÿfeld et al. 2011].

A terceira estratégia tem recebido bastante atenção [Yoshida and Nakao 2011b, Cuevas et al. 2010, Blond et al. 2010], o que pode ser explicado pela maior autonomia que ela oferece; ao contrário das demais, ela não requer acesso privilegiado a nenhuma entidade do sistema. Por esse motivo, neste artigo nos concentramos na terceira estratégia.

**Definição do problema.** Vários estudos tem mostrado as potencialidades da utilização da estratégia baseada em *snapshots* em uma série de sistemas de larga escala na Internet [Hoãÿfeld et al. 2011, Mansilha et al. 2010]. Apesar de promissora, essa estratégia torna-se suscetível a imprecisões quando *snapshots* falham em capturar parte dos usuários *online* no sistema. Usuários *online* podem não aparecer em snapshots por limitações técnicas do sistema alvo. Em BitTorrent, um exemplo é o tamanho máximo das listas de pares, $L_{max}$. Caso o número de usuários for maior que $L_{max}$, alguns deixarão de ser reportados. Mesmo obtendo várias listas, ainda há a chance de que algum usuário não apareça em alguma. Isso representa um importante problema, posto que traços gerados com tais imprecisões podem levar a conclusões erradas sobre as características desejáveis para um sistema.

**Contribuição.** Neste artigo, nós abordamos o problema da construção de traços mais acurados a partir de um conjunto de *snapshots* imprecisos. Para esse fim, nós propomos uma metodologia que mensura a probabilidade de que um determinado *snapshot* é impreciso (ou seja, informa incorretamente que um usuário está *offline*) em um dado instante; caso a probabilidade supere um limiar estabelecido, o *snapshot* é corrigido. Nossa metodologia é agnóstica às propriedades específicas de sistemas e ao comportamento sazonal de usuários, podendo ser aplicada em diversos contextos. São apresentadas avaliações a respeito de como essa metodologia permite melhorar a precisão de traços de sessões *online* de usuários. Os traços resultantes (corrigidos) são analisados considerando métricas como recorrência e duração.

O restante deste artigo está organizado como segue. Na Seção 2 é apresentada uma discussão sobre o desafio enfrentado neste artigo e a ideia investigada para superá-lo. Na Seção 3 é descrito o modelo de sistema, premissas e limitações relacionadas à metodologia proposta. Na Seção 4 é introduzida a metodologia para identificação e correção de falhas em *snapshots* de sessões *online* de usuários. Na Seção 5 é avaliada a eficácia da solução enquanto que na Seção 6 é dada ênfase ao seu impacto na análise de traços reais de enxames BitTorrent. Por fim, na Seção 7 é discutida a literatura relacionada, ao passo que na Seção 8 são apresentadas as conclusões e trabalhos em andamento.

## 2. Contextualização e Motivação

A seguir apresentamos uma visão geral do *modus operandis* da estratégia abordada neste artigo, usando a Figura 1 como base. Nesta figura, os rótulos $v_{1..3}$ representam os usuários que foram

[*]No contexto deste artigo, entende-se por sessão *online* o período de tempo compreendido entre a chegada do usuário no sistema (i.e., *join* ou *login* do usuário no sistema) e a sua saída (i.e., *leave* ou *logout*). Outro aspecto que caracteriza uma sessão de usuário é o instante em que esses eventos de chegada e saída ocorreram.

vistos pelo menos uma vez no sistema; os rótulos $s_{1..14}$ representam os *snapshots* capturados, em intervalos regulares $\Delta t$, pelos monitores. Na figura, 1 indica que um determinado usuário $v_i$ foi visto no sistema quando o *snapshot* $s_t$ foi capturado (um *snapshot* "positivo"), enquanto que 0 (um *snapshot* "nulo") representa o contrário. Construir o traço das sessões *online* (dos usuários) com base na informação representada na Figura 1 é uma tarefa trivial; para ilustrar, as sessões *online* identificadas nesse conjunto de *snapshots* é destacado em cinza na figura.

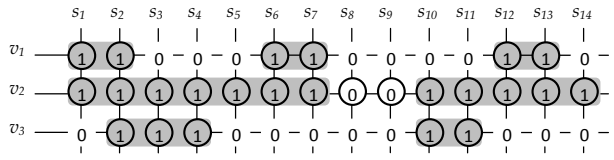
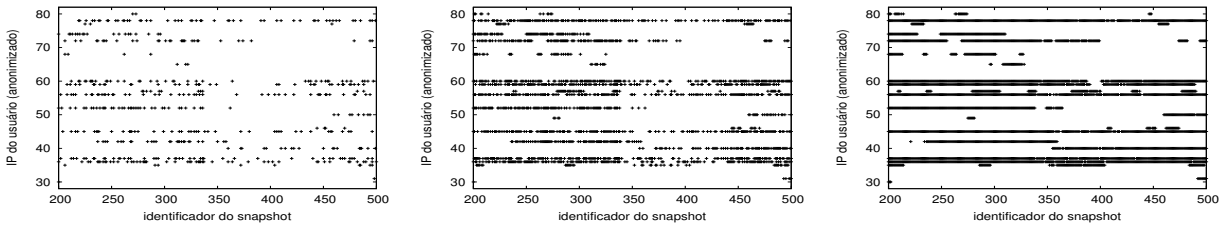



**Figura 1. Usuários ($v_i$) vistos *online* em *snaphots* ($s_t$)**

A Figura 1 ilustra também o problema abordado neste artigo. Considere que círculos na figura indicam que o usuário efetivamente estava *online* no sistema no instante da captura do *snapshot*. Note que os *snapshots* $s_8$ e $s_9$ falharam em capturar o usuário $v_2$. Nesse caso, o traço resultante indicará duas sessões para $v_2$, enquanto que na realidade o mesmo teve apenas uma.




(a) 1 monitor (b) 5 monitores (c) 50 monitores

**Figura 2. Conjunto de *snapshots* apresentado em três diferentes resoluções**

Para mitigar esse problema, uma alternativa seria aumentar a quantidade de recursos (no caso, monitores) empregados na captura de *snapshots*. Como cada monitor é capaz de "fotografar" apenas uma fração dos usuários *online*, a precisão do traço resultante torna-se diretamente proporcional à quantidade de monitores utilizados; esse aspecto é ilustrado na Figura 2. Uma direção proposta por Zhang et al. [Zhang et al. 2011] para resolver esse problema é super-dimensionar a quantidade de monitores usados para "fotografar" o sistema. Portanto, um desafio adicional abordado neste artigo é melhorar a resolução de traços sem, no entanto, aumentar a quantidade de recursos necessária para se obtê-los.

## 3. Modelo e Notação do Sistema

Nesta seção são apresentados o modelo de sistema e a definição de *snapshot* adotados. Além disso, são discutidas premissas e limitações relacionadas à metodologia proposta.

### 3.1. Componentes e Operação do Sistema

Seja $G = \langle V_t, E_t, B \rangle$ um sistema formado por nós $V_t = \{v_1, \ldots v_m\}$, arestas $E_t \subseteq V_t \times V_t$ e um conjunto de *bootstraps* $B = \{b_1, \ldots b_n\}$. Cada nó corresponde a um usuário *online* no sistema no instante $t$. As arestas indicam ciência de presença *online*: uma aresta de $v_i$ para $v_j$ indica que $v_i$ está ciente que $v_j$ está *online* no instante $t$. Por "ciente" entende-se que $v_i$ se comunicou com $v_j$ recentemente, ou que $v_i$ aprendeu essa informação de outra entidade (*bootstrap* ou usuário). O *bootstrap* é responsável pela admissão/registro de usuários no sistema. Assume-se que $b_k$ é ciente sobre a presença de $v_i$ no sistema no instante $t$ se e somente se $b_k$ foi responsável pela

admissão/registro de $v_i$ ou $v_i$ contatou $b_k$ (por outro motivo). Seja $e(v_i) \subseteq V_t$ ($e(b_k) \subseteq V_t$) o conjunto de usuários *online* sobre os quais $v_i$ ($b_k$) possui conhecimento.

**Obter lista de pares.** Essa requisição, lançada por $v_i$ para $b_k$ (ou $v_j$), consiste em obter uma (sub)lista $L$ dos usuários *online* que $b_k$ (ou $v_j$) está ciente; $L(v_i, x)$ denota a (sub)lista obtida por $v_i$ de $x$ ($x$ pode ser $b_k$ ou $v_j$). O tamanho máximo de tal lista é $L_{max}$. Note que $v_i$ pode consultar qualquer usuário $v_j$ desde que $v_i$ esteja ciente de $v_j$. Por definição, $L(v_i, b_k) \subseteq e(b_k)$; similarmente, $L(v_i, v_j) \subseteq e(v_j)$.

### 3.2. Construção de Traço

Seja $\mathcal{S}$ um conjunto de *snapshots* dos usuários em $G$, coletados em um dado período. Um *snapshot* $s_j \in \mathcal{S}$ é uma lista dos usuários vistos *online* no instante $t_j$. Um *snapshot* é capturado a cada $\Delta t$ unidades de tempo, nos instantes $t_0, t_1, \ldots$, de modo que $t_j = t_0 + j \cdot \Delta t$. Seja $V$ o conjunto dos nós vistos *online* pelo menos uma vez no sistema; por definição, $V_t \subseteq V$. A partir de $\mathcal{S}$, é possível construir uma matriz $M_{|V|,|\mathcal{S}|}$, na qual $m_{i,j}$ representa o estado do usuário $v_i$ no *snaphot* $s_j$: 1 se $v_i \in s_j$ ("*snapshot* positivo"), ou 0 caso contrário ("*snapshot* nulo").

Para construir traços de sessões *online* de usuários, são empregados um conjunto $V' \subseteq V$ de usuários especiais – chamados de *monitores* – para capturar *snapshots*. Um monitor $v_i \in V'$ pode contactar *bootstraps* e/ou outros usuários (ou ambos) para construir (parte de) um *snapshot*. Um *snapshot* $s_t$ é formado pela união das listas obtidas pelos monitores: $s_t = \bigcup L(v_i, x)$. Note que o custo para obter $L(v_i, x)$ é não desprezível (por exemplo, devido ao consumo de banda, latência, frequência máxima permitida de solicitações, entre outros).

### 3.3. Premissas do Modelo

**Duração das Sessões *Online*.** Com base no estudo de Yao et al. [Yao et al. 2009] e Steiner et al. [Steiner et al. 2009], consideramos que quanto maior a duração de uma sessão *online* é (no passado), maior ela tende a ser (no futuro). Esse é o caso dos usuários assistindo a *streaming* de vídeo ou baixando algum conteúdo grande (e.g. a imagem de uma distribuição do GNU/Linux).

**Composição de Listas de Pares.** Assume-se que a probabilidade de um usuário compor uma dada lista de pares é uniforme. Essa premissa está relacionada a uma variação do problema do coletor de cupons com tamanhos de amostras aleatórias [Kobza et al. 2007], e tem sido considerada em vários casos [Hoãÿfeld et al. 2011, Zhang et al. 2011]. A investigação de casos onde as listas são formadas de forma enviesadas (ou seguindo uma distribuição não uniforme) é vislumbrada como direção para trabalhos futuros.

### 3.4. Limitações Específicas do Sistema Alvo

**Ciência de Usuários *Online*.** Idealmente, um *snapshot* capturado no instante $t$ deveria ser igual a $s_t \subseteq V_t$. Contudo, pode haver casos em que *snapshots* reportem incorretamente um usuário como *online*. Os motivos para isso ocorrer são vários; por exemplo, $v_i$ pode desconectar do sistema de forma "não graciosa" (e.g. travar), ou $b_k$ pode não receber a mensagem de desconexão de $v_i$. Esses casos são incomuns, podendo ocorrer devido a um aspecto específico no projeto do sistema. Por esse motivo, esse aspecto é vislumbrado como direção futura de pesquisa.

**Abrangência da Lista de Pares.** Idealmente, $L = V_t$ (ou $s_t = V_t$) para qualquer instante $t$. Contudo, limitações específicas do sistema alvo podem impedir que isso aconteça. Por exemplo, dispositivos limitados (*hardware* embarcado ou sensores) podem não ter memória suficiente para manter grandes listas de pares. Além disso, alguns sistemas limitam o tamanho da lista de pares $L_{max}$ enviada por razões de otimização ou restrições de rede [Mansilha et al. 2010]. Pode até mesmo ser inviável para uma entidade construir e transmitir a lista completa de pares conhecidos quanto $|V_t|$ atinge a ordem de milhares ou milhões.

## 4. Reconstruindo Sessões de Usuários

Nesta seção apresentamos a metodologia para identificar *snapshotst* imprecisos. Em seguida, descrevemos o algoritmo proposto para reconstruir traços mais acurados de sessões *online*.

### 4.1. Estimando a Probabilidade de Falha dos *Snapshots*

Assumindo que $(i)$ a estrutura do sistema (grafo de relações entre os usuários, entre usuários e entidades de *bootstrap*, etc.) não é conhecida, e $(ii)$ o processo de captura de *snapshots* não permite concluir qualquer informação relevante para a correção dos mesmos, nossa proposta baseia-se exclusivamente em utilizar a informação contida no conjunto de *snapshots* para estimar uma *probabilidade de falha* média $p$. O uso de fontes de informação específicas do sistema alvo para estimar $p$ é vislumbrado como perspectiva de trabalho futuro.

Seja $X_i$ o evento no qual um determinado usuário $v_i$ está *online* e presente no *snapshot* $s_j$ ($m_{i,j} = 1$); $Y_i$ o evento no qual $v_i$ não está presente em $s_j$ ($m_{i,j} = 0$), embora *online*; e $Z_i$ o evento no qual $v_i$ está *offline* ($m_{i,j} = 0$). A probabilidade de termos um *snapshot* falho para $v_i$ é dada por $P(Y_i) = p$, enquanto que a probabilidade de termos um *snapshot* correto é $P(X_i) + P(Z_i) = 1 - p$. O desafio enfrentado nesse artigo é estimar $P(Y_i)$, posto que os eventos $Y_i$ e $Z_i$ são indistinguíveis no *snapshot*.

Considerando $p$ a probabilidade de um usuário falhar em aparecer em um *snapshot*, o objetivo é estimar $p$ ($\hat{p}$) e usá-lo para corrigir o traço. Para esse fim exploramos duas observações chave. A primeira observação é que, quanto mais tempo um usuário permanece no sistema, mais se espera sua permanência [Yao et al. 2009, Steiner et al. 2009]; consequentemente, existe uma probabilidade maior de que um usuário *offline* em um *snapshot* após um período longo no sistema seja um evento do tipo $Y$ (ao invés de $Z$):

**Proposição 1.** *Seja $k$ uma sequência de* snapshots *"positivos" ($k \in \mathbb{N}^*$). A probabilidade de um* snapshot *"nulo" $k+1$ ser incorreto aumenta geometricamente com $k$ (processo de Bernoulli).*

*Demonstração.* Considere cada *snapshot* um ensaio de Bernoulli. Seja *falha* um ensaio no qual um determinado usuário *está online* e presente no *snapshot* (amostra positiva), e *sucesso* um ensaio no qual o mesmo usuário está *online*, mas *ausente* no *snapshot* (amostra falso negativa). Nesse caso, a probabilidade de ocorrer uma amostra falso positiva após $k$ amostras positivas consecutivas segue um processo de Bernoulli, com uma distribuição geométrica, e é definida por $1 - (1-p)^{k+1}$; por definição, $1 - (1-p)^{k+1}$ aumenta geometricamente com $k$. □

A segunda observação é que, quanto maior uma sequência de *snapshots* nulos, maior é probabilidade de que o usuário realmente tenha saído do sistema durante aquele período:

**Proposição 2.** *Seja $k$ uma sequência de* snapshots *nulos ($k \in \mathbb{N}^*$). A probabilidade de que todos eles sejam falhos diminui exponencialmente com $k$.*

*Demonstração.* Seja $Y_{i,j}$ o evento no qual $v_i$ está *online* mas ausente no *snapshot* $s_j$, e $P(Y_{i,j}) = p$ a probabilidade de tal evento. Considerando que os usuários possuem chance uniforme de serem selecionados para compor uma lista $L$, temos que os eventos de falha são estatisticamente independentes. Logo, usando as regras de probabilidade condicional e da cadeia, temos

$$P\left(\bigcap_{j=1}^{k} Y_{i,j}\right) = \prod_{j=1}^{k} P\left(Y_{i,j} \,\middle|\, \bigcap_{l=1}^{j-1} Y_{i,l}\right) \therefore p^n \tag{1}$$

, que por definição diminui exponencialmente com $n$. □

Essas proposições são empregadas para se obter $\hat{p}$ com erro $1-\alpha$, onde $\alpha \in (0,1)$. Nesse contexto, $\alpha \to 0$ representa uma abordagem mais conservadora na busca por *snapshots* falhos, diminuindo assim a chance de falsos positivos. De maneira oposta, $\alpha \to 1$ representa uma abordagem mais agressiva, na qual mais *snaphots* são corrigidos.

Começando pela Proposição 1, que valores de $p$ permitiriam um *snapshot* $k+1$ ser falho para um dado usuário? Para isso, fazemos $1-(1-p)^{k+1} \geq 1-(1-\alpha)$. A probabilidade $p$ (para $k$ *snapshots* consecutivos) que satisfaz essa desigualdade é dada pela Equação 2.

$$p \geq 1-(1-\alpha)^{\frac{1}{k+1}} \tag{2}$$

Os efeitos da Proposição 2 são ilustrados na Figura 3. A probabilidade de um dado *snapshot* nulo ser um falso negativo, para um usuário $v_i$, decresce com a quantidade de *snapshots* nulos precedentes. Note que essa probabilidade pode ser mensurada em ambas as direções (curvas 1 e 2). Assim, para verificar se $v_i$ tornou-se *offline* é necessário medir a probabilidade do *snapshot* $s_i$ (mediana) ser um falso negativo. O motivo é que $s_i$ possuirá a menor probabilidade de falha considerando a soma, nesse ponto, de ambas as curvas. Admitindo uma margem de erro de $1-\alpha$, uma sequência $k$ deve ser corrigida se $(1-p)^{\frac{k+1}{2}} \geq 1-\alpha$. A probabilidade $p$ que satisfaz essa desigualdade é dada pela Equação 3.

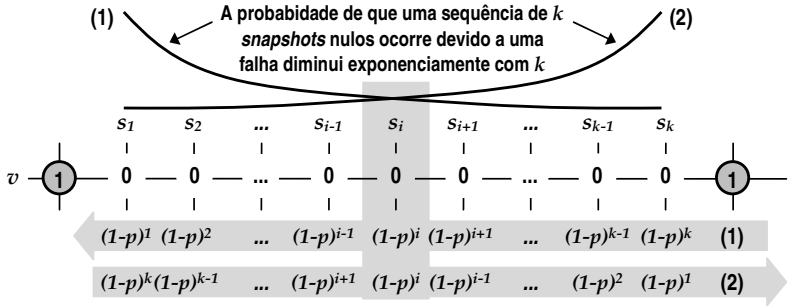



**Figura 3. Probabilidade condicional**

$$p \leq 1-(1-\alpha)^{\frac{2}{k+1}} \tag{3}$$

A probabilidade de falha para quaisquer sequências de *snapshots* é então calculada através da soma ponderada das probabilidades para cada sequência de tamanho $k$. Nesse caso, interessa buscar a menor probabilidade possível para essas sequências. A partir das Equações 2 e 3, define-se sistema não-linear abaixo

$$\begin{aligned} &\text{minimizar } \hat{p} \\ &\text{sujeito a } \hat{p} \geq \sum_{x=1}^{n} P[X=x] \cdot \left(1-(1-\alpha)^{\frac{1}{x+1}}\right) \\ &\hat{p} \leq \sum_{y=1}^{m} P[Y=y] \cdot \left(1-(1-\alpha)^{\frac{2}{y+1}}\right) \end{aligned}$$

Nesse sistema, $P[K=k]$ é a fração (normalizada) de sequências de tamanho $k$, $n$ é o tamanho da maior sequência de *snapshots* positivos, e $m$ é o tamanho da maior sequência de *snapshots* nulos; esses valores são obtidos a partir dos *snapshots* em análise. O valor de $\hat{p}$ que satisfaz o sistema é então a probabilidade de falha nesses *snapshots*. Note que o sistema

não pode ser solucionado quando não existir um valor para $\hat{p}$ que satisfaça as duas restrições simultaneamente. Nesse caso, não há dados suficientes para estimar a probabilidade de falha.

### 4.2. Corrigindo Traços

Para aprimorar a acurácia de um conjunto de *snapshots* $\mathcal{S}$, basicamente aplica-se a Proposição 2 usando o valor obtido de $\hat{p}$ para as sequências de *snapshots* nulos de um dado usuário $v_i$. Para entender esse processo, considere um conjunto $\mathcal{S}$ dado como entrada. Baseado na metodologia descrita na subseção anterior, propõe-se os passos descritos no Algoritmo 1 para aprimorar $\mathcal{S}$.

**Algorithm 1** Algoritmo para corrigir conjunto de *snapshots*

```
1: procedure CorrectSnapshotSet(S, α)                    ▷ conjunto de entrada e fator de erro
2:   pos_st.hist[] ← frequência tabulada de sequências de snapshots positivos em S
3:   nul_st.hist[] ← frequência tabulada de sequências de snapshots nulos em S
4:   pos_st.total ← número total de sequências de snapshots positivos em S
5:   nul_st.total ← número total de sequências de snapshots nulos em S
6:   pos_st.max_len ← comprimento da maior sequência de snapshots positivos em S
7:   nul_st.max_len ← comprimento da maior sequência de snapshots nulos em S
8:   p_pos ← 0
9:   for i ← 1 ... pos_st.max_len do
10:    p_pos ← p_pos + (pos_st.hist[i] · (1 − (1 − α)^(1/(i+1))))
11:  end for
12:  p_nul ← 0
13:  for j ← 1 ... nul_st.max_len do
14:    p_nul ← p_nul + (nul_st.hist[j] · (1 − (1 − α)^(2/(j+1))))
15:  end for
16:  if p_pos > p_nul then
17:    return null                                       ▷ é impossível estimar a probabilidade de falha
18:  else
19:    p̂ ← p_pos
20:    len ← ⌊ ln(1 − α) / ln(p̂) ⌋
21:    for all sequências s_{v,1...n} ∈ S de snapshots nulos para um dado usuário v_i do
22:      if length (s_{v,1...n}) ≤ len then              ▷ é provável que a sequência seja uma falha
23:        transforme s_{v,1...n} em uma sequência de snapshots positivos e substitua em S
24:      end if
25:    end for
26:    return S                                          ▷ conjunto de snapshots resultante
27:  end if
28: end procedure
```

O primeiro passo nesse procedimento (*procedure*) é obter as frequências tabuladas de tamanhos de sequências de *snapshots* positivos e nulos do conjunto de entrada $\mathcal{S}$ (linhas 2 e 3 do Algoritmo 1). Por exemplo, para o conjunto de *snapshots* apresentado na Figura 1, a frequência de sequências de *snapshots* positivos de tamanho 2 é 4 (*pos_st.hist* $[2] = 4$), e a de *snapshots* nulos com tamanho 5 é 1 (*pos_st.hist* $[5] = 1$).

O passo seguinte (linhas 4 e 5) é obter o número total de sequências de *snapshots* positivos e nulos do conjunto $\mathcal{S}$. No exemplo da Figura 1, esses totais são *pos_st.total* = 7 e *nul_st.total* = 4, respectivamente. O maior tamanho para cada uma dessas sequências também é extraído de $\mathcal{S}$ (linhas 6 e 7). No exemplo, eles são *pos_st.max_len* = 7 e *nul_st.max_len* = 5.

O próximo passo do Algoritmo 1 é estimar a probabilidade de falha para uma sequência de *snapshots* positivos ($p_{\hat{pos}}$, linhas 8-11) e nulos ($p_{\hat{nul}}$, linhas 12-15) de $\mathcal{S}$, e então resolver o sistema não-linear apresentado anteriormente (linhas 16-19). Note que caso o sistema não possa ser solucionado (devido às restrições relativas a $\hat{p}$ explicadas anteriormente), então o algoritmo retorna $\mathcal{S}$, sem modificá-lo (linha 17).

Caso o sistema possua solução, o passo seguinte é determinar qual é o tamanho *len* da maior sequência de valores nulos que pode ser considerada uma falha (linha 20). O raciocínio por trás da equação $len = \lfloor {\ln(1-\alpha)}/{\ln(\hat{p})} \rfloor$ explora a observação descrita na Proposição 2. Considerando $p$ obtido, para que tamanhos de sequências de snapshots nulos as mesmas podem ser

interpretadas como falhas, admitindo um erro de $1 - \alpha$? Para responder essa pergunta, fazemos $p^{len} \geq (1 - \alpha)$. Por fim, todas as sequências de valores nulos que possuem tamanho menor ou igual a *len* são alteradas para sequências de *snapshots* positivos (linhas 20 - 25).

Para ilustrar esse processo, considere a Figura 1 novamente. Assumindo que $\hat{p} = 0.26$ foi obtido utilizando $\alpha = 0.95$, tem-se que *len* = 2 (linha 20). Desse modo, apenas sequências de *snapshots* nulos com tamanho menor ou igual a 2 serão corrigidas. Esse é o caso da sequência $s_{8...9}$ para o usuário $v_2$. Esse usuário passará então a ter apenas uma sessão *online* no traço resultante ($s_1$ – $s_{14}$). Esse resultado corresponde a realidade, conforme discutido na Seção 2.

Após os snapshots falhos para um dado usuário $v$ terem sido identificados e corrigidos (admitindo erro $1 - \alpha$), o conjunto de *snapshots* $\mathcal{S}$ modificado é retornado (linha 26).

## 5. Avaliação

Para avaliar a metodologia proposta, procurou-se responder a quatro perguntas de pesquisa: qual a efetividade da nossa metodologia na correção dos conjuntos de *snapshots*? Segundo, qual a incidência de correções falso positivas? Terceiro, quais são os cenários onde o uso da metodologia proposta é mais vantajoso? E quarto, qual o comportamento da nossa metodologia em face de diferentes configurações para $\alpha$? Para responder essas perguntas, foram executados experimentos considerando diversos conjuntos de *snapshots* e configurações. Nesta seção são discutidos os resultados mais proeminentes.

### 5.1. Conjunto de Dados de Referência

Nesse experimento, foi utilizado um conjunto de *snapshots* capturados de vários enxames BitTorrent [†]. Em resumo, foram empregados 50 monitores ($|V'| = 50$) para capturar um total de 2.880 *snapshots* da população de usuários *online*, durante 30 dias (iniciado no dia 15 de junho de 2013); o intervalo entre *snapshots* foi $\Delta t = 15$ minutos. No maior enxame, o número médio de usuários *online* foi 3.035, o máximo foi 5.887, e o mínimo foi 1.636. O desvio padrão foi 794,78. Cada enxame era mantido por 4 *bootstraps* (*trackers*). Uma visão parcial do conjunto de *snapshots* para esse enxame é apresentada na Figura 2(c).

O valor $|V'| = 50$ foi obtido seguindo a metodologia proposta por Hoãÿfeld et al. [Hoãÿfeld et al. 2011], a qual é baseada em uma variação do problema do coletor de cupons com amostras de tamanho $k > 1$ [Kobza et al. 2007]. De acordo com essa metodologia, a quantidade aproximada de respostas do *tracker* necessárias para se obter um *snapshot* completo de um determinado enxame ($E[X]$) pode ser obtida pela Equação 4.

$$E[X] \approx \frac{N \cdot h_N}{L_{max} \cdot |B|} \tag{4}$$

Na equação acima, $N$ corresponde ao número esperado de usuários e $h_N$ o N-$simo$ número harmônico. A resposta do *tracker* ("lista de pares" na terminologia usada neste artigo) possui um tamanho determinado por $L_{max}$. Considerando uma estimativa inicial de 4.000 usuários para a população do enxame, $L_{max} = 200$ e $|B| = 4$ (ou seja, 800 usuários obtidos a cada intervalo), obteve-se $E[X] \approx 44$ monitores; para tolerar eventuais falhas, foram adicionados 6 monitores extras, totalizando assim 50 monitores.

### 5.2. Métricas de Avaliação

Note que um *snapshot* $s_t$ é composto pela união de *snapshots* parciais capturados por cada monitor $v_i \in V'$. Assim, no restante desta seção denota-se como $\mathcal{S}_{|V'|}$ o conjunto de *snapshots*

[†]Informações sobre a metodologia de captura podem ser encontradas em [Mansilha et al. 2010]. O conjunto de dados usado no experimento está disponível em `http://www.inf.ufrgs.br/~wlccordeiro/idmgmt/`

capturados usando $|V'|$ monitores. Além disso, denota-se como $\mathcal{S}'_{|V'|,\alpha}$ o conjunto de *snapshots* corrigido usando a metodologia proposta, aplicando-se $\alpha = x$, e $\mathcal{S}_{gt}$ o conjunto de *snapshots* tomado como referência (*ground-truth*) para medir o desempenho da metodologia proposta. Para fins de avaliação, as seguintes métricas foram definidas:

- **Quantidade de *snapshots* positivos faltantes ($\omega$)**. Conforme a Seção 3, $M_{|V|,|\mathcal{S}|}$ é a matriz binária construída a partir de $\mathcal{S}$. Seja $P(\mathcal{S}_{|V'|}) = \{m_{i,j} | m_{i,j} = 1\}$ o conjunto de *snapshots* positivos de $\mathcal{S}_{|V'|}$. Então, podemos definir $\omega$ por

$$\omega = \left| P(\mathcal{S}_{gt}) \setminus P(\mathcal{S}_{|V'|}) \right| \tag{5}$$

  Em outras palavras, $\omega$ é o número de *snapshots* positivos de $\mathcal{S}_{gt}$ (*ground-truth*) cujos correspondentes em $\mathcal{S}_{|V'|}$ (i.e. para um mesmo usuário) são *snapshots* nulos.
- **Quantidade de *snapshots* modificados ($\phi$)**. Essa métrica contabiliza os *snapshots* positivos em $\mathcal{S}'_{|V'|,\alpha}$ que eram inicialmente nulos no conjunto $\mathcal{S}_{|V'|}$

$$\phi = \left| P(\mathcal{S}'_{|V'|,\alpha}) \setminus P(\mathcal{S}_{|V'|}) \right| \tag{6}$$

- **Quantidade de *snapshots* corrigidos corretamente ($\tau$)**. Essa métrica contabiliza os *snapshots* positivos em $\mathcal{S}'_{|V'|,\alpha}$ que, inicialmente nulos em $\mathcal{S}_{|V'|}$, coincidem com *snapshots* positivos em $\mathcal{S}_{gt}$ (*ground-truth*). Formalmente, temos

$$\tau = \left| \left( P(\mathcal{S}_{gt}) \setminus P(\mathcal{S}_{|V'|}) \right) \cap P(\mathcal{S}'_{|V'|,\alpha}) \right| \tag{7}$$

  Os valores possíveis para $\tau$ variam no intervalo $[0, \omega]$. O valor $\tau = \omega$ é um ótimo global; significa que todos os *snapshots* foram corrigidos corretamente.
- **Número de correções falso positivas ($\psi$)**. Uma correção falso positiva refere-se a um *snapshot* $s_t$ que foi erroneamente corrigido para um usuário $v_i$. Em outras palavras, $v_i$ de fato estava *offline* quando $s_t$ foi capturado, mas a metodologia considerou aquele *snapshot* nulo como um caso de falso negativo. Formalmente, temos

$$\psi = \left| P(\mathcal{S}'_{|V'|,\alpha}) \setminus P(\mathcal{S}_{gt}) \right| \tag{8}$$

  Os valores possíveis para $\psi$ variam no intervalo $[0, \phi]$. O valor $\psi = 0$ é um ótimo global; significa que nenhuma correção realizada foi incorreta.

### 5.3. Efetividade da Metodologia e Incidência de Falsos Positivos

Para avaliar a efetividade da proposta, tentou-se recriar o conjunto de *snapshots* $\mathcal{S}_{50}$ (conjunto de referência, ou *ground-truth*), tomando como entrada conjuntos de *snapshots* $\mathcal{S}_{|V'|}$ para $|V'|$ variados. Para fins de ilustração, é considerada uma visão parcial de $\mathcal{S}_{50}$ tal como mostrada na Figura 2(c). Nessa avaliação foi adotada $\alpha = 0.95$.

A Tabela 1 mostra um resumo dos resultados obtidos. Nessa tabela, a coluna *Conjunto* indica o conjunto de *snapshots* considerado.Os campos $p_{\hat{\text{pos}}}$ e $p_{\hat{\text{nul}}}$ indicam as probabilidades estimadas para as sequências de *snapshots* positivas e nulas, respectivamente. O campo *Corrigido?* indica se foi possível efetuar a correção. O campo *Comprimento* mostra o tamanho máximo para as sequências de *snapshots* nulos que foram corrigidas. Por fim, os campos *Métrica* $\tau/\omega$ e *Métrica* $\psi/\phi$ apresentam os respectivos resultados obtidos para cada conjunto.

**Avaliação dos *Snapshots* Corrigidos Corretamente.** Observe na Tabela 1 que nossa metodologia alcançou alto grau de precisão na correção daqueles *snapshots* construídos com menos monitores. A proporção de *snapshots* corrigidos corretamente variou entre 65% (para $\mathcal{S}_2$) e 87% (para $\mathcal{S}_{10}$). A maior acurácia medida nos experimentos foi 90%, para o conjunto de *snapshots* $\mathcal{S}_{15}$. Não foi possível corrigir $\mathcal{S}_1$, já que $p_{\hat{pos}} > p_{\hat{nul}}$ (conforme discutido na Seção 4).

A principal conclusão que pode ser extraída da discussão acima é que a nossa metodologia alcançou melhor acurácia justamente para os conjuntos de *snapshots* construídos com

**Tabela 1. Conjuntos de *snapshots* submetidos à correção, usando $\alpha = 0.95$**

| Conjunto | $p\hat{pos}$ | $p\hat{nul}$ | Corrigido? | Comprimento | Métrica $\tau/\omega$ | | Métrica $\psi/\phi$ | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Valores | Proporção | Valores | Proporção |
| $\mathcal{S}_1$ | 0.750257 | 0.656683 | Não | – | – | – | – | – |
| $\mathcal{S}_2$ | 0.720646 | 0.754073 | Sim | 9 | 1461/2242 | .6516 | 31/1492 | .0207 |
| $\mathcal{S}_3$ | 0.692409 | 0.817299 | Sim | 8 | 1442/1850 | .7794 | 44/1486 | .0296 |
| $\mathcal{S}_4$ | 0.664327 | 0.835471 | Sim | 7 | 1258/1617 | .7779 | 53/1311 | .0404 |
| $\mathcal{S}_5$ | 0.643048 | 0.849721 | Sim | 6 | 1107/1433 | .7725 | 53/1160 | .0456 |
| $\mathcal{S}_6$ | 0.623862 | 0.863142 | Sim | 6 | 1011/1238 | .8166 | 61/1072 | .0569 |
| $\mathcal{S}_7$ | 0.590806 | 0.868512 | Sim | 5 | 871/1023 | .8514 | 64/935 | .0684 |
| $\mathcal{S}_8$ | 0.555304 | 0.875651 | Sim | 5 | 717/814 | .8808 | 75/792 | .0946 |
| $\mathcal{S}_9$ | 0.543981 | 0.877639 | Sim | 4 | 636/741 | .8582 | 76/712 | .1067 |
| $\mathcal{S}_{10}$ | 0.522767 | 0.870914 | Sim | 4 | 540/618 | .8737 | 80/620 | .1290 |
| $\mathcal{S}_{15}$ | 0.473238 | 0.858767 | Sim | 4 | 324/360 | .9000 | 82/406 | .2019 |
| $\mathcal{S}_{20}$ | 0.449136 | 0.842509 | Sim | 3 | 193/226 | .8539 | 81/274 | .2956 |
| $\mathcal{S}_{25}$ | 0.420627 | 0.835212 | Sim | 3 | 140/159 | .8805 | 83/223 | .3721 |
| $\mathcal{S}_{30}$ | 0.391245 | 0.814221 | Sim | 3 | 70/85 | .8235 | 83/153 | .5424 |
| $\mathcal{S}_{35}$ | 0.337525 | 0.781824 | Sim | 2 | 29/35 | .8285 | 79/108 | .7314 |
| $\mathcal{S}_{40}$ | 0.320752 | 0.764094 | Sim | 2 | 14/17 | .8235 | 79/93 | .8494 |
| $\mathcal{S}_{45}$ | 0.302146 | 0.752152 | Sim | 2 | 3/5 | .6000 | 79/82 | .9634 |

menos recursos. Em outras palavras, nossa metodologia permite capturar traços mais acurados utilizando bem menos recursos que o estimado segundo Hoãÿfeld et al. [Hoãÿfeld et al. 2011].

**Falso-positivos.** Observe na Tabela 1 que a incidência de falso positivos é significativamente baixa. No caso de $\mathcal{S}_{10}$, a ocorrência de falso positivos foi de 12%, enquanto que para $\mathcal{S}_2$ a proporção foi de apenas 2%. De maneira similar à avaliação de *snapshots* corrigidos corretamente, a proporção de falso positivos diminui significativamente conforme diminui a resolução do conjunto de *snapshots*. Em resumo, nossa metodologia mostrou-se mais efetiva justamente nos cenários onde os *snapshots* foram capturados com menos recursos.

### 5.3.1. Avaliação dos Cenários Mais Vantajosos para Correção de *Snapshots*

A seguir é discutida uma abordagem para avaliar quais são os cenários com maior potencial para se beneficiar da nossa metodologia. Conforme discutido na Seção 4, o uso de menos monitores aumenta a probabilidade $p$ de algum usuário *online* não ser capturado. Um limite inferior para $p$ pode ser estimado para um sistema com população $|V|$, fotografada usando $|V'|$ monitores, e com lista de pares $L_{max}$, usando a Equação 9. Essa equação corresponde a uma formulação para estimar a probabilidade de falhas em um sistema ideal, com número de usuários online conhecido. Nesse caso, $p$ corresponde à fração de usuários que não é possível se enxergar em um *snapshot*, dado um número de monitores. Considerando o conjunto de *snapshots* descrito na Seção 5.1, o limite inferior de $p$ estimado para $\mathcal{S}_2$ é $0.90$, enquanto que para $\mathcal{S}_{45}$ é $0.0$.

$$p = 1 - \frac{\min(L_{max} \cdot |V'|, |V|)}{|V|} \tag{9}$$

A Equação 9 pode ser utilizada para mensurar quando a metodologia proposta pode (significativamente) aprimorar um dado conjunto de *snapshots*. Em resumo, é possível comparar a probabilidade de falha medida $p\hat{pos}$ com o limite inferior $p$. Caso $p \gg p\hat{pos}$, então $\mathcal{S}_{|V'|}$ é mais adequado para ser corrigido usando a metodologia. Isso porque a probabilidade que $\mathcal{S}_{|V'|}$ possa ter perdido algum usuário *online* é maior do que a probabilidade estimada pela nossa metodologia. Como resultado, é possível alcançar maior precisão com a nossa metodologia do que poderia ocorrer se $p \ll p\hat{pos}$. Por exemplo, considere-se $\mathcal{S}_2$. A partir da Tabela 1, tem-se $p\hat{pos} \approx 0.72$, menor que o limite inferior $p = 0.90$, e portanto um candidato para correção com alta precisão. De maneira oposta, considere-se $\mathcal{S}_{45}$. Nesse caso, tem-se $p\hat{pos} \approx 0.30$, signifi-

**Tabela 2. Influência do parâmetro $\alpha$ na correção de $\mathcal{S}_5$**

| Cenário | $\hat{p_{pos}}$ | $\hat{p_{nul}}$ | Comprimento | Métrica $\tau/\omega$ | Métrica $\psi/\phi$ |
|---|---|---|---|---|---|
| $\alpha = 0.05$ | 0.018886 | 0.038773 | 0 | $^{0}/_{1433}$ | – |
| $\alpha = 0.10$ | 0.038356 | 0.077806 | 0 | $^{0}/_{1433}$ | – |
| $\alpha = 0.20$ | 0.079275 | 0.156736 | 0 | $^{0}/_{1433}$ | – |
| $\alpha = 0.30$ | 0.123293 | 0.237008 | 0 | $^{0}/_{1433}$ | – |
| $\alpha = 0.40$ | 0.171157 | 0.318917 | 0 | $^{0}/_{1433}$ | – |
| $\alpha = 0.50$ | 0.223962 | 0.402891 | 0 | $^{0}/_{1433}$ | – |
| $\alpha = 0.60$ | 0.283424 | 0.489587 | 0 | $^{0}/_{1433}$ | – |
| $\alpha = 0.70$ | 0.352509 | 0.580117 | 1 | $^{383}/_{1433}$ | $^{8}/_{391}$ |
| $\alpha = 0.80$ | 0.437208 | 0.676659 | 1 | $^{383}/_{1433}$ | $^{8}/_{391}$ |
| $\alpha = 0.85$ | 0.489514 | 0.728632 | 2 | $^{656}/_{1433}$ | $^{17}/_{673}$ |
| $\alpha = 0.90$ | 0.553874 | 0.784924 | 3 | $^{856}/_{1433}$ | $^{27}/_{883}$ |
| $\alpha = 0.95$ | 0.643048 | 0.849721 | 6 | $^{1107}/_{1433}$ | $^{53}/_{1160}$ |
| $\alpha = 0.99$ | 0.779671 | 0.922288 | 18 | $^{1333}/_{1433}$ | $^{144}/_{1477}$ |

cativamente maior que o limite inferior $p = 0.0$. Portanto, $\mathcal{S}_{45}$ tende a ser significativamente menos beneficiado com a metodologia (como de fato ocorreu, conforme a Tabela 1).

### 5.4. Análise de Sensibilidade da Metodologia

A Tabela 2 mostra os resultados de uma análise de sensibilidade tomando $\mathcal{S}_5$ como base. Observe que $\alpha$ regula quão conservadora nossa metodologia é ao estimar $\hat{p_{pos}}$, e portanto definir o maior comprimento de sequências de *snapshots* nulos que serão corrigidos.

Um valor baixo para $\alpha$ leva a uma estimativa mais conservadora de $\hat{p_{pos}}$. Logo, menos sequências de *snapshots* nulos serão candidatos a correção. Note que para $\alpha = 0.60$ (e valores menores), o tamanho de sequência de *snapshots* nulos que são corrigidos é 0. Isso ocorre porque a probabilidade de que apenas um único *snapshot* tenha falhado $(0.28)$ é menor que o erro $1 - \alpha$ (ou seja, $0.40$) admitido. De maneira oposta, valores mais altos para $\alpha$ tornam nossa metodologia mais efetiva na correção, ao custo de alguns falso positivos. Por exemplo, no caso de $\alpha = 0.70$ (e maiores), a probabilidade de falha mensurada $(0.35)$ é maior que o erro permitido $(0.30)$. Em resumo, ao configurar $\alpha$ é importante considerar quão conservador se deseja ser ao estimar a probabilidade de falha de um conjunto de *snapshots*. Em outras palavras, é necessário encontrar um balanço entre precisão (na correção) e ocorrências de falsos negativos.

## 6. Comportamento de Usuários em Sistemas de Compartilhamento de Arquivo

Como parte da avaliação, procurou-se responder a seguinte pergunta de pesquisa: qual o impacto da metodologia nos traços corrigidos? Para respondê-la, foi considerada uma visão completa do conjunto de *snapshots* investigado na seção anterior, com resoluções $\mathcal{S}_5$ e $\mathcal{S}_{50}$ (referência). Como métricas, foram utilizadas a recorrência de usuários (i.e., número de chegadas por usuário) e a duração das sessões *online*.

A Tabela 3 mostra um resumo estatístico para os seguintes conjuntos de *snapshots*: $\mathcal{S}_5$, $\mathcal{S}'_5$ (conjunto corrigido, usando $\alpha = 0.85$), e $\mathcal{S}_{50}$. Observe que $\mathcal{S}'_5$ é mais similar ao conjunto de referência, quando comparado com o conjunto original $\mathcal{S}_5$. Isso pode ser constatado, por exemplo, comparando o número médio de sessões por usuário e desvio padrão medido.

Um aspecto interessante a ser notado é que o número médio de sessões foi reduzido por um fator de 2 (aprox.), comparando-se os conjuntos $\mathcal{S}_5$ e $\mathcal{S}_{50}$, e que a média de duração das sessões foi aumentada por um fator de 3,5 (aprox.). Isso pode ser explicado pelo fato de que ao corrigir um intervalo de *snapshots* nulos (ou seja, alterando-se o mesmo para "positivo") basicamente "fundem-se" duas sessões em uma. Logo, o número de sessões é decrementado por um, enquanto que a duração da nova sessão torna-se a soma da duração das sessões fundidas mais o comprimento do intervalo corrigido (que pode variar entre 1 e $\lfloor {}^{\ln(1-\alpha)}/_{\ln(\hat{p})} \rfloor$).

**Tabela 3. Resumo das estatísticas dos conjuntos de *snapshots* estudados**

| Conjunto | Mín. | $1^{\circ}$ Quartil | Mediana | Média | Desvio Padrão | $3^{\circ}$ Quartil | Máx. |
|---|---|---|---|---|---|---|---|
| Número de Sessões por Usuário | | | | | | | |
| $\mathcal{S}_5$ | 1 | 1 | 3 | 10.97 | 33.88 | 7 | 671 |
| $\mathcal{S}'_5$ | 1 | 1 | 2 | 4.95 | 13.45 | 4 | 303 |
| $\mathcal{S}_{50}$ | 1 | 1 | 2 | 5.36 | 14.83 | 4 | 552 |
| Duração de Sessões por Usuários (minutos) | | | | | | | |
| $\mathcal{S}_5$ | 15 | 15 | 15 | 28.28 | 25.49 | 30 | 825 |
| $\mathcal{S}'_5$ | 15 | 15 | 45 | 86.03 | 156.54 | 90 | 6,660 |
| $\mathcal{S}_{50}$ | 15 | 15 | 45 | 102.9 | 257.62 | 105 | 24,750 |

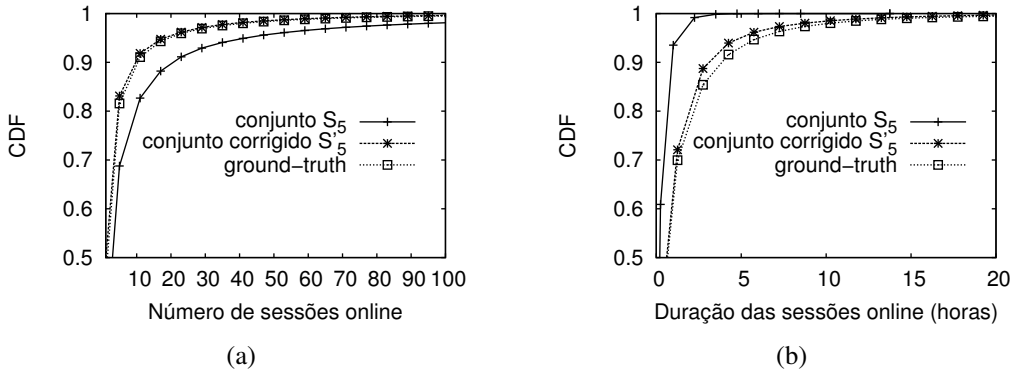



**Figura 4. Análise da recorrência de usuários (esquerda) e duração de sessões (direita)**

A Figura 4 mostra uma visão parcial da CDF da recorrência dos usuários e da duração das sessões. As curvas referentes a $\mathcal{S}'_5$ e a $\mathcal{S}_{50}$ são, de maneira aproximada, iguais. Isso é uma evidência de que nossa metodologia pode aprimorar significativamente um conjunto de *snapshots* capturado com poucos recursos. Mais importante, isso representa um importante avanço sobre as soluções que formam o estado da arte, as quais requerem um número significativo de monitores para capturar *snapshots* acurados [Hoãÿfeld et al. 2011, Zhang et al. 2011].

## 7. Trabalhos Relacionados

Ao que se sabe, nossa metodologia representa a primeira proposta formal e genérica para correção de traços. Em contraste, trabalhos anteriores tem considerado as mais variadas abordagens *ad-hoc* para correção de traços [Izal et al. 2004, Guo et al. 2005, Steiner et al. 2009]. Por exemplo, trabalhos no âmbito de BitTorrent tem considerado identificadores de sessões e informações sobre *download* e *upload* para reconstruir as sessões dos usuários [Izal et al. 2004, Guo et al. 2005]. Recentemente, Wang et al. [Wang and Kangasharju 2013] propuseram uma abordagem para calcular probabilidade de falhas em amostras de uma rede BitTorrent. No entanto, ela limita-se a estimar o número de usuários *online*, e é específica para o Mainline DHT.

Outro exemplo, mais similar à nossa metodologia, é a proposta por Steiner et al. [Steiner et al. 2009]. Nesse trabalho, o conceito de "lacunas na sessão" (noção similar ao *snapshot* nulo definido neste artigo) são identificados em um conjunto de dados, e corrigidos quando o tamanho da "lacuna" excede um limiar. Essa proposta foi usada para refatorar e analisar sessões de pares em uma rede KAD DHT. Embora essa solução possa corrigir traços de modo satisfatório, ela depende de propriedades e/ou atributos específicas do sistema alvo. Logo, ela torna-se pouco adequada para generalização e adoção em um leque maior de cenários.

Como alternativa para correção de traços, pesquisadores tem procurado superprovisionar o número de monitores dedicados para coletar *snapshots* [Hoãÿfeld et al. 2011,

Zhang et al. 2011]. Embora factível, essa opção pode tornar-se extremamente cara para adoção em cenários onde a população média estimada ultrapassa a ordem de milhares ou milhões de usuários. Por exemplo, de acordo com Zhang et al. [Zhang et al. 2011], monitorar um ecossistema BitTorrent com 10.000 usuários em média requer por volta de 489 monitores; para 100.000 usuários, em torno de 6.045. Quando o sistema alvo cresce em popularidade, torna-se um desafio garantir a coordenação dos monitores, sem contar que o super-provisionamento não oferece garantias sobre a precisão dos *snapshots* capturados.

## 8. Considerações Finais

A correção *a posteriori* de traços de sessões *online* de usuários tem sido considerada uma estratégia promissora para melhorar a qualidade dos mesmos. Contudo, a inexistência de uma abordagem sistemática, agnóstica de propriedades específicas do sistema alvo, para corrigir traços capturados usando estratégias baseadas *snapshots* pode ter dificultado sua mais ampla adoção. Para superar essa limitação, neste artigo é proposta uma metodologia baseada em uma análise probabilística para a correção de *snapshots*.

Em resumo, nossa metodologia mostrou-se efetiva, alcançando alto grau de precisão (variando entre 65% e 90% nos cenários avaliados) na correção de *snapshots* e baixo percentual de falso positivos. Mais importante, ela gerou traços com propriedades consistentes (por exemplo, recorrência e duração das sessões *online*) quando comparado com o conjunto de referência. A importância desses resultados é ressaltada pelo fato de que capturar traços precisos requer uma fração significativa de recursos [Hoãÿfeld et al. 2011, Zhang et al. 2011]. Nesse contexto, nossa metodologia representa uma importante contribuição para a comunidade, já que permite a captura de traços com grau similar de precisão enquanto utiliza muito menos recursos.

Apesar dos progressos alcançados, há muitas oportunidades de pesquisa na área. Como trabalhos futuros, pretendemos estender a metodologia para contemplar sistemas nos quais as listas de pares são construídas usando alguma heurística ou distribuição diferente da uniforme. Além disso, pretendemos investigar ajustes finos para o processo de estimativa da probabilidade de falha, tirando proveito de propriedades específicas do sistema alvo (por exemplo, tamanho de listas de pares e estimativas do número de usuários *online*), ou distribuição do tempo de permanência dos usuários [Yao et al. 2009, Steiner et al. 2009].

## Agradecimentos

Os autores estão em dívida com Jair Santanna (Univ. Twente, Holanda), Jon Howell e Jay Lorch (Microsoft Research Redmond, EUA) e Alberto Montresor (Univ. Trento, Itália), pelas discussões que ajudaram a melhorar este trabalho. A pesquisa foi financiada pelo Projeto 560226/2010-1 concedido pelo CNPq (*Edital MCT/CNPq no. 09/2010 PDI – Pequeno Porte*).

## Referências

Alexa Internet, Inc. (2012). Alexa toolbar. Disponível em: <http://www.alexa.com/toolbar>. Acesso em: junho 2012.

Barcellos, M. P. and Gaspary, L. P. (2006). Segurança em redes P2P: princípios, tecnologias e desafios. In *Minicursos do 24º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2006)*.

Blond, S. L., Legout, A., Lefessant, F., Dabbous, W., and Kaafar, M. A. (2010). Spying the world from your laptop. In *3rd USENIX conference on Large-scale exploits and emergent threats (LEET'10)*, pages 1–8.

Cuevas, R., Kryczka, M., Cuevas, A., Kaune, S., Guerrero, C., and Rejaie, R. (2010). Is content publishing in BitTorrent altruistic or profit-driven? In *6th International Conference on emerging Networking EXperiments and Technologies (Co-NEXT '10)*.

França, T. C., Pires, P., Pirmez, L., Delicato, F. C., and Miceli, C. (2011). Web das coisas: conectando dispositivos físicos ao mundo digital. In *Minicursos do 29º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2011)*, pages 211–264.

Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., and Zhang, X. (2005). Measurements, analysis, and modeling of bittorrent-like systems. In *5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 4–4, Berkeley, CA, USA. USENIX Association.

Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., and Zhang, X. (2007). A performance study of bittorrent-like peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 25(1):155–169.

Hoãÿfeld, T., Lehrieder, F., Hock, D., Oechsner, S., Despotovic, Z., Kellerer, W., and Michel, M. (2011). Characterization of BitTorrent swarms and their distribution in the Internet. *Computer Networks*, 55(5):1197–1215.

Izal, M., Urvoy-Keller, G., Biersack, E., Felber, P., Al Hamra, A., and Garcés-Erice, L. (2004). Dissecting bittorrent: Five months in a torrent's lifetime. In *Passive and Active Network Measurement*, volume 3015, pages 1–11. Springer Berlin Heidelberg.

Kobza, J., Jacobson, S., and Vaughan, D. (2007). A survey of the coupon collector's problem with random sample sizes. *Methodology and Computing in Applied Probability*, 9(4):573–584.

Mansilha, R. B., Mezzomo, A., Facchini, G., Gaspary, L. P., and Barcellos, M. P. (2010). Observando o universo BitTorrent através de telescópios. In *28º Simpósio Brasileiro de Redes de Computadores e de Sistemas Distribuídos (SBRC 2010)*, pages 1–14.

Rosensweig, E., Menasche, D., and Kurose, J. (2013). On the steady-state of cache networks. In *33rd IEEE International Conference on Computer Communications (INFOCOM 2013)*.

Ruiz, L. B., Correia, L. H., and Vieira, L. F. (2006). Arquitetura de redes de sensores sem fio. In *Minicursos do 22º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2004)*, page 167–218.

Sridharan, A. and Bolot, J. (2013). Location patterns of mobile users: A large scale study. In *33rd IEEE International Conference on Computer Communications (INFOCOM 2013)*.

Steiner, M., En-Najjary, T., and Biersack, E. W. (2009). Long term study of peer behavior in the kad dht. *IEEE/ACM Transactions on Networking*, 17(5):1371–1384.

Teles, A., Pinheiro, D. N., Gonçalves, J., Batista, R., Almeida, V., Endler, M., and Silva, F. J. (2013). Redes sociais móveis: conceitos, aplicações e aspectos de segurança e privacidade . In *Minicursos do 31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2013)*.

Wang, L. and Kangasharju, J. (2013). Measuring Large-Scale Distributed Systems: Case of BitTorrent Mainline DHT. In *IEEE Conference on Peer-to-Peer Computing (P2P 2013)*.

Yao, Z., Wang, X., Leonard, D., and Loguinov, D. (2009). Node isolation model and age-based neighbor selection in unstructured p2p networks. *IEEE/ACM Transactions on Networking*.

Yoshida, M. and Nakao, A. (2011a). A resource-efficient method for crawling swarm information in multiple BitTorrent networks. In *10th International Symposium on Autonomous Decentralized Systems (ISADS 2011)*, pages 497–502.

Yoshida, M. and Nakao, A. (2011b). Measuring BitTorrent swarms beyond reach. In *IEEE International Conference on Peer-to-Peer Computing (P2P 2011)*, pages 220–229.

Zhang, C., Dhungel, P., Wu, D., and Ross, K. W. (2011). Unraveling the bittorrent ecosystem. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1164–1177.

# Estratégia de Replicação Adaptativa para Tarefas de Computação por Humanos

**Lesandro Ponciano, Francisco Brasileiro, Guilherme Gadelha, Adabriand Furtado**

[1]Departamento de Sistemas e Computação
Universidade Federal de Campina Grande (UFCG)
Av. Aprígio Veloso, 882 – Bloco CO – Campina Grande – PB – Brasil

{lesandrop,fubica,guilherme,adabriand}@lsd.ufcg.edu.br

***Abstract.*** *Human computation systems are distributed systems in which the processors are humans beings, called workers. In such systems, task replication has been used as a way to obtain results redundancy and quality. The level of replication is usually defined before the tasks start executing. This approach, however, generates the problem of defining the suitable task replication level. If the level of replication is overestimated, an excess of workers are used and, therefore, there is an increase in the cost of executing all tasks. On the other hand, if the level of replication is underestimated, a desired level of quality cannot be achieved. Defining the appropriate level of replication involves considering a variety of aspects related to human computational behavior. This work proposes an adaptive replication strategy that defines the level of replication during execution time based on estimations of the degree of difficulty of tasks and the degree of credibility of human workers. Results from simulations using data from two real human computation applications show that, compared to non-adaptive replication, the proposed strategy reduces the number of replicas substantially, without compromising the accuracy of the obtained answers.*

***Resumo.*** *Sistemas de computação por humanos são sistemas distribuídos em que as tarefas são executadas por seres humanos, chamados trabalhadores. Nesses sistemas, replicação de tarefas tem sido utilizada como forma de se obter redundância de respostas e qualidade na execução de tarefas. A replicação geralmente é realizada de forma não adaptativa, i.e., o nível de replicação é definido antes da aplicação começar a ser executada. Essa abordagem, no entanto, gera o problema de se definir o nível de replicação adequado. Definir esse nível de replicação envolve considerar uma diversidade de fatores que afetam a acurácia do ser humano. Este trabalho propõe uma estratégia de replicação adaptativa que define o nível de replicação com base em estimativas do grau de dificuldade das tarefas e da credibilidade dos trabalhadores. Resultados obtidos em simulações que usam dados de duas aplicações reais mostram que, comparado à replicação não adaptativa, a estratégia proposta permite reduzir substancialmente o número de réplicas, sem prejuízo na acurácia das respostas.*

## 1. Introdução

Sistemas de computação por humanos[1] são sistemas distribuídos que utilizam a cognição e a capacidade de raciocínio de seres humanos para resolver problemas que os com-

[1] Do inglês *Human Computation*. Também tem sido traduzido como "computação humana".

putadores de silício ainda não podem resolver de forma satisfatória, mas que seres humanos são capazes de resolver de forma rápida e precisa [Von Ahn et al. 2008, Quinn and Bederson 2011]. Esses problemas incluem compreensão de linguagem natural, recuperação de informação em imagens e tarefas ligadas à criatividade. Aplicações de computação por humanos podem ser modeladas como uma aplicação distribuída composta por diversas tarefas que podem ser executadas por seres humanos diferentes. O uso de computação por humanos se popularizou com as tarefas reCAPTCHA [Von Ahn et al. 2008], mas atualmente o universo de aplicações baseadas em computação por humanos é amplo e diverso [Quinn and Bederson 2011, Oliveira et al. 2013].

Com o propósito de dar suporte à execução desse tipo de aplicação em larga escala, sistemas dedicados à computação por humanos têm sido desenvolvidos. Um sistema de computação por humanos é um sistema computacional distribuído onde os processadores são seres humanos, chamados de *trabalhadores*. Tal sistema orquestra o poder cognitivo de um grupo de trabalhadores conectados à Internet e gerência o poder computacional provido por eles de forma a executar as tarefas das aplicações. Dois tipos de sistemas de computação por humanos bastante difundidos atualmente são os sistemas de *trabalho online* [Ipeirotis et al. 2010] e os sistemas de *pensamento voluntário* [Ponciano et al. 2014]. Sistemas de trabalho online agregam trabalhadores que possuem uma motivação financeira. Um dos principais exemplos é a plataforma Amazon Mechanical Turk (Mturk)[2]. Sistemas de pensamento voluntário, por sua vez, agregam trabalhadores que executam tarefas como um trabalho voluntário. Um dos principais exemplos é o Zooniverse[3]. Sistemas como o Mturk e Zooniverse agregam trabalhadores não especialistas. Dessa forma, tem-se uma preocupação constante com a qualidade das respostas obtidas desses trabalhadores.

A forma como se avalia a qualidade das respostas geradas pelos trabalhadores em sistemas de computação por humanos depende do tipo de tarefa. Em tarefas que envolvem subjetividade e/ou criatividade, a qualidade das respostas não é avaliada em termos de corretude, i.e., não se define que uma resposta está correta ou incorreta. Entretanto, em tarefas ditas factuais as respostas podem ser avaliadas em termos de corretude. Por exemplo, define-se como factual uma tarefa que exibe uma fotografia de uma paisagem e pergunta ao trabalhador se existe ou não uma árvore na paisagem retratada na fotografia. Em tarefas desse tipo, busca-se a resposta correta. No entanto, diversos fatores não intencionais podem levar os trabalhadores a proverem respostas incorretas. Para se identificar e eliminar respostas incorretas, geralmente se utiliza redundância de respostas, que são providas por diversos trabalhadores diferentes. A ideia é que, sobre as respostas redundantes, pode-se aplicar uma estratégia de tratamento de incertezas e se obter a resposta correta. A estratégia mais simples e mais utilizada na prática é o voto majoritário [Sheshadri and Lease 2013, Quinn and Bederson 2011], em que se considera correta a resposta provida pela maioria dos trabalhadores.

Para se obter respostas redundantes em sistemas de computação por humanos, geralmente se utiliza *replicação de tarefas* realizada de forma ativa com nível de replicação definido de forma não adaptativa. Replicação ativa de tarefas, neste contexto, significa

[2] `mturk.com`, visitado pela última vez em 25/11/2013.
[3] `zooniverse.org`, visitado pela última vez em 25/11/2013.

que para cada tarefa são geradas diversas réplicas e que cada réplica é executada integralmente de forma independente por um trabalhador diferente. O nível de replicação definido de forma não adaptativa indica que a quantidade de trabalhadores que executarão diferentes réplicas das tarefas é definida antes delas começarem a serem executadas [Ponciano et al. 2013]. Essa abordagem de replicação, entretanto, gera o problema de se definir o nível de replicação adequado para as tarefas. Se o nível de replicação for subestimado, pode-se não obter a resposta correta. Por outro lado, se o nível de replicação for superestimado, aumenta-se desnecessariamente o custo de se executar todas as tarefas. Em mercados de trabalho *online*, esse custo significa maior gasto monetário em termos do valor pago aos trabalhadores para executar as diversas réplicas da tarefa. Em sistemas de pensamento voluntário, esse custo se manifesta pelo custo de recrutar e engajar voluntários para que executem as réplicas, ou de outra forma, em uma menor vazão do sistema.

Nesse contexto, o problema tratado neste trabalho é a *definição do nível de replicação adequado para tarefas de computação por humanos*. Definir o nível de replicação envolve considerar fatores adversos que podem levar os trabalhadores a gerarem respostas incorretas e, que em muitos casos, são difíceis de serem previstos antes da tarefa ser submetida para execução. Por exemplo, a acurácia dos trabalhadores tende a variar com o grau de dificuldade da tarefa. A percepção de dificuldade, por sua vez, pode variar de um trabalhador para outro. Além disso, existem tarefas para as quais aumentar o grau de replicação não aumenta necessariamente a probabilidade de se obter a resposta correta no conjunto de respostas redundantes [Sheng et al. 2008]. Esse é o caso, por exemplo, de tarefas que possuem algum complicador que leva trabalhadores não especialistas a gerarem respostas divergentes, tornando difícil identificar qual é a resposta representativa de escolha da maioria com alta confiança.

Este trabalho apresenta uma estratégia de *replicação de tarefas adaptativa* que visa otimizar o número de réplicas necessárias para se obter uma resposta que represente a escolha da maioria dos trabalhadores em sistemas de computação por humanos. A principal característica da estratégia proposta é usar o histórico de tarefas executadas pelos trabalhadores para estimar, em tarefas futuras, a probabilidade das respostas providas por alguns trabalhadores representarem uma resposta de escolha da maioria dos trabalhadores no sistema. Ao realizar essa estimativa, a estratégia leva em conta a ocorrência de respostas incorretas geradas de forma não intencional e sua relação com o grau de dificuldade das tarefas. A estratégia proposta é avaliada em simulações que usam dados de duas aplicações típicas de computação por humanos: análise de sentimentos e julgamento de fatos. Os resultados obtidos sugerem que a estratégia proposta é capaz de identificar a resposta da escolha da maioria com bem menos réplicas que uma replicação não adaptativa, sem prejuízo da acurácia obtida. Outro benefício da estratégia é destacar tarefas cujas respostas obtidas não são conclusivas. Isso permite, por exemplo, que o dono da aplicação conheça quais são as tarefas nas quais trabalhadores tendem a divergir. São tarefas que, por exemplo, podem requerer a execução por trabalhadores mais especializados.

O restante deste trabalho é dividido da seguinte forma. A Seção 2 apresenta o referencial teórico e os trabalhos relacionados. A Seção 3 apresenta a estratégia de replicação adaptativa. Em seguida, a Seção 4 apresenta os materiais e métodos utilizados para avaliar a estratégia proposta. Essa seção é seguida pela apresentação e discussão dos resultados.

Finalmente, a Seção 6 apresenta as conclusões e sugestões para trabalhos futuros.

## 2. Referencial Teórico e Trabalhos Relacionados

Esta seção apresenta duas classes de trabalhos relevantes a este estudo: os que tratam da acurácia dos trabalhadores e os que tratam de replicação de tarefas.

### 2.1. Acurácia dos Trabalhadores

A acurácia de respostas obtidas em sistemas de computação por humanos está diretamente relacionada à acurácia dos trabalhadores, ou seja, à probabilidade do trabalhador prover uma resposta correta. Naturalmente, trabalhadores podem prover respostas incorretas de forma intencional. Por exemplo, planejarem um conluio de modo a todos proverem uma mesma resposta errada para a tarefa. Entretanto, esse tipo de comportamento não tem sido reportado como algo comum nesses sistemas [Quinn and Bederson 2011, Kulkarni et al. 2012]. Geralmente, quando ele ocorre, se dá como uma reação dos trabalhadores a um comportamento inadequado do dono da aplicação, como submeter tarefas mal projetadas ou alta recusa dos trabalhos realizados pelos trabalhadores [Kulkarni et al. 2012]. Este trabalho foca em *fatores não intencionais*.

A Teoria do Erro Humano [Reason 1990] mostra que existem três causas de seres humanos proverem respostas incorretas de modo não intencional: ignorância (*mistake*), esquecimento (*lapse*) e deslize (*slip*). Ignorância ocorre em situações em que o ser humano não possui todo conhecimento necessário para executar a tarefa, assim ele não define corretamente os passos que precisam ser seguidos para se chegar à resposta correta. Em deslizes e esquecimentos o ser humano possui os conhecimentos necessários para executar a tarefa e ele define corretamente os passos que precisam ser seguidos para se chegar à resposta correta. Entretanto, no caso do deslize ele executada incorretamente algum dos passos e no esquecimento ele simplesmente se esquece de executar algum passo.

Para prevenir respostas incorretas causadas pelo fator ignorância, alguns estudos focam em tentar estimar a priori se o trabalhador possui os conhecimentos necessários para executar a tarefa. Isso é geralmente feito usando *qualificações* [Chen et al. 2011]. Qualificações são tarefas cujas respostas são conhecidas e são usadas para se avaliar a acurácia dos trabalhadores. Trabalhadores só são autorizados a executarem outras tarefas se obtiverem alta acurácia nas tarefas de qualificação. Essa estratégia requer que os itens das tarefas sejam muito parecidos com os itens da qualificação. Além disso, ela não previne esquecimentos e deslizes. Geralmente, a ocorrência de esquecimentos e deslizes é relacionada à atenção do trabalhador durante a execução da tarefa e à dificuldade que ele percebe na tarefa que está sendo executada [Reason 1990].

Existem diversos fatores que podem tornar uma tarefa de computação por humanos difícil de ser executada. Eles podem ser fatores inerentes aos dados a que a tarefa se refere (e.g., quando é requerida muita atenção para encontrar um determinado item em uma imagem) ou ao projeto da tarefa, como a carga cognitiva (e.g., quando é preciso seguir uma longa sequência de passos ou comparar muitos itens para se chegar à resposta correta). Neste trabalho, trata-se de cenários em que as tarefas são geradas a partir de uma grande quantidade de itens desconhecidos e que apresentam grande variação entre eles de modo que nem é possível preprocessá-los para estimar os graus de dificuldade e nem se conhece a priori as respostas corretas. Por exemplo, muitas das tarefas de computação

por humanos consistem basicamente de um item a ser julgado, de uma pergunta referente a esse item e de um conjunto de opções de resposta. As tarefas diferem umas das outras em termos apenas do item a ser julgado. Considere uma tarefa projetada para detectar ironia em mensagens de texto. Ela pode consistir da pergunta "A frase abaixo apresenta uma ironia?", e as opções de resposta podem ser "Sim" ou "Não". Grande quantidade dessas tarefas pode ser gerada coletando-se automaticamente frases em comentários feitos em notícias em páginas Web. Neste caso, não se tem qualquer controle do conteúdo de cada frase que será avaliada em cada tarefa. Algumas frases podem ser fáceis de serem avaliadas e outras serem muito difíceis.

### 2.2. Replicação de Tarefas

Replicação é um mecanismo bastante utilizado para se obter redundância em sistemas computacionais. Em sistemas distribuídos compostos por máquinas, muitas das primeiras abordagens que empregaram esse mecanismo são baseadas em máquina de estado (*machine state*) [Schneider 1990, Jalote 1994]. Nessas abordagens, replicação é dita ativa quando cada réplica de uma tarefa é executada integralmente por máquinas diferentes, cada uma partindo de um mesmo estado inicial até o mesmo estado final da tarefa. De outro modo, a replicação é dita passiva quando é mantido um *backup* do estado da tarefa durante sua execução por uma máquina. Esse *backup* mantém as computações já realizadas e ele pode ser utilizado para que a tarefa não precise ser re-executada integralmente caso uma falha ocorra durante sua execução.

Nos últimos anos, replicação tem sido utilizada com diversos propósitos e em diversos tipos de sistemas. Por exemplo, em grades computacionais oportunistas como OurGrid, replicação é utilizada com o propósito de reduzir o tempo de resposta das aplicações [Cirne et al. 2007]. Já em sistemas de computação voluntária como BOINC[4], replicação é utilizada como forma de identificar sabotadores e isolar as respostas providas por eles [Sarmenta 2002]. Em sistemas compostos por seres humanos, entretanto, replicação de tarefas é realizada para se obter respostas redundantes de modo que se possa identificar uma resposta que represente uma escolha social.

Em Escolha Social (*"Social Choice Theory"*) [Taylor and Pacelli 2008], parte-se de respostas providas por diversos indivíduos e, a partir da agregação dessas respostas, obtém-se uma resposta de preferência coletiva. A base dos algoritmos atuais de tratamento de incertezas em respostas obtidas em sistemas de computação por humanos é identificar uma resposta de preferência social, quando eliminadas respostas erradas [Sheshadri and Lease 2013]. Em tarefas em que há o conceito de corretude, essa resposta pode não ser necessariamente a resposta que seria julgada correta por um especialista. Geralmente quando isso ocorre o problema é atribuído ao nível de replicação utilizado, a algum fator dificultador no item que está sendo avaliado na tarefa ou à forma como a tarefa foi projetada [Kochhar et al. 2010].

O propósito deste trabalho é identificar uma resposta de escolha da maioria dos trabalhadores de modo que um número menor de trabalhadores tenham que ser consultados, i.e., menos réplicas sejam geradas. A estratégia de replicação proposta combina conceitos de escolha social com replicação de tarefas em sistemas com máquinas. A principal inspiração em escolha social é a *concordância* de um trabalhador com outros

[4] Do inglês *Berkeley Open Infrastructure for Network Computing* (http://boinc.berkeley.edu/).

trabalhadores. Neste trabalho, utiliza-se informação da concordância entre trabalhadores em tarefas executadas no passado para estimar concordâncias em tarefas futuras, levando-se em conta também estimativas do grau de dificuldade dessas tarefas. De replicação em sistemas compostos por máquinas, utiliza-se um arcabouço de medidas de credibilidade proposto para sistemas de computação voluntária [Sarmenta 2002].

A ideia de otimizar a redundância de respostas em sistemas de computação por humanos por meio de replicação de tarefas que leva em conta medidas de credibilidade foi publicada recentemente como um resumo de trabalho em andamento [Ponciano et al. 2013]. O presente artigo apresenta o estudo desenvolvido. Além de apresentar todos os fundamentos da estratégia, incorpora-se um novo método para estimar a acurácia dos trabalhadores, considerando o histórico de concordância com outros trabalhadores e estimativas de dificuldade das tarefas. Além disso, estende-se a avaliação da estratégia para medir a acurácia, economia de réplicas e proporção de tarefas cujas respostas obtidas dos trabalhadores não são conclusivas.

## 3. Estratégia de Replicação

Nesta seção, antes de apresentar nossa estratégia de replicação, são definidas a medida de dificuldade e as medidas de credibilidade que ela utiliza.

### 3.1. Medindo o Grau de Dificuldade das Tarefas

O grau de dificuldade de uma tarefa é medido pelo grau de divergência dos trabalhadores entre as opções de resposta disponíveis. A ideia principal é que quanto mais os trabalhadores se dividirem em diferentes opções de resposta, maior a chance da tarefa apresentar *(i)* alguma característica atípica, levando os trabalhadores a gerarem uma resposta incorreta por ignorância ou *(ii)* algum fator dificultador, gerando esquecimentos e deslizes. Dado $N$ o multiconjunto de todas as respostas recebidas dos trabalhadores para uma mesma tarefa e $f$ a frequência da resposta mais frequente em $N$, a Equação 1 apresenta o cálculo do grau de dificuldade $d$.

$$d = \frac{|N| - f}{|N|} \tag{1}$$

Esse grau de dificuldade é um valor real $d \in [0, 1]$. Ele assume o valor $0$ quando todos os trabalhadores apresentam a mesma resposta para a tarefa, i.e., $f = |N|$. Por outro lado, ele tende ao valor $1$ na proporção em que o número de opções de respostas aumenta e que os trabalhadores se dividem entre a opção mais frequente e as demais opções. Por simplicidade, nas análises realizadas neste trabalho, arredonda-se o grau de dificuldade da tarefa para $1$ casa decimal, de modo a trabalhar com apenas $11$ graus de dificuldade, de $0{,}0$ a $1{,}0$.

### 3.2. Medindo Concordância e Credibilidade

O grau de concordância de um trabalhador é a probabilidade dele, ao executar uma nova tarefa, prover uma resposta igual à resposta que a maioria dos trabalhadores proviria considerando seu histórico de tarefas. Um trabalhador pode concordar mais em tarefas com determinado grau de dificuldade e discordar mais em outros. Dessa forma, o grau de concordância do trabalhador varia com o grau de dificuldade da tarefa. Assim, define-se

como $c_{t,d}$ o grau de concordância do trabalhador $t$ em tarefas que possuem o grau de dificuldade $d$, $c_{t,d} \in [0, 1]$. Antes de executar qualquer tarefa, o grau de concordância do trabalhador $t$ é $c_{t,d} = 0{,}5$ para todos os valores de $d$. Ou seja, ele possui $50\%$ de chance de prover uma resposta igual à da maioria.

Sempre que a execução de uma tarefa é concluída atingindo um valor aceitável de credibilidade, calcula-se o seu grau de dificuldade pela Equação 1 e verifica-se os trabalhadores que proveram respostas iguais à resposta da maioria dos trabalhadores. Com isso, pode-se recalcular a proporção de convergência de cada trabalhador considerando todas as tarefas que ele executou no passado. Esse cálculo é dado por $p_{t,d} = m_{t,d}/n_{t,d}$, sendo $m_{t,d}$ o total de tarefas com grau de dificuldade $d$ que a resposta provida pelo trabalhador $t$ foi igual à resposta provida pela maioria e $n_{t,d}$ o total de tarefas com dificuldade $d$ que o trabalhador $t$ executou. O novo grau de concordância do trabalhador é computado pela Equação 2. Essa equação é uma média harmônica ponderada entre $p_{t,d}$ e $0{,}5$. Essa média visa impedir valores de concordância muito altos ou muito baixos quando poucas tarefas tiverem sido executadas. Por ser ponderada, essa média faz com que o valor inicial $0{,}5$ tenha menor peso no grau de concordância do trabalhador, quanto maior for o número de tarefas que ele tiver executado ($n_{t,d}$).

$$c_{t,d} = \frac{n_{t,d} + 1}{\frac{n_{t,d}}{p_{t,d}} + \frac{1}{0{,}5}} \tag{2}$$

A resposta gerada por um trabalhador $t$ tem probabilidade $c_{t,d}$ de ser a resposta de escolha da maioria dos trabalhadores. Respostas iguais geradas por diferentes trabalhadores para diferentes réplicas de uma mesma tarefa são agregadas em grupos. O número de grupos é definido como $g$ e ele varia com a diversidade de respostas geradas, sendo o grupo $G_a$ o multiconjunto das probabilidades das respostas $a$ recebidas de diferentes trabalhadores. Pode-se computar a credibilidade de cada grupo de respostas. Essa credibilidade é a probabilidade condicional da resposta do grupo ser a resposta de escolha da maioria dos trabalhadores no sistema e das outras possíveis respostas para a mesma tarefa não serem a escolha desses trabalhadores. O cálculo dessa probabilidade condicional é formalizado na Equação 3. Essa equação apresenta o cálculo da probabilidade condicional da resposta $a$ do grupo de respostas $G_a$ representar a escolha da maioria dos trabalhadores e das demais respostas dos outros grupos não representarem. Esse cálculo é inspirado na credibilidade de grupos de respostas em sistemas de computação voluntária [Sarmenta 2002].

$$C(G_a) = \frac{P(G_a \text{ good}) \prod_{i \neq a} P(G_i \text{ bad})}{\prod_{j=1}^{g} P(G_j \text{ bad}) + \sum_{j=1}^{g} P(G_j \text{ good}) \prod_{k \neq j} P(G_k \text{ bad})} \tag{3}$$

O grupo $G_a$ com maior valor de $C(G_a)$ é o grupo que representa a resposta candidata da tarefa. Por sua vez, a credibilidade requerida $r$, definida pelo dono da aplicação, indica um limiar mínimo para $C(G_a)$ de modo que essa resposta candidata possa ser aceita como a escolha da maioria. Dessa forma, uma resposta $a$ é dita ser de escolha da maioria dos trabalhadores se, e somente se, a condição na Equação 4 for satisfeita. Essa condição indica que $C(G_a)$ precisa ser maior ou igual a $r$ e que $C(G_a)$ também precisa ser maior que a credibilidade dos demais grupos. Assim, se $a$ é dita ser a resposta de escolha da maioria, nenhuma outra resposta para a mesma tarefa pode atingir essa condição.

$$C(G_a) \geq r \wedge C(G_a) > C(G_i), 1 \leq i \leq g, i \neq a \quad (4)$$

### 3.3. Estratégia de Replicação

A principal ideia dessa estratégia é que novas réplicas de uma tarefa precisam ser geradas apenas se as respostas já obtidas não atingirem um nível requerido de credibilidade ($r$). Isso é realizado utilizando os conceitos de credibilidade apresentados na seção anterior.

Para toda tarefa, sempre gera-se uma réplica inicial. Após a execução de cada réplica de uma tarefa, as respostas iguais obtidas de diferentes trabalhadores são agrupadas e o grau de credibilidade de cada grupo de respostas é recalculado pela Equação 3. Novas réplicas da tarefa são geradas até que pelo menos uma das condições seguintes seja satisfeita:

- Critério de credibilidade é atingido. Isso ocorre quando a condição descrita na Equação 4 é satisfeita, o que indica que uma resposta final para a tarefa foi obtida.
- Limite máximo de réplicas é atingido. Isso ocorre quando um limite máximo de réplicas definido pelo dono da aplicação é atingido.

Quando a replicação termina com o critério de credibilidade satisfeito, calcula-se o grau de dificuldade da tarefa ($d$) usando a Equação 1 e identifica-se os trabalhadores que proveram respostas que coincidiram com a maioria e os que proveram respostas que não coincidiram com a maioria. Esses dados são utilizados para atualizar os graus de concordância dos trabalhadores que participaram da execução. Como já discutido anteriormente, esse cálculo é realizado pela Equação 2.

Quando a replicação termina porque o número máximo de réplicas foi atingido, mas sem satisfazer o critério de credibilidade, duas situações são possíveis, dependendo de como a aplicação que usa replicação adaptativa é configurada. Em uma *configuração conservadora*, as tarefas que não atingem o limiar de credibilidade requerida são marcadas como "sem conclusão" e não têm uma resposta associada. Por outro lado, em uma *configuração não conservadora*, a resposta para a tarefa é aquela que obteve o maior valor de credibilidade, mesmo que abaixo do limiar requerido.

Considerando a literatura de replicação de tarefas discutida na Seção 2.2, essa estratégia de replicação pode ser definida como ativa e adaptativa. A estratégia de replicação é ativa porque cada réplica de uma tarefa é executada integralmente por cada trabalhador, não há *backup* do estado da tarefa durante sua execução. A estratégia é dita adaptativa no sentido de que o número de réplicas geradas não é pré-definido. Ele é calculado em tempo de execução e é dependente das respostas recebidas dos trabalhadores.

## 4. Materiais e Métodos

Esta seção apresenta as bases de dados, métricas e os cenários avaliados.

### 4.1. Bases de Dados

A estratégia proposta neste trabalho foi avaliada por meio de simulações que utilizam dados de duas aplicações de computação por humanos: Julgamento de Fatos[5] e Análise

[5] Disponível em `https://code.google.com/p/relation-extraction-corpus/`

de Sentimentos[6]. A base de dados Julgamento de Fatos foi disponibilizada pelo Google. Ela consiste em um conjunto de julgamentos realizados por seres humanos sobre relações referentes a pessoas públicas na Wikipedia. As relações são do tipo "a pessoa X se graduou na universidade Y". Os trabalhadores eram solicitados a julgar se a relação é "verdadeira", "falsa" ou "não responder". A base de dados Análise de Sentimentos, por sua vez, foi disponibilizada pelo CrowdFlower (crowdflower.com). Ela consiste em um conjunto de julgamento de seres humanos sobre a condição climática relatada em *tweets*[7]. Cada tarefa apresenta aos trabalhadores um tweet e eles respondem se a informação sobre o clima constante no tweet é "negativa", "neutra (o autor do tweet está apenas compartilhando informação)", "positiva", "O tweet não é relacionado à condição do clima", ou "não sei responder". Em ambas as bases, $95\%$ das tarefas foram executas por $5$ trabalhadores e as restantes foram executadas por mais trabalhadores.

A Tabela 1 apresenta um sumário estatístico das bases. Duas importantes distinções entre essas bases no contexto deste trabalho são: *(i)* número de opções de resposta por tarefa e *(ii)* proporção de tarefas por trabalhador. A aplicação Análise de Sentimentos apresenta $5$ opções de resposta e aplicação Julgamento de Fatos $3$. Quanto mais opções de resposta, maior a chance de divergência entre os trabalhadores em tarefas difíceis. Quanto à proporção de tarefa por trabalhadores, ela é de $748$ na aplicação Julgamento de Fatos e $51$ na aplicação Análise de Sentimentos. Para o algoritmo proposto, quando mais tarefas o trabalhador executa, mais informação histórica se mantém e mais acurada pode se tornar a estimativa da concordância dele com outros trabalhadores. Essas bases também contêm um conjunto de tarefas *Ground truth*, i.e., tarefas para as quais as respostas corretas são conhecidas. Essas tarefas foram geradas de modo a serem representativas de toda a base de dados. Elas consistem em $576$ tarefas na base de dados Julgamento de Fatos e em $300$ tarefas na base de dados Análise de Sentimentos. Neste estudo, essas tarefas são utilizadas para avaliar a acurácia das respostas obtidas pelas estratégias de replicação.

**Tab. 1. Sumário Estatístico das Bases de Dados**

| Característica | Julgamento de Fatos | Análise de Sentimentos |
|---|---|---|
| #Trabalhadores únicos | 57 | 1.960 |
| #Tarefas diferentes | 42.624 | 98.980 |
| #Opções de resposta por tarefa | 3 | 5 |
| #Réplicas | 220.000 | 500.000 |
| #Tarefas *ground truth* | 576 | 300 |

A estratégia de replicação proposta foi simulada usando como entrada os dados das tarefas e as respostas providas pelos trabalhadores para cada réplica. A ordem em que as tarefas são executadas e as respostas que os trabalhadores retornam são conforme registados nas bases de dados. No entanto, a ordem em que os trabalhadores geram as respostas para as réplicas das tarefas não é conhecida. Como o algoritmo proposto pode terminar a replicação sem que as respostas de todas as réplicas sejam utilizadas, a ordem em que as respostas armazenadas nas bases de dados são utilizadas tem impacto nos resultados. Esse impacto foi medido por $5$ simulações usando as respostas ordenadas de forma aleatória. Na análise dos resultados, sempre se apresenta a média dos resultados obtidos nessas simulações com barras de erros para um nível de confiança de $95\%$.

[6] Disponível em `https://minbox.com/gi/Uh0vixtBNw`

[7] Tweets são mensagens de texto de até $140$ caracteres compartilhadas na rede social `twitter.com`.

Naturalmente, o número de réplicas que o algoritmo proposto pode gerar é limitado ao existente na base de dados. Um dos principais objetivos da avaliação é verificar em que situações a estratégia de replicação proposta é capaz de gerar menos réplicas.

### 4.2. Métricas

Três métricas são utilizadas na avaliação realizada.

- **Economia de réplicas:** Dado que na base de dados são geradas $x$ réplicas e a estratégia proposta gerou $y$ réplicas, a economia de réplicas é dada por $\frac{x-y}{x}$. Essa economia é calculada em duas situações: por tarefa da aplicação e em toda a aplicação. Quando é calculada por tarefa, $x$ depende da tarefa, sendo que $95\%$ das tarefas têm $5$ réplicas e o restante tem mais que $5$. Quando a economia é calculada em toda a aplicação, $x$ é o total de réplicas na aplicação apresentada na Tabela 1.
- **Acurácia:** É a taxa de acerto nas tarefas *ground truth*. Ela é a razão entre o total de tarefas cujas respostas obtidas coincidem com as existentes na base *ground truth* e o total de tarefas *ground truth*.
- **Tarefas sem conclusão:** É a proporção de tarefas que o algoritmo proposto não atingiu o limiar de credibilidade definido pelo dono da aplicação. É calculado como a razão entre o número de tarefas que não atingiram o limiar de credibilidade e o número total de tarefas.

### 4.3. Cenários

Os cenários de avaliação foram projetados de modo a permitir uma análise custo/benefício da estratégia proposta. São dois os cenários: um que assume uma configuração conservadora da aplicação, e outra que não. No cenário com configuração não conservadora, a estratégia sempre gera uma resposta para as tarefas, que é a resposta do grupo de respostas de maior credibilidade. Utiliza-se o valor de $0{,}95$ como condição de parada com conclusão. Neste cenário se analisa: acurácia por aplicação; economia de réplicas por tarefa e por aplicação e distribuição da credibilidade das respostas obtidas. No segundo cenário, a configuração é conservadora e uma resposta só é gerada se o grupo de respostas com maior credibilidade for maior ou igual à credibilidade requerida, que é variada entre $0{,}90$ a $0{,}99$. Neste cenário se analisa: acurácia das respostas obtidas com conclusão, proporção de tarefas sem conclusão e economia de réplicas.

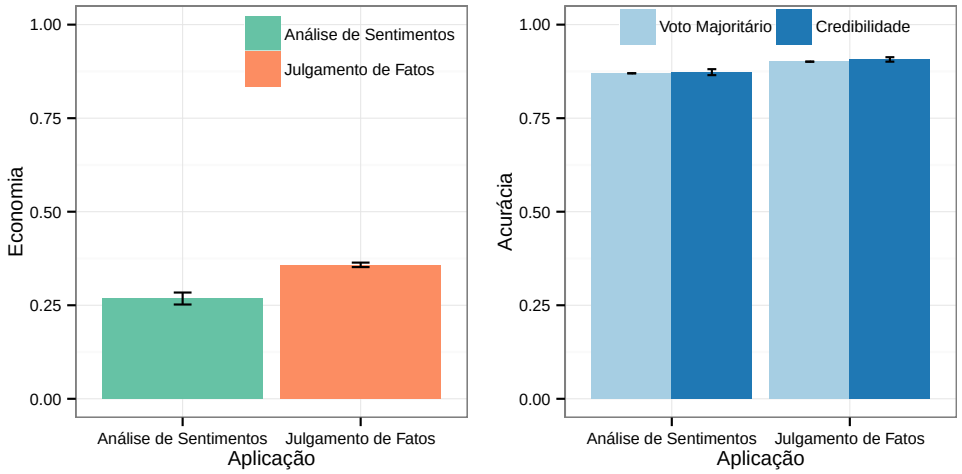
## 5. Apresentação e Análise dos Resultados

Esta seção apresenta os resultados obtidos nos cenários de avaliação.

### 5.1. Configuração não Conservadora da Aplicação

A Figura 1 apresenta o total de economia obtida e da acurácia das respostas em todas as tarefas das aplicações avaliadas. A Figura 1(a) mostra a economia de réplicas que a estratégia proposta gera em relação à replicação não adaptativa. Em média, a economia obtida é de $27\%$ na aplicação Análise de Sentimentos e de $36\%$ na aplicação Julgamento de Fatos. A Figura 1(b), por sua vez, mostra a acurácia que se obtém quando a resposta final para cada tarefa é escolhida pelo *voto majoritário* sobre todas as respostas obtidas na replicação não adaptativa e a acurácia que se obtém quando a resposta final para cada tarefa é escolhida pelo seu grau de *credibilidade* calculado pelo algoritmo adaptativo proposto neste trabalho. Essa figura mostra que as duas estratégias atingem acurácia similar,

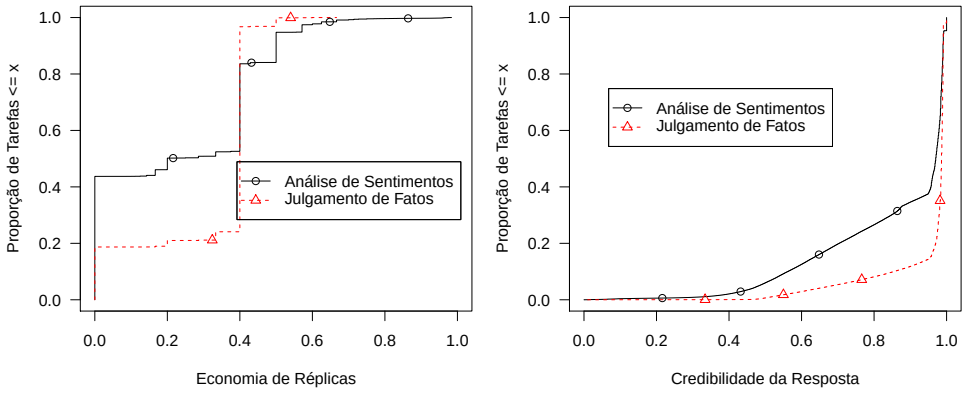
sendo em média 87% na aplicação Análise de Sentimentos e 90% na aplicação Julgamento de Fatos. Pelos resultados apresentados na Figura 1, pode-se concluir que, para as aplicações avaliadas, o algoritmo proposto permite reduzir substancialmente o número de réplicas das tarefas sem afetar negativamente a acurácia das respostas.




(a) Economia de réplicas (b) Acurácia

**Fig. 1. Economia de réplicas e acurácia das respostas por aplicação.**

Além do resultado agregado apresentado na Figura 1, também é importante verificar como o algoritmo proposto se comporta em cada tarefa. A Figura 2 tem como objetivo permitir essa análise. Essa figura apresenta as funções de distribuição acumulada da economia de réplicas obtida em cada tarefa (Fig. 2(a)) e do nível de credibilidade obtido na resposta final de cada tarefa (Fig. 2(b)). Na Figura 2(a) a economia de réplicas obtida é indicada no eixo horizontal e no eixo vertical é indicada a proporção de tarefas que atingiram economia menor ou igual à correspondente no eixo horizontal. O máximo de economia obtida em uma mesma tarefa é 0,66, na aplicação Julgamento de Fatos, e 0,98 na aplicação Análise de Sentimentos. Nas duas aplicações existem tarefas para as quais a economia de réplicas foi nula. Porém, a estratégia proposta economiza réplicas em 81% das tarefas na aplicação Julgamento de Fatos e em 56% das tarefas na aplicação Análise de Sentimentos.



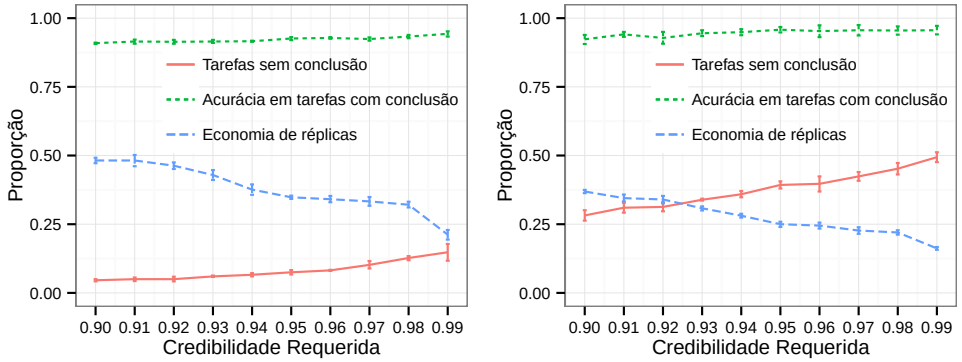

(a) Economia de réplicas em cada tarefa (b) Credibilidade da resposta final de cada tarefa

**Fig. 2. Funções de Distribuição Acumulada (FDAs) da (a) economia de réplicas obtida em cada tarefa e da (b) credibilidade da resposta final de cada tarefa.**

Na Figura 2(b), pode-se ver a distribuição da credibilidade da resposta final das tarefas obtida pelo algoritmo proposto. As tarefas cujas respostas não atingiram credibilidade de $0{,}95$ são uma proporção $0{,}10$ na aplicação Julgamento de Fatos e $0{,}45$ na aplicação Análise de Sentimentos. Ambos os resultados mostrados na Figura 2 indicam que a estratégia de replicação proposta apresenta pior desempenho em termos de economia de réplicas e da credibilidade final das respostas na aplicação Análise de Sentimentos. Provavelmente isso ocorre porque essa aplicação tem maior número de opções de resposta por tarefa e requer uma avaliação mais subjetiva do que a outra aplicação. Essas duas características contribuem para aumentar as chances dos trabalhadores divergirem e mais réplicas serem necessárias para tentar atingir maior credibilidade na resposta.

### 5.2. Configuração Conservadora da Aplicação

Neste cenário, apenas respostas que atingem o limiar de credibilidade requerido são consideradas na análise da acurácia. Tarefas em que a resposta final não atingiu o limiar são marcadas como "sem conclusão", porque são tarefas que podem requer uma avaliação mais especializada. Na Figura 3 são apresentados os resultados obtidos, sendo que na Figura 3(a) estão os resultado na aplicação Julgamento de Fatos e na Figura 3(b) estão os resultados na aplicação Análise de Sentimentos. Em ambos os gráficos, o eixo horizontal é a credibilidade requerida pelo dono da aplicação. O eixo vertical, por sua vez, é o valor obtido para cada uma das métricas que são apresentadas como linhas nos gráficos.




(a) Julgamento de Fatos (b) Análise de Sentimentos

**Fig. 3. Impacto da variação da credibilidade requerida na proporção de: i) economia de réplicas, ii) tarefas com conclusão iii) tarefas sem conclusão.**

As Figuras 3(a) e 3(b) mostram que na medida em que se aumenta a credibilidade requerida na resposta, ocorre uma redução na proporção de economia em relação à replicação não adaptativa e um aumento na proporção de tarefas sem conclusão. Na Figura 3(b) é possível ver o ponto em que o número de tarefas que não atingiram o limiar de credibilidade ultrapassa a economia obtida, isso ocorre entre a credibilidade requerida de $0{,}92$ e $0{,}93$. A proporção de acurácia, por sua vez, não sofre grandes variações quando a credibilidade requerida muda. Quando a credibilidade requerida varia de $0{,}90$ a $0{,}99$, a acurácia das respostas varia de $0{,}93$ para $0{,}97$ na aplicação Análise de Sentimentos e de $0{,}91$ para $0{,}93$ na aplicação de Julgamento de Fatos. Esses resultados indicam que, para os níveis de credibilidade requerida avaliados, quanto mais rigoroso o dono da aplicação é em termos da credibilidade requerida, menor é a economia de réplicas, maior é o número de tarefas sem conclusão e ocorre baixa variação na acurácia das tarefas com conclusão.

Além de obter economia de réplicas e manter a acurácia das respostas, outra vantagem da estratégia proposta é destacar como "sem conclusão" as tarefas em que nenhum grupo de resposta atingiu o limiar de credibilidade requerido. São tarefas cujas respostas não são conclusivas dado que os trabalhadores apresentam divergência entre as opções de resposta. Em computação por humanos, esse tipo de tarefa pode ter grande importância. Uma tarefa em que os trabalhadores divergem sobre a resposta correta pode revelar algo contraditório ou fora do padrão que o dono da aplicação pode ter interesse em investigar com maior cuidado. Dessa forma, a estratégia proposta também contribui para facilitar a análise dos dados obtidos nesses sistemas e também dá suporte a uma nova decisão como, por exemplo, usar trabalhadores mais especialistas para executar as tarefas sem conclusão.

### 5.3. Limitações e Implicações

A estratégia de replicação proposta neste trabalho possui algumas limitações que precisam ser destacadas. Ela foi projetada para ambientes em que os trabalhadores sempre executam os mesmos tipos de tarefas mudando apenas o item que está sendo avaliado. Esse é o caso, por exemplo, de sistemas de pensamento voluntário como Galaxy Zoo em que os voluntários sempre executam o mesmo tipo de tarefa, mudando-se apenas a imagem da galáxia a que a tarefa se refere. Esse também é o caso de grupos de tarefas que requerem inteligência humana (HITs, do inglês *Human Intelligence Tasks*) no Mturk, se eles admitirem que cada trabalhador execute mais de um HIT do grupo.

Para ser utilizada em sistemas de pensamento voluntário, a estratégia proposta pode ser implementada no nível do sistema de *middleware*, como o PyBossa[8]. Usuários do Mturk podem implementar a estratégia no nível da aplicação por meio de interfaces como o TurkIt[9]. Usuários de diversos sistemas de computação por humanos também podem usar a estratégia proposta para combinar diferentes sistemas. Por exemplo, primeiro as tarefas podem ser submetidas para serem executadas em sistemas como Mturk, que agregam trabalhadores não especialistas e que geralmente aceitam receber menos que trabalhadores especialistas. As tarefas cujos resultados não atingirem o limiar de credibilidade desejado, podem então ser submetidas para serem executadas em sistemas de computação por humanos que agregam trabalhadores mais especialistas e que, geralmente, cobram mais para executá-las, como Mobile Works (`mobileworks.com`) e InnoCentive (`innocentive.com`).

## 6. Conclusões

Este trabalho apresenta uma estratégia de replicação adaptativa que visa reduzir o número de réplicas necessárias para se obter uma resposta confiável em sistemas de computação por humanos. A estratégia é construída a partir de conceitos da Teoria de Erro Humano, da Teoria de Escolha Social e de replicação de tarefas em sistemas distribuídos compostos por máquinas. Resultados obtidos em simulações usando dados de duas aplicações reais mostram que a estratégia proposta permite uma considerável economia de réplicas, com acurácia similar à obtida quando se realiza voto majoritário sobre respostas redundantes obtidas por replicação não adaptativa. Outro benefício da estratégia proposta é destacar tarefas sem conclusão. Conhecer essas tarefas permite ao dono da aplicação tomar novas ações como usar trabalhadores mais especialistas em outros sistemas para executá-las.

[8] `https://github.com/PyBossa/pybossa`, último acesso em 25/11/2013.
[9] `https://code.google.com/p/turkit`, último acesso em 25/11/2013.

Trabalhos futuros podem investigar formas alternativas de medir o grau de dificuldade das tarefas e a convergência entre os trabalhadores, considerando medidas como entropia de Shannon e Cohen's kappa. Nesse estudo, pode-se identificar se variações na forma de medir o grau de dificuldade de tarefas e a convergência entre os trabalhadores permite melhorar o desempenho da estratégia de replicação proposta neste trabalho. Em futuras extensões da estratégia, além da economia de réplicas, acurácia das respostas e tarefas sem conclusão, outras métricas de desempenho como tempo de resposta das aplicações e vazão do sistema devem ser avaliadas.

## Referências

Chen, J. J., Menezes, N. J., Bradley, A. D., and North, T. (2011). Opportunities for crowdsourcing research on amazon mechanical turk. *Interfaces*, 5(3).

Cirne, W., Brasileiro, F., Paranhos, D., Góes, L., and Voorsluys, W. (2007). On the efficacy, efficiency and emergent behavior of task replication in large distributed systems. *Parallel Computing*, 33(3):213–234.

Ipeirotis, P. G., Provost, F., and Wang, J. (2010). Quality management on amazon mechanical turk. In *Human Computation Workshop*, pages 64–67. ACM.

Jalote, P. (1994). *Fault tolerance in distributed systems*. Prentice Hall.

Kochhar, S., Mazzocchi, S., and Paritosh, P. (2010). The anatomy of a large-scale human computation engine. In *Human Computation Workshop*, pages 10–17. ACM.

Kulkarni, A., Can, M., and Hartmann, B. (2012). Collaboratively crowdsourcing workflows with turkomatic. In *CSCW*, pages 1003–1012. ACM.

Oliveira, J., Ponciano, L., Andrade, N., and Brasileiro, F. (2013). Estratégias de obtencão de um item máximo em computacão por humanos. In *SBRC*, pages 253–266. SBC.

Ponciano, L., Brasileiro, F., and Gadelha, G. (2013). Task redundancy strategy based on volunteers' credibility for volunteer thinking projects. In *Proc. First AAAI Conference on Human Computation and Crowdsourcing*, pages 60–61. AAAI.

Ponciano, L., Brasileiro, F., Simpson, R., and Smith, A. (2014). Volunteers' engagement in human computation astronomy projects. *Computing in Science and Engineering*, 99:1.

Quinn, A. J. and Bederson, B. B. (2011). Human computation: a survey and taxonomy of a growing field. In *CHI*, pages 1403–1412. ACM.

Reason, J. (1990). *Human error*. Cambridge University Press Cambridge.

Sarmenta, L. F. (2002). Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572.

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

Sheng, V. S., Provost, F., and Ipeirotis, P. G. (2008). Get another label? improving data quality and data mining using multiple, noisy labelers. In *SIGKDD*, pages 614–622. ACM.

Sheshadri, A. and Lease, M. (2013). Square: A benchmark for research on computing crowd consensus. In *Proc. First AAAI Conference on Human Computation and Crowdsourcing*, pages 156 – 164. AAAI.

Taylor, A. and Pacelli, A. (2008). *Mathematics and Politics: Strategy, Voting, Power, and Proof*. Springer.

Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., and Blum, M. (2008). recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468.

# Influência do Facebook em Enxames Bittorrent.

**Thiago A. Guarnieri**[1] **Ana Paula C. da Silva**[2]
**Jussara M. Almeida**[2] **Alex Borges Vieira**[1]

[1]Departamento de Ciência da Computação, Universidade Federal de Juiz de Fora
[2]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais

{thiago.guarnieri, alex.borges}@ufjf.edu.br {ana.coutosilva, jussara}@dcc.ufmg.br

***Resumo.*** *O Bittorrent é uma das aplicações mais populares para compartilhamento de arquivos na Internet. Entretanto, o seu sucesso, assim como de outros sistemas Par-a-Par (P2P), depende da cooperação de seus usuários. Sistemas com usuários pouco altruístas ou sem mecanismos de incentivo estão condenados ao fracasso. Este trabalho discute a influência do Facebook em enxames Bittorrent. São analisados dados de mais de 16.600 enxames Bittorrent disseminados tanto pelo Facebook quanto por sistemas tradicionais. Os resultados encontrados indicam mudanças na rede P2P e melhorias quando um torrent é anunciado em redes sociais. Por exemplo, torrents sociais apresentam até 50% a mais de semeadores e quatro vezes mais peers que participam de múltiplos enxames. A proporção de usuários em um enxame pertencentes a uma mesma localização geográfica ou AS também é maior nos torrents sociais. Logo, caso o protocolo Bittorrent utilize conhecimento das redes físicas ou da localização geográfica dos peers na construção da rede P2P, torrents sociais apresentarão melhor localidade nas parcerias estabelecidas entre peers. Consequentemente, torrents sociais poderão ter melhor taxa de transferência, difusão de dados mais rápida e menor tráfego de dados entre ASes.*

***Abstract.*** *Bittorrent is one of the most popular file sharing applications on the Internet. It is well-known that its success mainly depends on its user cooperation. Systems with a small number of altruist users or without incentive mechanisms are fated to failure. This paper discusses the influence of Facebook on Bittorrent swarms. We analyze data from more than 16,600 Bittorrent swarms disseminated by Facebook and by traditional torrent websites. Our results show that the dissemination process is enhanced when a torrent is announced on a social network. Social torrents have up to* $50\%$ *more seeders and four times more peers that participate on multiple swarms. Furthermore, peers in social torrents exhibit high locality, both geographically and in terms of ASes. If the Bittorrent protocol leverages the knowledge of physical networks for building the P2P overlay network, social torrents may have better locality in the partnerships established between peers. As consequence, social torrents may have better throughput, faster data dissemination and less data traffic between ASes.*

## 1. Introdução

Desde a década passada, observa-se a massiva adoção do paradigma *peer-to-peer* (P2P) na Internet. De fato, o tráfego gerado por suas aplicações é de grande representatividade nas redes de computadores. Estimativas mostram que 36% de todo tráfego de *upload* da

América do Norte tem como origem o *Bitorrent*, atualmente a principal aplicação para compartilhamento de arquivos em redes P2P. Além disso a parcela deste protocolo no tráfego de pico desta região ultrapassa 10%[1].

O sucesso de sistemas P2P depende fortemente do nível de colaboração de seus participantes (*peers*). A falta de cooperação impõe um grande desafio à garantia de qualidade de serviço e por isso, existem vários mecanismos que procuram incentivar a cooperação em sistemas P2P. O *Bittorrent* implementa um esquema descentralizado conhecido por *tit-for-tat* [Cohen 2003]. Por esse esquema, *peers* monitoram-se mutuamente e assim, recursos são oferecidos a um *peer* de acordo com sua contribuição.

Apesar da larga adoção e uso do *tit-for-tat*, observa-se que em muitos casos, a rede P2P não se beneficia dele [Kaune et al. 2010]. Esse mecanismo não favorece relações baseadas em localização geográfica (ou topológica de redes) dos *peers* e nem relacionamentos de longa duração: de acordo com [Wang et al. 2011] a taxa de reencontro entre peers é menor que 5%. Mais ainda, não são considerados fatores sociais, como o relacionamento dos usuários do Bittorrent em outros tipos de redes (e.g., redes sociais).

É de se esperar uma melhor disseminação dos dados pelo Bittorrent caso se considere esses fatores [Andrade et al. 2012]. A intuição é simples: pode-se esperar de forma razoável forte cooperação entre parceiros, ou amigos, e assim, o *tit-for-tat* não seria necessário [Wang et al. 2013]. De fato, há indícios que a disseminação de dados no Bittorrent pode ser melhorada quando se utiliza redes sociais para fazer o seu anúncio inicial. Por exemplo, [Wang et al. 2011, Wang et al. 2013] mostram que *torrents* anunciados via Twitter, uma rede social popular, apresentam um número maior de *peers* estáveis, quando comparados a *torrents* anunciados de maneira tradicional. Esses *peers* estáveis entram no enxame e seguem padrões bem previsíveis. Eles se mantêm compartilhando por um período maior, melhorando a disseminação do conteúdo na rede P2P.

Adicionalmente, a localidade de um *peer*, mesmo em um único enxame, pode ser importante para redução global do tráfego entre sistemas autônomos. Por exemplo, [Wang and Liu 2012, Wang et al. 2011] mostram que 85% dos *peers* participam de múltiplos enxames e essa característica pode ser usada para reunir peers próximos em comunidades que compartilham o mesmo conjunto de arquivos.

Neste contexto, este trabalho avalia a influência de uma rede social, o Facebook, na disseminação de dados em redes Bittorrent. São analisados dados coletados de mais de 16.600 enxames Bittorrent reais, disseminados tanto pelo Facebook quanto por sistemas tradicionais (e.g., sistemas *web* como *pirate bay* e *btmon*). Diferente dos trabalhos anteriores, que focam no estudo das parcerias e como os arquivos são disseminados mais rapidamente, neste trabalho as comparações entre *torrents* tradicionais e os disseminados pelo Facebook são abordados sob 3 aspectos: (I) as características gerais da rede P2P, (II) as características que indicam melhor disseminação de dados (definidas como saúde de um *torrent*) e (III) a localidade dos *peers* da rede P2P.

Os resultados deste trabalho indicam mudanças na rede P2P e melhorias quando um *torrent* é anunciado pelo Facebook. Por exemplo, indicadores de saúde, como proporção de semeadores (*seeders*) e quantidade de peers que participam de múltiplos enxames são significantemente maiores. Os *torrents* sociais apresentam até 50% a

[1]Global Internet Phenomena Report: 2H 2012, www.sandvine.com

mais de *seeders* e quatro vezes mais *peers* que participam de múltiplos enxames. Essa característica pode ser explorada em mecanismos de incentivo que considerem a permanência em múltiplos enxames, premiando o usuário com, por exemplo, um aumento em seu *share-ratio*. A proporção de usuários em um enxame pertencentes a uma mesma localização geográfica ou AS também é maior nos *torrents* sociais. Dessa forma, caso o protocolo Bittorrent utilize conhecimento das redes físicas na construção da rede sobreposta, *torrents* sociais apresentarão melhores resultados no agrupamento de peers. Resumindo, os resultados apontam na direção de uma difusão mais eficiente de conteúdo, quando um *torrent* é disseminado pelo Facebook.

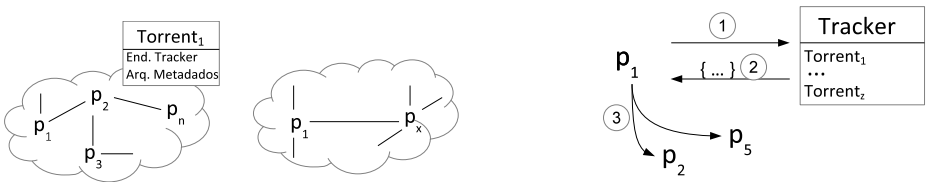
## 2. Bittorrent e Metodologia de Coleta de Dados

Nesta seção, são apresentados conceitos gerais sobre o Bittorrent, o principal sistema de compartilhamento de arquivos em P2P na atualidade. Além disso, é apresentada a metodologia de coleta de dados e as definições relativas aos *torrents* sociais.

### 2.1. Bittorrent Revisitado

O Bittorrent é, atualmente, a principal aplicação para compartilhamento de arquivos via P2P. Arquivos são compartilhados por ele em redes P2P sobreposta à rede física existente. Essa rede sobreposta não apresenta hierarquias explícitas. Ela é baseada em malhas com troca de dados realizadas por pedidos explícitos entre seus participantes (*peers*).

Cada conjunto de arquivos, compartilhado via um *torrent*, apresenta uma rede sobreposta correspondente (enxame *torrent*). Um arquivo *torrent* contém informações sobre possíveis entidades centralizadoras (*trackers*) que contabilizam estatísticas sobre o compartilhamento e gerenciam os *peers* do enxame. Além disso, um arquivo *torrent* possui informações sobre os arquivos compartilhados, os *chunks* de arquivos, assim como códigos *hash* de cada *chunk* (utilizado para verificação de integridade).




(a) Enxames Torrent.

(b) Processo de entrada no enxame.

**Figura 1. Usuários de BitTorrent são agrupados em enxames P2P. Cada enxame corresponde a uma rede sobreposta onde são compartilhados arquivos relacionados a um único *torrent*. Para iniciar o processo de compartilhamento, usuários do Bittorrent, geralmente, contatam uma entidade centralizadora.**

Como mostra a Figura 1-a, um enxame *torrent* contém $n$ *peers* que colaboram entre si para disseminar o conteúdo dos arquivos compartilhados. Cada *peer* $p_i$ possui uma lista com $m_i$ parceiros e troca dados (*chunks*) somente com eles. Os *peers* são livres para abandonar parcerias, além de entrar e sair do enxame a qualquer momento. Periodicamente, os *peers* trocam entre si mapas de *chunks*. Assim, $p_i$ conhece os *chunks* disponíveis em seus parceiros, e vice-versa. Assim, $p_i$ faz requisições explícitas a seus parceiros pelos *chunks* que ele necessita para completar o(s) arquivo(s) compartilhado(s).

Para entrar em um enxame, um *peer* novato $p_i$, geralmente, contata uma entidade centralizadora - o *tracker* (Figura 1-b[1]). O *tracker* retorna $p_i$ uma lista com *peers* candidatos a parceria (Figura 1-b[2]). Finalmente, $p_i$ tenta contatar cada um dos candidatos

e envia requisições de parceria (Figura 1-b[3]). Uma vez estabelecida a parceria, $p_i$ troca mapas de *chunks* e dados com seus parceiros.

Um *peer* $p_i$ pode participar de múltiplos enxames. Em outras palavras, quando $p_i$ realiza múltiplos compartilhamentos (cada um associado a um *torrent*), ele participa de diferentes redes sobrepostas. Em cada enxame, $p_i$ pode apresentar diferentes parcerias, assim como diferentes níveis de cooperação. Um dos mecanismos mais importantes do Bittorrent é o tit-for-tat [Cohen 2003]. Nesse esquema descentralizado, os *peers* se monitoram mutuamente, e assim, recursos são oferecidos a um parceiro de acordo com sua contribuição prévia. De forma simples, quando $p_i$ doa algum *chunk* a um parceiro $p_j$, ele fica com créditos com esse parceiro. Assim, $p_j$ prioriza requisições realizadas por $p_i$ até que seu débito com $p_i$ se anule.

### 2.2. Coleta de Dados

Para a condução deste trabalho, foram coletados 16.698 arquivos *torrent*. Estes arquivos foram divididos em duas classes como segue: *torrents* tradicionais, que são os arquivos *torrents* coletados de sistemas web tradicionais (www.btmon.com) e *torrents* sociais, que são arquivos *torrents* coletados a partir de comunidades no Facebook.

O conjunto de *torrents* tradicionais foi obtido por um processo de busca recursiva em um sistema web dedicado a anúncios de *torrent* (www.btmon.com). A partir da página principal, arquivos *torrents* foram capturados para posterior análise. A coleta foi interrompida no primeiro nível de recursividade com um número de *torrents* acima de 15 mil. Assim, Foram coletados 15.086 *torrents* a partir desse sistema web.

Os *torrents* sociais foram obtidos em comunidades do Facebook, a partir de um coletor desenvolvido com a API pública do Facebook. O coletor varre uma lista de comunidades inicialmente usada como semente. Essa lista foi definida a partir de buscas no Facebook por termos como "*Bittorrent*", "*torrent*"e "*P2P sharing*". Adicionalmente, são varridas as comunidades relacionadas às encontradas na lista inicial. Foram encontrados 1.612 *torrents* nessas comunidades. Foram encontrados apenas 38 *torrents* pertencentes a ambos os conjuntos. Assim, por representarem uma parcela pequena do total de dados coletados, esses *torrents* foram excluídos das análises apresentadas.

A coleta dos dados dos enxames de cada um dos *torrents* capturados foi efetuada utilizando-se a rede do campus da Universidade Federal de Juiz de Fora, por um período de 60 dias (julho a agosto de 2013). Foi desenvolvido um coletor de informações (*crawler*), utilizando a linguagem de programação *python* (python.org) em conjunto com a biblioteca *libtorrent* (libtorrent.org). O *crawler* permanece continuamente conectado aos enxames durante todo o período de coleta. Nesse período, ele apenas faz requisições aos *trackers* e estabelece parcerias. Por respeito as propriedades e direitos intelectuais, nenhum arquivo compartilhado nos *torrents* é capturado e armazenado em disco.

Nesse trabalho, informações e metadados a respeito dos enxames são obtidas diretamente dos *trackers* associados aos *torrents* (e.g., tamanho do enxame e quantidade de *seeders*). A data de criação, embora mportante para medir popularidade, não foi armazenada por não ser de preenchimento obrigatório no conjunto de metadados. Além disso um arquivo *torrent* pode ser republicado, o que torna essa informação imprecisa. Além disso, a popularidade também é influenciada pelo tipo de conteúdo disseminado. Já informações a respeito da localização dos *peers* não são providas pelos *trackers*. Essas informações

foram obtidas a partir da consulta dos endereços IP dos *peers* na base de dados Geolite (www.maxmind.com), por onde é possível obter dados a respeito do país, do AS e da empresa que gerencia o endereço IP do *peer torrent*.

Para descrever as propriedades gerais dos enxames (e.g., o tamanho dos enxames e a quantidade de arquivos compartilhadas) foram utilizadas todas as amostras coletadas. Entretanto, para comparar a localização dos *peers* dos enxames e a saúde dos enxames, foi utilizado um subconjunto aleatório dos *torrents* tradicionais, de tamanho similar à amostra social. O tamanho dos *swarms* também foi utilizado como critério de escolha. Assim, as comparações são realizadas em cojuntos de mesma ordem de grandeza.

## 3. Caracterização e Resultados

Esta seção discute os resultados da comparação entre *torrents* anunciados via rede social e *torrents* anunciados de forma tradicional. A comparação é realizada em três dimensões. Na seção 3.1, são discutidas as propriedades gerais dos *torrents*, como a quantidade e os tipos de arquivos disseminados em cada enxame e os tamanhos das redes de compartilhamento. Na seção 3.2, a saúde de um enxame em cada tipo de *torrent* é avaliada. No contexto desse trabalho, a saúde de um enxame refere-se a um conjunto de métricas que indicam bons níveis de compartilhamento como proporções entre cópias completas e incompletas e a fração de participantes que fazem compartilhamento em múltiplos enxames. Finalmente, na seção 3.3 são comparadas as características de localização geográfica e topológica dos participantes dos dois tipos de enxames estudados.

A tabela 1 resume o conjunto de dados coletado para realizar esse trabalho. Nas análises apresentadas, foram identificados quase 2 milhões de *peers*, espalhados por mais de 11 mil sistemas autônomos e 230 países diferentes. Os *torrents* analisados cobrem praticamente 1 milhão de arquivos com quase 1800 extensões diferentes.

| | *Torrent* tradicional | *Torrent* social |
|---|---|---|
| Número de IPs únicos | 1.280.688 | 358.463 |
| Número de Participantes | 1.434.393 | 410.858 |
| Número de países | 230 | 222 |
| Número de ASs | 11.366 | 7.567 |
| Número de Trackers | 3.602 | 821 |
| Número de Arquivos | 962.997 | 23.398 |
| Número de Extensões de arquivo | 1.747 | 457 |

**Tabela 1. Resumo da Amostra de Dados Coletada.**

### 3.1. Propriedades Gerais

Os enxames analisados apresentam uma população que pode variar de apenas alguns participantes até dezenas de milhares. A Figura 2 apresenta a distribuição dos tamanhos dos enxames. A linha azul, marcada com quadrado, apresenta a distribuição de todos dos enxames avaliados (*torrents* sociais e tradicionais); a linha em preto, marcada com círculo, apresenta a distribuição somente dos enxames de *torrents* tradicionais; e a linha vermelha, marcada com asterisco, apresenta a distribuição para os *torrents* sociais.

Quando se avalia todos enxames (linha azul), observa-se que, apesar de existirem enxames com milhares de *peers*, a grande maioria apresenta na ordem de até 100 participantes. De fato, mais de 40% apresentam menos de 10 participantes. Há também uma fração não desprezível de enxames com apenas 1 participante (por volta de 2,7%).

Os enxames sociais e tradicionais apresentam propriedades gerais diferentes. Por exemplo, os enxames sociais avaliados tendem a ser maiores que os tradicionais. De

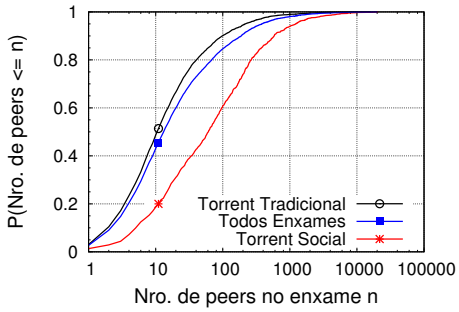



**Figura 2. Distribuição do tamanho dos enxames de *torrents*.**

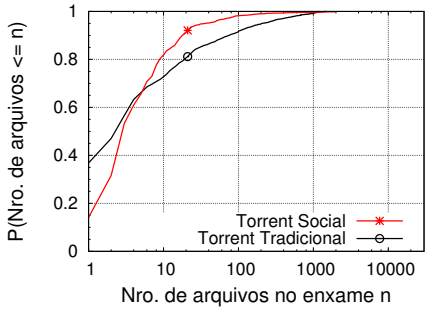



**Figura 3. Distribuição da quantidade de arquivos compartilhados.**

acordo com a Figura 2 enquanto 50% dos enxames tradicionais apresentam menos de 10 *peers* durante a observação, a mesma fração de enxames de *torrents* sociais apresentam até 60 *peers*. Na média, durante o período de observação, um *torrent* social tem três vezes mais *peers* que um *torrent* tradicional. Além disso, cerca de 3% dos *torrents* tradicionais apresentam somente 1 *peer*. Para *torrents* sociais esse número cai para cerca de 1%.

O número de enxames com baixo número de participantes possivelmente reflete a distribuição de popularidade entre os *torrents*. Geralmente, arquivos compartilhados atraem a atenção apenas nos períodos de recente divulgação. Com o passar do tempo, esses arquivos são disseminados e perdem popularidade. Seus enxames perdem participantes e podem chegar ao caso extremo de se tornarem *torrents* mortos (*dead torrents*), isto é, enxames sem cópias completas dos arquivos compartilhados.

A Figura 3 apresenta as distribuições do tamanho do conjunto de arquivos compartilhados em um único *torrent*. Para avaliar estas distribuições, os *torrents* (sociais e tradicionais) são divididos em 2 grupos: *torrents* que compartilham até 7 arquivos e *torrents* que compartilham mais de 7 arquivos. Considerando somente os *torrents* que compartilham até 7 arquivos, nota-se que os *torrents* sociais tendem a ser um pouco maiores que os tradicionais. Por exemplo, 36% dos *torrents* tradicionais neste grupo compartilham apenas 1 arquivo versus somente 15% dos *torrents* sociais. Já para os *torrents* que compartilham mais de 7 arquivos, nota-se uma inversão de padrões: os *torrents* tradicionais tendem a ser maiores. Por exemplo, cerca de 9% dos *torrents* tradicionais têm mais de 100 arquivos, mas menos de 2% dos sociais têm essa mesma quantidade. Na média geral, os *torrents* tradicionais têm 63,26 arquivos, enquanto os sociais somente 14,51.

Entre os *torrents* avaliados, os dois *torrents* tradicionais com maior número de arquivos compartilhados apresentam 29.780 e 9.845 arquivos. Ambos têm mais de 50

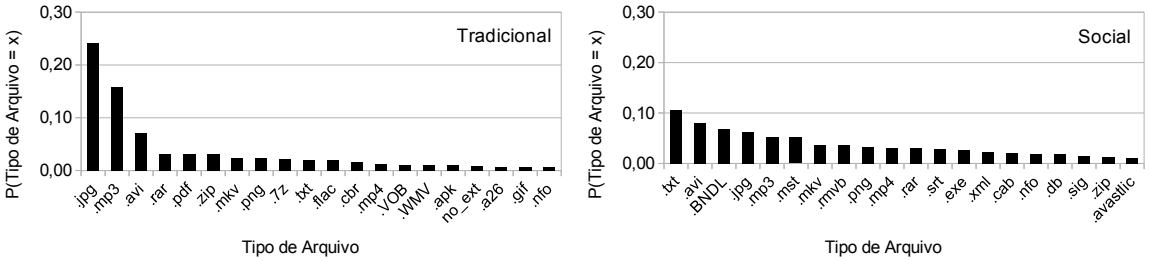



**Figura 4. Top 20 extensões compartilhadas.**

GB de tamanho e compartilham jogos e músicas respectivamente. Em contrapartida, os *torrents* sociais, os dois enxames com maior número de arquivos compartilhados têm somente 1.973 e 1.421 arquivos. Os dois conjuntos têm 6 GB e 3 GB respectivamente e ambos compartilham jogos e programas (com extensões dll, cab, exe, .dat, msi, etc).

Finalmente, a Figura 4 apresenta as 20 extensões de arquivo mais comuns em cada tipo de *torrent*. Note que, os arquivos mais comuns dos enxames de *torrents* tradicionais são diferentes dos encontrados nos *torrents* sociais (i.e. os eixos "x"dos gráficos são diferentes). Especificamente, as comunidades sociais tendem a compartilhar temas específicos de arquivos (e.g. jogos e filmes). Por esse motivo, as 20 extensões nesse tipo de *torrent* são praticamente todas relativas a vídeo e áudio (mp3, wmv, avi, etc.) e jogos (cab, zip, BNDL, etc.). Enquanto isso, *torrents* tradicionais não são enviesados por tema, tendo uma diversidade maior nas extensões dos arquivos compartilhados. Desta forma, arquivos de interesse restrito tendem a ser encontrados mais facilmente via mecanismos tradicionais de disseminação.

Tanto para os *torrents* sociais quanto para os tradicionais, é comum existir arquivos de texto descrevendo o conteúdo compartilhado. Assim, observa-se que em ambos os conjuntos há um número expressivo de arquivos "txt" e "nfo".

### 3.2. Saúde dos Enxames *Torrent*

Neste trabalho, a saúde de um *torrent* indica o quanto esse *torrent* pode ter sucesso em ser compartilhado. Por exemplo, quanto maior a quantidade de cópias completas, ou maior o número de *seeders*, maiores as chances de se difundir os dados de um *torrent*.

| | Tradicional | Social |
|---|---|---|
| % de *peers* sem *tracker* | 82% | 84% |
| % de enxames *multitracker* | 81% | 92% |
| % de *torrents* mortos (*dead torrents*) | 47,45% | 28,16% |

**Tabela 2. Aspectos gerais de saúde dos enxames.**

A Tabela 2 resume aspectos gerais relacionados à saúde das duas classes de *torrents*. Tanto os *torrents* sociais quanto os tradicionais apresentam proporções semelhantes de *peers* sem *tracker* associado. Como discutido anteriormente (Seção 2), um *tracker* serve para gerenciar um enxame. Porém, por ser um ponto de centralização, ele pode ser alvo de ataques ou impedimentos judiciais[2]. Atualmente, como corroborado pelos números da Tabela 2, os *torrents* tendem a ser disseminados sem ajuda de *trackers*, o que pode aumentar sua resiliência. A tabela também mostra um número significativamente maior de *torrents* sociais que são auxiliados por mais de um *tracker*. Em outras palavras, por

[2]No caso de compartilhamento de conteúdo ilegal ou com restrições de *copyright*.

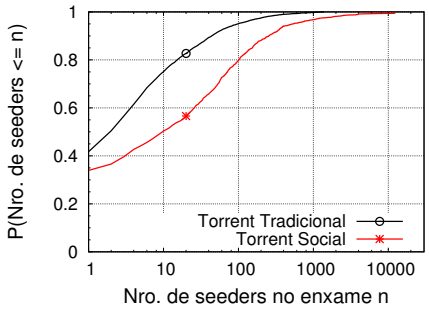



**Figura 5. Distribuição do número de *seeders* em um enxame.**

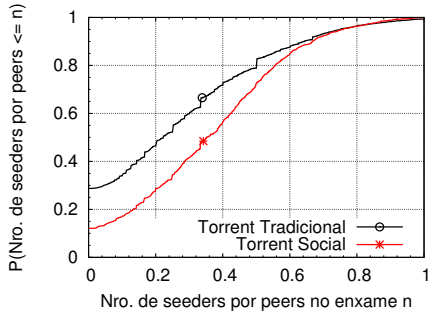



**Figura 6. Distribuição da proporção *seeders*/*peers* em um enxame.**

terem redundância de assistência, os *torrents* sociais podem apresentar melhor resiliência que os *torrents* tradicionais. Finalmente, há um número maior de *torrents* tradicionais mortos, comparados aos *torrents* sociais. Quase metade dos *torrents* tradicionais analisados já haviam perdido interesse do público e não tinham nenhum *seeder* no enxame. Esse número cai para 28% entre os *torrents* sociais.

Em números absolutos, os *torrents* sociais tendem a ter mais *seeders* que os tradicionais. A Figura 5 apresenta as distribuições acumuladas dos números de *seeders* de um enxame. Note que 80% dos *torrents* tradicionais têm menos de 16 *seeders*. Em comparação, a mesma fração de *torrents* sociais chega a ter 102 *seeders*. Em média, *torrents* sociais têm 137,3 *seeders* enquanto os tradicionais têm apenas 39,06.

Um número maior de *seeders* pode ser reflexo de um maior número de participantes no enxame. Assim, uma outra métrica que reflete a saúde de um *torrent* é a razão entre o número de *seeders* e o número total de *peers* no enxame. A Figura 6 apresenta as distribuições dessa proporção para *torrents* sociais e tradicionais. Os *torrents* sociais também tendem a ser melhores que os tradicionais quanto à esta métrica: 60% dos *torrents* sociais têm mais de 28% de seus participantes atuando como *seeders*, enquanto que somente 40% dos *torrents* tradicionais têm uma proporção de *seeders* nesta faixa.

A incidência de grupos de *peers* que participam de mais de um enxame também pode ser um indicador de saúde de um *torrent* uma vez que, neste caso, realocação e agrupamento dinâmico de enxames podem melhorar o desempenho do compartilhamento dos arquivos pertencentes aos enxames com *peers* em comum [Dán and Carlsson 2009].

A Figura 7 mostra a distribuição acumulada do número de *peers* que participam de mais de um enxame em cada tipo *torrent* analisado. Nitidamente, observa-se que os *torrents* sociais apresentam um número maior de *peers* com esse tipo de característica. Na

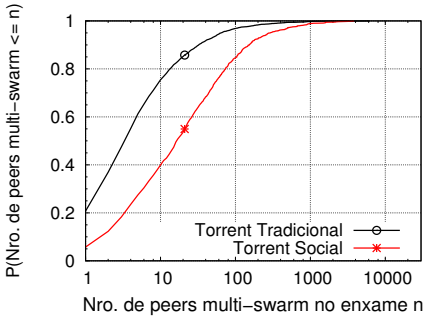



**Figura 7. Distribuição de *peers* que participam de múltiplos *swarms*.**

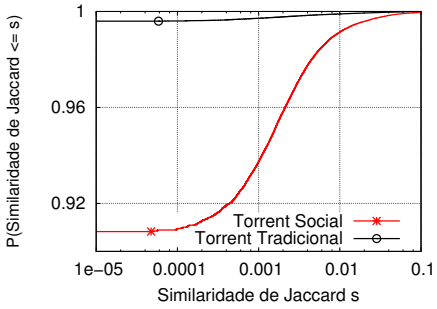



**Figura 8. Distribuição do coeficiente de similaridade de Jaccard.**

média, os *torrents* sociais têm cerca de 81 *peers* que participam de múltiplos enxames, e os tradicionais apenas 20. Em termos percentuais, cerca de 20% dos *peers* de *torrents* tradicionais participam de apenas 1 enxame, enquanto nos *torrents* sociais, esta proporção é de 5%.

Intuitivamente, *peers* com interesses em comum tendem a compartilhar também a participação em múltiplos enxames e assim, o protocolo Bittorrent poderia usar isso para estabelecer parcerias de longa duração. Além da melhora das parcerias formadas, a participação em múltiplos enxames pode ser explorada para balancear a carga (contribuição) entre os *peers* dos vários enxames envolvidos.

Para avaliar a coparticipação de *peers* em múltiplos enxames, foi analisado o coeficiente de Jaccard entre os conjuntos de participantes de cada par de enxames, para cada tipo de *torrent*. Mais precisamente, seja $A$ e $B$ dois enxames. O coeficiente de Jaccard é uma medida de similaridade entre esses dois conjuntos $A$ e $B$, sendo definido como:

$$\text{Coeficiente de similaridade de Jaccard(A,B)} = \frac{|A \cap B|}{|A \cup B|}.$$

A Figura 8 apresenta a distribuição dos coeficientes de Jaccard encontrada entre os enxames sociais e tradicionais. Claramente, observa-se o efeito da disseminação social no surgimento de grupos de peers em comum entre enxames: enquanto essa característica é praticamente ausente em enxames tradicionais, a mesma se torna evidente na difusão social, mesmo levando-se em conta o fato de que o *Facebook* não tem foco em distribuição de conteúdo e que os enxames sociais também podem ser disseminados em sites de indexação, o que poderia reduzir suas interseções.

Para melhor visualização de como a disseminação social faz surgir grupos de peers comuns, a Figura 9 apresenta as matrizes de similaridade para os enxames tradicionais e sociais. Cada cruzamento de linha e coluna dessa matriz representa a semelhança entre


(a) Torrents tradicionais (b) Torrents sociais.

**Figura 9. Matriz de similaridade.**

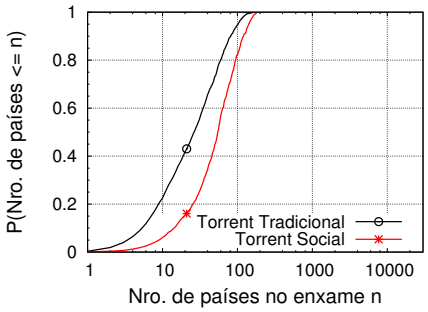



**Figura 10. Número de países distintos em um enxame.**

2 enxames. O coeficiente de similaridade entre 2 enxames varia de baixo (branco) a alto (azul escuro). Enquanto a matriz dos *torrents* tradicionais indica baixa similaridade entre todos os pares de enxame desse tipo, a matriz de *torrents* sociais indica o contrário: uma grande proporção de pares de enxame de *torrent* social tem evidente similaridade. Observa-se então a superposição entre o grafo de amigos do Facebook e os *peers* do enxame, o que poderia estar induzindo a formação de proto-comunidades de peers em decorrência da disseminação social.

### 3.3. Localização do *peers* nos enxames *torrent*

De forma geral, aplicações P2P não consideram localização na rede física para criar as parcerias da rede sobreposta [Seedorf et al. 2009]. O mesmo ocorre com o Bittorrent. Seu protocolo tenta maximizar o volume de dados trocados entre os *peers*, sem tomar conhecimento de suas localizações geográficas ou outras propriedades da rede físicas [Varvello and Steiner 2011]. Isso pode causar um grande volume de tráfego entre ASes ou ISPs, e consequentemente, alto custo.

Há várias propostas para lidar com os problemas que o desconhecimento da rede física pode causar na construção da rede P2P. Uma abordagem comum é enviesar a seleção de parcerias e favorecer aquelas entre *peers* que são localizados próximos uns aos outros (i.e. *peers* localizados em um mesmo ISP, AS ou país) [Seedorf et al. 2009, Varvello and Steiner 2011]. Assim, uma comparação da localização dos *peers* nos enxames *torrent* permite avaliar o potencial de eficácia desta estratégia.

Os *torrents* sociais e tradicionais apresentam diferenças substanciais com relação à localização de seus *peers*. Por exemplo, observa-se um número maior de países entre os *peers* de um *torrent* social. A Figura 10 mostra a distribuição do número de países identificados nos enxames analisados. Em média, os *peers* nos *torrents* sociais estão

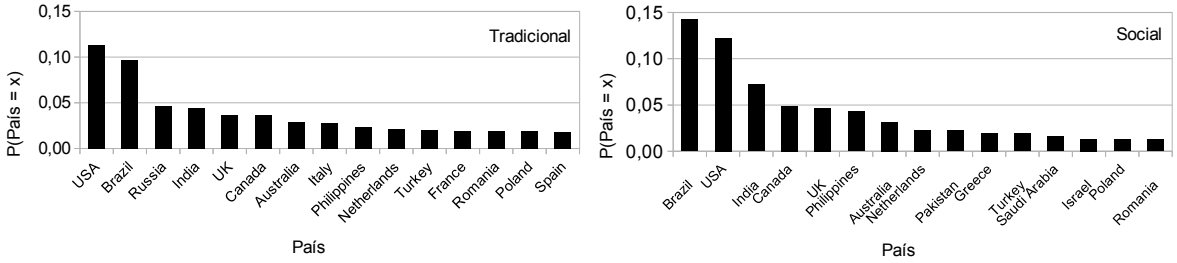



**Figura 11. Top 15 países mais frequentemente encontrados nos enxames.**

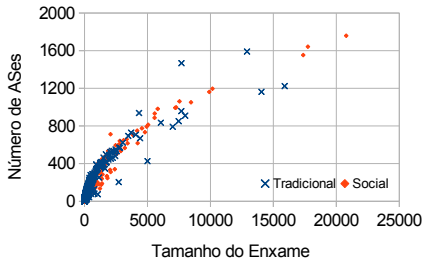



**Figura 12. Número de ASes por enxame.**

localizados em até 30 países distintos, contra apenas 14 países nos *torrents* tradicionais.

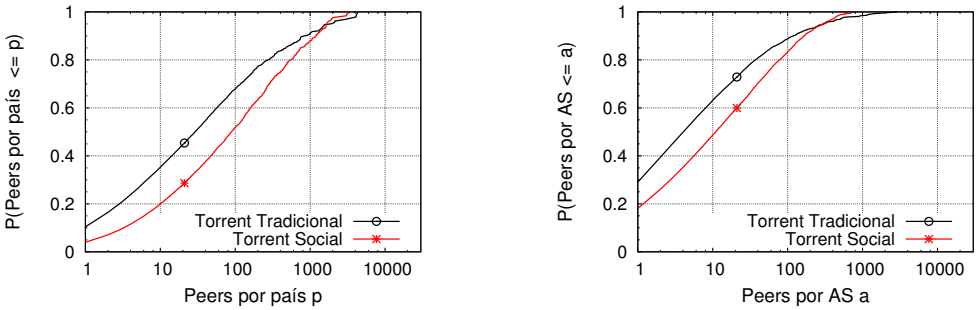
A Figura 11 apresenta os top-15 países mais frequentemente encontrados em cada tipo de *torrent*. Note que, embora as distribuições sejam razoavelmente semelhantes, porém, os países são diferentes. A distribuição para *torrents* sociais é influenciada pela distribuição dos países mais comuns no *Facebook*[3]. Países como Índia, Filipinas e Brasil tiveram suas posições melhoradas no *ranking* para *torrents* sociais, o que indica uma sobreposição entre as redes social e P2P.

Como mencionado, a localização topológica dos *peers* na rede física também pode ser explorada para favorecer estabelecimento de parcerias entre pares localizados no mesmo AS, por exemplo, reduzindo assim o tráfego entre ASes diferentes. Logo, a localização dos *peers* nos enxames em termos de ASes também é analisada.

A Figura 12 apresenta o número de ASes encontrados em cada enxame e seu respectivo tamanho (em número de *peers*). Não há grandes diferenças entre os enxames sociais e os tradicionais. Para ambos tipos de *torrents*, pode se observar uma correlação não linear entre o tamanho dos enxames e a quantidade de ASes encontrados entre seus participantes. Além disto, em ambos casos, os enxames apresentam tipicamente até 800 ASes diferentes entre seus participantes. Entretanto, há enxames com *peers* distribuídos por até 1800 ASes diferentes.

De acordo com a Figura 13, *torrents* sociais apresentam maiores proporções de *peers* por país e por AS. Em outras palavras, a localidade (por país e por AS) é maior nos *torrents* sociais. Praticamente 50% dos *torrents* tradicionais tem de menos de 29 *peers* por país. Esse número é significativamente maior *torrents* sociais: 50% dos *torrents* chega ter até 89 *peers* por país. A diferença na relação de *peers* por AS é mais estreita, mas ainda assim impactante. Enquanto 60% dos enxames tradicionais têm menos de 10 *peers* por AS, a mesma fração de enxames de *torrents* sociais apresenta até 22 *peers* por AS.

[3]http://www.quintly.com/blog/2013/03/facebook-country-statistics-march-2013/




(a) Proporção de *peers* por país (b) Proporção de *peers* por AS

**Figura 13. Localização dos *peers* de um enxame *torrent*.**

Esses números deixam claro que, caso o protocolo Bittorrent utilize conhecimento das redes físicas na construção da rede P2P, *torrents* sociais apresentarão melhores resultados no agrupamento de *peers*. Consequentemente, *torrents* sociais poderão ter melhor taxa de transferência, difusão de dados mais rápida e menor tráfego de dados entre ASes.

## 4. Trabalhos Relacionados

Há um grande número de trabalhos que foca na análise e melhoria da disseminação de conteúdo no Bittorrent. Notadamente, um dos mecanismos mais importantes nessa linha é o tit-for-tat [Cohen 2003]. Esse mecanismo incentiva a cooperação mútua nos enxames *torrent* a partir de monitorações distribuídas, realizadas localmente pelos *peers*. De certa forma, a eficiência do Bittorrent é associada a esse mecanismo. Porém, há um grande número de enxames *torrents* que não se beneficiam dele.

De fato, cerca de 40% dos torrents apresentam um desempenho baixo, com um grande número de usuários incapazes de obter o conteúdo compartilhado por completo [Kaune et al. 2010]. Esse baixo desempenho é explicado pela falta de incentivo a um *peer* para que ele permaneça contribuindo com o enxame após ele ter conseguido capturar o conteúdo completo. O tit-for-tat, que é o principal mecanismo de incentivo no Bittorrent, não utiliza informações de longa duração, não explora as interações sociais dos *peers* e não extrapola as avaliações entre enxames diferentes. Assim, tão logo um *peer* consegue baixar os arquivos compartilhados, ele costuma abandonar o enxame.

Há um número ainda pequeno de trabalhos que extrapola os incentivos providos pelo tit-for-tat e tenta utilizar o contexto social dos *peers* para melhorar a eficiência de sistemas P2P. Por exemplo, [Cheng and Liu 2009] explora o contexto social do Youtube[4] e mostra a forte correlação existente entre os vídeos Youtube e os que são sugeridos na página de exibição. Assim, eles propõem mecanismos para melhorar a difusão desses vídeos em redes P2P. Os *peers* fazem cache dos vídeos e redistribuem o entre seus parceiros, inclusive aqueles que estão assistindo os vídeos sugeridos.

Wang et al. apresentam três trabalhos que tratam o contexto social na difusão de *torrents* [Wang et al. 2011, Wang and Liu 2012, Wang et al. 2013]. Nesses trabalhos, os autores avaliam contatos de longa duração entre *peers* de enxames *torrents* que são divulgados pelo Twitter. Segundo os autores, em *torrents* divulgados pelo Twitter, há uma grande localidade temporal entre os *peers*. Enquanto nos enxames de *torrents* tra-

[4]youtube.com

dicionais, menos de 5% dos *peers* se reencontraram durante os 80 dias de experimento, esse número chega a 35% nos *torrents* divulgados pelo Twitter. Essa localidade temporal oferece uma grande oportunidade para melhorar o grau de compartilhamento. Por exemplo, em seus experimentos, os autores mostram que a latência inicial de *download* dos arquivos e o tempo para completar o *download* são melhorados.

Outros fatores, além da localidade temporal, também são impactados pelo contexto social. Anteriormente, [Guarnieri et al. 2013] mostraram que o perfil dos arquivos compartilhados e o agrupamento dos *peers* em torno de determinados países é diferente quando o *torrent* é disseminado via Facebook. Os resultados preliminares desse trabalho indicam uma melhor localização dos *peers* de um enxame, com possível formação de grupos em torno de países e ASes.

Neste trabalho, diferente dos anteriores, são analisados três aspectos diferentes do impacto da divulgação em redes sociais de *torrent*. Inicialmente, são avaliadas as características gerais da rede P2P e dos arquivos compartilhados. Em segundo lugar, é quantificada a diferença na saúde dos *torrents* sociais e dos tradicionais. Nesse caso, são verificadas métricas que indicam que um *torrent* social tem melhor desempenho que um tradicional. Mesmo sem nenhuma modificação nos mecanismos existente do Bittorrent, as métricas de saúde indicam que os *torrents* sociais podem ter melhores tempos de disseminação dos arquivos, melhores proporções de (*seeders*) e menos enxames mortos (*dead torrents*). Finalmente, é abordado a localidade dos *peers* da rede P2P, principalmente quanto ao agrupamento de peers em países e ASes. O estudo da localização dos *peers* estende o trabalho anterior [Guarnieri et al. 2013] e avalia não só o aspecto de agrupamento em torno de países ou ASes específicos, mas quantifica esses agrupamentos.

## 5. Conclusões e Trabalhos Futuros

Nesse trabalho foi avaliada a influência de uma rede social, o Facebook, na disseminação de arquivos pelo Bittorrent. Foram analisados dados coletados de mais de 16.600 enxames Bittorrent reais, disseminados tanto pelo Facebook quanto por sistemas web tradicionais, como o *pirate bay* e *btmon*.

As propriedades gerais dos *torrents* são diferentes entres os sociais e os tradicionais. Enxames de *torrents* sociais tendem a ser maiores. Na média, observou-se que enxames sociais são 3 vezes maiores que os tradicionais. O número de *dead torrents* é maior entre os enxames tradicionais. Além disso, cerca de 3% dos *torrents* tradicionais apresentam somente um *peer*. Para *torrents* sociais esse número cai para cerca de 1%. Os *torrents* sociais também apresentam indicadores de saúde melhores que os tradicionais. Por exemplo, os *torrents* divulgados pelo Facebook apresentam até 50% a mais de semeadores (*seeders*) e quatro vezes mais *peers* que participam de múltiplos enxames. É relevante ressaltar que, como os grupos sociais tendem a se organizar em torno de um interesse comum, a variedade de conteúdo ofertada é menor. Portanto, o usuário que busca por um conteúdo de interesse restrito, terá mais probabilidade de sucesso se recorrer a mecanismos de disseminação tradicionais.

A quantidade e a distribuição de países nos enxames sociais e tradicionais são, no geral, similares. Porém, os países envolvidos mudam. Em particular, países populares no Facebook tendem a ocupar posições de maior destaque nos *torrents* sociais, o que indica uma sobreposição entre as redes social e P2P.

Finalmente, a proporção de usuários em um enxame pertencentes a uma mesma localização geográfica ou AS também é maior nos *torrents* sociais. Isto implica que o uso de propriedades das redes físicas ou da localização geográfica pode levar a um maior agrupamento dos *peers* nos *torrents* sociais e menor tráfego de dados entre ASes distintos. A localidade, associada à maior interseção entre exames indica também uma possível melhoria do desempenho global de sistemas P2P, pois peers podem colaborar entre si em diversos enxames, promovendo relações de maior duração.

Como trabalho futuro, pretende-se avaliar as diferenças entre as topologias de rede de enxames de *torrents* sociais e tradicionais. Também é de interesse a investigação da formação das comunidades em torno dos *peers* que participam de múltiplos enxames (*multi-swarm*). A identificação de padrões nestas comunidades pode ajudar no desenvolvimento de um protocolo de escolha de parceiros baseado em interesses comuns.

**Agradecimentos**
Os autores agradecem o apoio do CNPq, CAPES, FAPEMIG, Instituto Nacional de Ciência e Tecnologia para Web (InWeb) e do projeto EU-IP mPlane (n-318627).

## Referências

Andrade, N., Mowbray, M., Lima, A., Wagner, G., and Ripeanu, M. (2012). Influences on cooperation in bittorrent communities. In *Proc. ACM SIGCOMM Workshop on Economics of peer-to-peer systems*.

Cheng, X. and Liu, J. (2009). Nettube: Exploring social networks for peer-to-peer short video sharing. In *Proc. IEEE INFOCOM*.

Cohen, B. (2003). Incentives build robustness in BitTorrent. In *Proc. Workshop on Economics of Peer-to-Peer Systems (P2PECON)*.

Dán, G. and Carlsson, N. (2009). Dynamic swarm management for improved BitTorrent performance. In *Proc. IPTPS*.

Guarnieri, T., Vieira, A. B., and da Silva, A. P. C. (2013). Impacto das relações sociais em sistemas de compartilhamento de arquivos. In *Proc. WP2P+ SBRC*.

Kaune, S., Tyson, G., Pussep, K., Mauthe, A., and Steinmetz, R. (2010). The seeder promotion problem: Measurements, analysis and solution space. In *Proc. IEEE ICCCN*.

Seedorf, J., Kiesel, S., and Stiemerling, M. (2009). Traffic localization for p2p-applications: The alto approach. In *Proc. IEEE Peer-to-Peer Computing*.

Varvello, M. and Steiner, M. (2011). Traffic localization for DHT-based BitTorrent networks. In *Proc. IFIP NETWORKING*.

Wang, H. and Liu, J. (2012). Exploring peer-to-peer locality in multiple torrent environment. *IEEE Transactions on Parallel and Distributed Systems*, 23(7):1216–1226.

Wang, H., Liu, J., Xu, K., , and Wu, D. (2013). Torrents on twitter: Explore long-term social relationships in peer-to-peer systems. *IEEE TNSM*, 10(1):1–10.

Wang, H., Wang, F., and Liu, J. (2011). On long-term social relationships in peer-to-peer systems. *Proc. IEEE 19th International Workshop on Quality of Service*.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 13*

## *Caracterização de Carga e Monitoração*

# Caracterização e Modelagem da Carga de Trabalho do Dropbox

**Glauber Gonçalves[2], Idilio Drago[3], Ana Paula Couto da Silva[2], Jussara Marques de Almeida[2], Alex Borges Vieira[1]**

[1] Departamento de Ciência da Computação - Universidade Federal de Juiz de Fora
[2]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais
[3]Politecnico di Torino - Italy
{ggoncalves, ana.coutosilva, jussara}@dcc.ufmg.br; idilio.drago@polito.it; alex.borges@ufjf.edu.br

***Resumo.*** *Este artigo apresenta a caracterização e modelagem de padrões de carga de trabalho de um sistema de armazenamento em nuvem muito popular atualmente, o Dropbox. A carga é analisada sob dois pontos de vista complementares. Por um lado, propriedades dos repositórios compartilhados por um conjunto de 333 usuários voluntários, tais como número, tipo e tamanho dos arquivos, nível de replicação e dinâmica de atualizações, são analisadas. Por outro lado, os padrões de comportamento de clientes são modelados a partir de dados de tráfego coletados em 3 redes universitárias. Nossos resultados podem ser aplicados na geração de cargas sintéticas realistas e servir como substrato para o desenvolvimento e avaliação de novas soluções de armazenamento em nuvem mais eficazes.*

***Abstract.*** *This paper presents a characterization and modeling of the workload of the currently most popular cloud storage system, Dropbox. The workload is analyzed from two complementary perspectives. On one side, characteristics of the Dropbox folders of a set of 333 volunteers, including number, type and sizes of files, replication level, and file update dynamics are analyzed. On the other side, the client behavior patterns are modeled based on the analysis of Dropbox traffic collected from 3 universities. Our results can be explored to design realistic synthetic workloads, which in turn can support the design and evaluation of more cost-effective cloud storage services.*

**Palavras-chave:** Dropbox, Caracterização e Modelagem, Carga de Trabalho.

## 1. Introdução

Computação na nuvem (*cloud computing*) [Zhang et al. 2010] tem atraído um grande interesse da indústria e da academia, servindo como plataforma para uma variedade de serviços. Em particular, armazenamento na nuvem (*cloud storage*) tem ganhado popularidade entre empresas e usuários domésticos por ser um mecanismo simples, prático e seguro de armazenar dados. Tal popularidade crescente se evidencia com a entrada recente de grandes empresas da Internet no mercado de armazenamento na nuvem (e.g., Google e Microsoft). Como consequência, o volume de tráfego gerado por essas aplicações tem crescido a uma taxa muito grande. Por exemplo, o Dropbox, uma das aplicações mais populares, trata mais de 1 bilhão de *uploads* de arquivos por dia[1].

[1]`https://www.dropbox.com/news/company-info`

O conhecimento da carga típica imposta a um serviço é fundamental para o projeto de soluções com melhor relação custo-benefício. No contexto de serviços de armazenamento na nuvem, diversos aspectos tornam a análise da carga de trabalho uma tarefa desafiadora. Como o conteúdo armazenado é privado e os protocolos de sincronização são proprietários, o conhecimento sobre o funcionamento dessas aplicações ainda é limitado. Além disso, o uso de criptografia torna difícil a captura e a análise de dados gerados por essas aplicações.

De fato, existem apenas alguns trabalhos recentes que analisam características de aplicações de armazenamento na nuvem [Mager et al. 2012, Gracia-Tinedo et al. 2013], com foco na arquitetura [Lenk et al. 2009] e na privacidade e segurança dos dados [Zhou et al. 2010]. A despeito de alguns estudos de desempenho [Drago et al. 2013a, Li et al. 2010, Wang and Ng 2010] e de padrões de tráfego [Drago et al. 2012], ainda existe uma lacuna na literatura quanto às cargas típicas dessas aplicações.

Neste artigo, nós apresentamos a caracterização e a modelagem da carga de trabalho de uma aplicação de armazenamento na nuvem de grande popularidade, o Dropbox (Seção 3). A carga é analisada sob dois pontos de vista complementares: os conteúdos armazenados nos repositórios pessoais e os padrões de comportamento de clientes (Seções 4.1 e 4.2, respectivamente). Especificamente, nós caracterizamos várias propriedades dos repositórios Dropbox, incluindo tamanho, número e tipo dos arquivos, nível de replicação e dinâmica de alterações a partir de dados coletados de 333 voluntários. Nós também propomos um modelo hierárquico de duas camadas que captura o comportamento de um cliente Dropbox. A *camada de sessão* (superior) captura as sucessivas sessões que um usuário pode ter durante um período de tempo. A *camada de transmissão de dados* (inferior) captura as interações que um cliente tem com os servidores do Dropbox durante uma sessão. Utilizando dados de tráfego coletados de 3 campi universitários, nós caracterizamos várias propriedades de cada camada, incluindo duração e tempo entre sessões, número de transferências de dados por sessão, duração das transferências, tempos entre transferências consecutivas e volume e duração de cada fluxo de dados em uma transferência. Os nossos resultados podem ser explorados no futuro para a geração de cargas sintéticas realistas, que por sua vez podem ser úteis no desenvolvimento e avaliação de soluções de armazenamento em nuvem mais eficazes (Seção 5).

## 2. Trabalhos Relacionados

Este artigo caracteriza e modela a carga de trabalho do Drobpox considerando os repositórios pessoais e os padrões de comportamento de clientes. Apesar de alguns esforços recentes de avaliação do desempenho de aplicações de armazenamento em nuvens, detalhados a seguir, a literatura não apresenta modelos de carga de trabalhos dessas aplicações. Em particular, este artigo é inovador quanto à modelagem dos repositórios pessoais e suas implicações para o funcionamento da aplicação.

Alguns estudos recentes analisam soluções específicas de armazenamento na nuvem [Mager et al. 2012] ou realizam comparações entres os provedores destes serviços [Li et al. 2010], focando em aspectos relacionados ao desempenho, segurança e privacidade. Por exemplo, a arquitetura de 5 sistemas populares e seus desempenhos são avaliados em [Drago et al. 2013a], enquanto que em [Hu et al. 2010] os autores estudam

o desempenho do armazenamento e da recuperação dos dados e questões relacionadas à privacidade de 5 aplicações. Gracia-Tinedo *et al.* [Gracia-Tinedo et al. 2013] apresentam um estudo baseado em medição ativa de 3 sistemas diferentes, provendo distribuições estatísticas que modelam aspectos de desempenho, como o tempo de transferência e taxa de falhas. Já em [Drago et al. 2012], os autores apresentam uma caracterização da aplicação Dropbox, focando no uso típico, padrões de tráfego e possíveis gargalos de desempenho. Nenhum desses trabalhos anteriores focam na caracterização e modelagem dos repositórios dos usuários e do comportamento dos clientes.

Outros trabalhos focam nos gargalos de desempenho de sistemas existentes e propõem novos mecanismos para reduzir os atrasos de sincronização, incluindo estratégias mais eficientes de alocação de tarefas [Wang et al. 2012] e de agrupamento de atualizações (*bundling*) [Li et al. 2013]. Nessa linha, nosso trabalho provê novos elementos que suportam a geração futura de cargas sintéticas realistas, o que também pode contribuir para o desenvolvimento e a avaliação de novos sistemas.

Este artigo estende dois trabalhos recentes [Drago et al. 2013b, Gonçalves et al. 2014], incluindo novos resultados e discussões sobre as implicações dos mesmos no funcionamento do Dropbox[2]. Particularmente, em [Drago et al. 2013b] nós realizamos uma análise preliminar de repositórios Dropbox. Este estudo é aqui revisitado, após uma limpeza dos dados para remoção de arquivos automaticamente criados pelo sistema e dados duplicados, o que levou a conclusões mais robustas. Além disto, nossa caracterização estende os estudos anteriores ao incluir aspectos relacionados à dinâmica de atualizações e as distribuições estatísticas que melhor modelam cada propriedade analisada. Estas distribuições são essenciais para a geração de cargas sintéticas.

## 3. Conceituação e Metodologia de Coleta de Dados

De acordo com o Google Trends[3], o volume de buscas por Dropbox ultrapassa o volume por serviços similares desde 2010. Isto sugere que o Dropbox é a aplicação de armazenamento na nuvem mais popular atualmente, o que motivou o foco deste estudo nesta aplicação. Assim, esta seção apresenta os conceitos básicos do Dropbox e descreve a metodologia adotada em nosso estudo, incluindo o modelo hierárquico para caracterização do comportamento dos clientes e a metodologia de coleta de dados.

### 3.1. Dropbox

Cada usuário do Dropbox tem um repositório local associado à sua conta, contendo arquivos que podem ser compartilhados com outros usuários. O Dropbox identifica múltiplas réplicas de um arquivo em um repositório (aplicando técnicas de deduplicação de arquivos), e armazena apenas uma cópia em seus servidores de armazenamento. Porém, a economia gerada por esse processo não é repassada ao usuário: arquivos replicados ocupam espaço no repositório local proporcional ao número de réplicas.

A arquitetura do Drobpox apresenta dois componentes principais: (i) *servidores de controle*, que são gerenciados pelo Dropbox em sua infraestrutura privada; e (ii) *servidores de armazenamento*, que são hospedados pela Amazon. Logo, os arquivos

[2]Resultados da modelagem de comportamento de clientes foram submetidos para o *IEEE International Conference on Communications* [Gonçalves et al. 2014].

[3]http://www.google.com/trends/

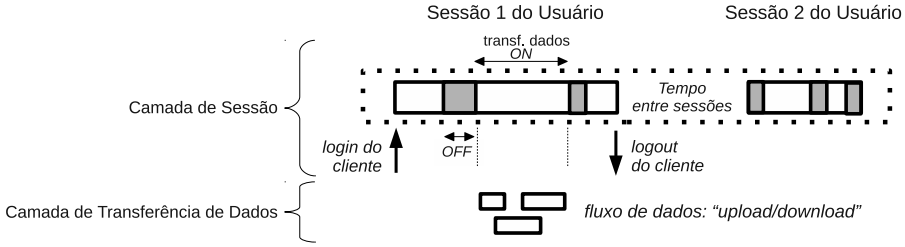



**Figura 1. Modelo de comportamento de um cliente Dropbox.**

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 as diferentes partes do serviço oferecido e suas funcionalidades específicas.

Os arquivos transferidos entre clientes e servidores são comprimidos no lado do cliente para reduzir o consumo de banda de rede e o tempo de transferência [Hu et al. 2010]. Somente a diferença entre versões consecutivas de um mesmo arquivo é transferida entre cliente e servidor e arquivos duplicados são transferidos somente uma vez. Todas as transferências são criptografadas com TLS/SSL. Mais detalhes sobre o protocolo Dropbox estão disponíveis em [Drago et al. 2012, Drago et al. 2013a].

### 3.2. Modelo Hierárquico do Comportamento do Cliente Dropbox

Para caracterizar o comportamento de um cliente Dropbox, nós propomos um modelo hierárquico de duas camadas composto por sessões (camada superior) e transferências de dados durante uma mesma sessão (camada inferior), como mostrado na Figura 1.

Uma sessão é iniciada com uma ação de *login* de um dispositivo particular, identificado pelo endereço IP [4], e é finalizada com uma ação de *logout*. Nós denominamos o tempo entre sessões consecutivas do cliente como *tempo entre sessões*. Durante uma sessão, o cliente mantém uma conexão TCP continuamente aberta com um servidor de notificação (e.g., `notify1.dropbox.com`), utilizada para receber informações sobre mudanças realizadas nos arquivos por outros dispositivos/compartilhamentos. Os clientes iniciam transferências de dados utilizando outra conexão TCP. Logo, os endereços IP dos servidores de notificação podem ser utilizados para identificar sessões de clientes na rede.

Durante uma sessão, o cliente alterna entre períodos de atividade (transferência) e de inatividade. Uma transferência inicia com um contato do cliente a um subdomínio específico do Dropbox para iniciar o processo de sincronização. Uma transferência é subdividida em múltiplos fluxos de dados que são iniciados juntamente com o processo de sincronização. Um fluxo pode ser mantido aberto, ainda que inativo, por um período curto, à espera de mais dados a serem transferidos. Logo, nós utilizamos o *timeout* aplicado pelo Dropbox (Seção 4.2.1) como um limiar mínimo para distinguir transferências consecutivas de um mesmo cliente: fluxos consecutivos de um mesmo endereço IP, separados por um intervalo de até 60 segundos(*timeout*), são agrupados como uma única transferência de dados. O tempo total de sincronização durante uma transferência de dados é denominado *período ON*, enquanto o intervalo de tempo entre transferências consecutivas é denominado *período OFF*.

[4]Note que, na presença de NAT, não é possível distinguir sessões distintas de dispositivos que compartilham o mesmo endereço IP.

Em suma, nosso modelo de comportamento do cliente é composto pelos seguintes componentes. A camada superior inclui a duração da sessão, tempo entre sessões, número de transferência de dados por sessão, assim como os tempos de *ON* e *OFF*. Já a camada inferior, relativa às transferências de dados, é analisada considerando o número de fluxos por transferência de dados, duração e volume de cada fluxo.

### 3.3. Coleta de Dados

Nossa caracterização é feita em conjuntos de dados complementares que descrevem, por um lado, os repositórios de arquivos de usuários voluntários, e por outro, o comportamento dos clientes ao interagir com o Dropbox.

#### Repositórios de Arquivos

A coleta de dados sobre os arquivos compartilhados no Dropbox foi feita com a participação de voluntários, recrutados a partir de uma chamada à comunidade científica por meio de e-mails para diferentes listas e contatos em redes sociais.

Cada voluntário respondeu algumas perguntas sobre seu perfil e executou um programa desenvolvido por nós. Este programa analisa as características do repositório Dropbox do usuário, armazenando para cada arquivo encontrado: o tamanho, a extensão e o tipo *MIME* do arquivo, a data da última alteração e um identificador composto pela chave *Hash* dos 8 kB iniciais e dos 8 kB finais do arquivo. Arquivos com mesma chave composta, mesmo tamanho e mesmo tipo *MIME* são considerados *réplicas*. Essa heurística foi adotada para reduzir o tempo de coleta. Uma análise preliminar mostrou que a maioria das réplicas tem o mesmo nome, sugerindo que nossa heurística provê uma boa aproximação. Cada voluntário recebeu um identificador único, permitindo que um mesmo voluntário contribua mais de uma vez com a coleta. No entanto, apenas os dados mais recentes de cada usuário foram considerados.

Os dados coletados foram preliminarmente analisados em [Drago et al. 2013b]. Contudo, diferentemente daquele trabalho, nós fizemos uma limpeza dos dados para remover arquivos de sistema (e.g., arquivos `.dropbox` e `.ini`), que não refletem a carga pois não são transferidos durante sincronizações. O conjunto de dados aqui analisado cobre cerca de 3 milhões de arquivos (1,38 TB) de 333 usuários únicos. Estes usuários são principalmente do Brasil (45%), Europa (40%) e EUA (7%), sendo a maioria alunos e pesquisadores de instituições acadêmicas. Cerca de 88% deles são homens entre 20 e 30 anos, e apenas 4,5% declararam pagar pelo uso de *cloud storage*. A capacidade de armazenamento média declarada é de 23,4 GB.

Os repositórios dos voluntários são caracterizados quanto ao espaço total ocupado, número, tipos e tamanhos dos arquivos compartilhados, porcentagens de arquivos e bytes replicados e *idade* dos arquivos (i.e., tempo desde a última alteração).

#### Comportamento dos Clientes

O comportamento de clientes é analisado a partir de dados de tráfego coletados passivamente, conforme a metodologia proposta em [Drago et al. 2012]. Diferente de repositórios de arquivos, nessa coleta analisamos o tráfego de dados de 3 locais, especificamente campi universitários, descritos a seguir. A ferramenta tstat [Finamore et al. 2011] foi usada para coletar informações sobre todas as conexões TCP nas redes de interesse, incluindo os endereços IPs de clientes e servidores e volume de dados

trocados. 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, armazenamento de dados, etc), nós usamos o texto `*.dropbox.com` encontrado nos certificados TLS/SSL e no domínio de nome que os clientes requisitam aos servidores DNS. Uma lista completa dos domínios usados pelo Dropbox e detalhes da metodologia de isolamento e classificação de tráfego Dropbox estão disponíveis em [Drago et al. 2012].

O tstat foi executado nos roteadores de borda de 3 campi universitários, denominados campus 1, 2 e 3. O campus 1, de uma universidade europeia, tem uma população de 13 mil pessoas, entre estudantes, funcionários e docentes. Os campi 2 e 3 pertencem a universidades brasileiras, com 57 mil e 20 mil pessoas, respectivamente. Os 3 conjuntos de dados incluem tráfego gerado por laboratórios e escritórios de administração. Os dados do campus 1 também incluem o tráfego da moradia estudantil.

Como mencionado, várias heurísticas foram usadas para filtrar dados não relacionados ao Dropbox. Considerando o modelo de comportamento proposto, nós focamos em fluxos TCP relacionados a transferências de *dados* e sessões de clientes. O tráfego direcionado à interface Web do Dropbox foi desconsiderado, já que, como descrito em [Drago et al. 2012] e confirmado em nossos dados, a vasta maioria do tráfego Dropbox é produzida pela aplicação cliente. Fluxos de até 2 segundos ou com até 5 kB de dados, que em sua maioria refletem problemas de comunicação, foram também descartados.

Múltiplos fluxos de dados foram agrupados em uma sessão, conforme modelo proposto, avaliando o endereço IP do cliente e os tempos de início e fim associados à sessão e aos fluxos. Um fluxo $f_i$ foi atribuído à sessão $s_i$ se: os endereços IPs dos clientes de $f_i$ e $s_i$ eram iguais, $start(f_i) \geq start(s_i)$ e $end(f_i) \leq end(s_i)$. Porém, foram detectados casos em que o início de uma sessão ocorria antes do término explícito da sessão anterior do mesmo endereço IP, ou seja, casos de sobreposição de sessões de um mesmo endereço IP. Nestes casos, não foi possível atribuir fluxos de dados que ocorrem durante a sobreposição a uma única sessão. Testes com o Dropbox em ambiente controlado [Drago et al. 2013a] mostraram que tais sobreposições podem ocorrer devido a: (i) uso de NAT, o que faz com que sessões de múltiplos clientes apareçam com o mesmo endereço; ou (ii) falhas de comunicação entre clientes e servidores, que forçam o cliente Dropbox a abrir uma sessão antes do término da anterior.

Para o caso (ii), esperamos sobreposições de mais curta duração. Logo, foi aplicada a seguinte heurística para lidar com sobreposições entre sessões de um mesmo endereço. Como é sabido que o campus 1 não adota oficialmente NAT em suas sub-redes, é esperado que a maioria das sobreposições neste conjunto de dados seja devido a falhas de comunicação[5]. Foram observadas sobreposições em 42% das sessões do campus 1. A distribuição das durações dessas sobreposições apresenta um ponto de inflexão (joelho) claro por volta de 140 segundos. Logo, esse limiar foi adotado para identificar sobreposições causadas por problemas de comunicação. Isto é, nos 3 conjuntos de dados, sessões com sobreposição de até 140 segundos foram agrupadas, formando uma única sessão. Essa operação foi realizada em 36%, 38% e 50% das sessões coletadas nos campi 1, 2 e 3, respectivamente. Sessões com sobreposições maiores que esse limiar -

[5]Observamos alguns poucos casos de sessões com sobreposições longas no campus 1, possivelmente devido ao uso de NAT nas moradias estudantis.

**Tabela 1. Descrição dos Conjuntos de Dados de Tráfego.**

| Campus | Período | Volume total (TB) | Tráfego Dropbox | | | |
|---|---|---|---|---|---|---|
| | | | # IPs | # Sessões | # Fluxos de dados | Volume (TB) |
| 1 | 6/3-9/5/13 | 526,3 | 17.457 | 718.631 | 1.752.516 | 10,80 |
| 2 | 19/2-14/3/13 | 38,9 | 4.637 | 98.789 | 132.672 | 1,08 |
| 3 | 6/3-6/5/13 | 30,8 | 155 | 10.823 | 74.558 | 0,56 |

4%, 15% e 45% das sessões dos campi 1, 2 e 3, respectivamente - foram descartadas. O maior descarte nos campi 2 e 3 é devido ao uso mais frequente de NAT em suas redes. Embora esse descarte possa ter afetado quantitativamente algumas métricas de nosso modelo hierárquico, como discutimos na Seção 4.2, acreditamos que não tenha afetado qualitativamente, pois obtivemos as mesmas distribuições para os 3 campi com leve variação de parâmetros conforme particularidades de cada campus (população e estabilidade da rede), a despeito das taxas de descarte diferentes.

A Tabela 1 sumariza os 3 conjuntos de dados, apresentando o período de coleta, o tráfego total coletado, e, para o tráfego Dropbox analisado (após filtragem), os números de endereços de clientes únicos, sessões, fluxos de dados e o volume de dados transferidos.

Ressaltamos que existe um viés nos dados analisados (tanto de tráfego quanto de repositório) para ambientes acadêmicos e poucas regiões geográficas. Porém, considerando a falta de estudo semelhante na literatura, esperamos que os resultados obtidos, discutidos a seguir, forneçam uma referência para estudos futuros.

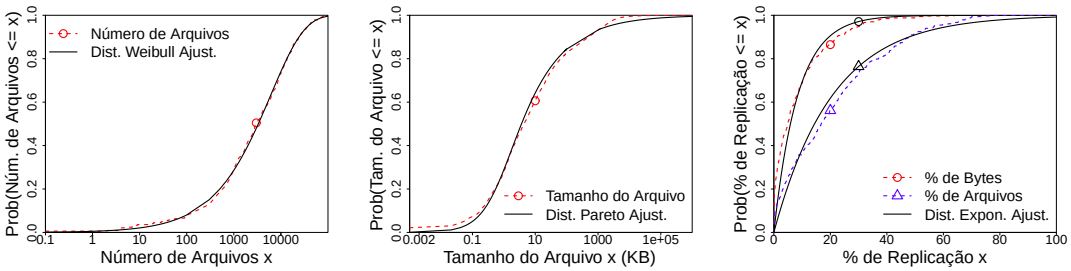
## 4. Resultados da Caracterização

Esta seção discute os resultados da caracterização dos repositórios de arquivos (Seção 4.1) e do comportamento dos clientes do Dropbox (Seção 4.2). Estes resultados provêem visões complementares da carga do sistema. Para cada componente analisado, nós apresentamos a distribuição estatística que melhor se adequa aos dados, escolhida entre os seguintes modelos usados na literatura: Normal, Log-Normal, Exponencial, Gamma, Logística, Beta, Uniforme, Weibull e Pareto para variáveis contínuas; Poisson, Binomial, Binomial Negativa, Geométrica e Hipergeométrica para variáveis discretas. Para cada modelo, os parâmetros da distribuição que mais se aproxima dos dados são determinados usando o método de estimativa por máxima verossimilhança. Após definição dos parâmetros de cada modelo, a distribuição com menor distância de Kolmogorov-Smirnov (distribuições contínuas) ou menor erro quadrático mínimo (LSE) (distribuições discretas) em relação aos dados é escolhida [Venables and Ripley 2002]. Esta escolha também é validada com uma avaliação visual do ajuste das curvas.

### 4.1. Repositórios de Arquivos

A maioria dos voluntários do nosso estudo tem um grande volume de dados em seus repositórios, talvez devido ao perfil acadêmico frequente entre eles. Mais de 70% dos voluntários armazenam pelo menos 1 GB de dados e praticamente 9% armazenam pelo menos 10 GB. O tamanho médio dos repositórios é 4,23 GB, bem acima do limite inicial para acesso gratuito (2 GB). Observa-se que uma distribuição Weibull[6] com parâmetros

[6]Função de densidade de probabilidade (PDF) da distribuição Weibull: $p_X(x) = \frac{\alpha}{\beta}(\frac{x}{\beta})^{\alpha-1}e^{-(x/\beta)^{\alpha}}$.




(a) Número (log no eixo x) (b) Tamanho (log no eixo x) (c) Nível de Replicação

**Figura 2. Características dos arquivos em repositórios Dropbox. (a) Número: distribuição Weibull[6] com $\alpha$=0,60 e $\beta$=6.080,77; (b) Tamanho: distribuição Pareto[7] com $\alpha$=0,37 e $\kappa$=0,66; (c) Porcentagem de replicação: distribuições Exponenciais[9] com $\lambda$=4,82 (arquivos) e $\lambda$=11,81 (bytes).**

$\alpha$=0, 69 e $\beta$=3, 32 provê o melhor modelo para os tamanhos dos repositórios. As curvas são omitidas por limitações de espaço e por serem produto das distribuições dos números e tamanhos de arquivos por repositório, discutidas a seguir.

A Figura 2(a) mostra a distribuição do número de arquivos nos repositórios (escala logarítmica no eixo x). Os voluntários tendem a armazenar muitos arquivos, possivelmente devido ao perfil acadêmico da maioria: 71% deles têm pelo menos 1.000 arquivos e cerca de 13% têm mais de 20 mil arquivos. Como nenhuma das distribuições discretas analisadas apresentou um bom modelo, foram avaliadas aproximações com distribuições contínuas. Como mostrado na figura, uma distribuição Weibull[6] (parâmetros $\alpha$ e $\beta$ na legenda da figura) apresentou uma aproximação justa.

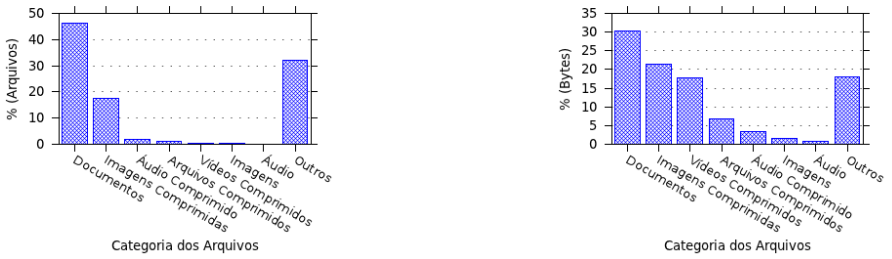
A Figura 2(b) apresenta a distribuição dos tamanhos dos arquivos nos repositórios. Mais de 93% dos arquivos são menores que 1 MB, e o tamanho médio é 493 KB. Logo, a vasta maioria dos arquivos é pequena, o que sugere uma sobrecarga na rede não desprezível imposta pelo uso de criptografia antes da transferência de cada arquivo. Uma distribuição Pareto[7], de cauda pesada, apresentou o melhor ajuste dos dados, implicando que, a despeito da tendência observada, alguns arquivos muito grandes (e.g., até 5.5 GB) são armazenados com frequência não desprezível.

A Figura 2(c) apresenta as distribuições das porcentagens de arquivos e bytes replicados nos repositórios[8]. Note que o volume de réplicas por repositório é significativo: na média, 20,7% dos arquivos e 8,4% dos bytes são réplicas. Pelo menos 40% dos arquivos de 18% dos voluntários são réplicas, e 20% dos voluntários têm mais de 17% do espaço total ocupado por réplicas. Mais ainda, os usuários replicam entre si 42% dos arquivos (14% dos bytes). A figura mostra que as duas distribuições são razoavelmente bem aproximadas por distribuições exponenciais[9], principalmente para valores mais altos que têm maior impacto na carga. Estes resultados são relevantes para geração de cargas pois, embora todas as réplicas sejam contabilizadas para o consumo de espaço do usuário, somente uma cópia de cada réplica é mantida nos servidores e transferida

[7]PDF da distribuição Pareto: $p_X(x) = \frac{\alpha\kappa^\alpha}{(x+\kappa)^{\alpha+1}}$. Note que esta é uma distribuição Pareto tipo II.

[8]São considerados apenas conteúdos repetidos, ou seja, são contabilizados todos os bytes/cópias de um mesmo arquivo exceto a primeira ocorrência.

[9]PDF da distribuição Exponencial: $p_X(x) = \lambda e^{-\lambda x}$.




(a) Porcentagem de arquivos (b) Espaço ocupado pelos arquivos

**Figura 3. Distribuição dos tipos de arquivos em repositórios Drobpox.**

durante atualizações. Nossos resultados mostram que o Dropbox tem muita oportunidade para economizar recursos ao identificar réplicas e tratá-las de forma especial.

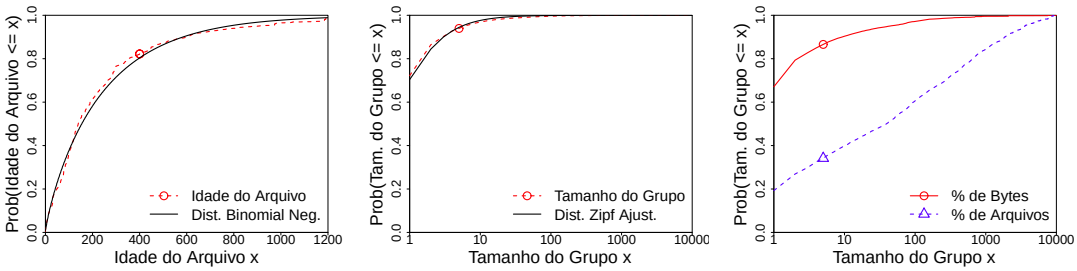
A Figura 3 mostra as distribuições de arquivos e bytes nos repositórios por tipo de arquivo. Mais de 45% dos arquivos são documentos (e.g., arquivos Word, Excel e PDF), e eles são responsáveis por mais de 30% do espaço total do repositório. Há também uma grande quantidade de conteúdo comprimido (imagens, áudio, etc), sugerindo que a compressão realizada antes de uma atualização pode ser um desperdício de recurso computacional, pelo menos para estes arquivos.

Por fim, analisamos a dinâmica de alteração dos arquivos, caracterizando a distribuição das idades dos arquivos. A idade de um arquivo é definida como o intervalo de tempo entre a sua última alteração e o momento da coleta. A Figura 4(a) mostra que a maioria dos arquivos dos voluntários não é constantemente alterada: quase 80% dos arquivos foram modificados em menos de 1 ano antes da coleta, e somente 14% dos arquivos foram modificados no último mês antes da coleta. Uma distribuição Binomial Negativa[10] se mostrou uma boa aproximação dos dados.

O Dropbox apresenta uma estratégia de agrupamento de arquivos - *bundling* – em que as atualizações são postergadas e agrupadas para reduzir latência e sobrecarga de controle [Drago et al. 2013a]. A idade dos arquivos nos permite especular até que ponto *bundling* pode ser feito. Para tal, os arquivos de cada repositório foram agrupados pelo instante da última alteração (com granularidade de 10 segundos), e a distribuição dos tamanhos dos grupos, em número de arquivos, foi caracterizada (Figura 4(b)). Foram também analisadas as porcentagens de bytes e de arquivos em grupos de determinado tamanho (Figura 4(c)). Note que 70% dos grupos têm apenas 1 arquivo e totalizam quase 70% dos dados. Porém, alguns grupos chegam a conter até 10.000 arquivos. Mais ainda, a maioria dos arquivos (80%) estão nos grupos com mais de um arquivo, sendo que 40% deles estão em grupos de pelo menos 100 arquivos. Logo, múltiplos arquivos pequenos são frequentemente adicionados/modificados ao mesmo tempo, sugerindo que o sistema deve estar preparado para lidar com transferências contendo muitos arquivos pequenos. A Figura 4(b) mostra que uma distribuição Zipf[11] aproxima bem os tamanhos dos grupos. As distribuições que aproximam os números de bytes e de arquivos por grupo são omitidas pois estas variáveis são função das distribuições das idades dos arquivos e dos tamanhos dos grupos (fornecidas), podendo então ser geradas sinteticamente a partir delas.

[10]Distribuição de probabilidade de massa (PMF) da Binomial Negativa: $p_X(x) = \frac{\Gamma(x+r)}{\Gamma(r)x!}p^r(1-p)^x$.

[11]PMF da distribuição Zipf: $p_X = \frac{1}{Z}x^{-\alpha}$, onde $Z$ é uma constante de normalização.




(a) Idade dos arquivos (b) Tamanhos dos grupos (c) % arquivos e bytes por grupo

**Figura 4. Dinâmica de alteração dos arquivos e potencial para *bundling*. (a) Idade dos arquivos: distribuição Binomial Negativa[10] com parâmetros $r$=0,828 e $p$=0.003; (b) Tamanho dos grupos (*bundles*): distribuição Zipf[11] com $\alpha$=2.32; (c) Porcentagens de arquivos e bytes por grupo. Log nos eixos x de (b) e (c).**

## 4.2. Comportamento de Clientes

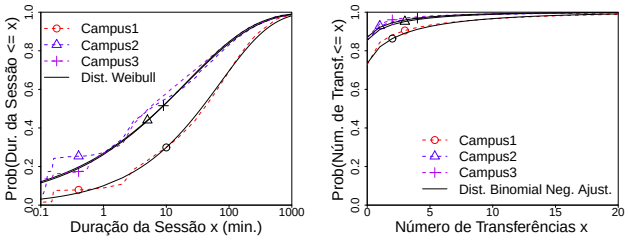
Esta seção apresenta os resultados da caracterização do comportamento dos clientes, conforme modelo apresentado na Seção 3.2.

### Camada de Sessão

A Figura 5(a) mostra as distribuições da duração das sessões nos 3 campi monitorados (escala logarítmica no eixo x). No geral, as seções nos campi 2 e 3 tendem a ser mais curtas que as do campus 1: a fração de sessões com até 200 minutos é de 83% no campus 1 chegando a 92% nos outros campi. As durações médias são 143,95, 84,65 e 93,75 minutos para os campi 1, 2 e 3 respectivamente. As durações mais curtas nos campi 2 e 3 podem ser devido ao uso frequente de NAT e uma maior instabilidade da rede durante o período de monitoração (i.e., quedas dos links internacionais), principalmente no campus 3, que geram interrupções mais frequentes das sessões. Já a rede do campus 1 tem melhor estabilidade e menor uso de NAT, o que pode explicar as sessões mais longas. Ainda assim, as durações de sessões nos 3 campi são bem modeladas por distribuições Weibull[6] (parâmetros na figura), um modelo previamente usado para aproximar as durações de sessões em outros sistemas (e.g., transmissão de vídeo ao vivo [Borges et al. 2012]).

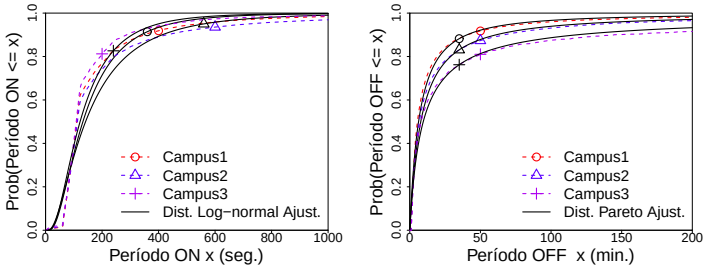
Durante uma sessão, o cliente alterna entre períodos de transferência de dados (*On*) e períodos de inatividade (*Off*). A Figura 5(b) mostra as distribuições do número de transferências (períodos *On*) por sessão para os 3 campi. A maioria das sessões não tem nenhuma transferência, principalmente nos campi 2 e 3 (85% das sessões). Nesses casos, os clientes conectam ao servidor Dropbox, sincronizam suas informações de conta, mas não transferem nenhum arquivo. Note que os usuários do campus 1 tendem a realizar mais transferências por sessão, possivelmente devido às sessões mais longas. Porém, o número médio de transferências de dados por sessão é, em geral, muito baixo: 1,3, 0,56 e 0,57 para os campi 1, 2 e 3, respectivamente. Apesar das diferenças quantitativas, distribuições binomiais negativas[10] aproximam bem os dados dos 3 campi. Note que a grande presença de sessões curtas e frequentemente sem nenhuma transferência de dados sugere que o uso de *caches* locais nos clientes durante sessões pode ter benefícios limitados.

A Figura 6(a) mostra que as distribuições das durações dos períodos de transferência (*ON*) nos 3 campi são muitos semelhantes, com uma maioria de




(a) Duração (log no eixo x) (b) Número de transferências

**Figura 5. Características das sessões. (a) Duração: distribuições Weibull[6] com $\alpha$=0,52,$\beta$=71,8 (campus 1), $\alpha$=0,38,$\beta$=20,8 (campus 2) e $\alpha$=0,39,$\beta$=20,4 (campus 3); (b) Número de transferências: distribuições binomiais negativas[10] com $r$=0,12,$p$=0,09 (campus 1), $r$=0,07,$p$=0,11 (campus 2) e $r$=0,06,$p$=0,12 (campus 3).**




(a) Transferência (ON) (b) Inatividade (OFF)

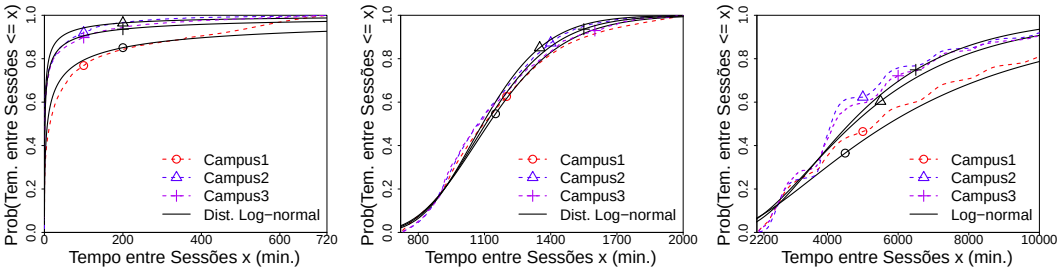
**Figura 6. Períodos de transferência e inatividade (ON/OFF). (a) ON: distribuições log-normais[12] com $\mu$=4,89,$\sigma$=0,71 (campus 1), $\mu$=4,95,$\sigma$=0,83 (campus 2) e $\mu$=4,80,$\sigma$=0,72 (campus 3); (b) OFF: distribuições Pareto[7] com $\alpha$=1,38,$\kappa$= 9.62 (campus 1), $\alpha$=1,13,$\kappa$=9,17 (campus 2) e $\alpha$=0,79,$\kappa$= 6,78 (campus 3).**

transferências muito curtas: aproximadamente 75% das transferências duram até 200 segundos. Nota-se também um joelho nas distribuições dos dados por volta de 60 segundos, que conjecturamos ser o valor padrão de *timeout* aplicado pelo Dropbox (corroborando resultados em [Drago et al. 2012]). Em média, as transferências duram 192, 247 e 179 segundos nos campi 1, 2 e 3. Todas as 3 distribuições são bem modeladas por distribuições log-normais[12], modelo que já foi utilizado para aproximar durações de transferências em outros contextos (e.g., vídeo ao vivo na Web [Veloso et al. 2002]).

A Figura 6(b) apresenta as distribuições para os períodos de inatividade (*OFF*) de uma sessão. Na média, um usuário permanece inativo por 29, 39 e 81 minutos nos campi 1, 2 e 3, respectivamente. A maior presença de clientes atrás de NAT e com IPs dinâmicos no campus 3 pode explicar os períodos *OFF* mais longos, já que isto dificulta a identificação de sessões consecutivas de um mesmo usuário. Apesar das diferenças, distribuições Pareto[7] aproximam bem os períodos *OFF* nos três campi. Note que, como esperado, os períodos *OFF* são bem mais longos que os períodos *ON*, já que os usuários gastam mais tempo em seus trabalhos locais que transferindo dados de/para os servidores. Alguns usuários podem também desabilitar a sincronização de seus clientes para evitar transferências. O uso de *bundling* também favorece períodos *OFF* mais longos.

Finalmente, discutimos o último componente da camada de sessão, os tempos

[12]PDF da distribuição Log-Normal: $p_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{-(ln(x)-\mu)^2}{2\sigma^2}}$.




(a) Tempos $\leq$ 720 min. (b) Tempos entre 720 e 2.000 (c) Tempos $>$ 2.000 min.

**Figura 7. Tempos entre sessões aproximados por distribuições log-normais[12]: (a) $\mu$=2.04,$\sigma$=3.18 (campus 1), $\mu$=-0.02,$\sigma$=2.94 (campus 2) e $\mu$=0.24,$\sigma$=3.33 (campus 3); (b) $\mu$=7.02,$\sigma$=0.24 (campus 1), $\mu$=6.99,$\sigma$=0.20 (campus 2) e $\mu$=7.00,$\sigma$=0.22 (campus 3); (c) $\mu$=8.65,$\sigma$=0.70 (campus 1), $\mu$=8.46,$\sigma$=0.56 (campus 2) e $\mu$=8.44,$\sigma$=0.50 (campus 3).**

entre sessões. Nenhum modelo de distribuição analisado apresentou um bom ajuste para todos os dados medidos em cada campus. Logo, optamos por modelar separadamente os tempos nas três faixas de valores: até 720 minutos, entre 720 e 2.000 minutos e acima de 2.000 minutos. Estas faixas foram escolhidas a partir de inspeção visual dos dados.

A Figura 7 apresenta as distribuições dos tempos entre sessões para cada faixa. No geral, os tempos entre sessões tendem a ser curtos. Esse comportamento é mais frequente nos campi 2 e 3, onde o uso de NAT e instabilidade da rede causam desconexões mais frequentes. De fato, a maioria dos tempos medidos (69%, 81% e 79% para os campi 1, 2 and 3) são inferiores a 720 minutos (Figura 7(a)). Mais ainda, 52% (campi 2 e 3) e 27% (campus 1) dos tempos entre sessões nesta faixa são menores que 5 minutos. Quanto às outras duas faixas de valores, cerca de 12% dos tempos entre sessões nos campi 2 e 3 e 14% dos tempos no campus 1 estão na faixa intermediária (entre 720 e 2.000 minutos). Apesar da diferenças, para os três conjuntos de dados, distribuições log-normais[12] apresentaram os melhores ajustes para os dados medidos em cada faixa[13].

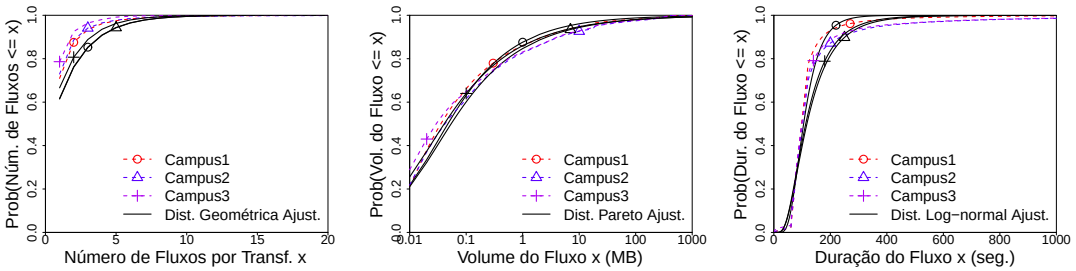
### Camada de Transmissão de Dados

A camada de transmissão de dados está relacionada aos múltiplos fluxos que uma transferência de dados (período *ON*) pode conter. As distribuições do número de fluxos por transferência são mostradas na Figura 8(a). As distribuições são semelhantes nos 3 campi: a maioria (pelo menos 71%) das transferências têm um único fluxo. Dentre todos modelos testados, a distribuição geométrica[14] foi a que apresentou o melhor ajuste para os 3 conjuntos de dados. Entretanto, como mostrado na figura, essa distribuição superestima um pouco o número de fluxos por transferência. Porém, ressaltamos que, para estudos de desempenho (e.g., planejamento de capacidade), é preferível superestimar o número de fluxos a subestimá-lo, já que superestimativas podem levar a decisões mais conservadoras.

A Figura 8(b) mostra que as distribuições de volume dos fluxos são muito semelhantes nos 3 campi: cerca de 82% dos fluxos carregam até 1 MB. Esta concentração em volumes pequenos está consistente com os resultados sobre tamanhos de arquivos nos

[13]Os ajustes são um pouco piores para tempos acima de 2.000 minutos. Entretanto, ressaltamos que somente uma minoria dos dados medidos estão neste intervalo.

[14]PMF da distribuição Geométrica: $p_X(x) = p(1-p)^x$.




(a) # fluxos por transferência (b) Volume (log no eixo x) (c) Duração

**Figura 8. Características dos fluxos. (a) Número: distribuições geométricas[14] com $p$=0,38 (campus 1), $p$=0,37 (campus 2) e $p$=0,42 (campus 3); (b) Volume: distribuições Pareto[7] com $\alpha$=0,50,$\kappa$=0,02 (campus 1), $\alpha$=0,44,$\kappa$= 0,01 (campus 2) e $\alpha$=0,43,$\kappa$=0,01 (campus 3); (c) Duração: distribuições log-normais[12] com $\mu$=0,50,$\sigma$=4,63 (campus 1), $\mu$=0,67,$\sigma$=4,78 (campus 2), $\mu$=0,75,$\sigma$=4,65 (campus 3).**

repositórios dos usuários voluntários (Figura 2(b)), apesar dos dados terem sido obtidos de fontes diferentes. Porém, pelo menos 5% dos fluxos têm mais de 10 MB, e o volume médio é em torno de 7MB. De fato, as distribuições dos dados apresentam cauda pesada e são bem ajustadas por distribuições Pareto[7]. Por fim, as distribuições das durações dos fluxos, mostradas na Figura 8(c), são semelhantes às das durações de períodos *ON* (Figura 6(a)), já que a maioria das transferências contêm apenas um fluxo.

## 5. Conclusões e Trabalhos Futuros

Este artigo apresentou a caracterização e modelagem de componentes da carga de trabalho do Dropbox relacionados aos repositórios de arquivos e ao comportamento dos clientes. O estudo foi feito a partir de dados coletados de repositórios de voluntários e dados de tráfego de 3 campi universitários. Para analisar o comportamento dos clientes foi proposto um modelo hierárquico composto das camadas de sessão e de transmissão de dados.

Observamos que a maioria dos voluntários tem um grande volume de dados em seus repositórios, embora muitos arquivos sejam pequenos. O nível de replicação por repositório é significativo, o que pode levar a uma economia significativa de recursos nos servidores do Dropbox. Além disto, a maioria dos arquivos não é frequentemente alterada embora, com frequência significativa, múltiplos arquivos pequenos sejam atualizados ao mesmo tempo, sugerindo que *bundling* de arquivos pode melhorar o desempenho.

Quanto ao comportamento dos clientes, apesar de diferenças quantitativas devido a variações no uso de NAT e na estabilidade da rede, os 3 campi analisados apresentam padrões semelhantes. Isto sugere que as distribuições encontradas descrevem bem o comportamento de clientes Dropbox, pelo menos no ambiente acadêmico. A maioria das sessões são curtas, e muitas delas não incluem nenhuma transferência de dados, o que sugere um benefício limitado do uso de *caches* nos clientes. As durações de sessão e dos períodos de transferência seguem distribuições previamente adotadas em medidas semelhantes em outros contextos [Borges et al. 2012, Veloso et al. 2002] mas com parâmetros bem diferentes, indicando padrões de carga significativamente distintos.

Trabalhos futuros incluem extensões para novos conjuntos de dados, coletados fora do ambiente acadêmico, e a construção de um gerador de cargas sintéticas.

**Agradecimentos**
Os autores agradecem o apoio do CNPq, CAPES, FAPEMIG, Instituto Nacional de Ciência e Tecnologia para Web (InWeb) e do projeto EU-IP mPlane (n-318627).

## Referências

Borges, A., Gomes, P., Nacif, J., Mantini, R., Almeida, J. M., and Campos, S. (2012). Characterizing SopCast Client Behavior. *Computer Communications*, 35(8):1004–1016.

Drago, I., Bocchi, E., Mellia, M., Slatman, H., and Pras, A. (2013a). Benchmarking personal cloud storage. In *Proc. IMC*.

Drago, I., Borges, A., and da Silva, A. P. C. (2013b). Caracterização dos arquivos armazenados no dropbox. In *Proc. WP2P+ - SBRC*.

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*.

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., Vieira, A. B., and Almeida, J. M. (2014). Modeling the dropbox client behavior. In *Submetido ao ICC*.

Gracia-Tinedo, R., Artigas, M., Moreno-Martínez, A., Cotes, C., and López, P. (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. *ACM SIGOPS Operating Systems Review*, 44(3):110–115.

Lenk, A., Klems, M., Nimis, J., Tai, S., and Sandholm, T. (2009). What's inside the Cloud? An architectural map of the Cloud landscape. In *Proc. ICSE*.

Li, A., Yang, X., Kandula, S., and Zhang, M. (2010). Cloudcmp: comparing public cloud providers. In *Proc. SIGCOMM*.

Li, Z., Wilson, C., Jiang, Z., Liu, Y., Zhao, B., Jin, C., Zhang, Z.-L., and Dai, Y. (2013). Efficient Batched Synchronization in Dropbox-like Cloud Storage Services. In *Proc. Middleware Conference*.

Mager, T., Biersack, E., and Michiardi, P. (2012). A Measurement Study of the Wuala On-line Storage Service. In *Proc. of the IEEE P2P*.

Veloso, E., Almeida, V., Meira, W., Bestavros, A., and Jin, S. (2002). A Hierarchical characterization of a live streaming media workload. In *Proc. SIGCOMM WIM*.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistic with S*. Springer.

Wang, G. and Ng, T. S. E. (2010). The impact of virtualization on network performance of amazon ec2 data center. In *Proc. INFOCOM*.

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. WQS*.

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.

Zhou, M., Zhang, R., Xie, W., Qian, W., and Zhou, A. (2010). Security and privacy in cloud computing: A survey. In *Proc. SKG*.

# Avaliação do Impacto de Falhas na Rede Nacional de Ensino e Pesquisa no Tráfego de um Campus Universitário

**Rodrigo Duarte**[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

{rodrigo.duarte, alex.borges}@ufjf.edu.br {cunha, jussara}@dcc.ufmg.br

***Abstract.*** *In this paper we characterize the impact of failures in Brazil's national education and research network on traffic at a client university. In particular, we study the impact of failures on traffic, user, and application behavior. The failures are interesting in that they persist for several hours and impact only international links, so destinations hosted in Brazil remain reachable. Our results show that although failures in international links have negligible impact on the performance of national traffic, users do adapt their behavior to the unavailability of services hosted abroad. For example, entertainment traffic migrates from Facebook to YouTube, which remains reachable during the analyzed failures; and the fraction of interactive traffic gradually decreases during failures, indicating that users may leave the campus early. We also show that asynchronous applications hosted abroad, like Dropbox and SMTP, queue up tasks during the failure and cause traffic bursts when the failure is restored.*

***Resumo.*** *Neste trabalho caracterizamos o impacto de falhas na Rede Nacional de Ensino e Pesquisa (RNP) no tráfego da Universidade Federal de Juiz de Fora. Nós estudamos o impacto das falhas no comportamento do tráfego, dos usuários e das aplicações na rede da universidade. As falhas estudadas são relevantes pois persistem por várias horas e afetam apenas enlaces internacionais da RNP, sem impedir acesso a destinos no Brasil. Nossos resultados mostram que embora falhas nos enlaces internacionais da RNP tenham impacto desprezível no desempenho de conexões nacionais, usuários modificam seu comportamento em função da indisponibilidade de serviços hospedados fora do Brasil. Por exemplo, o tráfego de entretenimento migra do Facebook para o YouTube, que permanece ativo durante as falhas; e a fração de tráfego interativo reduz gradativamente durante a falha, indicando evasão dos usuários da rede. Mostramos também que aplicações assíncronas com servidores fora do Brasil, como Dropbox e SMTP, acumulam tarefas durante a falha e causam rajadas de tráfego quando a falha é restaurada.*

## 1. Introdução

Falhas de comunicação na Internet podem ser causadas por problemas de *hardware*, como o rompimento de cabos, ou erros de *software*, como configuração inadequada de um roteador. A maioria destas falhas passa despercebida graças às mudanças automáticas de roteamento que as contornam usando rotas alternativas [Markopoulou et al. 2008]. Porém, falhas causadas por erros de configuração ou ausência de rotas alternativas requerem intervenção humana e suas soluções pode levar horas [Kompella et al. 2007].

Embora a literatura seja rica em esforços para *caracterizar* anomalias e falhas na Internet (e.g., [Markopoulou et al. 2008, Turner et al. 2010, Kompella et al. 2007, Zhang et al. 2008, Lakhina et al. 2004]), o *impacto* desses problemas no tráfego e no comportamento dos usuários ainda é pouco conhecido [Markopoulou et al. 2008]. Além disso, o comportamento dos usuários e o tráfego da Internet mudaram ao longo dos anos. O volume de tráfego P2P, antes fração dominante do tráfego total, reduziu frente à popularização da distribuição de vídeo sob demanda via HTTP [Sandvine 2013]. Aplicações interativas, como redes sociais, ferramentas colaborativas de edição de documentos e serviços bancários, são cada vez mais utilizadas pelos usuários de Internet. Várias novas aplicações estão disponíveis na nuvem e só podem ser utilizadas com acesso à rede. Em geral, a maior dependência da Internet para realização de várias tarefas diárias agrava o impacto de falhas de conectividade nos usuários.

Neste trabalho nós investigamos o impacto de falhas em enlaces internacionais no tráfego de um importante campus universitário. Para tal, nós caracterizamos o tráfego capturado do roteador de borda da Universidade Federal de Juiz de Fora (UFJF), conectada à Internet por meio da Rede Nacional de Ensino e Pesquisa (RNP). Nossa base de dados contém sumários de todos os fluxos de entrada e saída da rede da universidade durante o período letivo (entre janeiro e março de 2013), cobrindo mais de 40 TB de dados (seção 3). Em particular, caracterizamos mudanças no *comportamento*—volume, padrões e características do tráfego—dos usuários e aplicações durante períodos em que, sabidamente, houve queda dos enlaces internacionais da RNP. Nossos objetivos específicos são caracterizar o impacto das falhas no comportamento do tráfego (seção 4), dos usuários (seção 5) e das aplicações utilizadas no campus universitário (seção 6). Um dos diferenciais deste trabalho é que as falhas analisadas são parciais: apesar de destinos fora do Brasil terem ficado inacessíveis, destinos no Brasil praticamente não foram afetados.

Nossos resultados mostram que falhas nos enlaces internacionais da RNP não afetam o desempenho do tráfego nacional, possivelmente devido ao bom provisionamento dos enlaces nacionais. Nossa análise indica redução gradativa de tráfego interativo durante as falhas, o que pode ser resultante de usuários deixando o campus prematuramente devido aos problemas de conectividade. Outro padrão identificado é a migração das aplicações utilizadas, particularmente aplicações de redes sociais, de sites indisponíveis (hospedados no exterior) para sites hospedados nacionalmente (e.g., do Facebook para o YouTube). Mostramos também que aplicações assíncronas que executam em plano de fundo, e.g., Dropbox e SMTP, acumulam tarefas durante a falha e causam rajadas de tráfego após a restauração da falha.

Nossos resultados, mesmo limitados a uma universidade, representam mais um passo na direção da melhor compreensão do impacto de falhas no comportamento dos usuários. Acreditamos que os resultados apresentados são particularmente interessantes para a comunidade de pesquisa em redes de computadores e podem motivar mudanças práticas do uso e do gerenciamento dos recursos de rede existentes.

Por exemplo, de acordo com nossos resultados, rajadas de tráfego geradas por aplicações assíncronas podem comprometer o desempenho da rede após restauração de falhas. Assim, o uso de modeladores de tráfego (*traffic shaping*) ou priorização de tráfego pode ser necessário para manter os serviços de rede. Mais ainda, os resultados podem ser motivadores para potenciais mecanismos para redução da evasão dos usuários da rede

durante falhas. Por exemplo, o estabelecimento de parcerias de troca de tráfego com redes que hospedam serviços de produtividade.

## 2. Detecção, Caracterização e Impacto de Falhas na Internet

Nosso trabalho estuda o impacto que falhas na Internet causam no comportamento do tráfego, dos usuários e das aplicações. Nesta seção contextualizamos nosso trabalho apresentando técnicas de detecção e características de falhas na Internet disponíveis na literatura. Por último discutimos trabalhos relacionados sobre o impacto de falhas.

**Detecção.** A maioria das falhas são detectadas por equipamentos de rede e automaticamente reportadas a operadores via ferramentas como SNMP e *syslogs* [Turner et al. 2010]. Infelizmente, apenas operadores de rede têm acesso ao equipamento de rede e a essas ferramentas de detecção. Clientes de acesso, usuários finais e operadores de outras redes possuem pouca ou nenhuma visibilidade sobre falhas. Outro problema é que algumas falhas, como as causadas por *bugs* no *software* e erros de configuração, não são reportadas pelo equipamento de rede [Kompella et al. 2007].

Devido à dificuldade de se detectar alguns tipos de falhas, várias ferramentas de monitoramento foram propostas para complementar alarmes de equipamentos de rede. Essas ferramentas de monitoramento correlacionam informações de vários dispositivos na rede e injetam sondas de medição para obter informações mais precisas sobre o estado da rede. Além disso, essas ferramentas frequentemente detectam falhas num ambiente específico, limitando o escopo das falhas e do monitoramento realizado. Por exemplo, o SLAM é uma ferramenta que clientes de acesso à Internet podem utilizar para verificar se seus provedores estão provendo conectividade com a qualidade contratada [Sommers et al. 2007]; o NICE é uma ferramenta para detecção e diagnóstico de falhas intermitentes em *backbones* IP [Mahimkar et al. 2008]; e o NetMedic é uma ferramenta que coleta informações nos servidores e estações de trabalho de uma rede empresarial para detectar falhas [Kandula et al. 2009]. Outra frente para detecção de falhas é a análise do tráfego da rede em busca de anomalias. Estes métodos constroem um modelo do tráfego normal da rede ou enlace e depois detectam desvios deste modelo (e.g., [Lakhina et al. 2004], [Silveira and Diot 2010]).

Neste trabalho não fazemos detecção de falhas pois caracterizamos falhas reportadas pela RNP. Porém, nossos resultados podem servir de base para melhorias em sistemas de detecção de anomalias causadas por falhas parciais.

**Caracterização.** Trabalhos de caracterização de falhas de rede são menos comuns que os de detecção de falhas porque operadores em geral não publicam detalhes sobre falhas em suas redes. Pesquisadores já caracterizaram falhas na rede da CENIC (*Corporation for Education Network Initiatives in California*) e da Sprint [Turner et al. 2010, Markopoulou et al. 2008]. Nestas redes, dois tipos comuns de falha são resultantes de manutenção programada da rede e falhas intermitentes causadas por *hardware* defeituoso. Estes trabalhos também mostram que nenhum enlace está totalmente livre de falhas mas que alguns enlaces são mais propensos a falhas que outros (e.g., enlaces de tecnologia diferente). Outros trabalhos caracterizaram falhas detectadas com medições ativas na Internet (e.g., [Katz-Bassett et al. 2008, Quan et al. 2013]). Estes trabalhos enviam sondas de medição para vários destinos na Internet e inferem uma falha quando um prefixo antes acessível torna-se inacessível. Estes trabalhos encontram várias falhas dentre os prefixos monitorados, ge-

ralmente em redes menores longe do núcleo da Internet. Por exemplo, [Quan et al. 2013] encontraram que, na média, 0.15% dos prefixos normalmente acessíveis da Internet estão inacessíveis em um dado instante. Em geral, estes trabalhos indicam que a maioria das falhas na Internet duram poucos minutos, mas que poucas falhas de longa duração são responsáveis pela maior parte do tempo sem conectividade (*downtime*).

O foco deste trabalho não é caracterizar falhas, mas estudar o impacto de falhas no tráfego, usuários e aplicações. Ressaltamos que as falhas que estudamos estão entre as poucas falhas de longa duração e impacto prolongado no tráfego.

**Impacto.** Vários trabalhos caracterizaram o impacto de falhas em mudanças de roteamento na Internet [Feamster et al. 2003, Zhang et al. 2007, Li and Brooks 2011], mostrando que falhas são frequentemente seguidas (e às vezes precedidas) por mensagens BGP. Nosso trabalho é similar a outros trabalhos que tentam estimar o impacto de falhas e problemas de desempenho no comportamento do usuário em sítios Web (e.g., [Stefanov 2008]). Enquanto esses trabalhos mostram que usuários podem desistir de acessar um serviço devido a problemas de desempenho, nosso trabalho estuda mudanças no comportamento do usuário entre vários serviços durante falhas prolongadas. Não conhecemos nenhum trabalho que avalie o impacto de falhas parciais no comportamento do usuário e no tráfego de aplicações. Uma possível explicação para a existência de poucos trabalhos sobre o impacto de falhas é a necessidade de coletar dados em local que tenha visibilidade sobre o tráfego *durante* a falha, o que nem sempre é possível. Por exemplo, [Dainotti et al. 2011] caracterizaram o impacto do bloqueio de tráfego nos enlaces internacionais do Egito, em 2012. Como os autores não tinham dados coletados no Egito, eles usaram pacotes não requisitados recebidos em prefixos IP inativos (*Internet telescopes*) e mensagens BGP disponíveis publicamente. Infelizmente, como a falha no Egito bloqueou tráfego internacional, uma análise de mudança de comportamento do usuário com dados coletados externamente é impossível.

## 3. Conjunto de Dados de Tráfego e Descrição das Falhas Estudadas

Neste trabalho nós avaliamos o impacto de falhas na Rede Nacional de Ensino e Pesquisa (RNP) em dados trafegados na Universidade Federal de Juiz de Fora (UFJF). A figura 1 apresenta uma visão geral do ambiente de coleta de dados. Instalamos um chaveador (*switch*) entre o roteador de borda e o *firewall* da UFJF para espelhar o tráfego do enlace entre esses dispositivos. O roteador de borda e o *firewall* são responsáveis por rotear e filtrar, respectivamente, todo o tráfego de entrada e de saída do campus. O *firewall* da UFJF também faz tradução de endereços (NAT) para alguns nós da rede interna.

O tráfego espelhado é encaminhado a um servidor de coleta. Devido à quantidade de dados trafegados, aproximadamente 15,5 TB por mês, o servidor sumariza informações sobre o tráfego usando o TSTAT [Finamore et al. 2011]. TSTAT é uma ferramenta de código livre que coleta 111 métricas sobre as conexões, incluindo endereços IP e portos de origem e destino, horários de início e fim, número de pacotes, tráfego total, latência e um identificador da aplicação ou protocolo que criou a conexão. Os pacotes espelhados são descartados após sumarização, preservando a privacidade quanto aos dados dos usuários.

A rede da UFJF integra 22 unidades e provê conectividade para aproximadamente 6.000 computadores, conectados por rede cabeada em laboratórios de pesquisa, escritórios

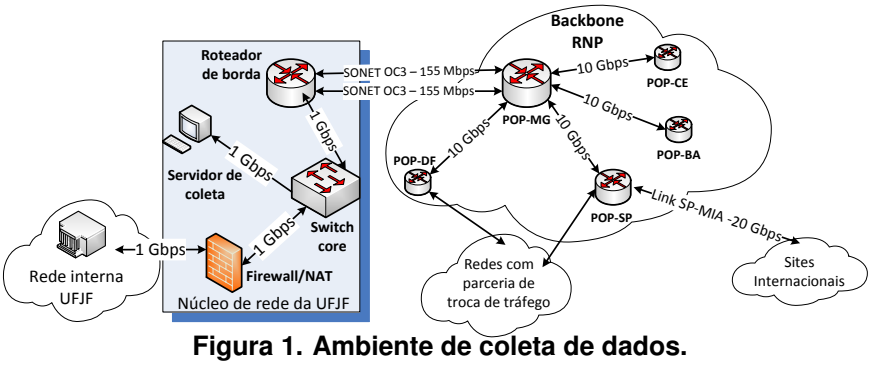



**Figura 1. Ambiente de coleta de dados.**

de administração, salas de aula e pontos de acesso sem fio. A UFJF possui aproximadamente 19.000 alunos, 1.500 funcionários e 1.400 professores. A coleta realizada não captura o tráfego interno da universidade. Apesar do tráfego interno ser interessante para análise, coletá-lo exigiria uma infraestrutura de coleta significativamente mais extensa, com um servidor de coleta em cada rede local.

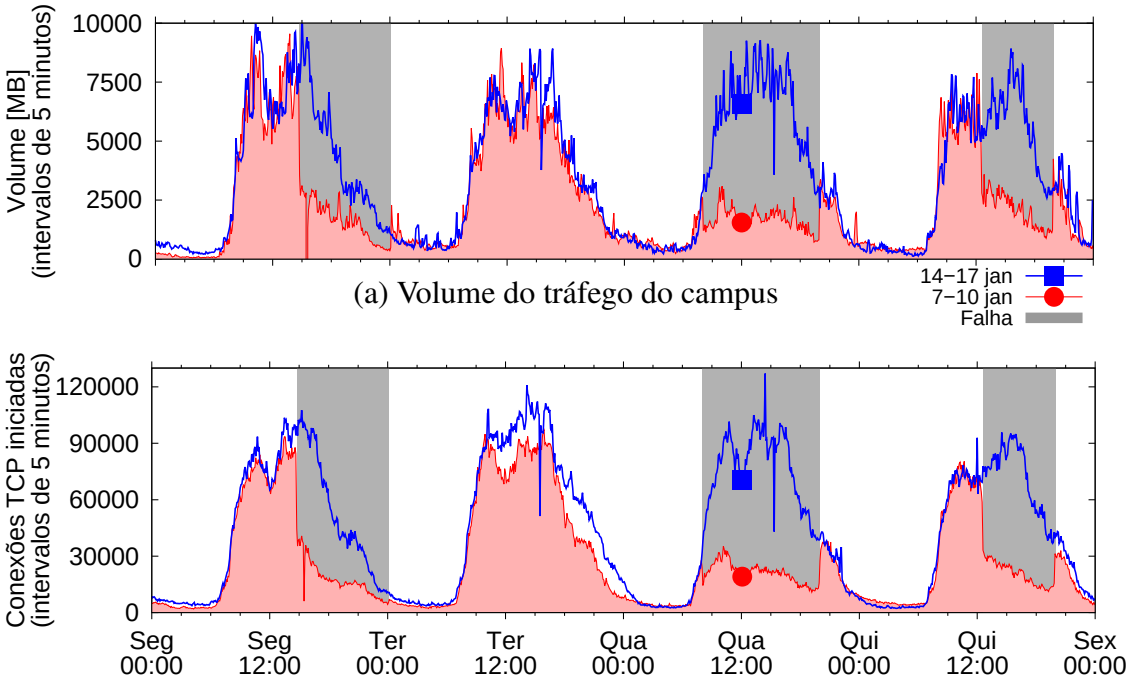
Todo o tráfego de dados da UFJF é enviado ao PoP-MG, em Belo Horizonte, por 2 enlaces ponto-a-ponto OC-3 com total de 310 Mbps de banda. A partir do PoP-MG, o tráfego da UFJF entra na RNP, que encaminha o tráfego ao seu destino. A RNP interliga praticamente todas as instituições públicas de ensino do Brasil, bem como algumas instituições governamentais (e.g., EMBRAPA). A infraestrutura da RNP é gerenciada em colaboração com pontos de troca de tráfego regionais operados por universidades, como o PoP-MG em Belo Horizonte. Empresas e redes comerciais podem se ligar à RNP nos pontos de troca de tráfego regionais, na maioria das vezes, via acordos de troca livre de tráfego (*peering*). O tráfego na RNP destinado a computadores fora do Brasil passa por enlaces internacionais. Atualmente, o mais importante desses enlaces liga São Paulo a Miami e tem banda de 20 Gbps. O tráfego da RNP destinado a empresas e redes conectadas aos pontos de troca de tráfego não utiliza os enlaces internacionais.

Focamos nossas análises em dias de falhas anunciadas pela RNP. Em particular, estudamos falhas parciais onde o acesso à Internet não é totalmente interrompido. De acordo com relatórios publicados pela RNP,[1] nos dias 7, 9 e 10 de janeiro de 2013 ocorreram falhas na infraestrutura de fibra óptica de algumas operadoras de telecomunicação. Essas falhas impossibilitaram o acesso aos enlaces internacionais da RNP. Consequentemente, destinos e serviços hospedados fora do Brasil ficaram inacessíveis, mas destinos conectados aos pontos de troca de tráfego da RNP continuaram acessíveis. Argumentamos que falhas globais têm impacto forte mas simples no tráfego—interrompendo-o por completo—e que falhas parciais são mais interessantes de analisar.

## 4. Impacto das Falhas no Tráfego

Nesta seção discutimos o impacto das falhas no tráfego da UFJF. Esta discussão servirá de base para compreensão dos resultados mais detalhados nas seções 5 e 6.

[1] `http://www.rnp.br/backbone/weblog/arquivo/arquivo_2013-m01.php`




(a) Volume do tráfego do campus

(b) Número de conexões TCP no campus

**Figura 2. Visão geral do tráfego de dados na rede da UFJF.**

### 4.1. Impacto das Falhas no Tráfego Agregado

A figura 2 apresenta uma visão geral do tráfego na rede da UFJF durante dois períodos distintos, cada um cobrindo quatro dias consecutivos. A figura 2(a) mostra o tráfego total, agregado em intervalos de cinco minutos. Como o TSTAT grava apenas o total de bytes trafegados e a duração de cada fluxo, nós distribuímos os bytes de um fluxo uniformemente ao longo de sua duração. A figura 2(b) mostra o número de conexões TCP iniciadas em intervalos de cinco minutos. As linhas azuis, marcadas com um quadrado, mostram o tráfego total e o número de conexões iniciadas no período de 14 a 17 de janeiro de 2013, quando nenhuma falha foi reportada pela RNP. As linhas vermelhas, marcadas com um círculo, mostram o tráfego e o número de conexões iniciadas entre os dias 7 e 10 de janeiro de 2013, quando a RNP reportou três falhas. Ambos períodos cobrem dias de semana de segunda a quinta-feira. As falhas ocorreram de 14:45 do dia 7 às 00:05 do dia 8 de janeiro, de 08:05 à 20:00 do dia 9 de janeiro e de 12:35 à 19:55 no dia 10 de janeiro (horários de verão de Brasília); nós sombreamos estes períodos na figura 2. Note que, devido à greve de 2012 e consequente adaptação do calendário acadêmico, as aulas na UFJF foram retomadas dia 7 de janeiro.

Em geral, o tráfego apresenta o típico padrão de uso diurno, com pico entre 10 e 16 horas e mínimo durante a madrugada. O aumento do tráfego a partir das 7 horas é mais acentuado que sua redução a partir das 18 horas. Isso ocorre devido à existência de cursos noturnos com quantidade de alunos menor que os cursos diurnos. Note que o impacto das falhas no volume total é imediato devido à interrupção do tráfego internacional. O volume de tráfego aumenta imediatamente após a restauração das falhas.

O comportamento do número de conexões TCP iniciadas é qualitativamente similar. Tentativas de conexões a destinos fora do Brasil durante a falha nunca são completadas com sucesso, são marcadas como conexões incompletas pelo TSTAT e não são contabilizadas na figura 2(b). O comportamento do número de conexões TCP ativas (ao

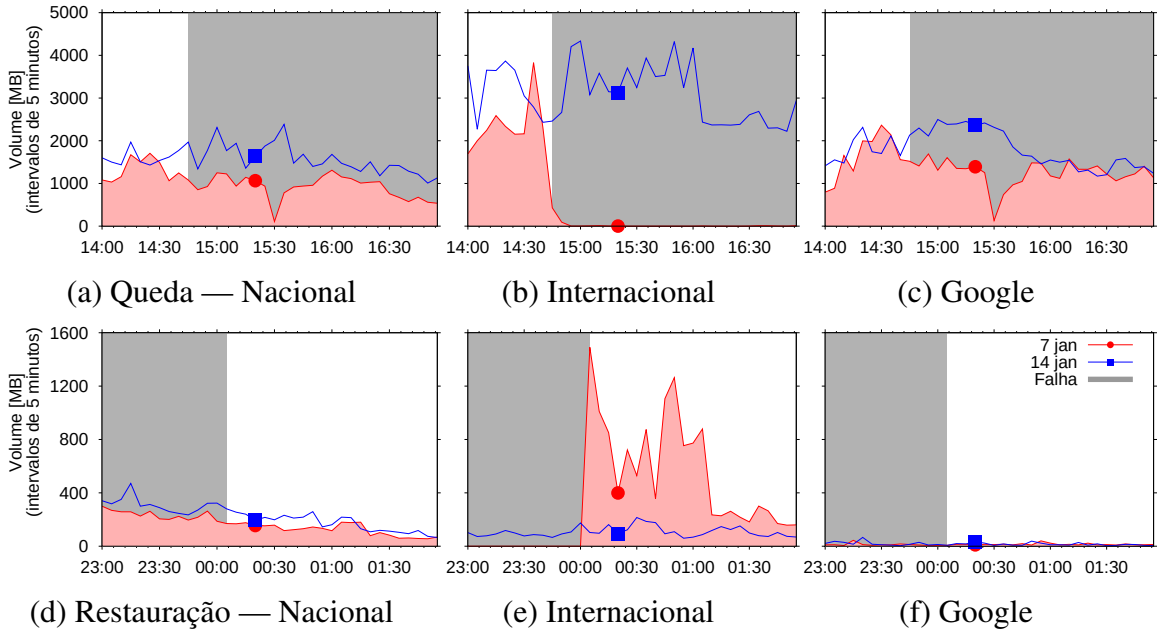



**Figura 3. Detalhe do impacto da falha no tráfego durante o início da falha (linha superior) e durante a recuperação da falha (linha inferior) no dia 7 de janeiro.**

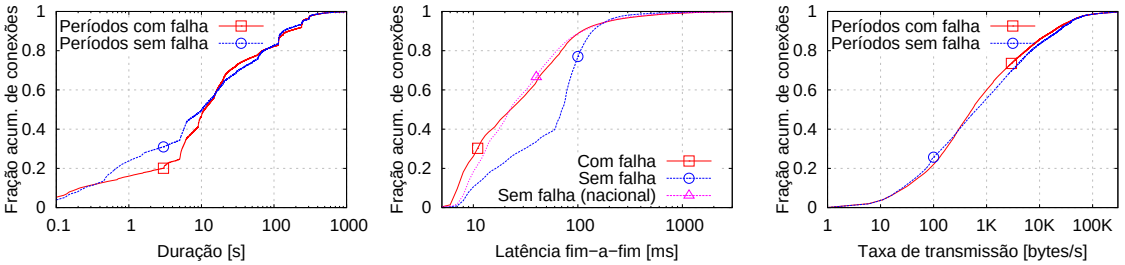
contrário de iniciadas), também é qualitativamente similar pois as conexões internacionais também são interrompidas pelas falhas (não mostrado).

### 4.2. Impacto das Falhas por Geolocalização dos Destinos

Nós separamos o tráfego total em três conjuntos: tráfego nacional, com destino no Brasil, tráfego internacional, com destino fora do Brasil, e tráfego para serviços do Google. Nós separamos tráfego para serviços do Google porque eles continuam acessíveis durante as falhas através do Ponto de Troca de Tráfego em São Paulo; além disso, grande parte do tráfego do campus é do YouTube (seção 5). Nós classificamos tráfego entre nacional e internacional usando a base de dados livre do MaxMind. Apesar das limitações conhecidas para bases de dados de geolocalização, a precisão é suficiente para a granularidade da nossa classificação [Poese et al. 2011]. Para classificar o tráfego do Google, nós resolvemos os endereços IPs associados a domínios de serviços do Google (e.g., youtube.com, gmail.com, google.com) na UFJF. Nós classificamos como tráfego para o Google qualquer conexão com endereço de origem ou destino nos prefixos referentes a estes IPs nas tabelas de roteamento do PTT-Metro em São Paulo.

A figura 3 mostra a variação dos tráfegos nacional, internacional e para serviços do Google durante períodos de três horas que cobrem o início (figuras 3(a–c)) e o término (figuras 3(d–f)) da falha do dia 7 de janeiro. Para fins de comparação, a figura mostra curvas correpondentes também para o dia 14 de janeiro, mesmo dia da semana mas sem falha reportada. Como na figura 2, nós mostramos o volume em intervalos de cinco minutos e o período de falha está sombreado.

As figuras 3(a) e 3(c) mostram que o início da falha não causa impacto imediato significativo nos tráfegos nacional e para o Google. Porém, o tráfego internacional rapidamente cai para zero. O vale mostrado em ambas as curvas por volta das 15:30 do dia 7 de janeiro foi causado por uma falha local na UFJF (reinício do roteador de borda). No momento de restauração da falha ocorre uma rajada de tráfego internacional (figura 3(e))




(a) Duração (b) Latência fim-a-fim (c) Taxa de transmissão

**Figura 4. Comparação do desempenho das conexões TCP durante a falha do dia 7 de janeiro com o mesmo período do dia 14 de janeiro (sem falha).**

gerada por aplicações assíncronas, como detalharemos na seção 6. Observamos impacto semelhante no número de conexões TCP (não mostrado).

### 4.3. Impacto das Falhas em Características e Desempenho de Conexões

Durante períodos de falha, as conexões TCP podem apresentar características diferentes das encontradas em períodos sem falha. A figura 4 mostra a distribuição acumulada da duração, latência fim-a-fim e taxa de transmissão das conexões durante um período que inclui do início à recuperação de uma falha e, para fins comparativos, um período de igual duração sem falhas. Os períodos mostrados são de 14:45 às 00:05 dos dias 7 (linha com quadrado) e 14 (linha com círculo) de janeiro.

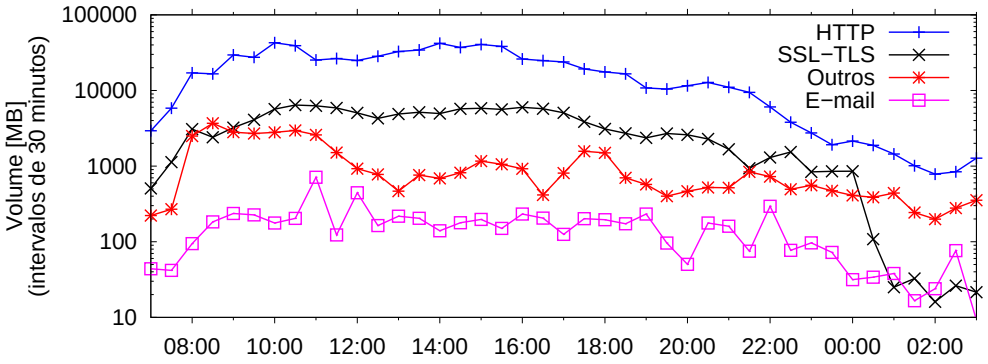
A Figura 4(a) mostra as distribuições acumuladas das durações das conexões TCP durante os períodos considerados. As distribuições são, em geral, similares exceto pela redução (em 54%) da fração de conexões com duração entre 0,5 e 3 segundos durante falhas e pelo aumento (em 16%) da fração de conexões com duração maior que 3 segundos. Como discutiremos na seção 5, uma causa para esse efeito está na mudança de comportamento dos usuários.

A figura 4(b) mostra as distribuições acumuladas da latência fim-a-fim das conexões TCP nos mesmos períodos. A diferença na latência entre os períodos com e sem falha resulta dos destinos acessados durante a falha estarem localizados em redes no Brasil e geograficamente mais próximos. Para uma comparação mais justa, mostramos também a latência fim-a-fim das conexões para destinos no Brasil durante o dia 14 (sem falha, linha marcada com um triângulo). Vemos que a falha não tem impacto na latência fim-a-fim das conexões com destino no Brasil.

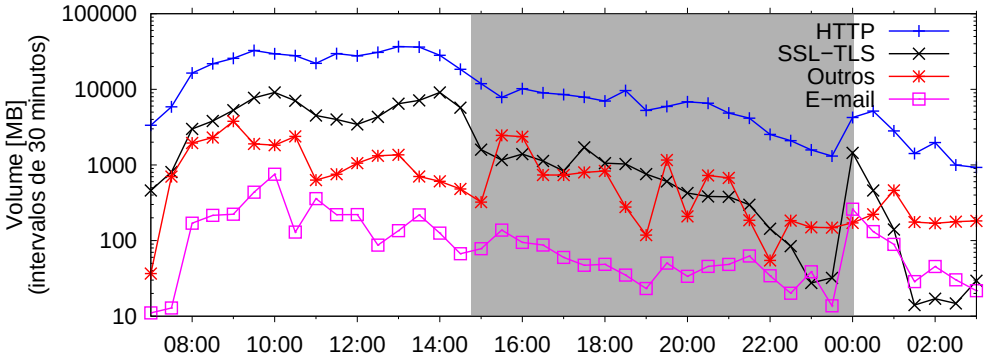
A figura 4(c) mostra que a distribuição acumulada da taxa de transmissão das conexões TCP durante o período considerado não é significativamente impactada pela falha. Isto indica que os enlaces não ficaram sobrecarregados durante a falha. Analisamos também a taxa de perda de pacotes das conexões TCP e não observamos diferenças significativas entre períodos com e sem falhas. Em ambos os casos, a porcentagem de tráfego transferido em retransmissões é menos de 1,5% do tráfego total.

## 5. Impacto das Falhas no Comportamento dos Usuários

Nesta seção analisamos o impacto das falhas em características do tráfego para inferir modificações no comportamento dos usuários da rede. Discutimos redução da quantidade de tráfego interativo e modificações da mistura de aplicações utilizadas durante falhas.




(a) Dia sem falha, 14 de janeiro




(b) Dia com falha, 7 de janeiro

**Figura 5. Volume de tráfego por protocolo.**

### 5.1. Redução de Tráfego Interativo Durante Falhas

A figura 5 mostra o volume total de tráfego de diferentes protocolos, segundo classificação do TSTAT, em intervalos de trinta minutos. A linha "E-mail" combina os protocolos POP, IMAP e SMTP. Todos os demais protocolos são agrupados na linha "Outros".

Nós mostramos na figura 5(a) tráfego normal no dia 14 de janeiro, sem falha, e na figura 5(b) o tráfego no dia 7 de janeiro, com falha. O comportamento da quantidade de conexões TCP por protocolo ao longo do tempo é qualitativamente similar (não mostrado). Apesar de mostrarmos apenas resultados para a falha do dia 7, os resultados apresentados também são válidos para as falhas dos dias 9 e 10 de janeiro.

Durante o período sem falhas, o volume de tráfego de cada protocolo se mantém relativamente estável entre 7:00 e 22:00 horas (figura 5(a)). O volume de tráfego criptografado (protocolo "SSL/TLS") diminui significativamente durante a madrugada. Em períodos de falhas, o volume de tráfego criptografado diminui gradativamente a partir do início da falha (figura 5(b)); compare, por exemplo, a quantidade de tráfego criptografado entre 20:00 e 20:30 dos dias 7 e 14 de janeiro (425 MB e 1.3 GB, respectivamente). Discutiremos os picos de tráfego criptografado e de e-mail após a restauração da falha (figura 5(b)) na seção 6.

Como o tráfego criptografado é resultante primariamente de atividades dos usuários na rede,[2] acreditamos que os *usuários saem do campus ou de suas estações de trabalho*

[2]Por exemplo, um dos destinos com maior quantidade de conexões criptografadas em nossos dados é o

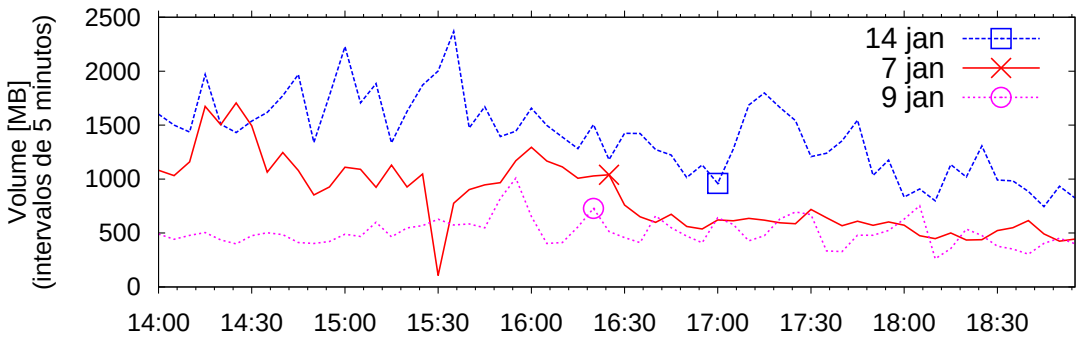



**Figura 6. Modificação no volume de tráfego nacional devido às falhas.**

*prematuramente* em função da falha de conectividade. Para testar a hipótese acima nós estimamos o número de usuários pela quantidade de conexões realizadas a serviços do Google (não mostrado). Não consideramos serviços como Twitter e Facebook porque estão hospedados fora do Brasil e, diferentemente de serviços do Google, ficam inacessíveis durante falhas. Às 17:00 horas do dia 7 de janeiro, aproximadamente 2 horas após o início da falha, a quantidade de conexões para serviços do Google era 45% menor que durante o mesmo período no dia 14, sem falha.

A figura 6 compara o volume de tráfego nacional nos dias 7 e 9 de janeiro (com falha) e no dia 14 de janeiro (sem falha). As falhas começaram às 14:45 e 8:05 nos dias 7 e 9 de janeiro, respectivamente (nesta figura não sombreamos as falhas). Vemos que mesmo o tráfego nacional, cujos serviços continuam acessíveis durante a falha, diminui no dia 7; mais um indicativo que usuários deixam o campus. Além disso, no dia 9 o tráfego nacional nunca alcança o volume normal em dias sem falha; indicando que usuários talvez nem vão ao campus ao receber notícia da falha de rede.

### 5.2. Modificações da Mistura de Aplicações Utilizadas

A tabela 1 compara tráfego das aplicações em um período de falha (entre 14:45 à 00:05 dos dias 7 e 8 de jan.) com período equivalente dos dias 14 e 15 de janeiro. O tráfego foi classificado em aplicações pelo TSTAT. Sumarizamos os resultados agregando aplicações como Twitter, MSN e Flickr em "Social"; MegaUpload, HotFile e RapidShare em "Hospedagem de Arquivos"; BitTorrent e eDonkey em "P2P" e provedores de anúncios em "Propaganda". Para cada um desses conjuntos, mostramos: a fração de suas conexões relativas ao total de conexões no período, a fração de seu tráfego relativo ao tráfego total no período, o volume de tráfego e o volume médio de suas conexões.

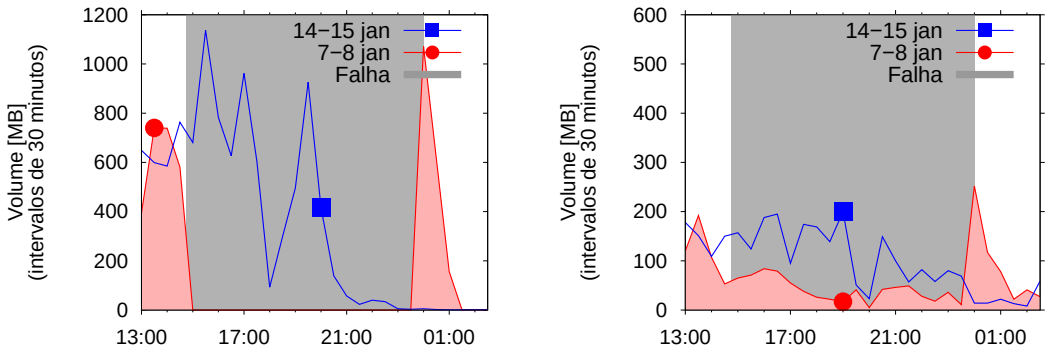
Como o Facebook é hospedado fora do Brasil, seu tráfego é reduzido a zero durante a falha. Em contrapartida, a fração de tráfego e conexões para o YouTube aumenta significativamente. Isso indica adaptabilidade dos usuários e uma migração das conexões de entretenimento para serviços que continuam disponíveis durante a falha. Notamos que a fração de conexões e tráfego nem sempre aumenta para aplicações que continuam ativas durante a falha (e.g., Google Maps, não mostrado).

Notamos que a fração do tráfego para serviços de hospedagem de arquivos é pe-

site da Prefeitura de Juiz de Fora (http://pjf.mg.gov.br).

**Tabela 1. Comparação do tráfego de aplicações entre 14:45 e 00:05 dos dias 7 e 8 de janeiro (período de falha) e o mesmo período nos dias 14 e 15 de janeiro (sem falha).**

| Tipo de Tráfego | % das conexões | | % do tráfego | | Volume de tráfego (GB) | | Volume por conexão (KB) | |
|---|---|---|---|---|---|---|---|---|
| | 07/01/13 | 14/01/13 | 07/01/13 | 14/01/13 | 07/01/13 | 14/01/13 | 07/01/13 | 14/01/13 |
| HTTP GET | 63,86 | 44,93 | 41,03 | 53,51 | 317,08 | 592,85 | 34,98 | 48,68 |
| HTTP POST | 2,20 | 2,77527 | 0,44 | 1,12 | 3,41 | 12,46 | 10,97 | 16,57 |
| Youtube | 3,06 | 1,07 | 39,43 | 17,53 | 304,72 | 194,24 | 702,29 | 668,79 |
| Propaganda | 2,23 | 2,15 | 0,33 | 0,41 | 2,59 | 4,54 | 8,18 | 7,79 |
| Social | 0,057 | 0,50 | 0,03 | 0,08 | 0,25 | 0,89 | 31,03 | 6,52 |
| Facebook | 0,007 | 6,46 | 0 | 4,21 | 0 | 46,65 | 0,00 | 26,62 |
| Hospedagem | 0 | 0,01 | 0 | 2,76 | 0 | 30,62 | 0,00 | 0,00 |
| P2P | 0,47 | 1,25 | 3,34 | 1,72 | 25,81 | 19,06 | 662,19 | 119,57 |
| Email | 1,38 | 0,82 | 0,66 | 0,78 | 5,10 | 8,64 | 44,56 | 82,66 |
| SSL/TLS | 10,83 | 20,6 | 9,26 | 15,21 | 71,56 | 168,53 | 79,67 | 64,16 |
| Demais | 15,90 | 19,44 | 0,9 | 0,8 | 6,95 | 8,86 | 5,39 | 3,81 |




(a) Dropbox (b) SMTP

**Figura 7. Impacto das falhas no volume de tráfego de aplicações assíncronas.**

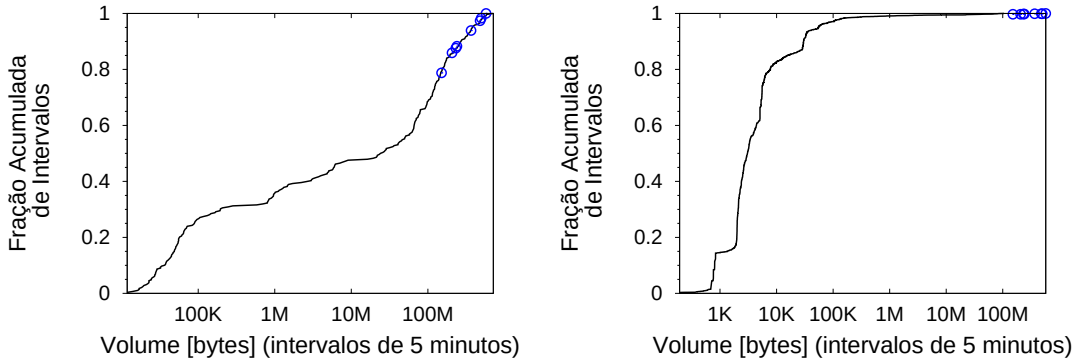
quena. O tráfego de aplicações P2P como Bittorrent e eDonkey também é pequeno, em parte devido a regras de bloqueio no *firewall* da UFJF. Como o tráfego dessas aplicações é pequeno com e sem falha, não conseguimos observar mudança de comportamento. Por último, a redução da proporção de conexões HTTP POST corrobora nosso resultado anterior de redução do tráfego interativo e possível evasão dos usuários da rede.

Também analisamos as frações de conexões e tráfego associadas a cada conjunto de aplicações 30 minutos após a restauração das falhas (não mostrado). Observamos que um intervalo de 30 minutos é suficiente para que usuários percebam a restauração da falha e retornem para o comportamento normal. Por exemplo, 30 minutos é suficiente para que a fração de conexões ao Facebook se normalize. Para as falhas analisadas, o volume de tráfego fica alterado por algumas horas, devido a modificações no comportamento de aplicações (seção 6). Infelizmente todas as falhas analisadas foram restauradas durante a madrugada e não pudemos analisar nenhuma falha restaurada em horário de pico.

## 6. Impacto das Falhas no Comportamento de Aplicações

Falhas têm impacto imediato no comportamento do tráfego e impacto gradativo no comportamento do usuário. Nesta seção avaliamos o impacto de falhas no comportamento de aplicações assíncronas que executam constantemente em plano de fundo.

A figura 7 mostra o volume de tráfego, em intervalos de trinta minutos para co-




(a) Tráfego no dia 17 de janeiro (b) Intervalos entre 19:55 e 20:35

**Figura 8. Distribuição do volume de tráfego Dropbox em intervalos de 5 minutos em diferentes períodos. Pontos em destaque são o volume de tráfego Dropbox nos 40 minutos seguintes à falha do dia 10 de janeiro, em intervalos de 5 minutos.**

nexões Dropbox[3] e SMTP (*Simple Mail Transport Protocol*), segundo classificação do TSTAT. Como anteriormente, as linhas azuis marcadas com um quadrado mostram o tráfego dos dias 14 e 15, sem falhas, e as linhas vermelhas marcadas com um círculo mostram o tráfego dos dias 7 e 8, com falha.

Como o Dropbox é hospedado na plataforma Amazon Web Services, todo o tráfego é interrompido durante a falha. De forma similar, parte do tráfego SMTP é internacional e interrompido durante a falha. Quando a falha é restaurada às 00:05 do dia 8, as tarefas acumuladas pelas duas aplicações ao longo da falha (i.e., arquivos criados e modificados no Dropbox bem como e-mails enfileirados) são disparadas. Para ambas aplicações, percebemos uma rajada de tráfego após a restauração da falha. A rajada aumenta o tráfego enviado pela aplicação e tem duração relativamente curta, entre 30 minutos e 1 hora.

A linha na figura 8(a) mostra a distribuição acumulada do volume de tráfego Dropbox durante o dia 17 de janeiro de 2013 (sem falha), em intervalos de cinco minutos. Nós também calculamos o tráfego Dropbox durante a rajada de tráfego após restauração de uma falha. Em particular, calculamos o tráfego em intervalos de cinco minutos durante os 40 minutos seguintes à restauração da falha do dia 10 de janeiro (i.e., oito intervalos de cinco minutos entre 19:55 e 20:35). Nós marcamos o tráfego relativo aos intervalos com rajada de tráfego com círculos azuis sobre a distribuição de tráfego Dropbox do dia 17 de janeiro. Focamos na falha do dia 10 de janeiro pois foi a falha restaurada mais cedo (às 19:55), de forma que uma comparação com o tráfego Dropbox em dias normais fosse mais realista. Os resultados mostram que os intervalos seguintes à falha têm volume de tráfego Dropbox maior que a maioria dos intervalos ao longo do dia, e comparável ao tráfego Dropbox em horários de pico.

A figura 8(b) é similar, mas a linha mostra a distribuição acumulado do volume de tráfego Dropbox apenas nos intervalos de 5 minutos entre 19:55 e 20:35 dos dias 14, 15, 16, 17 e 18 de janeiro (sem falhas). A figura 8(b) mostra que a rajada de tráfego Dropbox após restauração de uma falha é significativamente maior que o tráfego típico do horário, até 7 vezes maior no período de 40 minutos analisado.

[3]Na figura 5 e na tabela 1 a maior parte do tráfego Dropbox está classificado como “SSL/TLS”.

Atualmente, aplicações de armazenamento de arquivos em nuvem (*cloud storage*) são responsáveis por uma fração não desprezível do tráfego do campus. Apenas o Dropbox, uma das aplicações mais populares para tal fim, consome na média 4% da banda da universidade. Rajadas de tráfego combinadas a um possível aumento do volume de tráfego destas aplicações podem comprometer o desempenho da rede em períodos pós-falha e degradar o desempenho de aplicações interativas como teleconferências.

## 7. Discussão e Trabalhos Futuros

Este trabalho apresentou uma caracterização do impacto de falhas na rede da RNP no tráfego da Universidade Federal de Juiz de Fora. As falhas estudadas são parciais: enlaces e destinos internacionais ficaram inacessíveis durante horas enquanto destinos no Brasil praticamente não foram afetados. Nós focamos no impacto das falhas no comportamento dos usuários e no comportamento de aplicações assíncronas que rodam em plano de fundo. Observamos indicativos de evasão dos usuários da rede da universidade durante falhas. Observamos também que o tráfego de entretenimento migra de aplicações indisponíveis para aplicações disponíveis (em particular, do Facebook para o YouTube). Por último, discutimos que aplicações assíncronas com servidores fora do Brasil, como Dropbox e SMTP, podem acumular tarefas durante a falha e gerar uma rajada de tráfego imediatamente após restauração da falha.

Um potencial mecanismo para redução da evasão dos usuários da rede durante falhas é o estabelecimento de parcerias de troca de tráfego com redes que hospedam serviços de produtividade. Em particular, conectividade com o Google, Akamai e Amazon Web Services em pontos de troca de tráfego nacionais tornaria vários serviços disponíveis durante falhas. Esta solução depende da criação de centros de processamento de dados no Brasil, o que requer investimentos. Uma alternativa mais imediata é usar outras redes como provedores de acesso a serviços críticos durante falhas; por exemplo, outras redes educacionais da América Latina podem prover conectividade internacional temporariamente durante falhas.

Rajadas de tráfego geradas por aplicações assíncronas após restauração de falhas podem comprometer o desempenho da rede. Apesar das falhas que analisamos terem sido restauradas em período de baixa carga (após 19:55), rajadas após uma restauração em horário de pico podem levar a congestionamento de enlaces internacionais e comprometer aplicações como voz sobre IP. Este problema pode ser mitigado por modificações no *software*, modeladores de tráfego (*traffic shaping*) ou priorização de tráfego.

Atualmente estamos procurando outras universidades parceiras para obter dados de períodos com falha e estender nossa análise. Queremos também explorar alterações no comportamento de usuários e aplicações para melhorar técnicas de detecção de falhas. Por exemplo, um período com ausência de tráfego Dropbox seguido de uma rajada pode ser indicativo de uma falha (potencialmente parcial) de acesso a servidores do Dropbox.

## Referências

Dainotti, A., Squarcella, C., Aben, E., Claffy, K. C., Chiesa, M., Russo, M., and Pescapé, A. (2011). Analysis of Country-wide Internet Outages Caused by Censorship. In *Proc. IMC*.

Feamster, N., Andersen, D., Balakrishnan, H., and Kaashoek, F. (2003). Measuring the Effects of Internet Path Faults on Reactive Routing. In *Proc. ACM SIGMETRICS*.

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.

Kandula, S., Mahajan, R., Verkaik, P., Agarwal, S., Padhye, J., and Bahl, P. (2009). Detailed Diagnosis in Enterprise Networks. In *Proc. ACM SIGCOMM*.

Katz-Bassett, E., Madhyastha, H., John, J. P., Krishnamurthy, A., Wetherall, D., and Anderson, T. (2008). Studying Black Holes in the Internet with Hubble. In *Proc. USENIX NSDI*.

Kompella, R., Yates, J., Greenberg, A., and Snoeren, A. (2007). Detection and Localization of Network Blackholes. In *Proc. IEEE INFOCOM*.

Lakhina, A., Crovella, M., and Diot, C. (2004). Diagnosing Network-wide Traffic Anomalies. In *Proc. ACM SIGCOMM*.

Li, J. and Brooks, S. (2011). I-seismograph: Observing and Measuring Internet Earthquakes. In *Proc. IEEE INFOCOM*.

Mahimkar, A., Yates, J., Zhang, Y., Shaikh, A., Wang, J., Ge, Z., and Ee, C. (2008). Troubleshooting Chronic Conditions in Large IP Networks. In *Proc. ACM CoNEXT*.

Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C. N., Ganjali, Y., and Diot, C. (2008). Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Trans. Netw.*, 16(4):749–762.

Poese, I., , Uhlig, S., Kaafar, M. A., Donnet, B., and Gueye, B. (2011). IP Geolocation Databases: Unreliable? *SIGCOMM Comput. Commun. Rev.*, 41(2):53–56.

Quan, L., Heidemann, J., and Pradkin, Y. (2013). Trinocular: Understanding Internet Reliability Through Adaptive Probing. In *Proc. ACM SIGCOMM*.

Sandvine (2013). Global Internet Phenomena Report 2H2013. Available at: `http://www.sandvine.com/trends/global-internet-phenomena`.

Silveira, F. and Diot, C. (2010). URCA: Pulling out Anomalies by their Root Causes. In *Proc. IEEE INFOCOM*.

Sommers, J., Barford, P., Duffield, N., and Ron, A. (2007). Accurate and Efficient SLA Compliance Monitoring. In *Proc. ACM SIGCOMM*.

Stefanov, S. (2008). YSlow 2.0. In *CSDN SC2C*.

Turner, D., Levchenko, K., Snoeren, A., and Savage, S. (2010). California Fault Lines: Understanding the Causes and Impact of Network Failures. In *Proc. ACM SIGCOMM*.

Zhang, Y., Mao, Z., and Wang, J. (2007). A Framework for Measuring and Predicting the Impact of Routing Changes. In *Proc. IEEE INFOCOM*.

Zhang, Z., Zhang, Y., Hu, Y. C., Mao, Z. M., and Bush, R. (2008). iSPY: Detecting IP Prefix Hijacking On My Own. In *Proc. ACM SIGCOMM*.

# Vizinhanças ou condomínios: uma análise da origem de *spams* com base na organização de sistemas autônomos

**Osvaldo Fonseca[1], Pedro Henrique B. Las-Casas[1], Elverton Fazzion[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,pedro.lascasas,elverton,dorgival,meira}@dcc.ufmg.br

{cristine,jessen,mhp}@cert.br

***Abstract.*** *Despite the continuous efforts to mitigate spam, the volume of messages is huge and identifying spammers is still a challenge. Spam traffic analysis has been performed to determine the behavior of spammers, who are employing techniques more and more sophisticated to disseminate messages. This work analyzes the sources of spam towards understanding to what extent they explain the traffic. Our results show that, beyond a similar behavior among machines from the same autonomous system (AS), it is possible to separate them according to their sending characteristics. Further, the results also show that we may apply the concept of Internet BadNeighborhoods to ASes, once the machines from a given AS behave similarly.*

***Resumo.*** *Apesar dos esforços contínuos para combater o envio de spam, o volume de mensagens enviadas é muito grande e a identificação dos responsáveis ainda é uma tarefa muito difícil. Análises do tráfego de mensagens de spam têm sido realizadas para determinar o comportamento dos spammers, que têm utilizado técnicas cada vez mais sofisticadas na disseminação dessas mensagens. Este trabalho analisa a origem dos remetentes das mensagens de spam visando entender sua influência sobre o tráfego de spam propagado pela rede. Nossos resultados mostram que, além de existir um comportamento semelhante entre as máquinas provenientes de um mesmo sistema autônomo (AS), é possível dividi-los em grupos de acordo com suas características de envio de spam. Além disso, os resultados mostram que podemos aplicar o princípio de Internet BadNeighborhoods a ASes, uma vez que as máquinas de um mesmo AS apresentam comportamento semelhante.*

## 1. Introdução

Nas últimas duas décadas houve um aumento muito grande na utilização da Internet, o que levou a um aumento dos problemas relacionados ao envio de *spam* em proporções ainda maiores. Além do grande volume de dados gerado, uma vez que os provedores de serviços de correio eletrônico estimam que entre 40% e 80% das mensagens

eletrônicas são mensagens de *spam*, muitas vezes elas estão relacionadas ao envio de *phishing* [Orman 2013] e propagação de *malware* [Newman et al. 2002]. Por causa desses fatores, os prejuízos gerados pelo tráfego de *spam* são estimados na ordem de bilhões de dólares [Sipior et al. 2004].

A batalha contra os *spammers* se trava em diversas frentes. Muito já foi feito no sentido de desenvolver filtros baseados no conteúdo da mensagem, definindo regras para identificar os padrões de ofuscação observados nas mensagens de *spam* [Guerra et al. 2010]. Nos últimos anos, muitos esforços têm focado no entendimento do tráfego de *spam* dentro da rede, a fim de levantar elementos que possam ser usados para identificar as máquinas que enviam as mensagens antes que estas percorram a rede e consumam recursos dos servidores de correio no destino. Este trabalho se encaixa nesta linha, analisando onde se encontram as máquinas usadas pelos *spammers*.

Recentemente, o termo *Internet Bad Neighborhoods* (vizinhanças ruins) foi criado para identificar faixas contíguas do espaço de endereçamento IP que contenham um número significativo de máquinas com comportamento indesejado [van Wanrooij and Pras 2010]. O princípio por trás do conceito original seria que redes, que compartilham um prefixo IP, contendo muitas máquinas com tal comportamento sugeririam uma rede com problemas. Posteriormente, o conceito foi estendido para se referir a segmentos de rede que emanam tráfego indesejado, independente do número de máquinas envolvidas [Moreira Moura et al. 2011].

Nossa análise parte de princípios semelhantes, porém escolhemos o nível dos Sistemas Autônomos (ASes, *Autonomous Systems*) para identificar possíveis *bad neighborhoods*. Sistemas Autônomos, por natureza, identificam faixas do espaço de endereço IP sob o controle de uma única entidade, responsável por definir políticas de uso, roteamento e procedimentos administrativos que serão aplicados a todas as máquinas instaladas naquele espaço. Nesse sentido, dois endereços IP pertencentes a um mesmo AS teriam uma chance muito maior de exibirem comportamentos semelhantes que dois endereços IP em sub-redes com endereços adjacentes (no espaço IP), mas pertencentes a ASes distintos. Afinal, um AS com falhas de segurança em suas políticas torna-se uma potencial *Bad Neighborhood*, pois a probabilidade das máquinas pertencentes àquele AS ficarem comprometidas e começarem a enviar *spam* é elevada.

Nossos dados contêm tráfego de *spam* coletado em diversos pontos do globo. Ao agrupar as máquinas que geram esse tráfego com base em seus ASes de origem podemos criar um perfil do comportamento de cada sistema autônomo observado durante a coleta. Nossos resultados mostram que a maior parte das máquinas que enviam *spam* se concentra realmente em alguns ASes e que apenas 15 deles são responsáveis por mais de 80% do tráfego observado. Ao utilizar técnicas de mineração de dados sobre esse material, observamos que há semelhanças entre algumas delas, sendo possível agrupá-las em quatro categorias representativas de ASes do ponto de vista de seu papel na distribuição de *spam*. Essa constatação constitui uma das principais contribuições deste trabalho, uma vez que, até então, era comum supor que haveria apenas dois tipos de geradores de *spam*: transmissores "leves" e "pesados" (LVS, de *low-volume spammers* e HVS, de *high-volume spammers*) [Pathak et al. 2008]. Nossa caracterização indica que, além de más vizinhanças, onde o mau comportamento (envio de *spam*) é bastante comum entre as máquinas de um AS, há também boas vizinhanças, onde tal comportamento é raro, normalmente desapa-

recendo rapidamente logo após seu surgimento em uma ou outra máquina, e vizinhanças coniventes, onde não se observa o mau comportamento como algo generalizado, mas limitado a apenas algumas máquinas, que por sua vezes são transmissores pesados.

## 2. Trabalhos Relacionados

A definição de *Bad Neighborhoods*, apresentada na introdução, é devida a van Wanrooij e Pras [2010], que propõem o conceito como uma extensão do uso de *blacklists* na detecção de *spam*. Naquele trabalho, cada um dos prefixos de 24 bits de endereçamento IP (/24) seria uma vizinhança e vizinhanças ruins seriam aquelas com um grande número de máquinas enviando *spam*. Moreira Moura et al. [2011] focam na análise dessas vizinhanças e estendem a definição para incluir redes com poucos IPs transmissores, mas com um alto volume de tráfego, seguindo a classificação de transmissores "pesados" e "leves" proposta anteriormente por Pathak, Hu e Mao [2008]. Neste trabalho, consideramos cada AS como sendo uma vizinhança, pois um sistema autônomo é naturalmente uma boa forma de definir a área de cada vizinhança, uma vez que existe para cada AS uma gerência única e uma política de roteamento comum a todas as máquinas.

Muitos trabalhos avaliam o tráfego de *spam* a partir de mensagens coletadas nos servidores de correio de destino. Gomes et al. [2005] têm como objetivo encontrar características que podem ser usadas para separar mensagens legítimas de mensagens de *spam*, e o trabalho utiliza apenas um ponto específico da rede na coleta. Neste trabalho, as mensagens de *spam* são coletadas por *honeypots* de baixa interatividade instalados em 10 países diferentes e localizados em redes intermediárias. Isso nos permite ter uma visão mais global do tráfego de *spam*, além de oferecer uma perspectiva diferente.

Kokkodis and Faloutsos [2009] mostram alguns resultados que indicam que as atividades de *botnets* estão espalhadas no endereçamento IP, reduzindo a eficácia dos filtros *anti-spam* e dificultando o trabalho dos administradores das redes. O nosso trabalho, apesar de confirmar a existência de *spammers* em um número muito grande de redes, mostra que a maior parte das mensagens de *spam* são provenientes de um número pequeno de ASes, resultado que pode ser usado no desenvolvimento de novas técnicas para a detecção de *spam*.

Las-Casas et al. [2013] realizam uma análise detalhada do tráfego de mensagens de *spam*, que oferece uma visão geral sobre o tráfego e mostra as diferenças existentes entre os pontos de coleta utilizados, por um período de 3 meses. Neste trabalho, coletamos mensagens de *spam* durante um período de aproximadamente um ano, permitindo uma análise mais abrangente do tráfego de *spam*. Além disso, aprofundamos os estudos sobre os sistemas autônomos, a fim de entender melhor a origem dos remetentes das mensagens de *spam*.

## 3. Metodologia

Para coletar a base de dados utilizada no trabalho, foram usados doze *honeypots* de baixa interatividade [Steding-jessen et al. 2007] instalados em dez países: dois no Brasil, dois nos Estados Unidos e um coletor em cada um dos seguintes países: Argentina, Austrália, Áustria, Equador, Holanda, Taiwan, Noruega e Uruguai. Pode-se notar que temos coletores presentes em vários continentes, permitindo ao estudo uma visão mais global do tráfego de mensagens de *spam*. Com isso, eliminamos alguns problemas presentes em

vários trabalhos encontrados na literatura, cujos dados utilizados vêm de um único coletor. Além disso, nenhum dos coletores utilizados na análise apresentou indício de ter sido alvo de algum ataque específico e a possibilidade de *spoofing* é reduzida no caso de conexões estáveis, pois seria necessário interceptar a rota de retorno dos pacotes.

Os *honeypots* usados no trabalho são coletores que simulam servidores vulneráveis, como *open relays* SMTP e *proxies* abertos HTTP e SOCKS. Seu objetivo é exatamente capturar as mensagens de *spam* enviadas por *spammers* que vêem um coletor como uma máquina vulnerável, que pode ser abusada a fim de usá-la para a entrega de mensagens. Na prática, o coletor não entrega as mensagens de *spam* aos destinatários pretendidos; ao invés disso, elas são armazenadas localmente. O comportamento dos *honeypots*, entretanto, é configurado para levar o *spammer* a crer que a entrega está funcionando corretamente. As mensagens são periodicamente recolhidas por um sistema de coleta.

Junto com cada mensagem recebida são armazenadas informações adicionais obtidas pelo sistema no momento do envio. Essa informações incluem o protocolo usado pelo *spammer* (SMTP, SOCKS ou HTTP), o prefixo de rede e o AS de origem, o status do IP de origem em *blacklists* como Spamhaus PBL[1] e XBL[2] , entre outras. Dessa forma, a análise posterior considera o status dessas informações no momento do envio da mensagem e não através de uma consulta posterior, que poderia gerar erros de interpretação.

Posteriormente, durante a análise, alguns ASes mereceram um estudo mais aprofundado. Nesses casos, com base nos seus números de AS, buscamos por dados disponíveis na Internet por mais detalhes sobre as suas atividades. Com base nas atividades que foram identificadas durante essa busca, classificamos os ASes como sendo provedores, provedores DSL, serviços de *hosting*/*colocation*, provedores de acesso a empresas, etc.

Nas análises deste trabalho consideramos aproximadamente um ano de coleta, do dia 9 de maio de 2012 até o dia 31 de março de 2013, resultando em quase quatro bilhões de mensagens e 14TB em volume de dados. Ao analisar um período grande do tráfego de mensagens de *spam*, evitamos associar um comportamento atípico, que ocorreu em um curto período de tempo, ao comportamento geral das mensagens de *spam*. Portanto, os indícios encontrados, por serem recorrentes, possuem maior confiabilidade e relevância.

## 4. Resultados

A tabela 1 mostra uma visão geral das mensagens de *spam* utilizadas no estudo, que foram coletadas pelos 12 *honeypots*, discriminadas pelo protocolo que foi utilizado pelo *spammer* no envio. Durante o período de quase um ano de coleta foram obtidas 3,97 bilhões de mensagens, que corresponderam a 14 TB de dados. Os endereços das máquinas que enviaram *spam* estão associados a 149 *country codes* distintos, que correspondem a cerca de 60% de todos os países. Também é possível notar um número grande sistemas autônomos, 3226, mostrando que a coleta conseguiu alcançar muitas sub-redes, permitindo uma visão

[1]**Policy Block List (PBL):** endereços IP de usuários finais que não deveriam estar disseminando emails com SMTP não autenticados para qualquer servidor de email, exceto quando especificado pelo ISP.

[2]**Spamhaus Exploits List (XBL):** Banco de dados em tempo real de endereços endereços IP infectados por *exploits*, incluindo *open proxies* (HTTP, socks, AnalogX, etc), *worms/virus* com *spam engines* e outros tipos de *trojan-horse exploits.*

mais global do tráfego de *spam*.

**Tabela 1. Visão geral dos dados**

| | SMTP | SOCKS | HTTP | Total |
|---|---|---|---|---|
| Mensagens (x$10^6$) | 690 (17,4%) | 2.486 (62,5%) | 799 (20,1%) | 3.976 |
| Endereços IP | 294.072 | 34.397 | 11.449 | 328.050 |
| Sistemas Autônomos | 3.096 | 443 | 55 | 3.226 |
| *Contry Codes* | 146 | 66 | 14 | 149 |
| Volume (GB) | 2.564 | 8.522 | 3.3785 | 14.464 |

O volume de máquinas que utilizam o protocolo SOCKS e HTTP é bem menor, se comparado com o número de endereços IP que usam o protocolo SMTP. Apesar disso, o número de mensagens enviadas pelos protocolos SOCKS e HTTP é maior. Isso mostra que não há uma relação direta entre o número de máquinas e o número de mensagens enviadas, uma vez que aumentar o número de máquinas não significa aumentar na mesma proporção o volume de *spam*. Essa divisão é um indício da separação entre *spammers* de alto e baixo volume, mas durante a análise veremos que há mais elementos a considerar.

Pela tabela 1, podemos observar que as mensagens de spam são provenientes de muitas redes diferentes, uma vez que aparecem na coleta 3226 sistemas autônomos distintos. É interessante notar que grande parte dessas mensagens de *spam* são enviadas por um número muito pequeno de sistemas autônomos, sendo 50 deles reponsáveis por mais de 85% de todo o tráfego. Desta forma, analisar o comportamento das mensagens de *spam* na origem pode direcionar os esforços para a detecção de *spam*, pois podemos identificar quais são as vizinhanças que têm mal comportamento, e que, consequentemente, são mais propensas a enviar mensagens de *spam*.

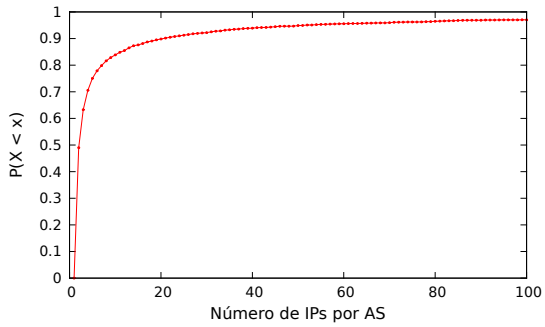
### 4.1. Distribuição dos endereços IP nos Sistemas Autônomos

A figura 1(a) mostra que, para a maior parte dos sistemas autônomos, poucas máquinas foram observadas contatando os *honeypots*. Em quase 90% dos ASes, o número de máquinas observadas foi inferior a 20. Pela tabela 2, podemos ver claramente a existência de um grupo grande de ASes com um número pequeno de endereços IP que enviam mensagens de *spam* e um outro grupo, muito menor, que contém a maior parte desses endereços. Isso mostra que as máquinas que enviam mensagens de *spam* não estão distribuídas uniformemente pelos sistemas autônomos.

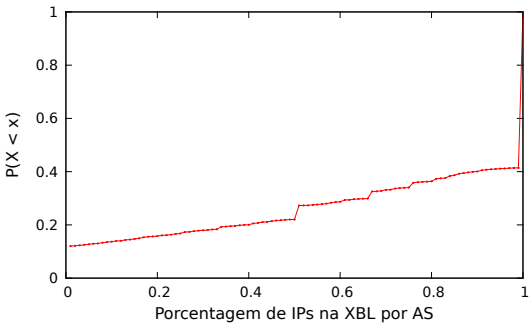
**Tabela 2. Número de endereços IP observados por AS**

| Nº de endereços IP por AS (x) | #ASes | #Mensagens (x$10^6$) | #endereços IP |
|---|---|---|---|
| x = 1 | 1581 (49,0%) | 108 (2,7%) | 1.581 (0,5%) |
| x <10 | 2705 (83,9%) | 305 (7,7%) | 5.635 (1,7%) |
| x <50 | 3061 (94,9%) | 797 (20,0%) | 13.309 (4,6%) |
| x <100 | 3131 (97,1%) | 1.150 (28,9%) | 18.159 (5,5%) |
| x $\geq$ 100 | 95 (2,9%) | 2.825 (71,1%) | 319.554 (97,4%) |

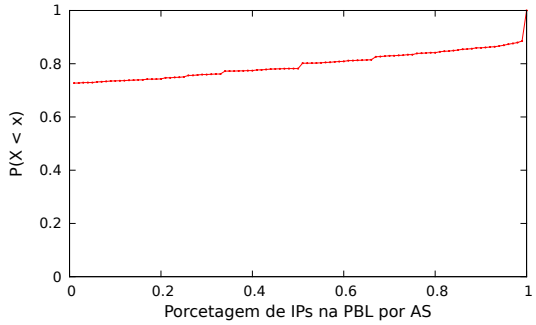
Aqueles ASes que têm menos de 10 máquinas que enviam *spam*, correspondem a mais de 83% do total. Apesar disso, eles enviam apenas 7,66% das mensagens e correspondem, em número de máquinas, a 1,7% do total. Desta forma, acreditamos que, em termos de vizinhanças, esses ASes não caracterizam uma má vizinhança, mas sim que suas políticas de segurança estão sendo bem implementadas, por causa do reduzido número



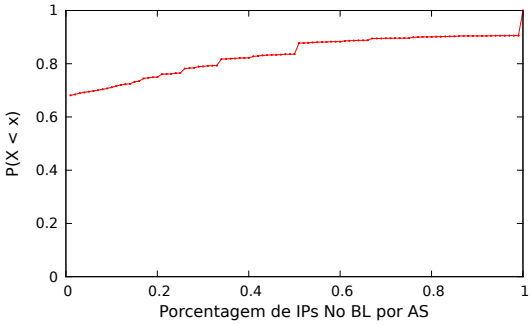

(a) CDF do número de IPs por AS




(b) CDF da porcentagem de IPs presentes na XBL por AS




(c) CDF da porcentagem de IPs presentes na PBL por AS




(d) CDF da porcentagem de IPs que não estão presentes em nenhuma *blacklist*

**Figura 1. CDFs sobre a distribuição dos endereços IP**

de endereços IP e baixo tráfego gerado por elas. Por outro lado, 95 sistemas autônomos (2,94%) contêm mais de 319 mil endereços IP (97,41%) e são responsáveis por 71% do tráfego das mensagens de *spam*, que corresponde a quase 3 bilhões de mensagens. Essas vizinhanças mostram um mau comportamento, possivelmente gerado devido a políticas de segurança fracas. Dessa forma, direcionar os esforços para entender o comportamento dessas redes é muito importante.

A figura 1 também mostra a distribuição dos endereços IP presentes em cada uma das *blacklists*. Existe uma quantidade muito pequena de sistemas autônomos que não têm endereços IP na XBL, cerca de 15%, como podemos ver na figura 1(b). Além disso, aproximadamente 60% dos ASes têm todos os seus endereços IP na XBL. Esse resultado nos faz acreditar que a maioria dos endereços IP estão na XBL, mas o que acontece é que uma boa parte dos ASes (49%) têm um único endereço IP e a maioria desses estão somente na XBL.

**Tabela 3. Visão geral das *blacklists***

| | **#IPs** | **#Msgs** ($10^6$) |
|---|---|---|
| **XBL** | 66.388 (20,2%) | 594 (14,9%) |
| **PBL** | 282.599 (86,2%) | 789 (19,8%) |
| **No BL** | 41005 (12,5%) | 2.714 (68,3%) |

Dessa forma, as informações sobre os endereços IP que estão na PBL e aqueles que não participam de nenhuma *blacklist* (NoBL) acabam ficando distorcidas, como apresentado nas figuras 1(c) e 1(d). No gráfico da figura 1(c), por exemplo, mais de 70% dos

sistemas autônomos não têm nenhum endereço IP presente na PBL, sendo que mais de 85% dos endereços IP encontrados no período de análise estão na PBL, conforme representado na tabela 3. Já os endereços IP que não estão em nenhuma *blacklist*, além de não pertencerem à maioria dos ASes, representam uma parcela muito pequena dos endereços IP, menos de 12%. Porém, esse baixo número de máquinas são responsáveis por mais de 68% de todo o tráfego das mensagens de *spam*, como apresentado na tabela 3.

### 4.2. Análise das vizinhanças com maior poder de envio

**Tabela 4. 15 principais ASes**

| AS | Msgs ($10^6$) | end. IP | IP SMTP | IP SOCKS | IP HTTP | IP XBL | IP PBL | IP No BL | vol.(GB) | Classificação |
|---|---|---|---|---|---|---|---|---|---|---|
| 10297 | 1.857 | 182 | 22,5% | 77,5% | 77,5% | 17,0% | 0,0% | 83,0% | 5.475 | hosting/colocation |
| 3462 | 359 | 100.395 | 78,7% | 22,1% | 6,8% | 7,7% | 99,8% | 0,1% | 1.320 | dsl/provedor de Internet |
| 29802 | 298 | 25 | 16,0% | 84,0% | 84,0% | 12,0% | 0,0% | 88,0% | 781 | hosting |
| 9299 | 148 | 82 | 39,0% | 61,0% | 26,8% | 34,2% | 1,2% | 65,9% | 342 | dsl/business |
| 2497 | 141 | 1.185 | 0,2% | 99,6% | 99,0% | 0,2% | 19,4% | 80,4% | 559 | hosting/clouding |
| 4134 | 126 | 110.123 | 99,7% | 0,3% | 0,2% | 15,8% | 81,8% | 17,1% | 2.446 | dsl |
| 6648 | 124 | 54 | 9,3% | 90,7% | 40,7% | 9,3% | 90,7% | 1,9% | 279 | dsl/business |
| 4725 | 31 | 215 | 0,9% | 99,1% | 98,6% | 0,9% | 95,8% | 3,3% | 120 | clouding/business |
| 27699 | 31 | 1.382 | 7,7% | 92,3% | 0,0% | 9,2% | 95,2% | 2,7% | 114 | dsl/business |
| 18881 | 28 | 3.091 | 10,7% | 89,3% | 0,0% | 8,6% | 96,1% | 2,0% | 97 | colocation/provedor |
| 8167 | 25 | 198 | 38,9% | 61,1% | 0,0% | 37,4% | 34,8% | 41,4% | 95 | clouding/provedor |
| 4837 | 23 | 20.551 | 99,9% | 0,1% | 0,0% | 26,5% | 78,4% | 18,5% | 97 | - |
| 9924 | 22 | 1.959 | 1,3% | 82,1% | 98,6% | 2,8% | 99,6% | 0,1% | 77 | - |
| 28573 | 21 | 700 | 75,7% | 24,3% | 0,0% | 46,9% | 98,4% | 0,9% | 76 | provedor |
| 4230 | 20 | 429 | 21,7% | 78,3% | 78,3% | 31,0% | 67,6% | 11,0% | 67 | hosting/provedor |

A tabela 4 mostra os 15 ASes que mais enviaram mensagens de *spam* no período analisado, sendo, sozinhos, responsáveis por mais de 80% de todo o tráfego de *spam*. Essa informação pode ser incorporada aos filtros de *spam* e utilizada pelos administradores das redes para reduzir o volume de *spam*, uma vez que mensagens enviadas por esses ASes tendem a ser *spam*. Ao analisar esses sistemas autônomos, apesar de todos enviarem um número alto de mensagens de *spam*, vemos que existem algumas vizinhanças que têm características muito diferentes umas das outras. Por outro lado, é possível notar que algumas delas são muito semelhantes, apesar de não terem nenhuma relação direta.

Alguns sistemas autônomos (10297 e 29802) têm características semelhantes em praticamente todos os aspectos. Esses dois ASes têm um número muito pequeno de endereços IP, sendo que a maioria deles utiliza os protocolos SOCKS e HTTP para enviar as mensagens de *spam* e não estão em nenhuma *blacklist*. O AS 2497 também é muito parecido, apesar de ter um número de máquinas maior. O AS 4725, por sua vez, diverge apenas por ter um número grande de endereços IP na *blacklist* PBL. As máquinas dessas vizinhanças se comportam de forma semelhante a servidores dedicados ao envio de *spam*, pois utilizam os protocolos SOCKS e HTTP, e a maior parte dos endereços IP não está em nenhuma *blacklist*, permitindo o envio de um número grande de mensagens pela mesma máquina.

Em contrapartida, podemos encontrar algumas vizinhanças com características completamente diferentes, como os sistemas autônomos 3462 e 4134. Ambos têm mais de 100 mil endereços IP, a grande maioria das máquinas observadas utilizou o protocolo SMTP para enviar *spam* e a maior parte deles estava em alguma *blacklist*. Além disso, o AS 4134 tem características muito marcantes, com mais de 99% dos seus endereços IP enviando mensagens de *spam* utilizando o protocolo SMTP e cerca de 17 mil deles estão

na XBL[3].

### 4.3. Agrupamento dos sistemas autônomos

Por conta dos indícios mencionandos na seção 4.2, buscamos uma forma de agrupar melhor os ASes observados e assim classificá-los de acordo com suas características. Para isso, utilizamos o algoritmo de agrupamento X-means, considerando as características das vizinhanças como atributos. O algoritmo tem a qualidade de definir automaticamente o número ideal de *clusters* a ser usado, ao contrário de outros algoritmos de agrupamento.

Para realização do agrupamento, utilizamos como atributos características que consideramos representar os sistemas autônomos. Os atributos envolvem diversas informações de cada AS, tais como: número de endereços IP, número de mensagens por dia, porcentagem de endereços IP em cada uma das *blacklists*, porcentagem de endereços IP que utilizam cada um dos protocolos e o número médio de mensagens enviadas por cada endereço IP. Esses atributos determinam bem as vizinhanças, pois definem os tipos de comportamento que as máquinas de uma rede podem assumir.

**Tabela 5. Atributos de cada um dos agrupamentos**

| | Grupo 1 | Grupo 2 | Grupo 3 | Grupo 4 |
|---|---|---|---|---|
| Nº ASes | 2.064 | 449 | 359 | 354 |
| Nº Msgs (x$10^6$) | 379 | 2.602 | 88 | 907 |
| Nº Endereços IP | 8.426 | 16.024 | 11.503 | 301.760 |
| Msgs/IP (x$10^3$) | 45,0 | 162,4 | 7,6 | 3,0 |
| Atividade[4] | 48,8 | 63,4 | 67,1 | 85,3 |
| Nº Msgs SMTP | 85,3% | 1,5% | 71,4% | 29,2% |
| Nº Msgs SOCKS | 13,5% | 75,3% | 23,2% | 50,3% |
| Nº Msgs HTTP | 1,2% | 23,2% | 5,4% | 20,5% |
| Nº Msgs Xbl | 81,9% | 1,7% | 68,6% | 20,0% |
| Nº Msgs Pbl | 2,8% | 1,4% | 52,0% | 76,8% |
| Nº Msgs No Bl | 17,2% | 97,2% | 15,0% | 11,7% |
| Nº IPs Xbl | 83,5% | 13,2% | 64,7% | 16,5% |
| Nº IPs Pbl | 5,2% | 12,9% | 86,6% | 89,5% |
| Nº IPs No Bl | 15,2% | 77,7% | 7,1% | 8,8% |
| Volume (TB) | 1,2 | 7,55 | 0,36 | 4,98 |
| Nº IPs SMTP | 97,43% | 84,62% | 99,19% | 89,01% |
| Nº IPs SOCKS | 2,67% | 15,27% | 0.82% | 11,12% |
| Nº IPs HTTP | 0,33% | 9,52% | 0,02% | 3,44% |

A tabela 5 exibe as propriedades de cada um dos quatro grupos gerados. O grupo 1 reúne 64% dos ASes, sendo que a maioria deles tem um número muito pequeno de endereços IP. A maior parte das máquinas desse grupo utiliza o protocolo SMTP e estão na XBL. Já o grupo 2, apesar de também ter um número pequeno de endereços IP, é responsável pela maior parte das mensagens de *spam* enviadas (65%). Além disso, mais de 98% das mensagens são enviadas utilizando os protocolos SOCKS e HTTP, apesar de grande parte dos endereços IP (84%) utilizar o protocolo SMTP para enviar *spam*. A grande maioria das mensagens (97%) foi enviada por máquinas que não estavam em nenhuma *blacklist*. O grupo 3 se diferencia dos demais por ter mais de 99% dos endereços IP enviando mensagens pelo protocolo SMTP e ter grande parte deles na PBL e XBL. Por último, o grupo 4, reúne os ASes que têm um número elevado de máquinas. A maioria de seus endereços IP estão na PBL, e apenas um número menor está na XBL. Apesar da

[3] *blacklist* que registra o endereço IP de máquinas infectadas por algum *malware*

[4] Média de dias, no período avaliado, em que as máquinas do grupo ficaram ativas.

maior parte das máquinas ter usado SMTP, o maior volume de *spam* foi enviado usando SOCKS.

Se considerarmos as vizinhanças que mais enviaram mensagens de *spam*, estudadas na seção 4.2, vemos que o agrupamento consegue colocar os ASes com características semelhantes em um mesmo grupo, e separar aqueles com comportamentos diferentes. Os sistemas autônomos 10297, 29802 e 2497, cujas máquinas se comportam como servidores dedicados, ficaram no grupo 2, responsável pela maior parte do tráfego de *spam*, mesmo tendo poucos endereços IP. Além disso, esse grupo tem poucas máquinas em *blacklist*, uma característica necessária à máquinas que enviam um grande volume de mensagens.

A maioria das máquinas dos sistemas autônomos 3462 e 4134 se comportam como *bots* e essas duas vizinhanças fazem parte do grupo 4. Esse grupo inclui os ASes que têm um número grande de endereços IP e que têm a maioria deles em *blacklists*. Observa-se também que, a maioria dos endereços IP desse grupo envia uma quantidade pequena de mensagens.

Ao analisar as 15 vizinhanças destacadas na seção 4.2, verificamos que nenhuma delas está no grupo 1 ou 3, como podemos verificar na tabela 4. Esse resultado já era esperado, uma vez que os ASes no grupo 1 têm o número de endereços IP muito reduzido e os ASes do grupo 3, além de terem poucas máquinas, enviam poucas mensagens de *spam*. Desta forma, as vizinhanças que mais enviam *spam* foram alocadas para os outros dois grupos: dos ASes que contêm um número elevado de endereços IP ou daqueles que enviam uma grande quantidade de mensagens de *spam*.

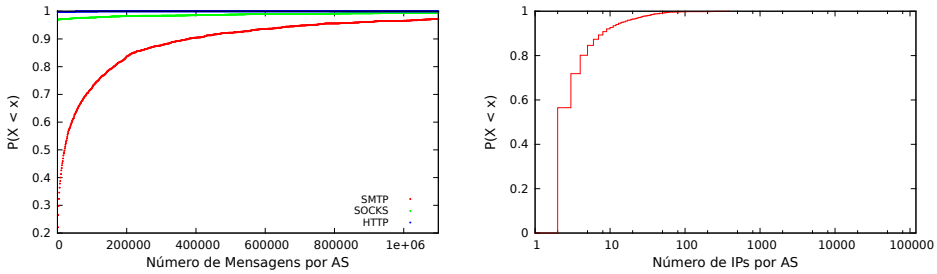
**Tabela 6. Grupos dos AS que mais enviam *spam***

| AS | Classificação | Grupo | AS | Classificação | Grupo |
|---|---|---|---|---|---|
| 10297 | hosting/colocation | 2 | 27699 | dsl/business | 4 |
| 3462 | dsl/provedor de Internet | 4 | 18881 | colocation/provedor | 4 |
| 29802 | hosting | 2 | 8167 | clouding/provedor | 2 |
| 9299 | dsl/business | 2 | 4837 | - | 4 |
| 2497 | hosting/clouding | 2 | 9924 | - | 4 |
| 4134 | dsl | 4 | 28573 | provedor | 4 |
| 6648 | dsl/business | 4 | 4230 | hosting/provedor | 4 |
| 4725 | clouding/business | 4 | | | |

### 4.3.1. Grupo 1

O gráfico da figura 2(a) mostra que uma porcentagem muito pequena das vizinhanças deste grupo utilizam os protocolos SOCKS e HTTP para enviar mensagens de *spam* e que mais de 70% dos ASes enviam menos de 100 mil mensagens em todo período. Apesar de não ser um número muito grande, uma vez que coletamos por quase um ano, ao observar a tabela 5 notamos que esse grupo tem um número muito pequeno de endereços IP, resultando em um número de mensagens por endereço IP relativamente alto.

A principal característica dos ASes desse grupo é o baixo número de máquinas, como podemos ver pela figura 2(b). Quase 60% dos sistemas autônomos têm apenas um único endereço IP e nenhum deles tem mais de cem endereços IP. Isso explica o fato desse grupo, mesmo englobando 64% dos AS, ser responsável por apenas 9,5% do tráfego de *spam* gerado. Além disso, a maioria dos endereços IP desse grupo estão na XBL, o que caracteriza máquinas infectadas, provavelmente pertencentes a *botnets*.

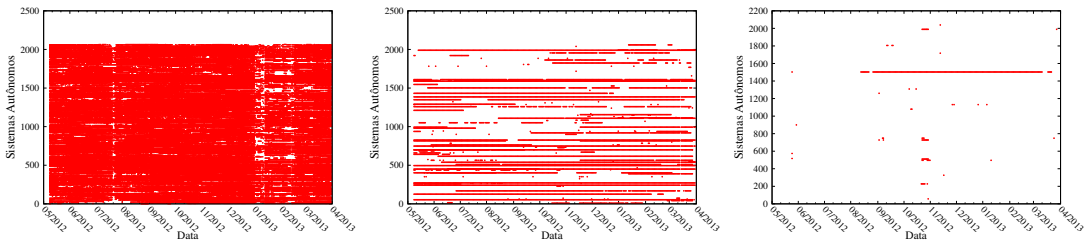



(a) CDF do número de mensagens enviadas por AS

(b) CDF do número de IPs por AS

**Figura 2. CDFs para análise do grupo 1**

O período de atividade das máquinas nos ASes desse grupo mostram atividade constante utilizando o protocolo SMTP em quase todos os ASes, como apresentado na figura 3(a). Por outro lado, o gráfico da figura 3(c) mostra que poucos ASes utilizam o protocolo HTTP e, em apenas um, esse protocolo esteve ativo por um período maior que dez dias. Essa grande quantidade de mensagens enviadas pelo protocolo SMTP, juntamente com um número alto de endereços IP na XBL, sugere a existência de *bots*. Como há poucos endereços comprometidos nesses ASes, é possível que essas máquinas constituam exceções na política de segurança de um sistema em geral seguro e que, por algum motivo, passem despercebidas à gerência dessas redes.




(a) SMTP

(b) SOCKS

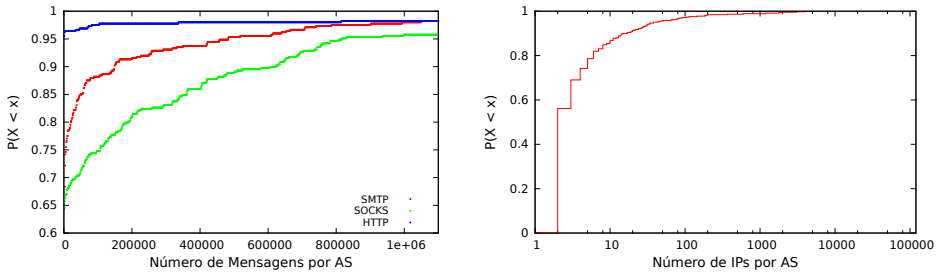
(c) HTTP

**Figura 3. Período de atividade dos ASes do grupo 1, por protocolo**

### 4.3.2. Grupo 2

Este grupo contém os ASes que mais enviam mensagens de *spam*, sendo responsáveis por mais de 65% de todas as mensagens. Como mostrado na figura 4(a), 5% dos ASes estão associados a um número maior que um milhão de mensagens e são eles os responsáveis pela maior parte do tráfego de *spam*. Neste grupo, quase nenhuma mensagem é enviada pelo protocolo SMTP, uma vez que 98% das mensagen são enviadas pelos protocolos SOCKS e HTTP.

Os sistemas autônomos desse grupo também têm um número muito baixo de endereços IP, pois cerca de 57% deles contêm um único endereço IP. Além disso, uma porcentagem muito pequena dessas vizinhanças têm um número de máquinas superior a cem, apenas 2%. Porém, mesmo tendo um número baixo de endereços IP, a quantidade média de mensagens de *spam* enviadas por cada um deles é muito grande, superior a 162



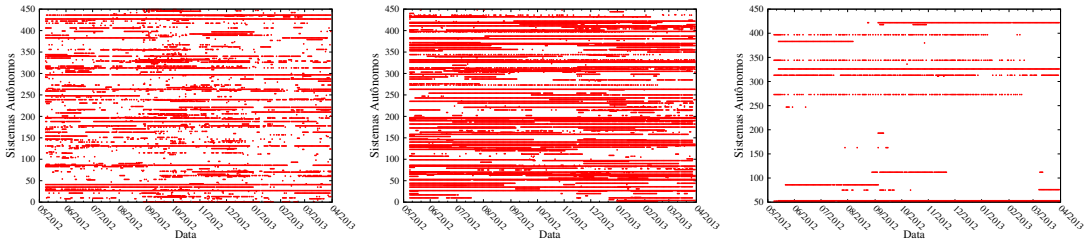

(a) CDF do número de mensagens enviadas por AS

(b) CDF do número de IPs por AS

**Figura 4. CDFs para análise do grupo 2**

mil, como podemos verificar na tabela 5. Um outro aspecto interessante é que, nesse grupo, um número muito grande de endereços IP não fazem parte de nenhuma *blacklist*. Todas as características encontradas para esse grupo sugerem que a maior parte dos ASes abrigam máquinas que funcionam como servidores dedicados ao envio de *spam*, provavelmente com a conivência dos administradores, já que não aparecem em *blacklists*.

Se comparado aos demais grupos, o período de atividade dos ASes utilizando os protocolos SOCKS e HTTP são maiores para esse grupo. Podemos ver que existe um número maior de ASes que utilizam esses protocolos, além de ficarem ativos por um período maior, como os gráficos da figura 5 mostram claramente.




(a) SMTP

(b) SOCKS

(c) HTTP

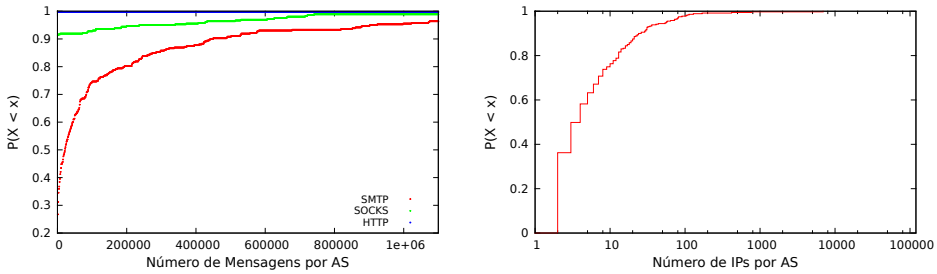
**Figura 5. Período de atividade dos ASes do grupo 2, por protocolo**

Como mencionado anteriormente, os AS 10297, 29802 e 2497 foram assinalados a esse grupo. Como outros do grupo que foram estudados, esses ASes se caracterizam por oferecerem serviços de *hosting* e *co-location*.

### 4.3.3. Grupo 3

Pela tabela 5, podemos ver que, nesse grupo, mais de 70% das mensagens de *spam* são enviadas utilizando o protocolo SMTP e que os ASes do grupo são responsáveis por apenas 2,2% do total das mensagens. O gráfico da figura 6(a) mostra que mais de 80% das vizinhanças enviam menos de 200 mil mensagens, explicando o fato deste grupo ser responsável por um número comparativamente menor de mensagens.

O gráfico da figura 6(b) mostra um comportamento semelhante ao visto nos grupos 1 e 2, mas o número de ASes com apenas um endereço IP é menor, pouco menos de 40%.



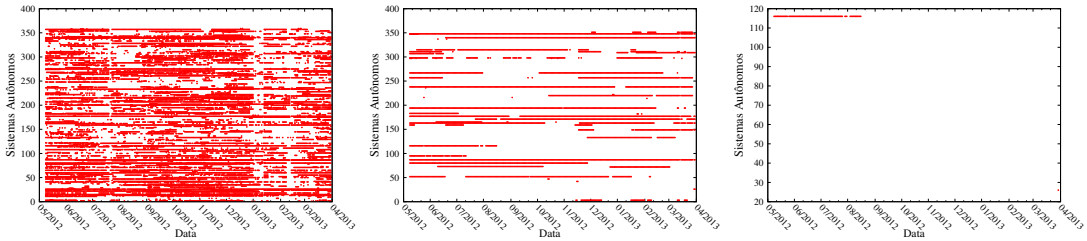

(a) CDF do número de mensagens enviadas por AS

(b) CDF do número de IPs por AS

**Figura 6. CDFs para análise do grupo 3**

O que marca esse grupo é o número muito alto de endereços IP que utilizam o protocolo SMTP, mais de 99% deles, superando qualquer outro grupo. Além disso, cerca de 64% das máquinas desse grupos estão na XBL. Isso sugere a presença de *bots*, mas o baixo número de endereços IP sugere que há menos máquinas comprometidas nesses ASes.

O gráfico da figura 7(c) mostra que apenas um AS desse grupo enviou mensagens de *spam* pelo protocolo HTTP, e mesmo assim por um curto período de tempo. Como era esperado, para o protocolo SMTP, todos os ASes são muito ativos durante todo o período, como pode ser visto na figura 7(a).




(a) SMTP

(b) SOCKS

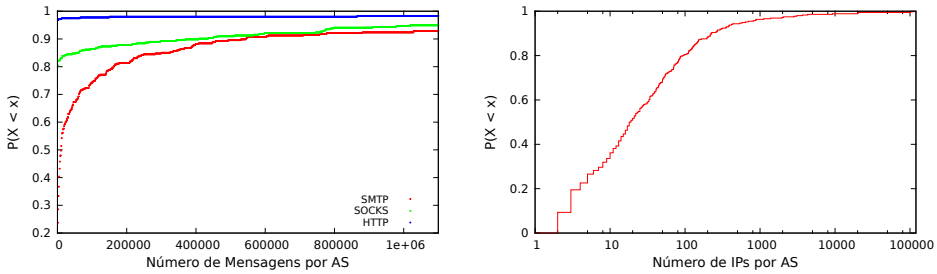
(c) HTTP

**Figura 7. Período de atividade dos ASes do grupo 3, por protocolo**

### 4.3.4. Grupo 4

Pela figura 8(a) podemos ver que a maioria dos ASes utiliza o protocolo SMTP para enviar mensagens de *spam*, mas os poucos sistemas autônomos que utilizam os protocolos SOCKS e HTTP enviam um número de mensagens maior que um milhão. As vizinhanças desse grupo são responsáveis por grande parte das mensagens de *spam*, aproximadamente 23% do total.

Esse grupo contém os ASes com o maior número de máquinas observadas, como podemos verificar na figura 8(b), com mais de 20% das vizinhanças com mais de 1000 endereços IP, em que algumas delas têm o número de máquinas superior a 100 mil. Dessa forma, mesmo sendo responsável por boa parte do tráfego de *spam*, o número de mensagens por endereço IP é o menor entre os grupos, apenas 3 mil. Além disso, a grande maioria dos endereços estão em *blacklists* e utilizam o protocolo SMTP. Por tudo isso, temos fortes indícios de que muitas das máquinas pertencentes a esse grupo fazem parte



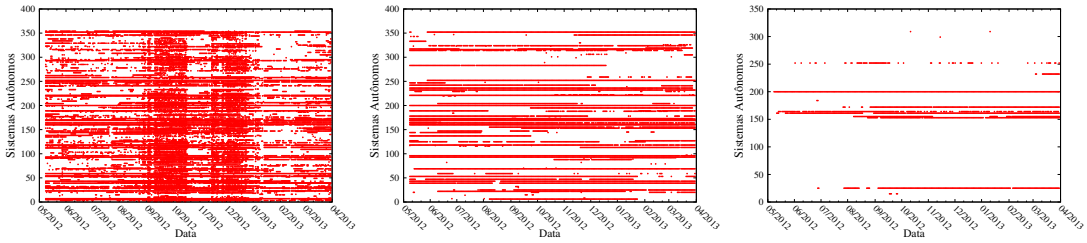

(a) CDF do número de mensagens enviadas por AS

(b) CDF do número de IPs por AS

**Figura 8. CDFs para análise do grupo 4**

de *botnets*. Pelo grande número de máquinas nessa situação, esses ASes se classificam na categoria de vizinhanças ruins, onde aparentemente as políticas de gerência e manutenção da rede não são capazes de evitar a proliferação de máquinas infectadas.

Como os ASes desse grupo têm um número muito grande de endereços IP, é comum que o mesmo AS utilize os três protocolos diferentes na disseminação de *spam*. Esse comportamento é explicitado pelas figuras 9(b) e 9(c). Já era de se esperar um período de atividade intenso no uso do protocolo SMTP, uma vez que máquinas pertencentes a *botnets* tendem a utilizar esse protocolo. Logo, o gráfico da figura 9(a) reforça as suspeitas sobre as *botnets*.




(a) SMTP

(b) SOCKS

(c) HTTP

**Figura 9. Período de atividade dos AS do grupo 4, por protocolo**

Os ASes 3462 e 4134, classificados como sendo de provedores de Internet com redes DSL, fazem parte desse grupo. Isso sugere que, realmente, a composição desse grupo é predominantemente de máquinas domésticas de usuários, infectadas por algum tipo de *malware*.

## 5. Conclusão

Vários esforços estão sendo aplicados na tentativa de combater o envio de *spam*, mas essa tarefa vem sendo dificultada devido à sofisticação das técnicas dos *spammers*. Este trabalho mostra que, apesar das mensagens de *spam* serem enviadas a partir de diversas redes, essa disseminação está concentrada em poucos sistemas autônomos e isso pode ser usado para direcionar os esforços na detecção de *spam*. Além disso, conseguimos agrupar os ASes em quatro categorias representativas, onde cada um dos grupos reúne os ASes com características de disseminação de *spam* semelhantes. Mostramos ainda que alguns

dos grupos definem ASes que são boas vizinhanças, uma vez que eles têm um número pequeno de endereços IP e enviam poucas mensagens de *spam*.

Como trabalhos futuros, pretendemos realizar análises mais profundas sobre cada uma das categorias encontradas, para entender melhor todas as diferenças entre os grupos. Também queremos entender melhor o comportamento da categoria de ASes considerada como sendo uma boa vizinhança e verificar se são as políticas de segurança utilizadas que definem o comportamento desses sistemas autônomos.

## Agradecimentos

Este trabalho foi parcialmente financiado por NIC.Br, Fapemig, CAPES, CNPq e InWeb.

## Referências

Gomes, L. H., Almeida, R. B., Bettencourt, L. M. A., Almeida, V., and Almeida, J. M. (2005). Comparative Graph Theoretical Characterization of Networks of Spam and Legitimate Email. In *Proceedings of the Second Conference on Email and Anti-Spam - CEAS 2005*, Stanford, CA, USA. CEAS.

Guerra, P. H. C., Guedes, D., Wagner Meira, J., Hoepers, C., Chaves, M. H. P. C., and Steding-Jessen, K. (2010). Exploring the spam arms race to characterize spam evolution. In *Proceedings of the 7th Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, Redmond, WA.

Kokkodis, M. and Faloutsos, M. (2009). Spamming botnets: Are we losing the war? ? In *Proceedings of the 6th Conference on e-mail and anti-spam (CEAS)*.

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

Moreira Moura, G. C., Sadre, R., and Pras, A. (2011). Internet bad neighborhoods: the spam case. In Festor, O. and Lupu, E., editors, *7th International Conference on Network and Services Management (CNSM 2011), Paris, France*, pages 1–8, USA. IEEE Communications Society.

Newman, M. E. J., Forrest, S., and Balthrop, J. (2002). Email networks and the spread of computer viruses. *Phys. Rev. E*, 66:035101.

Orman, H. (2013). The compleat story of phish. *Internet Computing, IEEE*, 17(1):87–91.

Pathak, A., Hu, Y. C., and Mao, Z. M. (2008). Peeking into spammer behavior from a unique vantage point. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET'08, pages 3:1–3:9, Berkeley, CA, USA. USENIX Association.

Sipior, J. C., Ward, B. T., and Bonner, P. G. (2004). Should spam be on the menu? *Commun. ACM*, 47(6):59–63.

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

van Wanrooij, W. and Pras, A. (2010). Filtering spam from bad neighborhoods. *International Journal of Network Management*, 20(6):433–444.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 14*

## *Segurança de Redes e Sistemas*

# Siot – Defendendo a Internet das Coisas contra Exploits

**Fernando A. Teixeira**[1], **Fernando Pereira**[1], **Gustavo Vieira**[1], **Pablo Marcondes**[1], **Hao Chi Wong**[2], **José Marcos S. Nogueira**[1], **Leonardo B. Oliveira**[1]

[1]Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG, Brazil

{fateixeira,fernando,gvieira,pablom,jmarcos,leob}@dcc.ufmg.br

Intel Corporation – Santa Clara, CA

hao-chi.wong@intel.com

***Abstract.*** *The Internet of Things (IoT) demands tailor-made security solutions. Today, there are a number of proposals able to meet IoT's demands in the context of attacks from outsiders. In the context of insiders, however, this does not hold true. Existing solutions to deal with this class of attacks not always take into consideration the IoT's idiosyncrasies and, therefore, they do not produce the best results. This work aims at coming up with tailor-made security schemes for thwarting attacks from insiders in the context of IoT systems. Our solution, SIoT, makes use of a pioneering solution to pinpoint vulnerabilities: we cross-check data from communicating nodes. SIoT provides the same guarantees as traditional security mechanisms, but it is about 83% more efficient, according to the experiments that this article describes.*

***Resumo.*** *A Internet das Coisas (IoT – Internet of Things) requer soluções de segurança feitas sob medida. Embora haja propostas que tratem ataques externos de forma satisfatória, o mesmo não ocorre para ataques internos. Sobre esses últimos, as propostas existentes nem sempre consideram as características de IoT e, portanto, não lhes são adequadas. O objetivo deste trabalho é conceber uma solução para proteger sistemas de IoT contra ataques internos. Nossa solução, SIoT, emprega uma abordagem pioneira na busca de vulnerabilidades: o cruzamento de dados de diferentes nós da rede. SIoT provê as mesmas garantias que mecanismos de segurança tradicionais, porém é cerca de 83% mais eficiente, de acordo com os experimentos que este artigo descreve.*

## 1. Introdução

A Internet das Coisas (IoT – *Internet of Things*) é uma infraestrutura de rede dinâmica e global com capacidades de autoconfiguração, baseada em protocolos de comunicação padronizados e interoperáveis, onde "coisas" físicas e virtuais têm identidades, atributos físicos e personalidades virtuais. Usam interfaces inteligentes bem como são naturalmente integradas à Internet [Atzori et al. 2010]. Na IoT, as *"coisas"* ou objetos devem se tornar participantes ativos em processos de negócio, informacionais e sociais, onde serão capazes de interagir e comunicar entre elas mesmas, trocar informações coletadas do ambiente, reagindo autonomamente aos eventos do mundo físico real, bem como influenciar esse contexto sem intervenção direta do ser humano.

As interfaces na forma de serviço facilitam as interações entre essas *"coisas inteligentes"* via Internet e permitem a consulta de informações e troca de estados associados às "coisas". Mas questões como segurança e privacidade devem ser consideradas. É nesse contexto que se insere o presente trabalho: a comunicação necessária entre as "coisas" ou objetos, efetivamente realizadas através das interfaces de rede, abre possibilidades de ataques externos ou internos de segurança. Os atacantes buscam então explorar vulnerabilidades para poderem influenciar o comportamento ou obter o controle do sistema.

A Internet das Coisas (IoT – Internet of Things) requer soluções de segurança feitas sob medida [Wangham et al. 2013]. Isso porque, ao contrário de um computador tradicional, os elementos de IoT usualmente: (i) são dotados de menor capacidade de processamento, memória e energia [Atzori et al. 2010]; (ii) realizam tarefas de forma colaborativa [Atzori et al. 2010]; e (iii) executam sistemas eminentemente desenvolvidos usando a linguagem C ou linguagens nela baseadas (TinyOS [Levis et al. 2004], ContikiOS [Dunkels et al. 2004] e Linux embarcado, por exemplo). Entretanto, as propostas de segurança existentes nem sempre levam em conta tais características e, portanto, nem sempre são adequadas para a IoT.

Em relação a ataques disparados por adversários externos (*outsiders*), por exemplo, já existem propostas que atendem a IoT satisfatoriamente ( [Oliveira et al. 2006, Simplício et al. 2010], por exemplo). Algumas delas são capazes de garantir propriedades como sigilo e autenticação de mensagens a um custo aceitável mesmo para a IoT [Karlof and Wagner 2003]. Assim, ataques de espionagem, personificação (*spoofing*) e retransmissão (*replay*) podem ser eficientemente evitados [Karlof and Wagner 2003].

No entanto, para algumas questões de segurança ainda não há propostas personalizadas para IoT. Um exemplo são alguns ataques disparados por adversários internos (*insiders*), ou seja, por aqueles cujas credenciais de acesso são válidas. As mensagens trocadas por esses adversários são legítimas na medida em que elas são autenticadas com sucesso pelos seus destinatários. Isso, contudo, não as torna menos maliciosas. Essas mensagens objetivam, por exemplo, explorar vulnerabilidades de sistemas para então obter o seu controle; ou seja, elas são, no fundo, *exploits* [Chess and West 2007]. Para essa classe de ataques, ainda não há soluções plenamente adequadas para o contexto de IoT.

**Objetivo.** O objetivo deste trabalho é o estudo da questão de segurança de IoT e proposição de soluções que envolvem algoritmos, técnicas e mecanismos para proteção dos módulos de IoT contra ações de exploração de vulnerabilidades de código (*exploits*). Mais precisamente, ela objetiva conceber uma solução de segurança de código para IoT capaz de defender seus sistemas contra ataques internos à rede, em especial contra Buffer Overflows. Nossa proposta, *SIoT* (*Secure IoT*), emprega uma abordagem pioneira na busca de vulnerabilidades: o cruzamento de dados de diferentes módulos de um mesmo sistema distribuído[1]. Esse cruzamento de dados resulta em mais informação e, portanto, maior precisão. Tal precisão é traduzida em eficiência. Até onde sabemos, não há propostas que usem técnicas de cruzamento de dados de diferentes módulos do sistema para este mesmo fim.

**Contribuição.** As principais contribuições deste trabalho são as seguintes. **(i)** Indicação das características de IoT que mais influenciam no projeto de soluções de

[1] Aqui, módulos são as partes do sistema distribuído que são executadas por diferentes nós da rede.

segurança; **(i)** Projeto de uma solução personalizada para IoT, que utiliza uma técnica inédita para apontar vulnerabilidades. **(iii)** Concepção de um algoritmo capaz de unir, eficientemente, grafos de diferentes módulos de um sistema. **(iv)** Desenvolvimento de uma ferramenta disponível *online*[2] que implementa nossa solução. **(v)** Avaliação da solução utilizando o *ContikiOS* [Dunkels et al. 2004], um sistema operacional para a IoT.

Acerca deste último ponto, em especial, utilizamos a SIoT para proteger as aplicações `udp-ipv6` e `webser/wget` do ContikiOS contra ataques de *Buffer Overflow*. Os resultados indicam que para um mesmo nível de segurança, a SIoT se mostra pelo menos 83% mais eficiente que por propostas hoje amplamente utilizadas.

**Organização do texto.** A Seção 2 discorre sobre as características de IoT que mais impactam a segurança. O modelo de ataque e os conceitos de segurança de código utilizados no artigo são apresentado na Seção 3. A Seção 4 descreve nossa solução. A avaliação da solução via um estudo de caso e resultados é apresentado na Seção 5. Os trabalhos relacionados são discutidos na Seção 6 e as conclusões na Seção 7.

## 2. IoT: Características & Segurança

A IoT se difere das redes de computadores tradicionais em inúmeros aspectos [Atzori et al. 2010]. Por exemplo, sua escala é maior, sendo composta por centenas ou milhares de nós. Os avanços na miniaturização permitem que as coisas menores tenham cada vez mais a capacidade de interagir e se conectar [Wangham et al. 2013].

Contudo, são as características que impactam a segurança que mais nos importam (Tabela 1). Neste contexto, é na escassez de recursos que reside a diferença chave entre IoT e redes tradicionais [Wangham et al. 2013]. Idealmente, os nós da IoT são compostos por Sistemas Embarcados (SEs) de baixo custo monetário, a fim de reduzir o valor final de produtos. Dessa forma, SEs possuem poucos recursos computacionais (processamento, memória, energia, etc.) e são incapazes de executar sistemas "pesados". Para mitigar essa questão, sistemas para IoT são frequentemente desenvolvidos em linguagens de programação eficientes, tal como C.

O problema é que sistemas desenvolvidos em C são inerentemente mais inseguros [Chess and West 2007]. A linguagem C torna os sistemas mais "leves" e, por isso, mais apropriados a SEs. No entanto, tal eficiência tem preço: sistemas desenvolvidos em C são comumente mais vulneráveis. Uma das razões da eficiência de C está no fato de que a linguagem não realiza certas verificações que Java, por exemplo, realiza[3]. Uma dessas verificações é a de limites de arranjos; e é justamente aí que surge o perigo do ataque de *Buffer Overflow – BOF* [Chess and West 2007], um dos ataques mais comuns a sistemas computacionais. E mecanismos de segurança para mitigar ataques de BOF presentes em PCs[4], muitas vezes não estão presentes em SEs.

Outro fator que impacta a segurança de forma preponderante é que tarefas na IoT são usualmente realizadas de forma colaborativa [Atzori et al. 2010]. Em virtude disso, a natureza dos sistemas executados é alterada, isto é, eles passam a ser eminentemente distribuídos. Isso, naturalmente, incrementa sua complexidade o que, por sua vez, torna a

[2]https://bitbucket.org/ecosoc/siot

[3]Formalmente, a ausência dessas verificações é porque C é fracamente tipada (*weakly typed*).

[4]Por exemplo o NOEXEC `http://pax.grsecurity.net/docs/noexec.txt`

| Características | Redes Tradicionais | IoT |
|---|---|---|
| Escala | dezenas ou centenas de nós | centenas ou milhares de nós |
| Linguagem de Programação | Python, Java, C, C++ e C# | Eminentemente C e C++ |
| Realização Tarefas | independentemente | colaborativamente |
| Recursos Computacionais | "rico" | "pobre" |
| Recurso Crítico | tempo ou memória | energia |
| Arquitetura dos Nós | 64,32 *bits* | 8, 16, ou 32 *bits* |

**Tabela 1. Redes Tradicionais versus IoT: aspectos que impactam a segurança.**

busca por vulnerabilidades mais desafiadora. Na medida em que os nós colaboram, eles também trocam mais mensagens e, portanto, aumenta-se a janela para potenciais ataques.

Por fim, a arquitetura dos nós de IoT também influência a segurança. Diferentemente de PCs, em que hoje a arquitetura padrão é de 64 *bits*, SEs são geralmente baseadas em arquiteturas de de 8, 16, ou 32 *bits*. Essa diferença altera a forma com a qual o *hardware* executa um sistema e, portanto, impacta a segurança [Chess and West 2007].

## 3. Conceitos Fundamentais

Nessa seção, descrevemos nossas premissas e modelo de ataque (Seção 3.1) e apresentamos os conceitos de segurança de sistemas (Seção 3.2) utilizados no artigo.

### 3.1. Premissas & Modelo de Ataque

Grande parte de ataques externos podem ser evitados com o uso de cifras (*ciphers*) e códigos de autenticação de mensagens [Karlof and Wagner 2003]. Assim, já há na literatura diversas propostas capazes de proteger uma rede contra essa classe de ataques – por exemplo, SSL/TLS[5] para comunicação fim-a-fim em redes tradicionais; e, por exemplo, o NEKAP [Oliveira et al. 2006], que fornece segurança na camada de enlace para redes *ad hoc* com escassez de recursos como a IoT. Neste modelo, quando uma mensagem é recebida, verifica-se sua validade e ela só é tratada pelo sistema caso a verificação seja bem sucedida. Tais propostas dificultam a ação de adversários que busquem forjar as mensagens do sistema. Nosso trabalho pressupõe que esses mecanismos tiveram sucesso e se concentra nos ataques realizados por adversários que conseguiram acesso ao sistema.

Consideramos que o adversário possui acesso às entradas dos dispositivos de rede e, portanto, logra com que outros membros da rede validem suas mensagens [Karlof and Wagner 2003]. Como o adversário possui os mesmos privilégios de um usuário comum ele pode realizar ataques internos e, por isso, a efetividade das propostas citadas no parágrafo anterior é reduzida. É necessário, portanto, que mecanismos de segurança protejam o sistema contra ataques que exploram falhas no código através da manipulação das entradas dos sistema.

Em particular, nossa solução pressupõe que: (i) as mensagens trocadas na rede são autenticadas; e (ii) sistemas executados pelos nós são conhecidos e estão íntegros. Assim, nosso adversário tem acesso às "coisas" como um usuário comum (*ordinary user*), mas não é capaz de trocar seu código ou injetar mensagens falsas diretamente no canal de rede.

[5] `http://tools.ietf.org/html/rfc5246`

Entretanto, o adversário é capaz de entrar com dados que levem o programa a um estado inconsistente e, dessa maneira, até mesmo fazer com as aplicações enviem mensagens espúrias para outros nós da rede. Esse cenário é bastante comum: em geral, os adversários primeiro obtêm privilégios menores para então obterem maiores privilégios. E como as "coisas inteligentes" possuem outros canais de interação com o mundo, como sensores ou controles remotos, é possível que o adversário manipule essas entradas de forma a induzir os 'coisas" a erros. Por exemplo, uma sequência de teclas em um controle remoto de uma TV inteligente pode ser usada para explorar uma falha de código e fazer com que a TV envie mensagens não desejadas na rede ou um isqueiro aceso perto de um sensor de temperatura pode ser usado para explorar uma falha de *integer overflow* no código.

### 3.2. Segurança de Sistemas

A maioria das propostas de defesa para sistemas computacionais são baseadas na Análise Estática [Chess and West 2007], na Análise Dinâmica [Serebryany et al. 2012], ou na combinação de ambas. A concepção dessas propostas, no entanto, é uma tarefa desafiadora já que qualquer propriedade não trivial de linguagens recursivamente enumeráveis é um problema indecidível. Ou, em outras palavras, não existe programa genérico capaz de decidir se um outro programa qualquer é ou não vulnerável.

Na Análise Estática [Chess and West 2007] o sistema é inspecionado antes do programa ser implantado (*deployed*). Por essa razão, é também conhecida como *análise de código*. A vantagem do método é que não há sobrecarga (*overhead*) durante a execução do sistema. Seu aspecto negativo é que a análise não possui informações que só estarão disponíveis em tempo de execução. Não raro tal "desinformação" impossibilita afirmar se há ou não uma vulnerabilidade em certos trechos do sistema. Na dúvida, a análise é conservadora e, assim, presume-se que a vulnerabilidade existe. Esse conservadorismo, por sua vez, traduz-se em falso-positivos[6].

Na Análise Dinâmica [Serebryany et al. 2012], por outro lado, o sistema é, sim, executado. Agora, a técnica pode tirar proveito das informações só disponíveis em tempo de execução. Isso, por um lado, mitiga a questão dos falso-positivos, comuns à Análise Estática. No entanto, seus resultados são pertinentes apenas às entradas testadas e, assim, não se pode tirar conclusões acerca do comportamento geral do programa.

Em razão das naturezas complementares das análises Estática e Dinâmica, é comum o emprego da Análise Híbrida, ou seja, a combinação de ambas. Usualmente, a Análise Estática é primeiro empregada para apontar-se as vulnerabilidades e, posteriormente, a Análise Dinâmica instrumenta o sistema para monitorá-lo em tempo de execução, em seus trechos supostamente vulneráveis. E eis o motivo pelo qual falso-positivos acarretam sobrecarga. Embora eles não apresentem perigo, eles são monitorados em tempo de execução, o que resulta em sobrecarga. Portanto, é fundamental para a eficiência de um sistema que os falso-positivos sejam descobertos e eliminados.

Diversos ferramentais são utilizados durante a análise de sistemas. Dentre elas, duas são especialmente importantes no contexto deste trabalho: os Grafos de Fluxo de Controle (CFG) e os Grafos de Dependências [Chess and West 2007]. O CFG é um grafo dirigido que representa as possíveis sequências de instruções que são processadas durante

[6]Posteriormente, mostraremos porque falso-positivos são particularmente indesejáveis na IoT

a execução de um programa. Um vértice do CFG é chamado um bloco básico. Um bloco básico é um conjunto maximal de instruções que sempre são executadas em sequência. Assim, um bloco básico possui um desvio, seja ele condicional ou não, somente no final da lista de suas instruções constituintes. Existe uma aresta entre um bloco básico $b_1$ e um bloco básico $b_2$ se, e somente se, o programa pode fluir de $b_1$ para $b_2$. O grafo de dependência, por sua vez, possui um nó para cada variável e cada operação do programa. O grafo de dependências também é direcionado. Nele, há uma aresta entre cada variável $u$ e uma instrução $i$ se $i$ representa uma instrução que usa $u$.

## 4. Siot

A fim de prover ao analisador estático uma visão holística do programa sob análise, este artigo introduz o conceito de 'Grafo de Controle de Fluxo de Sistemas Distribuídos', descrito na Seção 4.1. Em seguida, mostramos a arquitetura da Siot na Seção 4.2.

### 4.1. Grafo de Controle de Fluxo de Sistemas Distribuídos

Grafos de fluxo de controle têm, por quase cinco décadas, desfrutado de um papel central na análise de programas. Por outro lado, neste artigo clamamos que essa representação não é suficientemente expressiva para analisar programas distribuídos. A fim de contornar essa limitação, propomos uma forma de criar um Grafo de Fluxo de Controle Distribuído (*Distributed Control Flow Graph* – DCFG). Esse grafo conecta, em uma única representação, os CFGs dos vários programas que constituem uma aplicação distribuída.

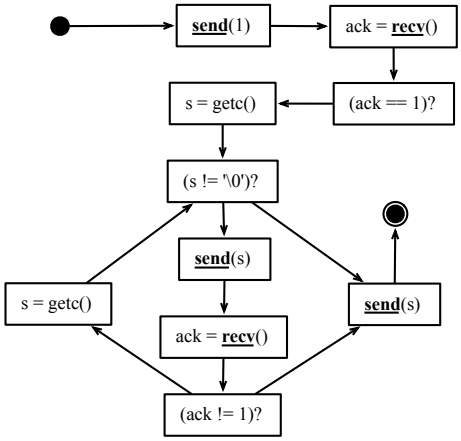
Cada módulo de um sistema distribuído possui seu próprio CFG. A Figura 1 mostra um sistema *echo*, com módulos cliente e servidor e seus respectivos CFGs. Na linha 5 do módulo servidor (Figura 1b) temos um exemplo de uma vulnerabilidade. Nessa linha, o arranjo `msg` recebe dados da rede via um `recv`. Esse `recv` recebe dados do segundo `send` do módulo cliente (linha 6 da Figura 1a). E esse `send`, por sua vez, depende do `getc` (linha 4) desse mesmo módulo. Logo há um caminho entre o usuário e o arranjo `msg`, o que o torna vulnerável.

Mas também temos um exemplo de um falso-positivo nesse sistema. A primeira vista, se analisarmos apenas o módulo servidor (Figura 1b) concluímos que o arranjo `msg` é vulnerável por depender do primeiro `recv` (linha 1 da Figura 1b). Mas ao analisar o módulo cliente (Figura 1a) verifica-se que esse `recv` só recebe informações do primeiro `send` (linha 1 da Figura 1a). Entretanto, esse `send` envia apenas uma constante (`send(1)`). Portanto, não há dependências com o usuário. Deste exemplo, concluímos que a análise de um sistema distribuído pode beneficiar-se de um DCFG.

Interessa-nos, portanto, uma maneira de unir CFGs de módulos individuais, de maneira tal que as análises já descritas na literatura possam ser aplicadas sobre o DCFG resultante sem modificações. A solução trivial desse problema é unir todos os `sends` com todos os `recvs` de um sistema. Entretanto, se assim o procedermos várias arestas desnecessárias serão incluídas. São desnecessárias as arestas que ligam um send $s_A$ de um módulo A a um receive $r_B$ de um módulo B sem no entanto haver um cenário que ao mesmo tempo leve o módulo A a $s_A$ e o módulo B a $s_B$. Vejamos novamente a Figura 1. Como exemplo, podemos citar que é desnecessário adicionar uma aresta entre o `A:send` e o `H:recv` porque a mensagem enviada por A será recebida por `F:recv` e não por `H:recv`. Como não há um caminho que permita que a mensagem enviada pelo

(a)

```
1  send(1);
2  ack = recv()
3  if (ack == 1) {
4      s = getc();
5      while (s != '\0') {
6          send(s)
7          ack = recv();
8          if (ack != 1) {
9              break;
10         } else {
11             s = getc();
12         }
13     }
14     send(s);
15 }
```




(b)

```
1  msg = recv();
2  if (msg == 1) {
3      send(1);
4      do {
5          msg = recv();
6          putc(msg);
7          if (msg != '\0')
8              send(1);
9          else
10             break;
11     } while (1);
12 } else {
13     send(0);
14 }
```

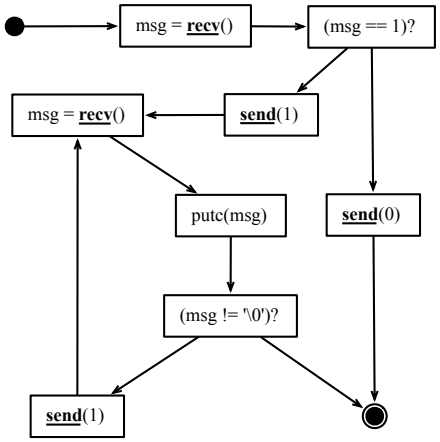



**Figura 1. Grafo de fluxo de controle de dois módulos do sistema *Echo*. (a) Módulo cliente. (b) Módulo servidor *echo*.**

`A:send` chegue a `H:recv`, uma aresta inserida entre esses nodos seria desnecessária. Para resolver esse problema, propomos um algoritmo que leva em consideração o CFG de cada módulo e como esses módulos interagem via rede de forma a reduzir a quantidade de arestas desnecessárias.

Com o objetivo de encontrar um DCFG melhor que a solução trivial, introduzimos a noção de nível. A Equação 1 define os níveis de um `send` ou `recv` no grafo como segue:

$$\begin{array}{rcl} \mathit{nível}(cfg, 0) & = & \{\mathit{root}\} \\ \mathit{nível}(cfg, n) & = & \{v \mid \overrightarrow{uv} \in cfg \wedge u \in \mathit{level}(cfg, n-1)\} \end{array} \quad (1)$$

Em outras palavras, consideremos inicialmente a definição de níveis dos `sends`. O nível zero ($L_0$) possui a raiz do grafo. O nível um ($L_1$) contém os `sends` sucessores diretos da raiz. O nível dois ($L_2$) contém os `sends` sucessores diretos de algum `send` do nível $L_1$. E assim recursivamente. O algoritmo termina quando alcança o fim do programa ou quando um nível possui o mesmo conjunto de `sends` de um nível anterior. O mesmo raciocínio se aplica aos `recvs`. A Figura 2 mostra os conjuntos de níveis dos `sends` do cliente *echo* (esquerda) e `recvs` do servidor (direita).

A partir da definição dos níveis, concluímos que devemos incluir no grafo uma

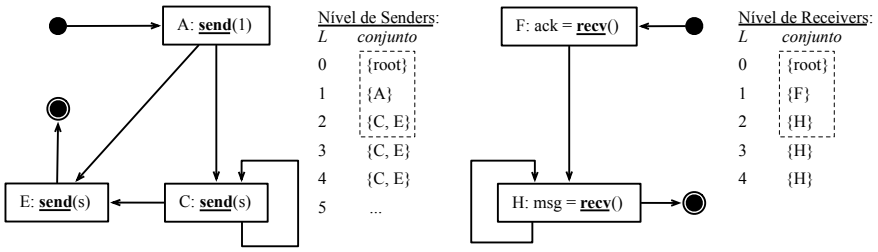



**Figura 2. Conjuntos de níveis dos sends do cliente *echo* (esquerda) e recvs do servidor (direita). A caixa tracejada delimita os níveis distintos.**

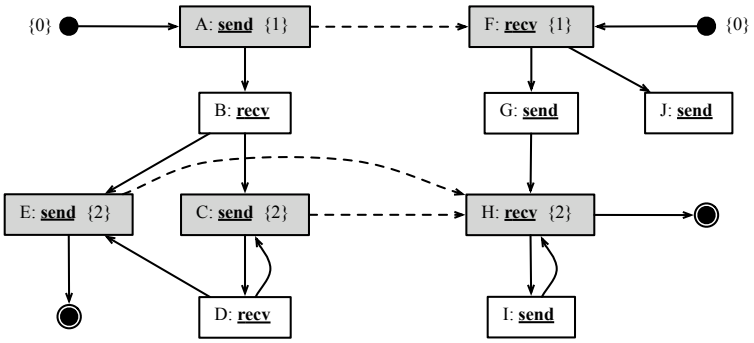



**Figura 3. Ligações entre sends do cliente *echo* e recvs do servidor *echo* do nosso exemplo.**

aresta entre "*A:send(1)*" ($s_A$) e "*F:recv(1)*" ($r_F$) pois esses nós pertencem ao nível $L_0$. Ou seja, teremos uma aresta $\overrightarrow{s_A r_F}$. Também devemos incluir as arestas $\overrightarrow{s_C r_H}$ e $\overrightarrow{s_E r_H}$ pois esses nós pertencem ao nível $L_1$. Não é necessário criar as arestas $\overrightarrow{s_A r_H}$, $\overrightarrow{s_C r_F}$ e $\overrightarrow{s_E r_F}$ porque esses nós pertencem a níveis diferentes. As ligações entre `sends` do cliente e `recvs` do servidor *echo* do nosso exemplo podem ser vistas na Figura 3. Os números próximos aos vértices representam o nível ao qual o nó pertence.

**Grafo de Dependência de Sistemas Distribuídos.** A partir do DCFG, podemos criar um grafo de dependências distribuído. Tal grafo pode ser construído a partir da união dos grafos de dependências individuais de cada módulo, via uma estratégia semelhante àquela que usamos para unir CFGs. Cada programa do sistema possui seu próprio grafo de dependência. Para cada instrução de acesso a rede, é criado um vértice no grafo para representar essa operação. O vértice é marcado como um nó de rede de leitura ou de escrita e inserido em uma lista correspondente. Insere-se, então, uma aresta de dependência entre o vértice de rede encontrado e os vértices correspondentes no outro programa de acordo com o DCFG do sistema construído anteriormente. Dessa forma, o grafo de dependências final é a união dos grafos de cada programa. Essa estrutura representa o grafo de dependências do sistema distribuído como se tal sistema fosse um único programa.

**Complexidade.** O algoritmo de conexão de nodos possui um pior caso de $O(2^N)$, sendo $N$ o número de operações de envio ou de recebimento de dados do grafo de protocolo. Para fundamentar esse limite, notamos que o algoritmo termina assim que um nível repetido é encontrado. Entretanto, a quantidade de níveis é finita, e limitada pelo número

possível de subconjuntos de vértices do grafo, que é $2^N$. Vale ressaltar, contudo, que não encontramos um exemplo que validasse esse pior caso. Em todos os nossos exemplos práticos, o algoritmo gerou uma quantidade de níveis inferior ao número de vértices do grafo de protocolo. Por fim, é importante observar que a complexidade da proposta não depende do número de nós da rede. Mas sim do número de programas diferentes que executam nos nós, que é pequena, pois vários executam o mesmo código.

### 4.2. Arquitetura e Implementação

A Siot foi implementada como uma extensão do compilador LLVM[7] [Lattner and Adve 2004] através de um conjunto de passos de otimização. A arquitetura da Siot é apresentada na Figura 4. O processo de análise é constituído por três etapas: (i) Compilação e preparação dos *bytecodes* (*.bc*) dos módulos do sistema – *Merge*; (ii) Análise das instruções do *bytecode* do sistema distribuído – *NetInputDep*, *NetLevel* e *NetDepGraph*; (iii) Análise de arranjos vulneráveis, instrumentação dos módulos e geração de estatísticas/grafo de arranjos vulneráveis – *NetVulArrays*.

Inicialmente, os módulos do sistema são compilados via *clang* (compilador do LLVM) e são gerados *bytecodes* de cada módulo. O passo *Merge* recebe os diversos módulos do sistema distribuído a ser analisado e gera como saída um *bytecode* único. O Merge analisa quais são as possíveis funções de acesso a rede e exibe para o usuário. O usuário determina quais realmente desempenham o papel de envio e recebimento de dados via rede e o Merge atualiza o arquivo de configuração – config.txt – que será utilizado nas próximas execuções. Uma vez definidas as funções de acesso à rede, são inseridas *tags* no *bytecode* para identificá-las. Também são inseridas *tags* para diferenciar cada função e variável global de cada módulo do sistema. Feito isso, é gerado um *bytecode* unificado contendo os módulos do sistema.

Em seguida, o passo *NetInputDep* avalia quais são as entradas do sistema, ou seja, quais instruções interagem com o usuário, com a rede ou com o sistema de arquivos. Nesse passo são geradas listas com todas as entradas do sistema classificadas como dependentes de rede ou não. Em paralelo, o passo *NetLevel* implementa o algoritmo de definição de níveis descrito na seção anterior ( 4.1). Uma vez definidos os níveis de cada interação com a rede e com auxílio das *tags* inseridas pelo Merge, o passo NetDepGraph constrói o Grafo de Dependências Distribuído.

De posse do Grafo de Dependências Distribuído, o passo *NetVulArrays* procura por caminhos entre entradas de um módulo e seus acessos à memória. O NetVulArrays verifica, também, quais são dependentes de rede e, desses, quais dependem de entradas do módulo que envia as informações via rede. Ao final, o NetVulArrays gera: (i) estatísticas de verdadeiros-positivos e falsos-positivos; (ii) grafo dos caminhos considerados vulneráveis; (iii) *bytecode* dos módulos instrumentados.

## 5. Estudo de Caso

BOFs (*Buffer Overflows*) são um dos ataques mais utilizados para tomada de controle de sistemas [Chess and West 2007]. Eles são evitados através de Verificações de Limites de Arranjo (*Array Bounds-Checks* – ABCs) [Chess and West 2007]. Ou seja,

[7]LLVM não é uma sigla, mas sim o nome de uma coleção de tecnologias e ferramentas de compilação modulares e reutilizáveis.

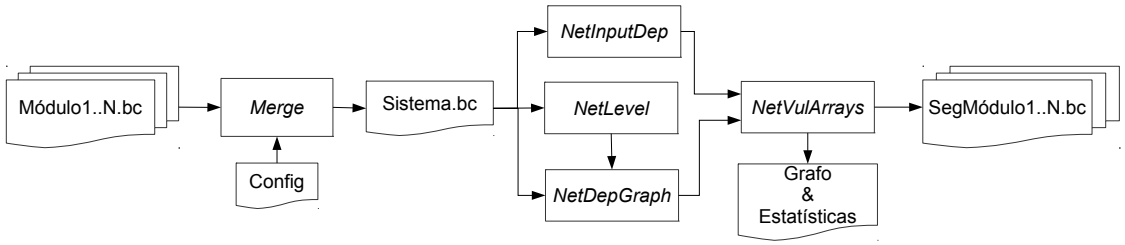



**Figura 4. Arquitetura da Siot.**

**UDP Server**

```
1  static void tcpip_handler(void) {
2    static int seq_id;
3    char buf[MAX_PAYLOAD_LEN];
4    if(uip_newdata()) {
5      ((char *)uip_appdata)[uip_datalen()] = 0;
6      PRINTF("Server received: '%s' from ", (char *)
           uip_appdata);
7      ...
8      uip_ipaddr_copy(&server_conn->ripaddr, &
           UIP_IP_BUF->srcipaddr);
9      PRINTF("Responding with message: ");
10     sprintf(buf, "Hello from the server! (%d)", ++
           seq_id);
11     PRINTF("%s\n", buf);
12     uip_udp_packet_send(server_conn, buf, strlen(buf)
           );
13     ...
14   }
15 }
```

**Figura 5. Trecho do Servidor udp-ipv6 do ContikiOS.**

**UDP Client**

```
1  static void tcpip_handler(void) {
2    char *str;
3    if(uip_newdata()) {
4      str = uip_appdata;
5      str[uip_datalen()] = '\0';
6      printf("Response from the server: '%s'\n", str);
7    }
8  }
9  static void timeout_handler(void) {
10   static int seq_id;
11   printf("Client sending to: ");
12   PRINT6ADDR(&client_conn->ripaddr);
13   sprintf(buf, "Hello %d from the client", ++seq_id);
14   printf(" (msg: %s)\n", buf);
15   uip_udp_packet_send(client_conn, buf,
         UIP_APPDATA_SIZE);
16   ...
17 }
```

**Figura 6. Trecho do Cliente udp-ipv6 do ContikiOS.**

instrumentações no código que checam antes de acessos a memória se um índice está dentro dos limites do arranjo. Até então, contudo, a utilização de ABCs no contexto de IoT era ineficiente. Isso porque as propostas existente são conservadoras e instrumentam um número considerável de falso-positivos. Nesta seção, utilizamos a Siot para proteger sistemas de IoT contra BOFs. Em particular, avaliamos a Siot sobre as seguintes aplicações do ContikiOS: (i) `udp-ipv6` cliente e servidor e (ii) `webserver` e `wget`.

A aplicação udp-ipv6[8] demonstra como construir um cliente e um servidor UDP usando a pilha de protocolos 6LoWPAN[9]. No servidor, (Figura 5), as mensagens são recebidas via rede através da função `uip_newdata()` (linha 4) e são acessadas através do arranjo `uip_appdata`. As ferramentas tradicionais consideram esse arranjo vulnerável e, para protegê-lo, utilizam um ABC que seria acionado sempre que uma mensagem for recebida. No módulo cliente (Figura 6), da mesma forma, um outro arranjo seria protegido por ferramentas tradicionais. Isso porque a função `uip_newdata()` (linha 3) é chamada para verificar se há mensagens recebidas e, em seguida, os dados são transferidos para o arranjo `str` (linha 4).

No entanto, ao se analisar o sistema como um todo – isto é, ambos os módulos –, nota-se que os arranjos supracitados não são de fato vulneráveis. Contudo, as ferramentas tendem a reportar tal situação como uma vulnerabilidade. Esse aviso é um falso positivo.

[8]https://github.com/contiki-os/contiki/tree/master/examples/udp-ipv6

[9]IPv6 over Low power Wireless Personal Area Networks – http://tools.ietf.org/html/rfc6282

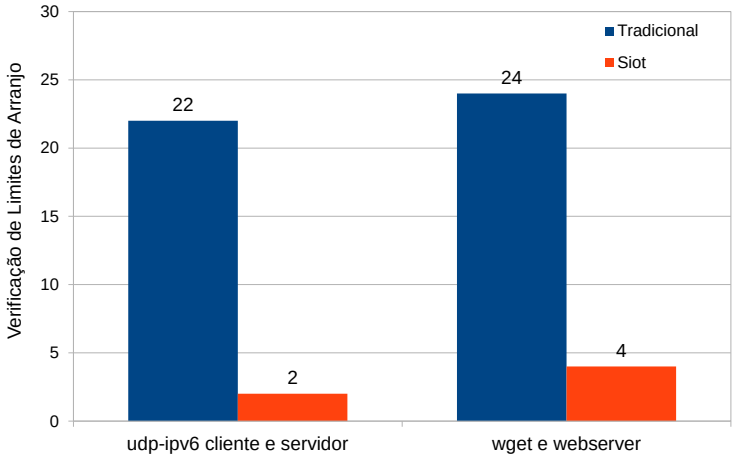



**Figura 7. Verificações de Limites de Arranjos inseridos em cada aplicação.**

Veja, a mensagem que o cliente prepara não possui qualquer dependência com fontes externas de dados (Figura 6, linhas 10 a 14). Ora, se tal mensagem não é vulnerável em sua origem, seguindo nosso modelo de ataque (Seção 3.1), ela também não é vulnerável no destino. No servidor, de forma análoga, a mensagem enviada para o cliente não possui dependência com entradas de dados (Figura 5, linhas de 8 a 11). Em suma, os ABCs que seriam inseridos eram, então, desnecessários.

Ao protegermos as aplicações com a Siot, reduzimos em 91% a quantidade de ABC a serem inseridos na aplicação udp-ipv6 (ou seja, de 22 para 2) e 83% na aplicação webserver/wget (isto é, de 24 para 4), como mostra a Figura 7. A redução é possível pois ao analisar os sistema como todo reduzimos a quantidade de entradas sensíveis e podemos fazer uma análise cruzada de dados. Dessa forma muitos falso positivos são evitados e o número de ABCs pode ser reduzido.

O número de ABCs que conseguimos evitar é significativo principalmente em relação ao custo enérgico de cada ABC inserido e a quantidade de vezes que são acionados. Para cada ABC é necessário incluir seis instruções assembly [Serebryany et al. 2012]. Se consideramos um dispositivo IoT típico com um microcontrolador Atmel AVR de 8 mA, 3V, 7,38 MHz e 1 instrução/clock teremos um gasto de energia de 3,25 mJ/instrução[10]. Portanto, cada ABC incluído aumentará o consumo da aplicação em 19,5 mJ. Se consideramos que uma aplicação udp-ipv6 envia e recebe uma mensagem por minuto teremos ao final de uma dia um consumo de energia de 618 mJ adicionais na abordagem tradicional versus apenas 56 mJ na Siot. A Tabela 3 mostra o impacto dessa economia em termos práticos. Se considerarmos uma rede com cem dispositivos executando essa aplicação durante uma semana, a economia gerada equivale a um quinto

[10] $Energia = I * V * tempo$ onde $I$ é a corrente, $V$ a tensão e $tempo = instruções/clock$.

| Aplicação | Instruções | DCFG Nós | DCFG Arestas | Tempo (s) | Memória (MB) |
|---|---|---|---|---|---|
| udp-ipv6 cliente e servidor | 42.904 | 65.218 | 95.237 | 33,25 | 45,40 |
| webserver e wget | 42.828 | 66.154 | 93.678 | 27,28 | 92,53 |

**Tabela 2. Número de instruções, tamanho dos DCFGs (nós e arestas) e a quantidade de tempo e memória gastos na análise estática.**

| Pilha Alcalina | Economia Energética (kJ) | Pilhas (AA) Economizadas | Economia Monetária (R$) |
|---|---|---|---|
| Uma semana | 0,393 | 0,04 | 0,20 |
| Um mês | 1,685 | 0,16 | 0,80 |
| Um ano | 20,223 | 1,86 | 9,27 |

**Tabela 3. Economia de energia.**

da capacidade de uma pilha alcalina[11], ou seja, 0,04 kJ ou R$ 1,00. Já em um ano, a economia chega 1,86 pilhas ou R$9,27.

Por fim, é importante frisar que as análises estáticas realizadas pela Siot gastaram em média 29,5 segundos em um notebook Intel core i7 2.20GHz (Tabela 2). Cada aplicação analisada possui mais de 49.000 instruções e cada grafo contém cerca de 65 mil nós e 90 mil arestas. Foram gastos 45,4 MBytes de memória *RAM* na análise da aplicação udp-ipv6 e 92,53 MBytes na análise da aplicação wget/webserver.

## 6. Trabalhos Correlatos

A maioria das propostas de defesa para sistemas computacionais de maneira geral são baseadas na Análise Estática [Chess and West 2007], na Análise Dinâmica [Serebryany et al. 2012], ou na Análise Híbrida. Em redes tradicionais, as técnicas usadas variam desde análise estática e instrumentação dinâmica de código até soluções que procuram executar o sistema simbolicamente [Chess and West 2007]. No entanto, tais propostas não levam em consideração as características de IoT e, portanto, nem sempre lhes são adequadas [Wangham et al. 2013].

O exemplo canônico é a a proposta de Serebryany *et al.* [Serebryany et al. 2012], ou seja, a ferramenta AddressSanitizer. Essa ferramenta realiza uma análise estática do sistema em busca de acessos ilegais à memória. O sistema é, então, instrumentado para que sua execução seja abortada na iminência de tais acessos. Desta forma, a AddressSanitizer é capaz de eliminar por completo tais vulnerabilidades e, por conseguinte, é chamada de consistente (*sound*). A AddressSanitizer, contudo, é conservadora na sua análise e considera que toda troca de mensagens é contaminada. Tal conservadorismo traduz-se em ineficiência o que a torna, portanto, inadequada para a IoT.

Entre as iniciativas de aplicação das técnicas de análise estática e dinâmica em aplicações que acessam a rede podemos destacar aquelas voltadas para aplicações Web que visam detectar fluxos contaminado e a fim de saneá-los [Huang et al. 2004, Jovanovic et al. 2006, Balzarotti et al. 2008]. Esses trabalhos buscam dependências entre entradas

[11] Custo da pilha R$5,00 e energia por pilha $3Wh$.

(fontes) e saídas sensíveis (sorvedouros), além de verificar se esses caminhos estão saneados, ou seja, se há algum tipo de verificação de vulnerabilidade ou transformação de código que impeça que *strings* enviadas pelos usuários através das entradas do sistema alcancem canais de saída. Mesmo nesses trabalhos não há um tratamento especial em relação às funções que fazem acesso à rede. Cada programa é analisado individualmente e qualquer informação recebida da rede é considerada contaminada. Através das técnicas que apresentamos nesse artigo é possível reduzir o número de entradas (fontes) e, por consequência, reduzir a quantidade de caminhos a serem saneados. Dessa forma é possível reduzir o custo de proteger tais sistemas.

Cumpre lembrar que há propostas para redes limitadas que analisam sistemas. Dentre essas, destacam-se os trabalhos de Sasnauskas *et al.* [Sasnauskas et al. 2010] e Li *et al.* [Li and Regehr 2010], respectivamente, o Kleenet e o T-Check. Tais ferramentas executam sistemas simbolicamente em busca de defeitos de software em geral (*bugs*). A complexidade dos algoritmos dessas soluções comprometem a escalabilidade e, por restrições computacionais, tornam-nas inconsistentes (unsound) já que não possível testar todos os cenários de execução possíveis de um programa.

Nossa solução utiliza o cruzamento de dados de módulos de sistemas distribuídos para ser mais eficiente. Nesta linha, há trabalhos como o de Yang *et al.* [Yang et al. 2009] – sobre modelos de verificação (*model checking*) – e o de Comparetti *et al.* [Comparetti et al. 2009] – sobre engenharia reversa de protocolos. Embora ambas as propostas objetivem a descoberta de vulnerabilidades de protocolos distribuídos, em suas análises, elas tratam os interlocutores da comunicação como caixas-pretas, sem no entanto examinar seus código fonte.

## 7. Conclusões

A IoT requer soluções de segurança feitas exclusivamente para o seu contexto. É fato que há propostas de segurança para IoT que buscam proteger a IoT contra ataques disparados por adversários externos. Contudo, o mesmo nem sempre ocorre para ataques disparados por adversários com acesso às entradas de dados dos sistema. As propostas para se combater *exploits* foram concebidas no contexto de redes de computadores tradicionais, não consideram as peculiaridades de IoT e, portanto, não lhes são plenamente adequadas. O objetivo deste trabalho foi, portanto, apresentar uma solução de segurança exclusivamente concebida para segurança de código de IoT ante ataques que visam explorar falhas no código como acontece nos ataques de Buffer Overflow.

Nossa proposta – Siot – emprega uma abordagem pioneira na busca de vulnerabilidades. Tal abordagem gera um Grafo de Controle de Fluxo Distribuído que provê ao analisador estático uma visão holística do programa sob análise. Desta forma, é possível cruzar dados de diferentes módulos de um sistema o que torna a solução menos conservadora e, por sua vez, mais eficiente. Os resultados indicam que a Siot provê as mesmas garantias que mecanismos de segurança tradicionais, porém é cerca de 83% mais eficiente em relação a quantidade de verificações de limites de arranjos.

Como trabalhos futuros pretendemos utilizar as técnicas aqui propostas para verificar outros tipos de vulnerabilidades de código como Integer Overfow, por exemplo.

## Referências

Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805.

Balzarotti, D., Cova, M., Felmetsger, V., Jovanovic, N., Kirda, E., Kruegel, C., and Vigna, G. (2008). Saner: Composing static and dynamic analysis to validate sanitization in web applications. In *IEEE Symposium on Security and Privacy*, pages 387–401. IEEE.

Chess, B. and West, J. (2007). *Secure Programming with Static Analysis*. Addison-Wesley Professional, first edition.

Comparetti, P. M., Wondracek, G., Kruegel, C., and Kirda, E. (2009). Prospex: Protocol specification extraction. In *IEEE Symposium on Security and Privacy*, pages 110–125.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *IEEE International Conference on Local Computer Networks (LCN'04)*, pages 455–462.

Huang, Y.-W., Yu, F., Hang, C., Tsai, C.-H., Lee, D. T., and Kuo, S.-Y. (2004). Securing web application code by static analysis and runtime protection. In *WWW*, pages 40–51.

Jovanovic, N., Kruegel, C., and Kirda, E. (2006). Pixy: A static analysis tool for detecting web application vulnerabilities (short paper). In *Symposium on Security and Privacy*, pages 258–263. IEEE.

Karlof, C. and Wagner, D. (2003). Secure routing in wireless sensor networks: Attacks and countermeasures. In *1st IEEE Int'l Workshop on Sensor Network Protocols and Applications*.

Lattner, C. and Adve, V. S. (2004). LLVM: A compilation framework for lifelong program analysis & transformation. In *CGO*, pages 75–88. IEEE.

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D. (2004). TinyOS: An operating system for wireless sensor networks. In Weber, Rabaey, and Aarts, editors, *Ambient Intelligence*. Springer-Verlag.

Li, P. and Regehr, J. (2010). T-check: bug finding for sensor networks. In *9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 174–185. ACM.

Oliveira, S., Wong, H. C., and Nogueira, J. M. S. (2006). Nekap: Estabelecimento de chaves resiliente a intrusos em RSSF. In *SBRC - 24th Brazilian Symposium on Computer Networks*.

Sasnauskas, R., Landsiedel, O., Alizai, M. H., Weise, C., Kowalewski, S., and Wehrle, K. (2010). Kleenet: discovering insidious interaction bugs in wireless sensor networks before deployment. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 186–196. ACM.

Serebryany, K., Bruening, D., Potapenko, A., and Vyukov, D. (2012). Addresssanitizer: a fast address sanity checker. In *USENIX*, pages 28–28. USENIX Association.

Simplício, Jr., M. A., Barreto, P. S. L. M., Margi, C. B., and Carvalho, T. C. M. B. (2010). A survey on key management mechanisms for distributed wireless sensor networks. *Computer Networks*, 54(15):2591–2612.

Wangham, M. S., Domenech, M. C., and de Mello, E. R. (2013). Infraestrutura de autenticação e de autorização para internet das coisas. In *Minicursos*, volume 1 of *13th Brazilian Symposium on Information and Computer System Security (SBSeg'13)*. SBC.

Yang, J., Chen, T., Wu, M., Xu, Z., Liu, X., Lin, H., Yang, M., Long, F., Zhang, L., and Zhou, L. (2009). MODIST: Transparent model checking of unmodified distributed systems. In *USENIX Symposium on Networked Systems Design and Implementation NSDI'09*, pages 213–228.

# Socialbots: Implicações na segurança e na credibilidade de serviços baseados no Twitter

**Carlos Freitas**[1], **Fabrício Benevenuto**[1], **Adriano Veloso**[1]

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

{alessandro, fabricio,adrianov}@dcc.ufmg.br

***Abstract.*** *More and more, data extracted from social networks is used to build new applications and services, such as traffic monitoring platforms, identification of epidemic outbreaks, as well as several other applications related to the creation of smart cities, for example. However, such services are vulnerable to attacks from bots − automatized accounts − seeking to tamper statistics of public perception posting an excessive number of messages generated automatically. Bots can invalidate many existing services, which makes it crucial to understand the main forms of attacks and to seek defense mechanisms. This work presents a wide characterization of the behavior of bots on Twitter. From a real data set containing 19,115 bots, several characteristics of bots were identified, extracted from behavior and writing patterns, that have discriminative power. From these features, we present an automatic detection method capable to detect 92% of the bots while only less than 1% of real users are misclassified.*

***Resumo.*** *Cada vez mais, dados extraídos de redes sociais são utilizados para a construção de novas aplicações e serviços, como plataformas para monitoramento de trânsito, identificação de surtos epidêmicos, bem como várias outras aplicações associadas à criação de cidades inteligentes, por exemplo. Entretanto, tais serviços são vulneráveis a ataques de bots − contas automatizadas − que buscam adulterar estatísticas de percepção pública postando um excessivo número de mensagens geradas automaticamente. Bots podem invalidar diversos serviços existentes, o que torna crucial entender as principais formas de ataque, bem como buscar mecanismos de defesa. Este trabalho apresenta uma ampla caracterização do comportamento de bots no Twitter. A partir de uma base de dados real contendo 19.115 bots, foram identificadas diversas características dos bots, extraídas de padrões de comportamento e de escrita de texto, que possuem alto poder discriminativo. A partir dessas características, apresentamos um método de detecção automática de bots capaz de detectar 92% deles, enquanto menos de 1% dos usuários reais são classificados erroneamente.*

## 1. Introdução

O Twitter é inegavelmente uma das redes sociais mais populares da atualidade, na qual seus usuários geram mais de 500 milhões de mensagens diariamente [Protalinski 2013], o que, aliado a sua API aberta, tem tornado a plataforma largamente utilizada para serviços de extração de conhecimento. Como exemplo podemos citar a predição de mudanças no mercado de ações [Zhang and Paxson 2011], a detecção de catástrofes em tempo

real [Sakaki et al. 2010], detecção de focos de epidemias [Gomide et al. 2011] e também análise de opinião [Tumasjan et al. 2010]. Geralmente, esses serviços usam amostras do Twitter, dessa forma tornando-se vulneráveis a ataques com o objetivo de adulterar suas estatísticas. Por exemplo, um ou mais usuários podem postar mensagens sobre um tópico específico para direcionar os resultados de um algoritmo de análise de opinião. Mais importante, robôs (ou bots) podem ser utilizados para postar mensagens enviesadas sobre um tópico específico (e.g., postar mensagens favorecendo algum candidato político).

Socialbots, bots desenvolvidos de forma a se passar por humanos, já são usados com o objetivo de enganar e influenciar outros usuários na rede [Messias et al. 2013, Harris 2013]. Esses bots possuem a capacidade de comprometer a estrutura da rede social [Boshmaf et al. 2011], permitindo assim que eles ganhem influência na rede. Bots podem ser explorados para a difusão de propaganda e informações erradas na rede. Por exemplo, uma rede de socialbots pode ser usada para a propagação de ações políticas ou publicitárias que tentam criar a impressão de que são movimentos espontâneos e populares [Ratkiewicz et al. 2011]. Além disso, bots já são usados por candidatos políticos durante campanhas eleitorais com o objetivo de alterar os "trending topics" [Orcutt 2012], ou para aumentar artificialmente seus números de seguidores, e consequentemente seus índices de popularidade [Calzolari 2012]. Este cenário só piora quando consideramos a existência de serviços de venda de bots.[1]

A quantidade exata de bots no Twitter é desconhecida. [Chu et al. 2012] estimam que 50% das contas sejam associadas a bots. Contudo, o Twitter afirma que contas falsas ou spammers representam apenas 5% dos seus 215 milhões de usuários ativos [Gara 2013]. Seja 5% ou 50%, a necessidade de estratégias para a detecção de bots no Twitter é crucial para garantir a credibilidade e segurança dos serviços que usam o Twitter como fonte de dados.

Neste artigo, abordamos o problema de detectar bots no Twitter. Nosso foco está na identificação de comportamentos de bots que extrapolam as estratégias de identificação de atividade automática. Como principais contribuições podemos mencionar: (i) a caracterização do comportamento de bots em uma grande base de dados, (ii) identificação de atributos linguísticos na postagem de bots, que até onde tenhamos conhecimento nunca foram utilizados para a detecção de bots, e finalmente, (iii) a criação de um método de detecção automática de bots que explora os atributos identificados.

Nossa abordagem foi capaz de detectar mais de 92% dos bots da nossa base de dados, classificando erroneamente menos de 1% dos usuários. Para isso, nós construímos uma coleção contendo 19.115 bots, identificados através de uma abordagem de identificação de padrões automáticos de postagem. A partir desses bots, nós investigamos diversos outros aspectos capazes de diferenciá-los de usuários comuns. De posse dessas características, nós investigamos a viabilidade de uma estratégia supervisionada de classificação para a identificação de bots.

O restante do artigo está organizado da seguinte forma: Na próxima seção apresentamos trabalhos relacionados. Na Seção 3 descrevemos a construção da base de dados de bots utilizada em nossos experimentos. Na Seção 4 apresentamos um estudo dos atributos usados por nosso método. Na Seção 5 apresentamos os resultados obtidos por nosso

[1] http://www.jetbots.com/

método. Finalmente, na Seção 6 apresentamos conclusões e direções para possíveis trabalhos futuros.

## 2. Trabalhos Relacionados

Existem vários estudos com foco na criação de bots. O projeto Realboy visa a criação de bots que imitam usuários reais de forma verossímil [Coburn and Marra 2008]. O Web Ecology Project[2] visa a criação de socialbots para interagirem com um grupo de usuários no Twitter. [Messias et al. 2013] criaram bots capazes de interagir com usuários legítimos no Twitter. Durante o período de 90 dias os mesmos conseguiram resultados significantes em sistemas medidores de influência como o Klout[3] e Twitalyzer.[4] Finalmente, [Boshmaf et al. 2011] projetaram uma rede social de bots com o intuito de realizar uma infiltração em larga escala. O estudo demonstrou que redes sociais podem ser infiltradas com uma taxa de sucesso de até 80%. De maneira geral, esses esforços demostram a vulnerabilidade do Twitter à infiltração de bots.

De forma complementar à detecção de bots, [Wagner et al. 2012] criaram um modelo de aprendizado de máquina para prever a suscetibilidade dos usuários a ataques de socialbots, utilizando três componentes diferentes de atributos (a rede do usuário, seu comportamento e características linguísticas). Seus resultados apontam que usuários mais "abertos" a interações sociais são mais suscetíveis a ataques. Posteriormente, [Wald et al. 2013] realizaram um estudo similar e encontraram que o Klout score, número de seguidores e de amigos, são bons preditores se um usuário irá interagir com um bot.

Apesar de bots não serem utilizados necessariamente para postar algum tipo de spam, existem diversos esforços nessa tarefa que são complementares ao nosso esforço. Em particular, [Grier et al. 2010, Pitsillidis et al. 2010, Stringhini et al. 2010, Benevenuto et al. 2010] desenvolveram técnicas para a detecção automáticas de spambots baseadas no seu comportamento anormal. [Lee et al. 2011] realizaram um estudo de longo prazo sobre poluidores de conteúdo no Twitter usando "honeypots", cujo modelo conseguiu detectar spammers com 98% de acurácia. Contudo, não é claro o desempenho destes métodos para a detecção de bots que não estejam envolvidos em atividades relacionadas a spam. Finalmente, [Thomas et al. 2013] investigaram, durante 10 meses, o mercado negro de venda de contas em serviços sociais e criaram um método para a detecção de contas fraudulentas. Esse método é capaz de detectar contas fraudulentas com 99% de precisão antes mesmo delas iniciarem qualquer atividade ilegal.

[Chu et al. 2012] usam técnicas de aprendizado de máquina para identificar três tipos de contas: usuários, bots e ciborgues (usuários assistidos por bots). Eles mostram que a regularidade de postagem, a fração de tweets com URLS e o meio de postagem (o uso de aplicativos externos) apresentam indícios de qual é o tipo da conta. [Zhang and Paxson 2011] desenvolveram um método para detecção de contas com atividade automatizada usando apenas o "timestamp" das mensagens utilizando um teste $\chi^2$. Apesar desses métodos apresentarem bons resultados, eles podem ser facilmente burlados por bots que: (i) postem com intervalos aleatórios, ou seguem uma distribuição

[2]http://www.webecologyproject.org/category/competition/
[3]http://klout.com/
[4]http://twitalyzer.com/

similar a comportamentos típicos de humanos, (ii) diminuam a fração de tweets com URLs, e (iii) usando ferramentas para automação web que imitem um navegador, (e.g., phantomjs [5] e o fake[6]). Dessa forma nossa abordagem visa a identificação de atributos mais difíceis de serem burlados por bots, como a estrutura dos tweets e o padrão de escrita, além das características do usuário.

## 3. Base de Dados

Para estudar o comportamento de bots no Twitter, precisamos de uma amostra ampla e representativa de bots e usuários legítimos. Até onde conhecemos, nenhuma coleção com tais características está disponível publicamente. Descrevemos a seguir como construímos a coleção para nossos experimentos. A base de de dados foi criada a partir de um "snapshot" completo da rede do Twitter e todos os tweets postados por todos os usuários até Agosto de 2009 [Cha et al. 2010]. Mais especificamente, o conjunto de dados contém 54.981.152 usuários ligados uns aos outros por 1.963.263.821 arestas. O conjunto de dados também contém todos os tweets postados pelos usuários coletados, que consiste em 1.755.925.520 tweets. Cerca de 8% das contas eram privadas, o que implica que apenas seus seguidores poderiam ver seus tweets. Posteriormente [Ghosh et al. 2012] recoletaram os usuários desta base de dados em Fevereiro de 2011, encontrando um total de 379.340 contas suspensas pelo Twitter.

Nossa estratégia consiste em investigar essas contas suspensas para identificar bots, através de um método de detecção de atividade automática no Twitter, proposto recentemente[Zhang and Paxson 2011]. Além disso, nós selecionamos uma amostra de um milhão de contas não suspensas que conjuntamente com as contas suspensas foram submetidas ao teste de atividade automática. Uma conta é reprovada no teste quando ela apresenta um comportamento altamente automatizado (e.g., postagem de tweets em intervalos regulares de tempo). Finalmente, como o método precisa de pelo menos 30 tweets para funcionar as contas com menos de 30 tweets foram consideradas "insuficientes". Apesar do método realizar uma análise simples, o mesmo nos permitiu criar uma grande coleção rotulada e assim realizar um estudo de comportamentos mais complexos dos bots no Twitter. Nossa abordagem consiste em investigar outros aspectos relativos ao comportamento e padrões de escrita dessas contas, na tentativa de identificar mesmo bots com comportamentos mais complexos.

**Tabela 1. Teste de atividade automática**

| | Reprovadas | Aprovadas | Insuficientes |
|---|---|---|---|
| Não suspensas | 5.755 | 91.118 | 903.127 |
| Suspensas | 19.115 | 25.355 | 334.869 |

Como podemos perceber pelos resultados da Tabela 1, cerca de 42% das contas suspensas com pelo menos 30 tweets utilizam algum método de atividade automática, enquanto menos de 6% das contas não suspensas com tweets suficientes usam um recurso similar.

Para compor nossa base de dados consideramos as contas não suspensas que não possuem nenhum método de automatização como usuários legítimos. De forma similar,

[5]http://phantomjs.org/

[6]http://fakeapp.com/

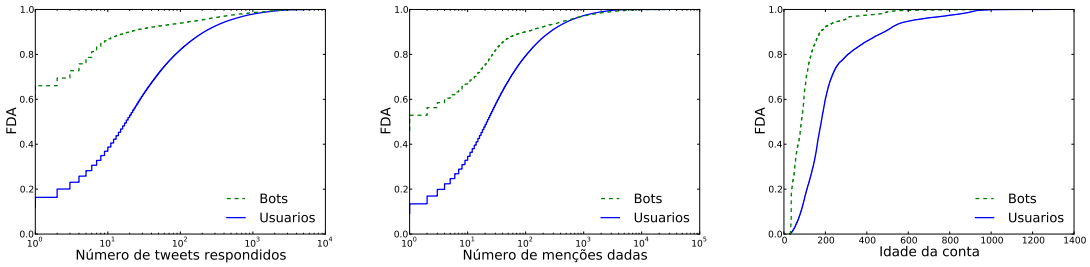
consideramos que as contas suspensas com atividade automática são bots. Dessa forma, nosso dataset contém **110.233** (91.118+19.115) contas e **42.773.272** de tweets.

## 4. Analisando Atributos de Usuários

De forma diferente dos humanos, bots geralmente são criados com algum objetivo específico: invadir um grupo de usuários, espalhar spam, postar mensagens sobre um tópico em particular, etc. Além disso, bots simples não são capazes de interagir inteligentemente com outros usuários (e.g., respondendo perguntas encaminhadas aos mesmos). Dessa forma, é esperado que usuários e bots possuam comportamentos diferentes. Intuitivamente, esperamos que humanos sejam mais sociais e ativos em conversas, enquanto que os bots postam mais tweets, enviesados para algum tópico em particular ou contendo URLs. Para comprovar isto, analisamos um grande conjunto de atributos extraídos de padrões de comportamento e de escrita do texto. Consideramos três conjuntos de atributos: (i) atributos de conteúdo, (ii) atributos do usuário e (iii) atributos linguísticos.

### 4.1. Atributos do Usuário

Atributos do usuário capturam características como a influência na rede do Twitter e as interações sociais do usuário. Foram consideradas as seguintes métricas como atributos de usuário: número de seguidores, número de amigos, a razão de seguidores por amigos, número de tweets, idade da conta do usuário − o número de dias entre a criação da conta e do último tweet analisado por nós, número de vezes que o usuário foi mencionado, número de vezes que o usuário foi respondido, número de vezes que o usuário mencionou alguém, número de vezes que o usuário respondeu alguém, número de amigos dos seguidores do usuário, número total de tweets dos amigos do usuário e a existência de palavras associadas a spam no nome do usuário. No total, temos 12 atributos de usuário.




**(a)** Número de tweets respondidos

**(b)** Número de menções dadas

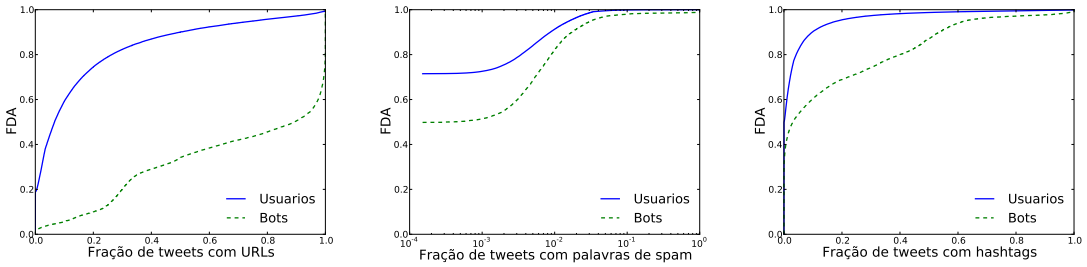
**(c)** Idade da Conta

**Figura 1. Funções de distribuição acumulada de três atributos do usuário.**

Em seguida, analisamos três características do usuário, que podem diferenciar bots de usuários legítimos. A Figura 1 mostra a função de distribuição acumulada (FDA) dos três atributos: número de tweets respondidos, número de menções dadas e idade da conta. A partir das Figuras 1(a) e 1(b) notamos que usuários legítimos são mais sociais e ativos em conversas do que bots. Finalmente, a Figura 1(c) mostra a idade da conta do usuário. Podemos observar que bots tendem a possuir contas mais novas, provavelmente pelo fato de serem bloqueados por outros usuários ou reportados para o Twitter por realizarem atividades ilícitas, e.g., postar links de spam.

### 4.2. Atributos de Conteúdo

Atributos de conteúdo são baseados em propriedades dos tweets postados pelos usuários, que capturam características específicas relacionadas a forma com que os mesmos escrevem seus tweets. Devido ao fato dos usuários geralmente postarem vários tweets, utilizamos o valor máximo, mínimo, médio e a mediana das seguintes métricas: número de hashtags por palavra em cada tweet, número de URLs por palavra em cada tweet, número de palavras em cada tweet, número de caracteres em cada tweet, número de URLs em cada tweet, número de hashtags em cada tweet, número de caracteres numéricos (e.g. 1,2,3) em cada tweet, número de usuários mencionados em cada tweet, número de vezes que o tweet foi retweetado. Também utilizamos a fração de tweets contendo pelo menos uma palavra relacionada a atividades de spam, a fração de mensagens que eram respostas, a fração de mensagens que mencionam um outro usuário, a fração de tweets que contem hashtags, a fração de mensagens que são retweets e a fração de mensagens que contem URLs. Ao todo temos 42 atributos de conteúdo.




**(a)** Fração de tweets com URLs

**(b)** Fração de tweets com palavras de spam

**(c)** Fração de tweets com hashtags

**Figura 2. Funções de distribuição acumulada de três atributos de conteúdo.**

A seguir, apresentamos uma análise de três atributos de conteúdo: fração de URLs, fração de tweets com palavras de spam e fração de hashtags. A Figura 2 mostra as FDAs destes atributos. A Figura 2(a) mostra que bots postam mais tweets com URLs que usuários legítimos. Contudo, como a Figura 2(b) indica, bots não são necessariamente spammers, o que aponta que eles possam postar URLs dos mais diversos tópicos (e.g., notícias sobre um determinado tópico). Finalmente, a Figura 2(c) revela que bots tendem a postar mais hashtags que usuários legítimos, talvez com o intuito de aparecer mais em buscas de determinados tópicos.

### 4.3. Atributos Linguísticos

Atributos linguísticos capturam propriedades específicas do padrão de escrita do usuário, visto que usuários que postam mensagens sobre vários tópicos geram conteúdo menos previsível do que aqueles que se restringem a um tópico em particular. Consideramos as seguintes métricas como atributos linguísticos:

- **Tamanho do Vocabulário:** Consideramos o tamanho do vocabulário do usuário, isto é, o número total de palavras diferentes usadas por ele, assim como a razão entre ele e o número de tweets do usuário.

- **N-gramas**: Dado um conjunto de tweets gerados por um usuário para cada tweet calculamos o número de n-gramas que já foram usados pelo usuário em outros tweets, além da sua razão com o número total de n-gramas já utilizados pelo usuário. Um n-grama é uma sequência contígua de $n$ itens de uma dada sequência de texto, os itens podem ser caracteres, palavras, sílabas etc. Um n-grama de tamanho 1 é conhecido como unigrama, de tamanho 2 como brigrama e de tamanho 3 como trigrama. Usamos a média destes valores como atributos de nosso classificador. Calculamos variações desta métrica usando n-gramas de palavras e caracteres, além de valores de n iguais a 2, 3 e 4.
- **Distância do Cosseno:** Dado um conjunto de tweets gerados por um usuário. Para cada tweet computamos a distância máxima do cosseno [Baeza-Yates and Ribeiro-Neto 1999] com o resto dos tweets do usuário. A distância de dois tweets é dada por

$$dist(t_j, q) = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}}$$

  Onde, $w_{t,d}$ é o produto da frequência do termo $t$ no tweet $d$ pela frequência inversa do termo nos tweets do usuário. Usamos a média destes valores como atributo no nosso classificador.
- **Índice de Jaccard:** Dado um conjunto de tweets gerados por um usuário para cada tweet é computado o máximo índice de Jaccard [Tan et al. 2005] com o resto dos tweets postados. O índice de dois tweets é dado por

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

  Usamos a média destes valores como atributo no nosso classificador. Além disso, calculamos três variações do índice usando unigramas, bigramas e trigramas.
- **Modelo de N-gramas**: Dado um conjunto de tweets gerados por um usuário calculamos a probabilidade de cada tweet ser gerado pelo usuário usando um modelo de linguagem [Manning and Schütze 1999], um modelo estatístico que atribui a probabilidade de uma sequência de $m$ palavras por meio de uma distribuição de probabilidade. Para isso, usamos um modelo de n-grama, no qual a probabilidade $P(w_1, ..., w_m)$ de observar a sequência $w_1, ..., w_m$ é aproximado por

$$P(w_1, ..., w_m) \approx \prod_{i=1}^{m} P(w_i | w_{i-(n-1)}, ..., w_{i-1})$$

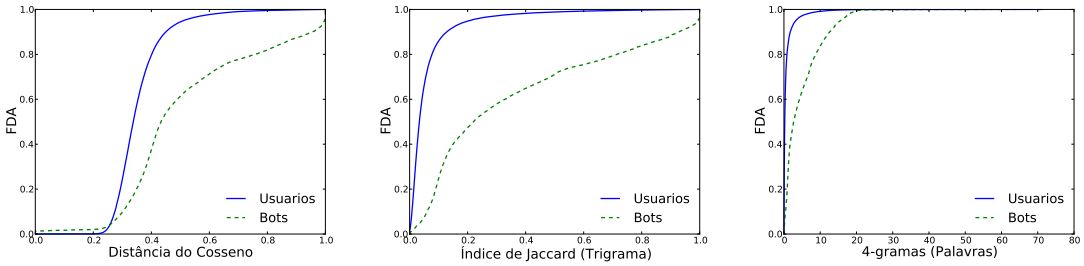
  Assumimos que a probabilidade de observar a palavra $w_i$ é dada por apenas as últimas $n-1$ palavras, propriedade Markoviana. Dessa forma a probabilidade condicional pode ser calculada a partir da contagem da frequência dos n-gramas nos tweets restantes do usuário.

$$P(w_i | w_{i-(n-1)}, ..., w_{i-1}) = \frac{freq(w_{i-(n-1)}, ..., w_{i-1}, w_i)}{freq(w_{i-(n-1)}, ..., w_{i-1})}$$

  Para cada usuário usamos a média das probabilidades de cada tweet como atributo no nosso classificador. Calculamos variações desta métrica usando bigramas e

trigramas de palavras, além de n-gramas de caracteres para valores de n iguais a 2, 3 e 4.

Devido ao custo computacional destas métricas foram analisados apenas os últimos 200 tweets de cada usuário. Ao todo temos 23 atributos linguísticos.




**(a)** Distância do Cosseno

**(b)** Índice de Jaccard (Trigrama)

**(c)** 4-gramas (Palavras)

**Figura 3. Funções de distribuição acumulada de três atributos linguísticos.**

A seguir, realizamos uma análise de três atributos linguísticos: A distância do cosseno, o índice de Jaccard (trigrama) e o 4-gramas (palavras). A figura 3 mostra as FDAs desses atributos. Podemos notar que o padrão de escrita dos bots é mais previsível que o dos usuários legítimos, visto que usuários legítimos usam o Twitter para conversar sobre diversos tópicos, enquanto bots tendem a postar mensagens com foco em um tópico específico.

## 5. Detectando Bots

Nesta seção, analisamos o desempenho dos atributos discutidos na seção anterior em conjunto com um algoritmo de aprendizado supervisionado para a tarefa de detectar bots no Twitter. Além disso, apresentamos na seção 5.1 as métricas usadas para avaliar os resultados da classificação. A seção 5.2 descreve o algoritmo de classificação, ou seja, o classificador, e ambiente experimental utilizado.

### 5.1. Métricas de Avaliação

Para avaliar o desempenho de nossa abordagem foram utilizadas as seguintes métricas: precisão, revocação, Micro-F1, Macro-F1 e Área sob a curva ROC (AUC). A revocação(r) de uma classe *X* é a razão entre o número de usuários corretamente classificados e o número de usuários na classe *X*. A precisão(p) de uma classe *X* é a razão do número de usuários corretamente classificados e o número total de usuários previstos como sendo da classe *X*. Para explicar essas métricas, usaremos uma matriz de confusão, ilustrada na Tabela 2. Cada uma das posições nesta matriz representa o número de elementos em cada classe original, e como eles foram previstos pelo classificador. Na Tabela 2, os valores de precisão ($p_{bot}$) e revocação ($r_{bot}$) para a classe bot são calculados como $p_{bot} = \frac{a}{(a+c)}$ e $r_{bot} = \frac{a}{(a+b)}$.

A medida F1 é a média harmônica entre a precisão e revocação, e é definida como $F1 = \frac{2pr}{(p+r)}$. Micro-F1 e Macro-F1 são duas variações da métrica geralmente utilizadas para avaliar a eficácia de um classificador. Micro-F1 é calculada computando os valores

**Tabela 2. Exemplo de Matriz de Confusão**

| | | Previsto | |
|---|---|---|---|
| | | Bot | Usuário |
| Verdadeiro | Bot | a | b |
| | Usuário | c | d |

globais de precisão e revocação para todas as classes, e em seguida calculando a medida F1. Micro-F1 considera igualmente importante a classificação de cada usuário, independentemente de sua classe, esta métrica basicamente mede a capacidade do classificador de prever corretamente a classe de um usuário. De forma contraria, Macro-F1 é calculado computando primeiro os valores F1 para cada classe de forma isolada, e posteriormente calcular a média destes valores. Macro-F1 considera igualmente importante a eficácia do classificador em cada classe, independentemente do tamanho relativo da classe no conjunto. Desta forma, essas métricas fornecem avaliações complementares da efetividade de um classificador. Finalmente, também foi usada a Área sob a curva ROC que mede a capacidade discriminativa do classificador.

### 5.2. Classificador e Ambiente Experimental

Nos nossos experimentos utilizamos o classificador Random Forest [Breiman 2001], visto que ele foi o que apresentou o melhor desempenho dentre os classificadores testados, dessa forma reportamos apenas seus resultados. A implementação utilizada em nossos experimentos é encontrada na biblioteca Scikit da linguagem de programação Python.[7] Todos os experimentos de classificação são realizados usando validação cruzada com 20 partições. Em cada teste, separamos nosso conjunto de dados em 20 amostras disjuntas, das quais uma é usada como teste e o restante como treino para nosso classificador. O processo é repetido 20 vezes, de forma que cada amostra é usada exatamente uma vez como teste. Isso gera 20 resultados diferentes, finalmente, reportamos os valores médios.

### 5.3. Resultados da Classificação

A Tabela 3 mostra a matriz de confusão obtida em nossos experimentos. Os números apresentados são as porcentagens relativas ao total de contas em cada classe. Aproximadamente 92% dos bots e 99% dos usuários foram classificados corretamente. Desta forma, apenas uma pequena fração - menos de 1% - de usuários foi erroneamente classificado.

**Tabela 3. Matriz de Confusão**

| | | Previsto | |
|---|---|---|---|
| | | Bot | Usuário |
| Verdadeiro | Bot | 92.67% | 7.33% |
| | Usuário | 0.94% | 99.16% |

Uma pequena fração (mais de 7%) dos bots foram classificados erroneamente como usuários legítimos. Após uma inspeção manual, percebemos que esses bots tendem a postar poucas URLs e hashtags, além de postarem tweets contendo citações. Este

[7] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

comportamento engana alguns aspectos importantes usados pelo classificador para diferenciar bots de usuários legítimos. Além disso, analisamos uma amostra dos usuários que foram classificados como bots. Notamos que esses usuários geralmente são bots que não foram suspensos pelo Twitter e cujo padrão temporal de postagem não foi detectado pelo algoritmo de detecção de atividade automática. Dessa forma comprovamos que a abordagem proposta é mais robusta para a detecção de bots.

### 5.4. Importância dos Atributos

Para medir a importância dos atributos calculamos o ganho de informação, isto é redução esperada na entropia, de cada um dos mesmos. A Tabela 4 apresenta o ranking com os 20 atributos mais importantes segundo esta métrica.

**Tabela 4. Ranking dos 20 melhores atributos**

| Posição | Atributo |
|---|---|
| 1 | Idade da conta |
| 2 | Fração de tweets com URLs |
| 3 | Número de URLs por tweet (média) |
| 4 | Índice de Jaccard (Trigrama) |
| 5 | Índice de Jaccard (Bigrama) |
| 6 | Índice de Jaccard (Unigrama) |
| 7 | 4-gramas (Palavra) |
| 8 | URLs por palavra (média) |
| 9 | Trigramas (Palavra) |
| 10 | Fração de respostas |
| 11 | Número de Amigos |
| 12 | Fração de mensagens que mencionam um usuário |
| 13 | Fração de respostas |
| 14 | URLs por palavra (mediana) |
| 15 | Número de menções por tweet (média) |
| 16 | Número de URLs (mediana) |
| 17 | Trigramas relativo (Palavras) |
| 18 | Número de dígitos por tweet (mediana) |
| 19 | Número de tweets dos amigos do usuário |
| 20 | Bigramas (Palavras) |

Entre os primeiros atributos do ranking temos a fração de tweets contendo URLs e o número médio de URLs por tweet, o que indica que bots postam links com maior frequência que os usuários legítimos (e.g., bots que postam links de notícias ou spam). Além disso, podemos notar que os atributos linguísticos apresentam um grande poder discriminativo, apesar de serem redundantes, isso revela que apesar de todas as limitações do Twitter os padrões linguísticos de seus usuários são bons atributos para detecção de bots. Finalmente, podemos notar que bots são geralmente associados a contas mais novas.

A Tabela 5 apresenta um resumo dos resultados, mostrando número de atributos de cada conjunto (usuário, conteúdo e linguísticos) no top 10, 20, 30, 40, 50, 60, 70 e 77 atributos mais discriminativos de acordo com o ranking de ganho de informação. Como podemos notar os atributos de conteúdo são os mais significativos no topo do ranking,

**Tabela 5. Número de atributos nas posições do topo do ranking**

| | Usuário | Conteúdo | Linguísticos |
|---|---|---|---|
| Top 10 | 1 | 4 | 5 |
| Top 20 | 3 | 9 | 8 |
| Top 30 | 8 | 12 | 10 |
| Top 40 | 8 | 19 | 13 |
| Top 50 | 9 | 24 | 17 |
| Top 60 | 9 | 30 | 21 |
| Top 70 | 10 | 37 | 23 |
| Top 77 | 12 | 42 | 23 |

seguidos pelos atributos linguísticos o que confirma que a estrutura dos tweets e o padrão de escrita do usuário são atributos fortemente discriminativos na detecção de bots.

### 5.5. Redução do Conjunto de Atributos

De forma similar a detecção de spammers no Twitter, a detecção de bots é uma constante luta entre os mecanismos de detecção de bots e seus criadores. Dessa forma, esperamos que novos bots sejam mais difíceis de ser detectados por estratégias atuais de detecção. Portanto, a importância dos atributos pode variar com o tempo, isto é, atributos importantes hoje podem se tornar pouco discriminativos. De modo que é importante que diferentes conjuntos de atributos possam ser usados para obter resultados de classificação precisos.

Com essa finalidade, computamos os resultados utilizando os diferentes conjuntos de atributos: do usuário (U), de conteúdo (C) e linguísticos (L), assim como a combinação dos mesmos. A Tabela 6 apresenta o desempenho do classificador usando diferentes conjuntos de atributos.

**Tabela 6. Resultados de nosso classificador**

| Atributos | Micro F1 | Macro F1 | AUC |
|---|---|---|---|
| L | 0.954 | 0.916 | 0.976 |
| U | 0.971 | 0.948 | 0.985 |
| C | 0.964 | 0.936 | 0.982 |
| L+U | 0.977 | 0.960 | 0.991 |
| U+C | 0.978 | 0.962 | 0.991 |
| L+C | 0.973 | 0.951 | 0.987 |
| L+U+C (Modelo proposto) | 0.980 | 0.969 | 0.992 |

Apesar dos atributos do usuário não serem individualmente os mais discriminativos, em conjunto foram os que apresentaram os melhores resultados nos nossos testes, o que pode ser explicado pelo fato que estes atributos são pouco redundantes entre si. De forma similar, os atributos linguísticos e de conteúdo por apresentarem grande redundância entre si apresentam desempenho inferior. Finalmente, notamos que nosso classificador possui um alto poder discriminativo independentemente do conjunto de atributos utilizado.

## 6. Conclusão

Neste trabalho abordamos o problema de detecção de bots no Twitter. Apresentamos uma ampla caracterização do comportamento de bots no Twitter usando três conjuntos de atributos: do usuário, de conteúdo e linguísticos. Nossa análise aponta que os bots tendem a postar mais tweets contendo URLs e hashtags que usuários, além de possuírem um padrão de escrita mais detectável que o de usuários. Além disso, usuários tendem a ser mais "sociais" e participativos em conversas do que os bots.

Com base em nossas medições e caracterização, criamos um método de detecção automática de bots usando um algoritmo de classificação supervisionado. Nosso método foi capaz de detectar 92% dos bots enquanto apenas menos de 1% dos usuários são classificados erroneamente. Posteriormente, estudamos o desempenho de cada atributo proposto e notamos que a idade da conta, a fração de URLs e o padrão de escrita possuem alto poder discriminativo em nossos experimentos. Finalmente, testamos o desempenho de nosso classificador ao utilizar apenas subconjuntos de atributos, nós observamos que nossa abordagem consegue ter um bom desempenho ainda quando apenas um grupo de nossos atributos é utilizado.

Nós acreditamos que esses resultados representam um importante passo na detecção de bots com estratégias complexas e que não podem ser detectados por algoritmos de detecção de atividade automática. No futuro pretendemos implementar um sistema Web de alerta de contas suspeitas de serem bots.

## 7. Agradecimentos

Este trabalho teve apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), da Fundação de Amparo à Pesquisa do estado de Minas Gerais (FAPEMIG), da da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e também do Instituto Nacional de Ciência e Tecnologia para a Web (InWeb). Agradecemos também à Saptarshi Ghosh por fornecer a base de dados usada na nossa pesquisa sem a qual o presente trabalho não teria sido possível.

## Referências

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

Benevenuto, F., Magno, G., Rodrigues, T., and Almeida, V. (2010). Detecting spammers on Twitter. In *Proceedings of the Seventh Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*.

Boshmaf, Y., Muslukhov, I., Beznosov, K., and Ripeanu, M. (2011). The socialbot network: when bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 93–102, New York, NY, USA. ACM.

Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5 – 32.

Calzolari, M. C. (2012). Analysis of twitter followers of the us presidential election candidates: Barack obama and mitt romney.

Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, K. P. (2010). Measuring User Influence in Twitter: The Million Follower Fallacy. In *In Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM)*, Washington DC, USA.

Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2012). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Trans. Dependable Secur. Comput.*, 9(6):811–824.

Coburn, Z. and Marra, G. (2008). Realboy: believable twitter bots. `http://ca.olin.edu/2008/realboy/index.html`.

Gara, T. (2013). One big doubt hanging over twitter's ipo: Fake accounts. `http://online.wsj.com/news/articles/SB10001424052702303492504579113754194762812`.

Ghosh, S., Viswanath, B., Kooti, F., Sharma, N. K., Gautam, K., Benevenuto, F., Ganguly, N., and Gummadi, K. (2012). Understanding and Combating Link Farming in the Twitter Social Network. In *Proceedings of the 21st International World Wide Web Conference (WWW'12)*, Lyon, France.

Gomide, J., Veloso, A., Jr., W. M., Almeida, V., Benevenuto, F., Ferraz, F., and Teixeira, M. (2011). Dengue surveillance based on a computational model of spatio-temporal locality of twitter. In *ACM Web Science Conference (WebSci)*.

Grier, C., Thomas, K., Paxson, V., and Zhang, M. (2010). @spam: The underground on 140 characters or less. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 27–37, New York, NY, USA. ACM.

Harris, D. (2013). Can evil data scientists fool us all with the world's best spam? `http://gigaom.com/2013/02/28/can-evil-data-scientists-fool-us-all-with-the-worlds-best-spam/`.

Lee, K., Eoff, B. D., and Caverlee, J. (2011). Seven months with the devils: A long-term study of content polluters on twitter. In Adamic, L. A., Baeza-Yates, R. A., and Counts, S., editors, *ICWSM*. The AAAI Press.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

Messias, J., Schmidt, L., Rabelo, R., and Benevenuto, F. (2013). You followed my bot! transforming robots into influential users in twitter. *First Monday*, 18(7).

Orcutt, M. (2012). Twitter mischief plagues mexico's election. `http://www.technologyreview.com/news/428286/twitter-mischief-plagues-mexicos-election/`.

Pitsillidis, A., Levchenko, K., Kreibich, C., Kanich, C., Voelker, G. M., Paxson, V., Weaver, N., and Savage, S. (2010). Botnet judo: Fighting spam with itself. In *NDSS*. The Internet Society.

Protalinski, E. (2013). Twitter sees 218m monthly active users, 163.5m monthly mobile users, 100m daily users, and 500m tweets per day.

Ratkiewicz, J., Conover, M., Meiss, M., Gonçalves, B., Patil, S., Flammini, A., and Menczer, F. (2011). Truthy: Mapping the spread of astroturf in microblog streams.

In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 249–252, New York, NY, USA. ACM.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 851–860, New York, NY, USA. ACM.

Stringhini, G., Kruegel, C., and Vigna, G. (2010). Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 1–9, New York, NY, USA. ACM.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Thomas, K., McCoy, D., Grier, C., Kolcz, A., and Paxson, V. (2013). Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In *Proceedings of the 22nd Usenix Security Symposium*.

Tumasjan, A., Sprenger, T. O., Sandner, P. G., and Welpe, I. M. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185.

Wagner, C., Mitter, S., Körner, C., and Strohmaier, M. (2012). When social bots attack: Modeling susceptibility of users in online social networks. In *2nd workshop on Making Sense of Microposts at WWW2012*.

Wald, R., Khoshgoftaar, T., Napolitano, A., and Sumner, C. (2013). Predicting susceptibility to social bots on twitter. In *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pages 6–13.

Zhang, C. M. and Paxson, V. (2011). Detecting and analyzing automated activity on twitter. In *Proceedings of the 12th International Conference on Passive and Active Measurement*, PAM'11, pages 102–111, Berlin, Heidelberg. Springer-Verlag.

# Integrando Plataformas de Nuvens a Federações de Identidade[1,2]

**Ioram S. Sette[I,II], Carlos A. G. Ferraz[I]**

[I] Centro de Informática – Universidade Federal de Pernambuco (UFPE) – Recife - PE
[II] Centro de Estudos e Sistemas Avançados do Recife (CESAR) – Recife – PE

{iss,cagf}@cin.ufpe.br

***Abstract.*** *Privacy of the data processed and stored in the cloud is a concern for the users of these services. Cloud platforms are exposed at the Internet, their usage is shared with other users, and third parties manage them. Identity and access control mechanisms are relevant for intending to protect data from improper access. Among them, identity federations let the user's authentication to be performed by entities which are closer to it. The proposal of this work is to integrate Openstack cloud platform, through its authentication module Keystone, to identity federations using SAML and OpenID Connect protocols. These protocols are compared observing the ease of integration and performance.*

***Resumo.*** *A privacidade dos dados processados e armazenados em nuvens de computadores é uma preocupação para os usuários destes serviços. Plataformas de nuvens estão expostas na Internet, seu uso é compartilhado com outros usuários e são gerenciadas por terceiros. Mecanismos de controle de identidade e acesso são importantes para proteger as informações de acessos indevidos. Dentre eles, as federações de identidade permitem que a autenticação dos usuários seja realizada por entidades mais próximas aos mesmos. A proposta deste trabalho é integrar a plataforma de nuvem Openstack, através de seu módulo de autenticação Keystone, a federações de identidade usando os protocolos SAML e OpenID Connect. Estes protocolos são comparados com relação à facilidade de integração e desempenho.*

## 1. Introdução

O uso de nuvens de computadores por usuários e empresas vem crescendo a cada dia. Usuários usam serviços de nuvem em suas modalidades IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*) e SaaS (*Software as a Service*), enviando seus dados privados para serem processados e armazenados nestas plataformas [Columbus 2013a; Columbus 2013b]. A autenticação e autorização dos usuários através de mecanismos de controle de acesso e de uso aos serviços e dados hospedados em plataformas de nuvens são importantes para evitar o uso indevido e a quebra da privacidade [Tavisi *et al* 2012; Karp *et al* 2009].

Federações de identidade permitem que a autenticação dos usuários seja realizada por entidades que possuem relacionamento próximo aos mesmos, por exemplo: universidade-alunos/professores/pesquisadores, empresas-colaboradores, banco-cliente, grandes provedores de serviço na Internet-usuários. Estas instituições por muitas vezes podem utilizar mecanismos sofisticados de autenticação como biometria, *smartcards*, *tokens*, ou uma combinação deles (autenticação multi-fator). O uso de federações também permite que o usuário se autentique uma única vez e obtenha acesso a vários serviços e sistemas diferentes através do mecanismo de *Single Sign-On* (SSO). Por fim, ao centralizar a autenticação, os usuários passam a ter menos senhas para lembrar, podendo escolher senhas mais fortes e seguras [Revar *et al* 2011].

[1] Projeto apoiado pela Rede Nacional de Pesquisa (RNP) - Edital PGId 2013.

[2] Este trabalho foi apoiado em parte pelo Instituto Nacional de Ciência e Tecnologia para Engenharia de Software (INES), financiado pelo CNPq e FACEPE, processos 573964/2008-4 e APQ-1037-1.03/08.

Dentre os protocolos de federação de identidade mais relevantes encontram-se o SAML (*Security Assertion Markup Language*) e OpenID Connect. Estes protocolos permitem a troca de dados de autenticação e autorização entre provedores de identidade (IdP – *Identity Provider*) e provedores de serviço (SP – *Service Provider*). O SAML (https://www.oasis-open.org/committees/security/) é bastante usado no meio acadêmico e por empresas que possuem uma relação de confiança entre si. Trata-se de um protocolo aberto baseado em XML. O OpenID Connect (http://openid.net/connect) também é um protocolo aberto, baseado em REST/JSON. Ele é mais recente e usado por grandes provedores de serviço na Internet, como Microsoft e Google [Lynch 2011].

Os principais provedores de computação em nuvem desenvolveram suas próprias plataformas, padrões e tecnologias. Em contrapartida, a comunidade de software livre conta com alguns projetos abertos de plataformas nos modelos IaaS (ex. Openstack e Eucalyptus) e PaaS (ex. Apache Hadoop). Plataformas de IaaS possuem em suas arquiteturas um módulo responsável pela gestão de identidade e acesso. Este módulo se integra aos demais, oferecendo a eles serviço de autenticação e autorização. Nas versões mais recentes das plataformas Eucalyptus (2013b) e Openstack (2013a), os módulos de gestão de identidade e acesso são capazes de se integrar com sistemas de diretório externos, por exemplo, através do protocolo LDAP (*Lightweight Directory Access Protocol*). Porém, apesar de permitirem extensões, eles não oferecem em suas distribuições oficiais mecanismos para integração com federações de identidade [Eucalyptus 2013b; Openstack 2013 p.109-113].

Neste trabalho, a plataforma Openstack, apoiada por mais de 200 empresas (ex. HP, IBM e Cisco) e adotada pelo GT-CNC (Computação em Nuvem para Ciência) da RNP [Diniz *et al* 2013; Silva *et al* 2013], é estendida para suportar autenticação através de federações de identidade utilizando o protocolo OpenID Connect. A versão adotada será a de Chadwick (2013a), que já se integra com provedores de identidade através de SAML. Desta forma, será possível comparar os protocolos de federação de identidade quanto à facilidade de integração com o Openstack e, uma vez integrados a esta plataforma, comparar os desempenhos dos referidos protocolos.

Este artigo se divide em mais 5 seções. A seção 2 discute os trabalhos relacionados. Um modelo para autenticação e autorização em plataformas de nuvem é discutido na seção 3. A seção 4 descreve como funciona a integração da plataforma Openstack com federações de identidade. A seção 5 compara os protocolos SAML e OpenID Connect em relação à integração com o Openstack. A seção 6 descreve os experimentos realizados e avalia os resultados. Por fim, a seção 7 apresenta as conclusões e trabalhos futuros.

## 2. Trabalhos Relacionados

Os trabalhos a seguir abordam a integração entre plataformas de nuvem e federações de identidade.

Um protótipo com intenção de integrar a plataforma Openstack com provedores de identidade utilizando o protocolo OpenID foi construído por Khan *et al* (2011). No entanto, a versão do Openstack utilizada na época não possuía um módulo de autenticação bem definido, como o Keystone presente nas versões atuais. Então, a API de serviços do Openstack foi alterada criando-se funções para autenticação através do protocolo OpenID, versão inicial que inspirou a criação do OpenID Connect.

Revar *et al* (2011) descrevem uma arquitetura de alto nível e trechos de códigos mostrando que é possível implementar um mecanismo de *Single Sign-On* utilizando infraestrutura de chaves públicas com certificados X.509, na plataforma Eucalyptus.

Em [Chadwick *et al* 2012], a plataforma Eucalyptus é integrada com provedores de identidade através do protocolo SAML. Em [Chadwick *et al* 2013], o mesmo autor integra a plataforma Openstack, em versão atual com o módulo de identidade (Keystone), com provedores de identidade também usando o protocolo SAML. O IdP utilizado foi o

SimpleSAMLphp, que além de implementar o protocolo SAML possui um componente chamado "Proxy IdP" que faz interface com outros provedores de identidade nos padrões SAML, OAuth, OpenID, dentre outros. Em [Chadwick 2013a], o mesmo autor explica como funciona a integração do Keystone com federações de identidade e define dois fluxos de autenticação, um para cliente simples e um outro para cliente inteligente, protegido contra ataques de *phishing*.

Diniz *et al* (2013) estendem a solução de Chadwick (2013a) para integrar o Openstack a um IdP Shibboleth, apresentando as dificuldades e soluções encontradas. Em [Silva *et al* 2013], os autores apresentam um estudo de caso para integração do serviço Swift do Openstack com autenticação federada através de IdP Shibboleth usando a solução de [Diniz *et al* 2013].

Resultados preliminares do nosso trabalho são apresentados em [Sette *et al* 2013]. A implementação de referência proposta em [Chadwick 2013a] é estendida para possibilitar que o "Keystone Federado" dê suporte ao protocolo OpenID Connect, sem a necessidade de *proxy* intermediário. Os resultados são apresentados na forma de uma avaliação de desempenho e comparações entre os protocolos SAML e OpensID Connect com perfil de cliente básico.

Neste artigo, o perfil de cliente implícito do OpenID Connect foi implementado no "Keystone Federado" e as análises e comparações adicionadas aos resultados obtidos em [Sette *et al* 2013]. Comparações mais aprofundadas entre os protocolos de federação de identidade SAML e OpenID Connect e uma análise sobre os mecanismos de autenticação e autorização em plataformas de nuvem são apresentadas, o que nenhum dos trabalhos relacionados aqui faz.

## 3. Autenticação e Autorização em Plataformas Abertas de IaaS

É comum, nas plataformas abertas de IaaS como Openstack e Eucalyptus, a existência de módulos dedicados a gestão da identidade e acesso dos usuários. Estes módulos costumam ser peças centrais e importantes nas arquiteturas, fazendo interface com todos os demais módulos.

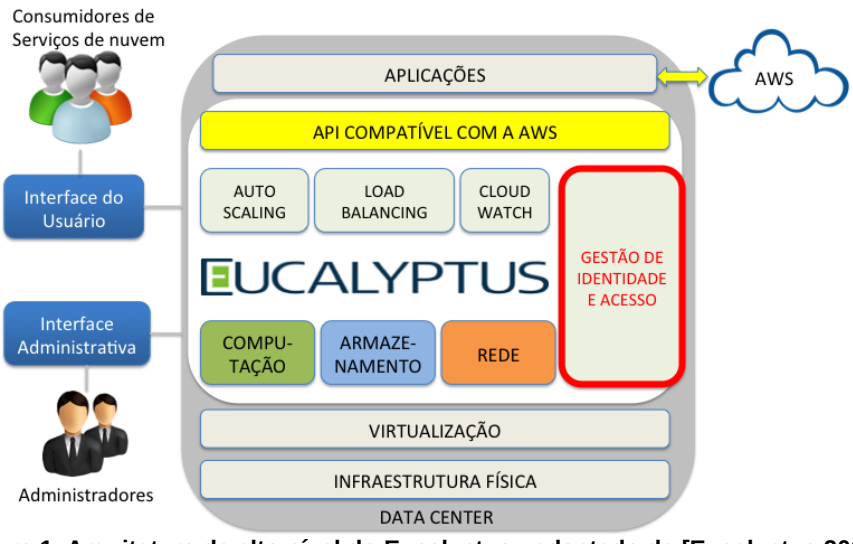



**Figura 1. Arquitetura de alto nível do Eucalyptus - adaptado de [Eucalyptus 2013a]**

No Eucalyptus, o módulo de Gestão de Identidade e Acesso, em destaque na Figura 1, tem esta função. Após a autenticação, regras (*policies*) são verificadas a cada tentativa de acesso, de forma centralizada. A solução é bastante flexível, inspirada na AWS (*Amazon Web Services*). Neste modelo, é possível criar regras refinadas baseadas nos atributos dos recursos (ABAC – *Attribute-Based Access Control*). Por exemplo, pode-se criar uma regra

que permite o acesso de leitura apenas a arquivos menores que 1MB para um determinado grupo de usuários [Eucalyptus 2013a; Eucalyptus 2013b].

O Openstack (Figura 2) também possui um módulo de identidade responsável pela autenticação, o Keystone (em destaque). Além da autenticação, consideramos que o Keystone faz uma autorização de "alto nível", ou seja, transforma os atributos de autenticação em papéis (RBAC - *Role-Based Access Control*). A autorização de fato ocorre posteriormente por cada módulo do Openstack de forma descentralizada com base nos papéis (*roles*), idenficadores do usuário e projetos (*tenants*) ao qual o usuário faz parte. Desta forma, o Keystone se integra a todos os módulos da plataforma Openstack. Por exemplo, o módulo de armazenamento (Swift), baseado na identificação do usuário, do papel e do projeto, decide se libera as ações de *upload* e *download* de arquivos em determinada pasta. Ao contrário da solução Eucalyptus/AWS, esta solução não permite regras mais refinadas [Openstack 2013 p.3-4; Openstack 2013 p.109-113].

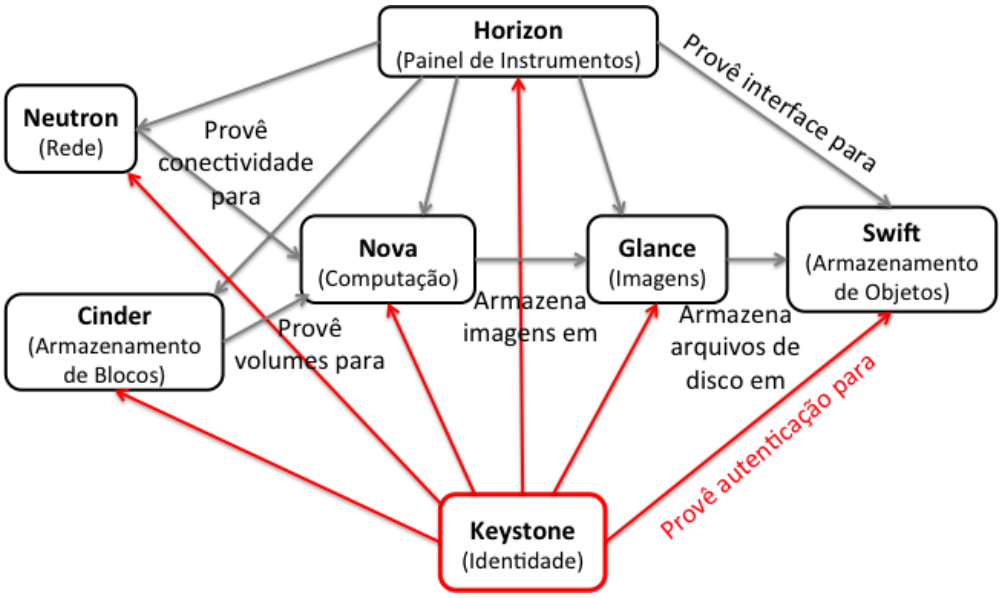



**Figura 2. Arquitetura de alto nível do Openstack – adaptado de [Openstack 2013 p.3-4]**

Apesar de ambas as soluções possuírem integração com serviços de diretórios através de consultas LDAP, nenhuma delas possuem nativamente integração com provedores de identidade externos, como os existentes numa federação de identidade.

Provedores de identidade podem fornecer informações valiosas sobre os usuários autenticados, como informações pessoais, acadêmicas, profissionais, culturais, dentre outras. As plataformas de nuvem utilizam estes atributos dos usuários em seus mecanismos de autorização. Regras de autorização (*policies*) são definidas nos mecanismos de controle de acesso como o ABAC e RBAC. No ABAC, o uso dos atributos é natural, podendo-se criar regras refinadas, por exemplo: apenas usuários de idade superior a 18 anos (atributo do usuário) pode assistir a filmes do gênero "violência" (atributo do recurso). Já no RBAC, os atributos são mapeados em papéis.

## 4. Plataforma Openstack Federada

A integração do Keystone com federações de identidade permite que uma relação de confiança seja estabelecida entre o Openstack, que atua como provedor de serviço (SP), e provedores de identidade (IdPs), onde os usuários são autenticados.

Desta forma, os provedores de identidade informam os atributos dos usuários autenticados ao Keystone, que os mapeia em um papel (*role*) e projeto (*tenant*) necessários para a autorização de acesso. Como o Keystone não conhece previamente todos os usuários

cadastrados nos IdPs, um identificador é gerado para cada usuário em cada primeiro acesso. Estes identificadores permitem a criação de regras específicas para determinado usuário.

A Figura 3 descreve o fluxo para uso de serviços da nuvem Openstack através de autenticação por IdPs federados. Em nosso estudo, usamos o serviço de armazenamento de dados (Swift) que permite o envio e recebimento de arquivos na nuvem. Para acessar o Swift, a ferramenta de linha de comando (CLI) chamada "cliente Swift" foi modificada para suportar federação de identidade. Para interagir com IdPs, os usuários utilizam navegadores web (*browsers*).

Quando o usuário realiza uma operação através do "cliente Swift", a ferramenta inicialmente requisita um *token* ao serviço Keystone (1). Na versão não federada, o Keystone realiza a autenticação localmente, checando as credenciais em banco de dados ou serviço de diretório. Já na versão federada, o Keystone solicita ao usuário que escolha um provedor de identidade com o qual o serviço de nuvem possui uma relação de confiança estabelecida, através de uma lista. Na figura 3, duas opções são apresentadas: o IdP SAML (UFRN) e o IdP OpenID Connect, ou OIDC (RNP GID Lab), implementando os respectivos protocolos.

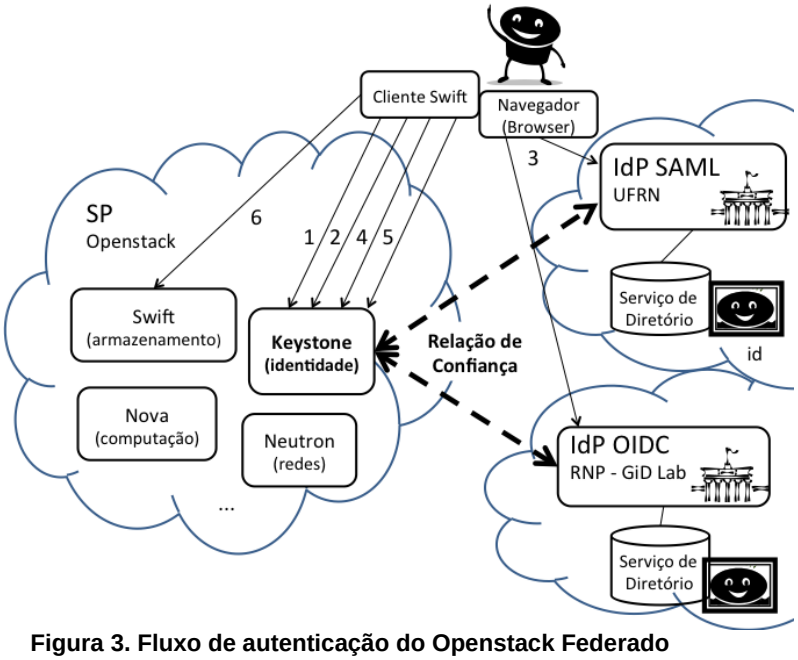



**Figura 3. Fluxo de autenticação do Openstack Federado**

Ao escolher um IdP (2) onde possui credenciais de acesso, o usuário se autentica por meio de um navegador web (3). Vários mecanismos de autenticação podem ser usados pelo IdP para autenticar os usuários, por exemplo, usuário/senha, *tokens* OTP (*one-time-password*), *smartcards*, biometria etc.

Ao final da autenticação, o IdP gera uma mensagem contendo o resultado da autenticação e os atributos do usuário e encaminha ao "cliente Swift", que por sua vez entrega ao serviço Keystone (4). O Keystone valida a mensagem emitida pelo IdP e, baseado nos atributos do usuário, seleciona os projetos (*tenants)* e papéis que o usuário pode exercer em cada projeto. A lista de projetos é apresentada ao usuário, que deve escolher qual pretende utilizar. Por exemplo, apenas os usuários que se autenticam no IdP da UFRN podem acessar os serviços e recursos alocados na nuvem para esta instituição. De forma similar, existe o projeto RNP acessível apenas aos usuários autenticados pelos usuários do IdP da RNP. É possível ainda que exista um projeto público ou compartilhado entre instituições na nuvem. Neste caso, o projeto aceitaria o acesso de usuários autenticados em qualquer um dos dois IdPs.

Ao escolher o projeto (5), os papéis (*roles*) que o usuário possui para o projeto escolhido são definidos e um *token* é gerado e entregue ao usuário. Este *token* deve ser usado para acessar os serviços do Openstack. Um detalhe importante é que o usuário federado não precisa estar previamente cadastrado na nuvem Openstack para usar os seus serviços.

Por fim, o "cliente Swift" acessa o serviço Swift (6) solicitando a operação desejada, usando o *token* recebido. Um novo controle de acesso é realizado internamente pelo serviço Swift através de regras definidas em ACLs (*Access Control Lists*). Papéis administrativos podem ser definidos no Swift para dar aos usuários permissões plenas dentro do projeto, por exemplo: criar pastas, listar, enviar ou receber arquivos, além de criar as regras (ACL) que definem quais outros papéis ou usuários possuem que tipo de acesso (leitura/*download*/listagem ou escrita/*upload*) para cada pasta.

## 5. SAML e OpenID Connect

Nesta seção, uma comparação entre os protocolos SAML e OpenID Connect é realizada apontando as principais semelhanças e diferenças entre eles considerando o contexto da integração com o Openstack. Os fluxos de autenticação destes protocolos também são apresentados para cada protocolo e perfil testados nos experimentos realizados.

### 5.1. Comparações entre SAML e OpenID Connect

Do ponto de vista do usuário, os dois protocolos são bem semelhantes e realizam bem a proposta de autenticação SSO. A sequência de passos para os dois protocolos é exatamente a mesma: o SP redireciona o cliente para o IdP, que autentica o usuário e redireciona o cliente de volta ao SP passando dados que comprovam a autenticação do mesmo. Porém, nesta seção listaremos algumas diferenças que existem entre eles, resumidas na Tabela 1.

O protocolo SAML, criado em 2001, baseia-se na troca de mensagens de autenticação e autorização no padrão XML entre usuário e provedores de serviço e de identidade. É um protocolo maduro, com a sua última versão (2.0) lançada em 2005. Esta versão suporta vários mecanismos de transporte através de diferentes *bindings* [OASIS 2005a], por exemplo: SOAP (única opção na versão 1.x), HTTP Redirect (GET) e HTTP POST. O uso do HTTP sem o SOAP tornou o protocolo mais leve, apesar das mensagens ainda serem XML contendo certificados e assinaturas digitais. Além dos *bindings*, a nova versão permite diferentes fluxos de autenticação, ou *profiles* [OASIS 2005b]. No perfil *Web Browser SSO*, toda troca de dados entre IdP e SP passa pelo cliente através de mensagens assinadas, não havendo tráfego entre SP e IdP. Já no perfil *Artifact Resolution*, SP e IdP conversam diretamente para trocar dados de autenticação.

O OpenID Connect é mais recente (2012) e foi construído sobre o protocolo OAuth 2.0 [IETF 2012]. Trata-se de um protocolo RESTful transportando mensagens JSON entre usuário e provedores de serviço e de identidade. Apesar do pouco tempo, é utilizado por grandes empresas como Google, Yahoo, Facebook e Microsoft. Dois fluxos de autorização são definidos. O perfil de cliente básico é o mais usado e funciona de forma similar ao *Artifact Resolution* do SAML, onde SP e IdP trocam mensagens com dados da autenticação [Sakimura *et al* 2013a]. O perfil implícito [Sakimura *et al* 2013b] é um pouco mais leve, com menos mensagens trocadas entre SP e IdP, porém exige que o cliente (*browser*) seja alterado para enviar os dados enviados pelo IdP ao SP, uma vez que ele os envia através de fragmentos de URL.

A relação de confiança entre SP e IdP é estabelecida de diferentes formas entre os protocolos. No SAML, chaves públicas são trocadas previamente e usadas para assinar as mensagens trafegadas. Desta forma, o IdP pode verificar se a solicitação de autenticação veio de um SP confiável e o SP também pode conferir se a mensagem de autenticação foi emitida pelo IdP e se está íntegra. Isto possibilita a conversa entre SP e IdP por intermédio

do usuário. A opção de criptografia foi introduzida na versão 2.0. No OpenID Connect, o SP cadastra-se previamente no IdP e recebe um par de credenciais (*clientId* e *clientSecret*). Quando SP e IdP se comunicam o SP se autentica junto ao IdP através delas. Opcionalmente, as mensagens podem ser assinadas e/ou criptografadas pelo IdP através dos mecanismos JWS (*JSON Web Signature*) e JWE (*JSON Web Encryption*), respectivamente. Apesar de ambos os protocolos suportarem assinatura e criptografia em suas especificações, é prática comum o uso de assinaturas em IdPs SAML, mas não em IdPs OpenID Connect.

Os atributos dos usuários usados nas mensagens SAML são definidos em outras especificações, como X.500, LDAP, Internet2 eduPerson, ou esquemas privados baseados no perfil de atributo X.500/LDAP. No OpenID Connect, os atributos são agrupados em escopos (*scopes*). O único escopo obrigatório é o "openid" que possui o atributo (*claim*) "sub", identificador do usuário no IdP. Os escopos opcionais são "profile", "email", "address" e "phone", com atributos bem definidos na especificação. Escopos adicionais definidos pelo usuário também são permitidos.

**Tabela 1. Resumo das comparações entre protocolos SAML e OpenID Connect**

| | SAML | OpenID Connect/OAuth2 |
|---|---|---|
| Versão atual | 2.0 (2005) | 1.0 (2012) / 2.0 (2012) |
| Data da primeira versão | 2001 | 2012 / 2006 |
| Protocolo/Mensagens | Vários/XML | REST/JSON |
| Suporta mensagens assinadas? | Sim (XML Signature) | Sim (JWS) |
| Suporta mensagens criptografadas? | Sim (XML Encryption) | Sim (JWE) |
| Atributos | X.500/LDAP | Scopes/Claims |
| No. Interações Usuário x IdP | 2 | 2 |
| No. Mínimo de Interações SP x IdP | 0 (perfil Web SSO) | 1 (perfil cliente implícito) |

Os experimentos deste trabalho comparam os protocolos SAML com *binding HTTP Redirect*, perfil *Web Browser SSO* (mais eficiente) e mensagens assinadas e OpenID Connect com perfis de usuário básico e implícito, sem assinatura ou criptografia.

### 5.2. Fluxos de Autenticação

Os diagramas de sequência a seguir apresentam os fluxos de autenticação através do Keystone não federado (Figura 4) e federado usando os protocolos SAML com perfil *Web Browser SSO* (Figura 5), OpenID Connect com perfis de cliente básico (Figura 6) e implícito (Figura 7), respectivamente. Nos exemplos, o usuário solicita a listagem dos arquivos em uma pasta no Swift.

O diagrama da Figura 4 apresenta o fluxo de autenticação do Openstack não federado. Ao usar o "cliente swift" neste modo, o usuário informa o projeto, credenciais do usuário e a pasta que deseja listar (1). O "cliente swift" solicita ao Keystone um *token* (1.1), usado para acessar o serviço Swift (1.2).

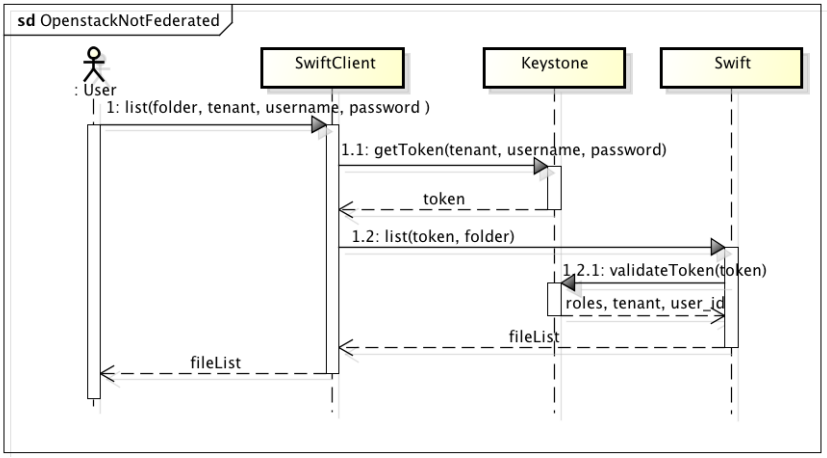



**Figura 4. Fluxo de Autenticação do Openstack não federado**

Na Figura 5, o diagrama apresenta o fluxo de autenticação do Openstack federado utilizando um IdP SAML com *binding* HTTP Redirect. Ao usar o "cliente swift", o usuário informa apenas a pasta que deseja listar (1). O "cliente swift" solicita ao Keystone a lista de IdPs disponíveis para autenticação (1.1). O usuário escolhe o IdP (2) e o Keystone devolve a URL de autenticação (1.2). O "cliente swift" abre um browser chamando a URL (1.3) e o usuário se autentica no IdP (3). O IdP SAML responde com uma mensagem do tipo SAMLResponse, assinada pelo SP (3.2). O Keystone então valida a mensagem e descobre os atributos do usuário autenticado (1.4.1). Neste momento, um ID de usuário é gerado a partir do identificador principal do usuário e criado na base de usuários do Keystone, caso não exista. Em seguida, o Keystone usa um mapa entre atributos do IdP e do SP, e descobre quais os projetos que o usuário pode acessar (1.4.2). O usuário escolhe um projeto (4) e o "cliente swift" solicita um *token* válido para acessar o serviço Swift (1.5). Finalmente, o "cliente swift" solicita ao serviço Swift (1.6) a listagem da pasta usando o *token*.

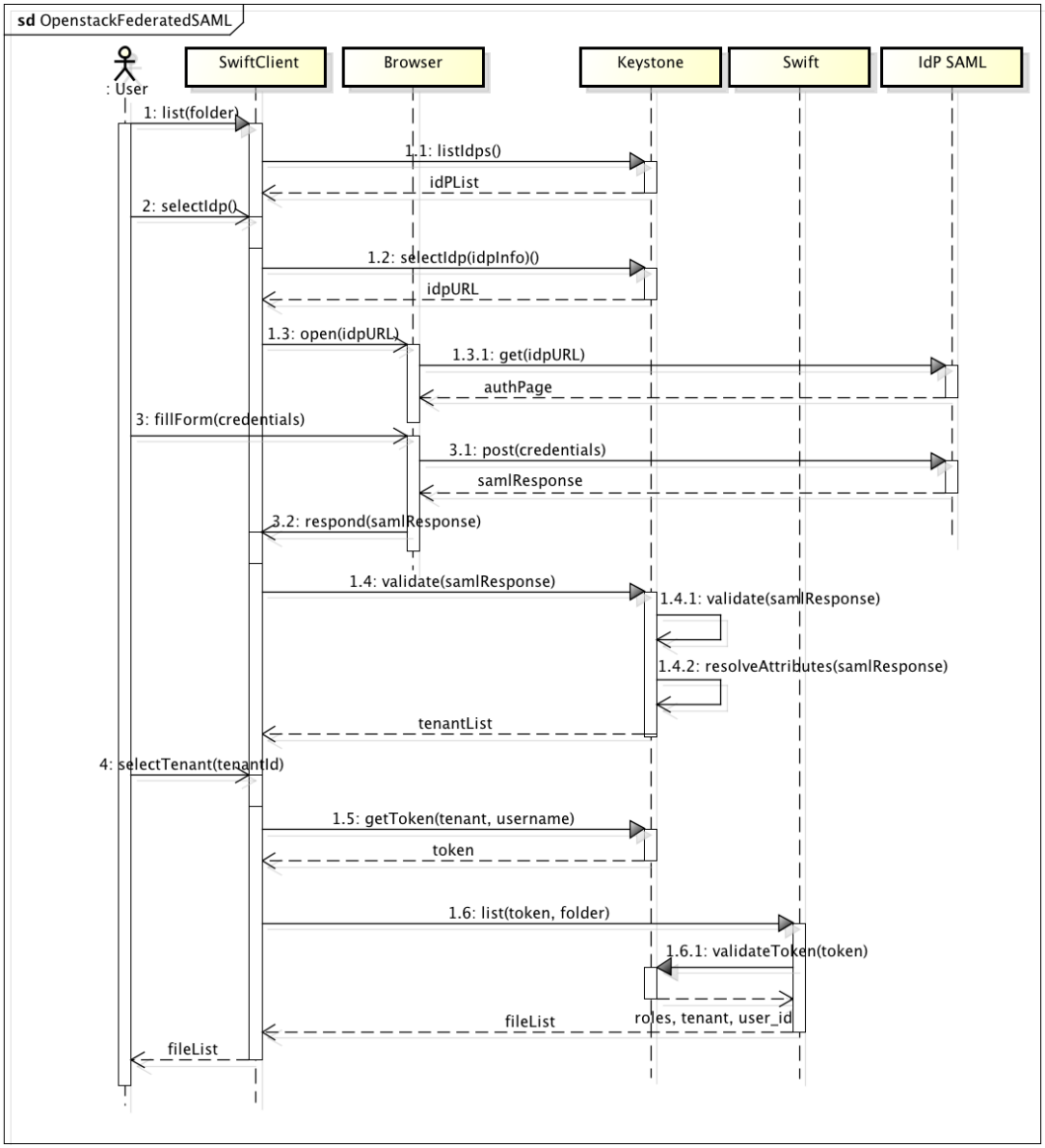



**Figura 5. Fluxo de Autenticação do Openstack federado com protocolo SAML**

O fluxo de autenticação do Openstack federado utilizando um IdP OpenID Connect com perfil de cliente básico é similar ao diagrama da Figura 5. A Figura 6 mostra apenas o

trecho onde ocorrem as diferenças. No passo 1.4, no lugar da mensagem SAMLResponse, o IdP OpenID Connect devolve um código (*code*) que identifica a autenticação (3.1). O SP possui credenciais (*clientId* e *clientSecret*) cadastradas previamente no IdP. No momento da validação, o Keystone solicita ao IdP um *token* para obter acesso aos dados do usuário, enviando suas credenciais e o código identificador da autenticação (1.4.1). Além do *token* de acesso, o Keystone recebe do IdP um *token* de identificação que serve para o SP garantir que está se comunicando com o IdP correto. O protocolo OpenID Connect define uma série de validações que o SP deve fazer sobre o *token* de identificação (1.4.2). Na sequência, o Keystone solicita ao IdP os atributos do usuário, passando o *token* de acesso recebido (1.4.3). Daí em diante, o fluxo continua de forma similar ao protocolo SAML.

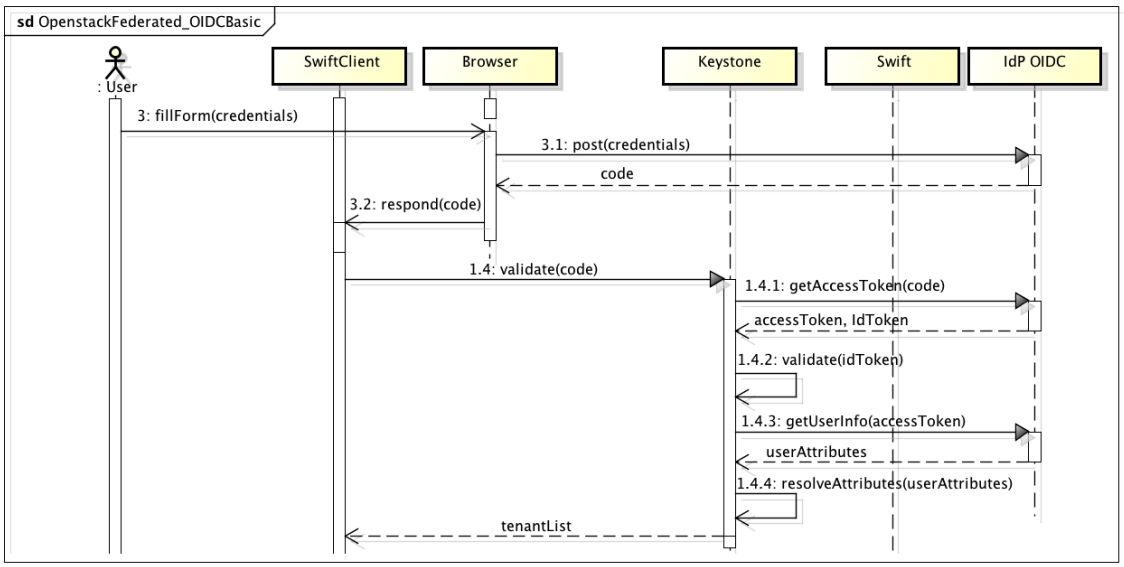



**Figura 6. Autenticação do Openstack federado usando OpenID Connect Basic Profile**

Para o Openstack federado utilizando um IdP OpenID Connect com perfil de cliente implícito, como fizemos no caso anterior, mostramos apenas o trecho com as diferenças na Figura 7. Em relação ao perfil de cliente básico, a diferença está no passo 1.4 que possui uma mensagem a menos. Desta vez, em vez de passar o código (*code*) para receber o *token* de acesso e *token* de identificação do IdP, o Keystone já recebe essas informações do "cliente swift".

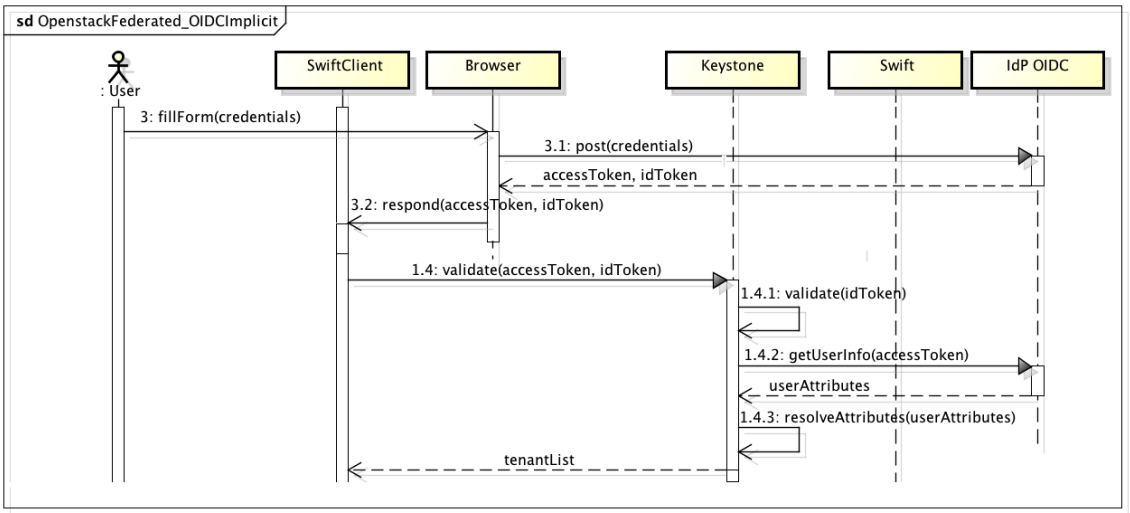



**Figura 7. Autenticação do Openstack federado usando OpenID Connect Implict Profile**

## 6. Experimentos

Esta seção descreve os experimentos realizados para comparação de desempenho entre os protocolos SAML com *binding* HTTP Redirect e perfil *Web Browser SSO*, e OpenID Connect, nos perfis de usuário básico e implícito, integrados ao Openstack.

### 6.1. Ambiente

O ambiente usado para a realização do experimento é composto por um provedor de serviços de nuvem Openstack (SP), um IdP SAML e um IdP OpenID Connect. Os aplicativos usados pelo usuário ("cliente swift" e navegador) foram simulados a partir da ferramenta JMeter (http://jmeter.apache.org), executada no mesmo ambiente do SP.

O SP é composto por um computador desktop com 3GB de RAM, processador Athlon dual-core de 2 GHz, sistema Linux Ubuntu 12.04 LTS. Os seguintes módulos da plataforma Openstack foram instalados: Swift na versão Grizzly seguindo a configuração Swift All-In-One (http://docs.openstack.org/developer/swift/development_saio.html) e Keystone, na versão federada proposta por Chadwick (2013a). O computador está instalado no Centro de Informática (CIn) da UFPE, conectado à RNP através de um link de 10 Gbps.

O IdP SAML foi instalado em uma VM com 512 MB de RAM, processador Xeon de 2.4 GHz, sistema Linux Ubuntu 12.04 LTS. O sistema utilizado é o SimpleSAMLphp (http://simplesamlphp.org), configurado para funcionar com *binding* HTTP Redirect. O servidor pertence ao Grupo de Trabalho de Computação em Nuvem para Ciência (GT-CNC) da RNP e está no DIMAP/UFRN, também conectado à RNP através de link de 10 Gbps.

O IdP OpenID Connect foi instalado em uma VM com 1GB de RAM, processador Xeon de 2.4GHz, sistema Linux Ubuntu 12.04 LTS. O sistema usado é o MITREid Connect (http://id.mitre.org/connect), configurado para funcionar com perfis de usuário básico e implícito. O servidor faz parte do Laboratório de Gestão de Identidade (GID Lab) da RNP e está localizado no PoP-MA, conectado ao backbone da RNP através de link de 3Gbps.

### 6.2. Cenários de Teste

O Keystone foi configurado com dois projetos para usuários autenticados pelos IdP SAML e IdP OpenID Connect respectivamente. No projeto UFRN, os usuários com atributo "brEduAffiliationType" igual a "employee" possuem acesso irrestrito por possuírem papel federated_admin, enquanto os com atributo "student" apenas podem listar e receber arquivos, com papel federated_member. Já no projeto GIDLab, o usuário "josegidlab" possui o papel federated_admin e todos os outros, federated_member.

Para configurar este cenário, os passos descritos em [Siu 2012] e [Chadwick 2013b] foram seguidos. Eles servem para mapear atributos do IdP (id do usuário, email etc.) com os atributos do SP (papéis, projetos etc.).

O JMeter foi configurado para realizar 300 repetições sequenciais da listagem de uma pasta no Swift para cada IdP. Os tempos e a quantidade de bytes trafegados são medidos para cada um dos 6 passos definidos na Figura 3.

A comunicação entre o navegador e o IdP SAML é criptografada utilizando o protocolo HTTPS. Todas as demais conexões são realizadas com o protocolo HTTP, sem criptografia. A quantidade de bytes medida não contempla a criptografia quando o HTTP é usado.

Não consideramos no teste o tempo de autenticação nos IdPs. Para isto, os usuários são previamente autenticados nos dois IdPs e os *cookies* de seção dos provedores são configurados no JMeter. Desta forma, os IdPs, através do mecanismo de SSO, retornam as mensagens de autenticação sem solicitar novamente as credenciais do usuário.

### 6.3. Resultados

Os tempos para realização de cada passo (1 a 6) da Figura 3 e a quantidade de bytes recebidos em cada passo são apresentados na Tabela 2 e discutidos na sequência. Em relação aos tempos medidos, observamos que os mesmos não apresentaram uma distribuição normal. Portanto, comparamos as medianas das amostras coletadas para os protocolos SAML e OpenID Connect utilizando o teste de hipótese não paramétrico de Kruskal-Wallis [Montgomery 2003 p.589-591], uma extensão do teste de Mann-Whitney-Wilcoxon [Montgomery 2003 p.585-588] para mais de duas amostras. Consideramos um intervalo de confiança (IC) de 95% nas análises.

**Tabela 2. Tempo e quantidade de bytes recebidos em 300 repetições de chamadas ao servidor Swift para listar uma pasta vazia usando protocolos SAML e OpenID Connect**

| *Variável* | *Passo* | *SAML WebApp SSO profile* | *OpenID Connect Basic profile* | *OpenID Connect Implicit profile* | *p* |
|---|---|---|---|---|---|
| Tempo (em ms) | 1 | 8 (7-8) | 8 (7-8) | 8 (7-8) | 0,00 |
| | 2 | 20 (19-20) | 13 (13-14) | 13 (13-14) | 0,00 |
| | 3 | 47 (46-48) | 40 (38-55) | 95,5 (77,75-171) | 0,00 |
| | 4 | 674,5 (620-997,5) | 1106 (886-1414) | 821 (764-1260) | 0,00 |
| | 5 | 73 (61-83,25) | 75 (61-88) | 75 (61-84) | 0,14 |
| | 6 | 76 (67-83) | 75,5 (68-83) | 76 (69-83) | 0,98 |
| | **Total** | **920 (854,8-1202)** | **1540 (1120-1656)** | **1172 (1061-1557)** | **0,00** |
| Bytes recebidos | 1 | 1553 (1553-1553) | 1553 (1553-1553) | 1554 (1554-1554) | 0,00 |
| | 2 | 1271 (1265-1279) | 256 (256-256) | 285 (285-285) | 0,00 |
| | 3 | 11930 (11930-11930) | 316 (316-316) | 1294 (1294-1294) | 0,00 |
| | 4 | 992 (992-992) | 968 (968-968) | 968 (968-968) | 0,00 |
| | 5 | 2610 (2610-2610) | 2610 (2610-2610) | 2609 (2609-2609) | 0,00 |
| | 6 | 168 (168-168) | 168 (168-168) | 168 (168-168) | 1 |
| | **Total** | **18530 (18520-18540)** | **5871 (5871-5871)** | **6878 (6878-6878)** | **0,00** |

O passo 1 representa a listagem dos IdPs disponíveis. Apesar dos testes de hipótese apontarem que as distribuições são diferentes (p=0), podemos observar que as medianas e intervalos interquartílicos dos tempos e da quantidade de bytes recebidos são bastante próximos. O passo 2 representa a seleção do IdP, onde os dados do IdP selecionado são retornados. No caso do SAML, os dados incluem a chave pública, o que faz com que sejam bem maiores comparados aos do OpenID Connect.

Nos passos 3 e 4, o cliente, que se comunicava com o SP (Keystone) nos passos anteriores, passa a se comunicar com o IdP. Como os IdPs estão fisicamente em localidades distintas e conectados com velocidades distintas (seção 6.1), medimos a latência entre o cliente e o SP, localizados na UFPE, e os IdPs SAML (UFRN) e OpenID Connect (PoP-MA), obtendo os resultados de 10ms e 32ms, respectivamente. Estes tempos foram descontados dos tempos medidos para cada chamada realizada aos IdPs.

O passo 3 representa a autenticação do usuário no IdP. O tempo maior do perfil implícito do OpenID Connect é justificável pelos dados retornados virem como fragmento, que precisam ser processados e uma requisição HTTP preparada para enviar os dados ao Keystone. Nos outros dois protocolos, os dados já vem formatados na URL de redirecionamento. O volume trafegado para o perfil implícito também é maior em relação ao perfil básico, uma vez que os *tokens* de autenticação vêm na resposta em lugar do identificador (*code*). O protocolo SAML é o que apresenta maior volume trafegado, uma vez que a mensagem no formato XML é maior, contendo chaves públicas, assinaturas e todos os dados de autenticação do usuário.

O passo 4 é o mais representativo em relação ao tempo total. Ele contempla a validação dos dados da autenticação, o mapeamento de atributos, e a listagem dos projetos associados ao usuário. As validações do protocolo SAML são computacionalmente

custosas, principalmente por incluir a validação da assinatura através de algoritmo RSA. Porém, o protocolo OpenID Connect apresenta um tempo maior devido às conexões entre SP e IdP, duas no perfil básico e uma no perfil implícito. O OpenID Connect, nos dois perfis, também realiza validações do *token* de identificação (*id token*) conforme explicado na seção 4.1. A resposta desta chamada é a lista de projetos disponíveis para o usuário, apresentando valores próximos apesar de estatisticamente diferentes.

A partir do passo 5, o cliente volta a falar com o SP (Keystone e Swift). As chamadas são as mesmas, independentemente do protocolo do IdP. O passo 5 representa a requisição do *token* Openstack, apresentando tempos e quantidades de bytes recebidos similares. O passo 6 representa a listagem de uma pasta contendo o mesmo arquivo, também apresentando tempos e bytes trafegados iguais (p=0,98 e p=1).

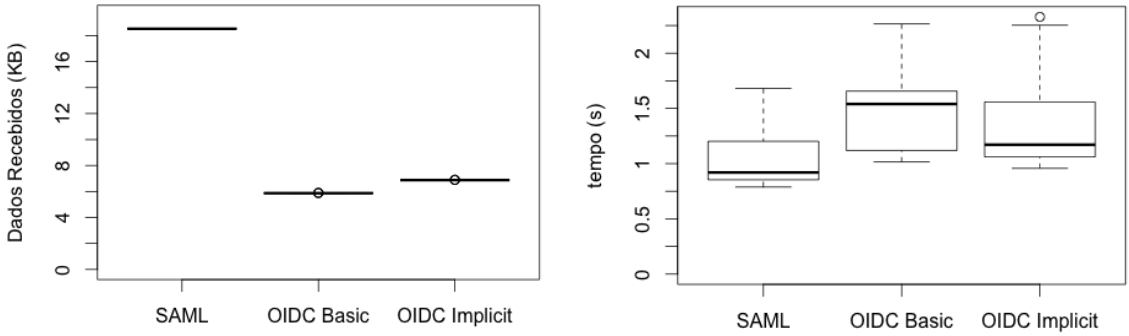



**Figura 8. Quantidade de bytes recebidos pelo cliente (a) e tempo total (b) em experimentos com protocolos SAML e OpenID Connect (OIDC) nos perfis básico e implícito.**

Podemos concluir que, no total dos passos, o protocolo SAML gera um maior volume de dados recebidos pelo cliente (18,5KB contra 5,9 e 6,9KB, respectivamente), mas realiza suas operações num tempo menor que o OpenID Connect (0,9s contra 1,5 e 1,2s), conforme visto na Figura 8. A diferença de tempo podia ser ainda maior em favor do SAML se o protocolo HTTPS fosse utilizado para as requisições ao IdP OpenID Connect. Entre os perfis básico e implícito do OpenID Connect, o implícito apresentou melhor desempenho com menor tempo e apenas 1KB de diferença. O tráfego entre SP e IdP existente no protocolo OpenID Connect não foi contabilizado, pois a medição foi realizada apenas nas interações entre cliente e provedores (SP e IdP). Se considerado, a diferença no total de tráfego gerado pelos protocolos diminuiria.

## 7. Conclusões e Trabalhos Futuros

Este trabalho considera que é vantajoso se integrar serviços, como plataformas de nuvem, a federações de identidades. Estas, possuem relacionamento próximo ao usuários, podendo fornecer informações importantes sobre eles.

Ao integrarmos a plataforma Openstack com federações através dos protocolos SAML e OpenID Connect, observamos que o SAML gera uma quantidade grande de informações no padrão XML, apesar dos experimentos utilizarem *binding HTTP Redirect* e perfil *Web Browser SSO*. A integração do SAML também exige maior esforço, por exemplo com a necessidade de geração e armazenamento de chaves criptográficas para assinatura das mensagens.

Da forma como foi configurado, o SAML comprova, entretanto, que o esforço para integração compensa, obtendo-se tempos finais menores devido à ausência de tráfego de rede entre IdP e SP. Além disso, o protocolo é mais maduro, suportando esquemas de atributos bem definidos no padrão X.500/LDAP.

A integração do Openstack com o protocolo OpenID Connect é simples, facilitada pelo protocolo em comum, o REST. Apesar de mais recente que o SAML, o OpenID Connect é utilizado por grandes provedores de serviços na Internet como Google, Paypal e Microsoft. O perfil de cliente implícito do OpenID Connect mostrou-se mais eficiente que o perfil básico, com tempo final próximo ao protocolo SAML. No entanto, sua implementação é mais complexa, necessitando de desenvolvimento no cliente (*browser*) para encaminhar os *tokens* ao SP. Ele também não elimina totalmente a comunicação entre SP e IdP.

Como extensões deste trabalho, serão consideradas uma análise de segurança, bem como a inclusão novos protocolos e padrões, por exemplo, o WS-Federation. Novos cenários também serão avaliados, como o uso de assinatura e criptografia das mensagens no protocolo OpenID Connect, a integração com a plataforma Shibboleth, possivelmente usando a solução de Diniz *et al* (2013) e a repetição dos testes usando o protocolo HTTPS no lugar do HTTP, tornando o cenário mais próximo de um caso real, onde a criptografia é necessária. Pretendemos ainda propor um modelo para autenticação e autorização de plataformas de nuvens através da integração com federações de identidade através de mecanismos como ABAC e $\text{UCON}_{\text{ABC}}$ [Tavizi *et al* 2012; Lazouski *et al* 2012; Marcon Junior *et al* 2013].

## Referências

Chadwick, D. W. (2013) “Federated Keystone”, http://wiki.openstack.org/wiki/Keystone/Federation/Blueprint, April.

Chadwick, D. W. (2013) “Handling ACLs that use UserIDs in Federated Keystone”, https://blueprints.launchpad.net/keystone/+spec/acls-userids-federation, April.

Chadwick, D. W. and Fatema, K. (2012) “A privacy preserving authorisation system for the cloud”, In: *Journal of Computer and System Sciences*, i.78, p.1359-1373. Elsevier.

Chadwick, D. W., Casenove, M. and Siu, K. (2013) “My private cloud – granting federated access to cloud resources” In: *Journal of Cloud Computing*, p.1-16. SpringerOpen.

Columbus, L. (2013) “Gartner Predicts Infrastructure Services Will Accelerate Cloud Computing Growth”, http://www.forbes.com/sites/louiscolumbus/2013/02/19/gartner-predicts-infrastructure-services-will-accelerate-cloud-computing-growth/, February.

Columbus, L. (2013) “Predicting Enterprise Cloud Computing Growth”, http://www.forbes.com/sites/louiscolumbus/2013/09/04/predicting-enterprise-cloud-computing-growth/, September.

Diniz, T. F. S., Silva, C. E. and Araujo, R. (2013) “Integrando o Openstack Keystone com Federações de Identidades”, In: *XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, p.465-474. SBC.

Eucalyptus (2013) “The Eucalyptus Cloud”, http://www.eucalyptus.com/eucalyptus-cloud/iaas, December.

Eucalyptus (2013) “Eucalyptus 3.4.0 Administration Guide”, http://www.eucalyptus.com/docs/eucalyptus/3.4/admin-guide-3.4.0.pdf, p.38-45, November.

IETF (2012) “The OAuth 2.0 Authorization Framework”, http://tools.ietf.org/html/rfc6749, October.

Karp, A., Haury, H. and Davis, M. H. (2009) "From ABAC to ZBAC: The Evolution of Access Control Models", http://www.hpl.hp.com/techreports/2009/HPL-2009-30.pdf, February.

Khan, R. H., Ylitalo, J. and Ahmed, A. S. (2011) "OpenID authentication as a service in OpenStack", In: *7th International Conference on Information Assurance and Security*, p.372-377. IEEE.

Lazouski, A. et al. (2012) "Usage Control in Cloud Systems", In: *The 7th International Conference for Internet Technology and Secured Transactions*, p.202-207. IEEE.

Lynch, L. (2011) "Inside the Identity Management Game", In: *IEEE Internet Computing*, v.15 (5), p.78-82. IEEE.

Marcon Junior, A. L. et al. (2013) "A UCON ABC Resilient Authorization Evaluation for Cloud Computing", In: *IEEE Transactions on Parallel and Distributed Systems*, p.1-11. IEEE.

Montgomery, D. C. and Runger, G. C. (2003) "Applied Statistics and Probability for Engineers", John Wiley & Sons, Inc., 3rd edition.

OASIS (2005) "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0", http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf, March.

OASIS (2005) "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf, March.

Openstack (2013) "Administration Guide", http://docs.openstack.org/grizzly/openstack-compute/admin/bk-compute-adminguide-grizzly.pdf, December.

Revar, A. G. and Bhavsar, M. D. (2011) "Securing User Authentication using Single Sign-On in Cloud Computing", In: *Nirma University International Conference on Engineering*, p.1-4. IEEE.

Sakimura, N. et al. (2013) "OpenID Connect Basic Client Implementer's Guide 1.0 - draft 29", http://openid.net/specs/openid-connect-basic-1_0.html, October.

Sakimura, N. et al. (2013) "OpenID Connect Implicit Client Implementer's Guide 1.0 - draft 12", http://openid.net/specs/openid-connect-implicit-1_0.html, October.

Sette, I. S. and Ferraz, C. A. G. (2013) "Integrando Openstack com Provedores de Identidade OpenID Connect e SAML: Uma Análise Comparativa" In: *XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, p.497-506, SBC.

Silva, L. M. et al. (2013) "Estudo de Caso: Integração de Clientes da Nuvem Openstack Swift Com Uma Federação de Identidade", In: *XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, p.455-464. SBC.

Siu, K. (2012) "A Role Mapping Service for the Keystone Identity Server", https://blueprints.launchpad.net/keystone/+spec/role-mapping-service-keystone, November.

Tavizi, T., Shajari, M. and Dodangeh, P. (2012) "A Usage Control Based Architecture for Cloud Environments", In: *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, p.1534-1539. IEEE.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 15*

## *Internet do Futuro*

# Um Algoritmo de Alocação de Largura de Banda para Tráfegos Elásticos em Redes de Circuito Dinâmico

**Diêgo B. M. de Moura**[1], **Leobino Sampaio**[1], **Gustavo Bittencourt**[1]

[1] Programa de Mestrado de Ciência da Computação (PMCC)
Instituto de Matemática - Universidade Federal da Bahia (UFBA)
Salvador, BA – Brazil

{diego.braga,gustavo}@dcc.ufba {leobino}@ufba.br

***Abstract.*** *Different profiles of applications may be supported by dynamic circuit network (DCN), which makes the problem of bandwidth allocation to highly challenging requests and of fundamental importance to the proper network performance. This article proposes a new algorithm for bandwidth allocation in DCN networks called Band Residual balancing (BBR). The BBR considers elastic applications and seeks to balance the residual band within the range of a requisition, thereby reducing the impact of advance reservations on the immediate reservations. Experiments performed through simulation showed that the BBR was 27% higher than the commonly used heuristics in the literature.*

***Resumo.*** *Diferentes perfis de aplicações poderão ser suportados pelas redes de circuito dinâmico (DCN), o que torna o problema de alocação de banda às requisições altamente desafiador e de fundamental importância para o bom desempenho da rede. Este artigo propõe um novo algoritmo para alocação de banda em redes DCN denominado de Balanceamento de Banda Residual (BBR). O BBR considera aplicações elásticas e procura balancear a banda residual dentro do intervalo de uma requisição, reduzindo assim o impacto das reservas antecipadas sobre as reservas imediatas. Experimentos realizados através de simulação mostraram que o BBR chegou a ser superior em 27% à heurística comumente utilizada na literatura.*

## 1. Introdução

Aplicações emergentes como distribuição de conteúdo e *backup* com transferências massivas de dados demandam grande largura de banda e possuem requisitos não compatíveis com o modelo de *melhor esforço* adotado na internet. Entre as soluções propostas para resolver esse problema, estão as redes de circuito dinâmico (DCN), as quais oferecem um serviço garantido e seguro, a partir de canais dedicados com alta largura de banda. A relevância desse serviço é percebida através da implantação dele em algumas das atuais redes de ensino e pesquisa, bem como em provedores de serviço [Internet2 2013, Veeraragha and Jukan 2010, eXtreme 2013, Verizon 2013].

Por meio da DCN, o usuário pode agendar sua requisição, especificando a origem e o destino do circuito, a taxa de transmissão e o tempo de início e fim no uso dos recursos. Entretanto, esse tipo de requisição, denominado em [Burchard et al. 2003] de ***reserva fixa***, favorece o problema de fragmentação da banda, fazendo com que a rede subutilize seus recursos. Para superar esse problema, Burchard et al. em [Burchard et al. 2003]

propuseram o modelo de ***reserva maleável***. Nesse modelo, ao invés dos usuários informarem exatamente o tempo de início e fim da transmissão, estes informam um intervalo dentro do qual seja aceitável realizar o escalonamento. Contudo, esse modelo de reserva não permite que a largura de banda seja variável, o que impede o aproveitamento de características específicas de algumas aplicações, dificultando um uso mais eficiente dos recursos. Dentro dos trabalhos que eliminam a necessidade da largura de banda ser fixa, a heurística mais comum utilizada pelos algoritmos de alocação de banda é o *Quick-Finish* (QF) [Naiksatam and Figueira 2007, Rajah et al. 2009, Patel et al. 2009]. Porém, essa heurística não é muito adequada para um ambiente em que coexistam aplicações heterogêneas, porque tende a criar instantes de tempo com completa ausência de largura de banda, acarretando o bloqueio de requisições que demandam uso imediato do recurso.

Diferentes perfis de aplicações poderão ser suportados pelas DCN's, possibilitando a coexistência de aplicações que requerem garantias mínimas de largura de banda com aplicações *elásticas*, em que o atraso e a flutuação na largura de banda são tolerados. Esta heterogeneidade, gerada pela coexistência de diferentes tipos de aplicações, faz com que as soluções para os problemas de alocação de banda às requisições exerçam um papel fundamental para o bom desempenho da rede em acomodar novas aplicações. Por tais motivos, novas estratégias para o provisionamento de recursos em DCN são bastante oportunas, uma vez que aplicações elásticas são bastante flexíveis e, por isso, permitem muitas opções de escalonamento. O desafio está em definir qual dessas opções utilizar, de forma a permitir uma maior utilização dos recursos, minimizando, assim, as chances de bloqueio de requisições futuras.

Este trabalho propõe um Algoritmo alternativo ao QF, denominado de Balanceamento de Banda Residual (BBR), para alocação de largura de banda em redes de circuito dinâmico. O algoritmo fornece um serviço determinístico para todas as aplicações e, ao mesmo tempo, reduz o impacto das reservas antecipadas sobre as reservas imediatas. Os resultados mostraram que o algoritmo proposto chega a ser superior em até 27% a heurística tradicional.

O restante do artigo está organizado como segue: A Seção 2 são apresentados os conceitos sobre redes de circuito dinâmico, os modelos de reserva que podem ser suportados pelas DCN e alguns trabalhos relacionados. A Seção 3 mostra o funcionamento do algoritmo proposto. Na Seção 4 são expostos os resultados da simulação e, por fim, na Seção 5 está a conclusão do artigo.

## 2. Redes de Circuito Dinâmico

Uma rede de circuito dinâmico é uma rede que oferece um *serviço* de transmissão que cria circuitos de curto prazo, com duração de minutos a dias, sob demanda, entre os usuários finais que necessitam de canais de largura de banda dedicada [Internet2 2013]. A maioria das redes que oferecem o serviço de circuito dinâmico, o fazem como complemento a sua rede existente. A Internet2 e a ESnet nos EUA são exemplos desse tipo de rede. Implantações similares também ocorrem na rede Européia GÉANT, e na asiática JGN-X [eXtreme 2013]. Para dar suporte ao estabelecimento de canais dedicados, essas infraestruturas utilizam *frameworks* que implementam um plano de controle, como é o caso por exemplo do OSCARS [Oscars 2014], amplamente difundido e implantado em redes que oferecem esse tipo de serviço. Assim como o OSCARS, existe o AutoBAHN

[AutoBAHN 2013], o UCLP [UCLP 2013] e outros.

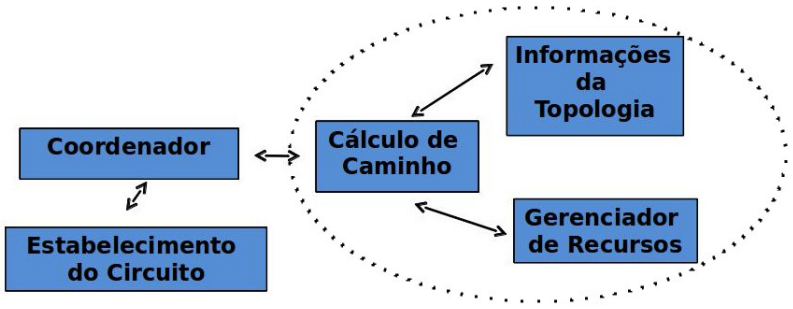



**Figura 1. Arquitetura Genérica para Alocação de Largura de Banda.**

Na Figura 1 é possível observar a arquitetura genérica para a alocação de largura de banda em DCN. Um módulo principal, denominado de coordenador é o responsável por receber solicitações de requisição e acionar o módulo de Cálculo de Caminho, que por sua vez utilizará os módulos de Informações da Topologia e Gerenciador de Recursos para saber quais as alocações existentes na rede e, decidir, se é possível ou não computar um caminho considerando as restrições da requisição. Uma vez calculado o caminho, o módulo Coordenador aciona o módulo Estabelecimento de Circuito que de fato irá configurar o circuito no plano de dados. O contexto desse trabalho só não abrange este último módulo.

### 2.1. Reserva de Largura de Banda

A reserva de recursos é uma maneira comum de prover qualidade de serviço (QoS) para aplicações. Existem dois tipos de reserva: reservas imediatas (RI) no qual os recursos são reservados no momento em que o pedido por eles é feito, e as reservas antecipadas (RA) em que os recursos são reservados muito tempo antes de serem utilizados. A RI é a forma tradicional de reserva em redes de circuito como é o caso da telefonia. Nesse tipo de reserva, a alocação do recurso coincide com a chegada da requisição. Já na RA, o recurso é alocado não no momento em que a requisição chega, mas em um tempo futuro. As reservas antecipadas tem recebido muita atenção nos últimos anos, sendo amplamente implementadas nas DCN's.

Um dos motivos para a maioria das DCN's adotarem a RA é devido ao fato desse modo de alocação diminuir a probabilidade da requisição ser bloqueada. Isso acontece, pois normalmente as aplicações destinadas ao modo RA demandam muito recurso, e quanto menor for a antecedência da reserva, maiores são as chances destes estarem sendo utilizados por outras requisições.

Para implementar o sistema de reserva antecipada é necessário definir o intervalo de tempo entre a chegada da requisição e o início do uso dos recursos. Esse intervalo é conhecido como *book-ahead time*. Os sistemas que utilizam a reserva antecipada normalmente definem um valor mínimo e máximo para o *book-ahead time*. O valor mínimo é um modo de evitar que a requisição seja tão curta a ponto de não ter diferença para uma RI. O valor máximo é para previnir um armazenamento muito grande de estados das requisições no sistema. O intervalo total em que as requisições podem ser feitas é comumente denominado de *horizon*. Esses elementos podem ser visto na Figura 2. O *horizon*

foi discretizado em intervalos de comprimento fixo chamados de *slots*, que é definido pelo operador da rede baseado nas características da sua topologia.

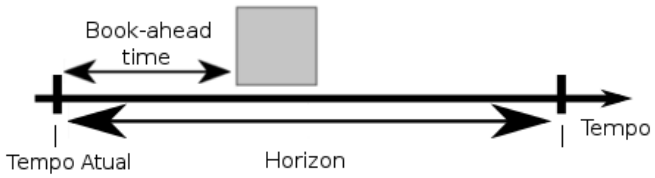
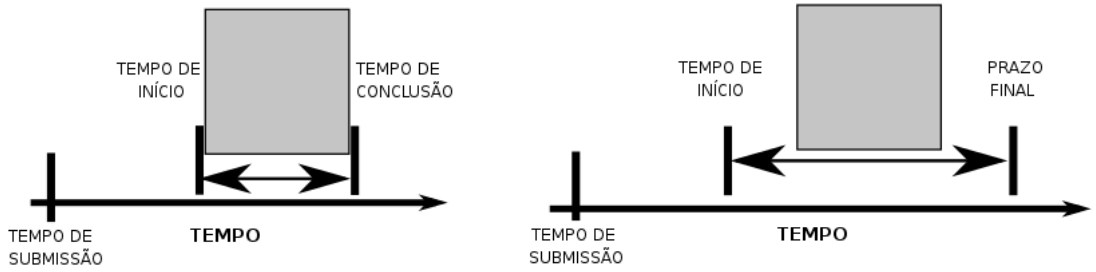



**Figura 2. Book-ahead time e o Horizon.**

A forma mais tradicional de realizar uma reserva antecipada é através da reserva fixa, em que o usuário define todos os parâmetros necessário para a aplicação, como por exemplo o tempo de inicio e fim da transmissão e a taxa de transmissão que permance sempre constante. Nesse modelo, representado pela Figura 3(a), como cada requisição define o seu tempo de início, é provável que existam intervalos ociosos entre requisições consecutivas. Quando esse intervalo é tão pequeno a ponto de não permitir uma nova alocação, esse recurso torna-se desperdiçado. Esse fenômeno, denominado de fragmentação da banda, foi estudado em [Burchard et al. 2003] sendo considerado um dos maiores elementos para diminuir o desempenho das redes com reservas antecipadas.

O conceito de reserva maleável proposto por Burchard et al. em [Burchard et al. 2003] trata o problema da fragmentação da banda. Nesse novo modelo de reserva, ao invés de especificar exatamente o tempo de início e fim no uso dos recursos, especifica-se um intervalo dentro do qual a requisição precisa ser satisfeita, como mostra a Figura 3(b). Entretanto, reservas maleáveis não permitem o uso de largura de banda variável.




(a) Reserva Fixa (b) Reserva Maleável

**Figura 3. Modelos de reserva, adaptada de [Netto et al. 2007].**

As ***reservas flexíveis*** por sua vez são aquelas que possuem as características das reservas maleáveis adicionando a possibilidade de a largura de banda ser variável [Naiksatam and Figueira 2007]. Embora o modelo de reserva flexível não possua restrições fixa de tempo de início, nem uma taxa de transmissão constante, esse modelo tem restrição com relação ao prazo de término da transmissão que precisa ser atendido. Esse tipo de reserva permite ao algoritmo de alocação de banda a possibilidade de preencher instantes do tempo em que não há qualquer alocação de recursos, minimizando assim a fragmentação da largura de banda.

Como é possível observar na Figura 3(b), a requisição possui um tempo de início e fim no qual a transmissão dos dados precisa ser concluída. Dentro desse intervalo, o algoritmo de alocação de banda pode aproveitar as características específicas do tráfego elástico, como é o caso da taxa de transmissão ser variável, e escolher-lá de modo a otimizar a utilização dos recursos da rede. Para qualquer tamanho de arquivo a ser transferido existe um tempo mínimo para conclusão dessa transmissão, que é baseado na máxima capacidade de rede. Quanto maior for o intervalo de uma requisição comparado a esse tempo mínimo, maior é a flexibilidade da requisição. Por exemplo, se uma requisição exigir uma demanda de 50 Gpbs para concluir a transmissão de um arquivo e a rede suporta apenas 10 Gbps, consequentemente serão necessários no mínimo 5 instantes de tempo para concluir a transmissão. Caso a flexibilidade da requisição seja 3, o intervalo para concluir a transmissão será 15 instantes de tempo, ou seja, a flexibilidade é multiplicada a partir desse valor mínimo.

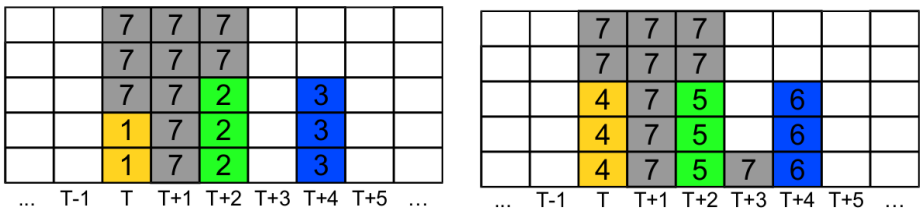
## 2.2. O Algoritmo Quick-Finish

Uma heurística bastante utilizada pelos algoritmos de alocação de largura de banda que permitem a largura de banda ser variável é o *Quick-Finish* (QF) [Naiksatam and Figueira 2007, Rajah et al. 2009, Patel et al. 2009]. Essa heurística tem como objetivo concluir o mais cedo possível a transmissão de uma dada requisição.

Uma reserva antecipada flexível é representada por $R$ = ($n^s$, $n^d$, $t^I$, $t^F$, D), em que $n^s$ e $n^d$ referem-se aos nós de origem e de destino do circuito, respectivamente; $t^I$ é o tempo mais cedo para iniciar a transmissão, $t^F$ é o tempo máximo para finalizar a transmissão e , por fim, $D$ representando a demanda da reserva representada através de *slots de largura de banda*. Vale destacar que a flexibilidade está embutida no intervalo [$t^I$, $t^F$]. Assim, a flexibilidade não é um parâmetro visível nos argumentos dos algoritmos.

Na heurística QF, o algoritmo inicialmente calcula os *k*-menores caminhos entre os nós de origem e de destino. Para cada caminho, inicia-se a busca por recursos a partir do instante $t^I$ até o $t^F$. Em cada instante, o algoritmo aloca a máxima quantidade de banda disponível. Essa busca continua nos instantes seguintes, até a quantidade de banda ser suficiente para atender a demanda da requisição ou até alcançar o prazo máximo de término da mesma. Ao analisar cada caminho, apenas os que puderem concluir a transmissão antes do instante $t^F$ são adicionados a uma lista com os respectivos valores de tempo de término da transmissão. O caminho que possuir o menor valor para o término é o escolhido.

Um exemplo simples de como o algoritmo QF escalona as requisições é descrito a seguir. Assume-se que em um sistema a cada instante de tempo seja possível alocar até 5 Gbps de largura de banda, abstraídos em *slots de largura de banda* de 1 Gbps cada. Além disso existem apenas 2 caminhos, sendo cada um deles com três requisições já aceitas. Essas requisições estão enumeradas de 1 à 6 como mostra a Figura 4(a) e 4(b).




(a) Caminho 1 (b) Caminho 2

**Figura 4. Requisições aceitas nos caminhos usando QF.**

Suponha que uma nova requisição enumerada de 7 chegue a esse sistema com os intervalos para início e término sendo iguais a T e T+5 e que demande 10 Gbps. Conforme mostra a Figura 4, no caminho 1 apenas o primeiro instante do intervalo da requisição possui mais recursos que o caminho 2. Porém, esse primeiro instante é suficiente para fazer com que a requisição no caminho 1 termine no instante T+2 e, por isso, seja escolhido tendo em vista que no caminho 2 a requisição só concluiria no instante T+3.

### 2.3. Trabalhos Relacionados

A utilização das reservas antecipadas normalmente degrada a utilização dos recursos devido à fragmentação da largura de banda discutida na Seção 2.1. Para resolver o problema, Burchard et al. em [Burchard et al. 2003] propuseram o conceito de reserva maleável, que também é utilizado neste trabalho, porém sem a restrição da largura de banda ser constante e sendo permitido a alocação em instantes não contíguos.

Em [Balman et al. 2010], os autores assumem utilizar reservas maleáveis para agendar larguras de banda sem que seja permitido que a mesma seja variável. Consequentemente isto impede o aproveitamento das características específicas de algumas aplicações, de modo a melhorar a eficiência na utilização dos recursos.

Em uma evolução do trabalho desenvolvido, em [Balman 2013] permite-se que requisições já aceitas possam ser realocadas para admitir novas requisições. A realocação consiste em analisar cada requisição já alocada dentro do intervalo de tempo da nova requisição, e verificar se é possível rearranjá-la de tal forma que se garanta que as requisições já aceitas somada a requisição mais recente possam ser admitidas no sistema, sem que nenhuma restrição das mesmas sejam violadas. O problema com essa abordagem é a questão de escalabilidade, uma vez que o reescalonamento é feito no modo *online*.

Em [Rajah et al. 2009], os autores utilizam a heurística QF com e sem reescalonamento online. Porém, o trabalho não faz uso das reservas antecipadas em conjunto com as reservas imediatas. Em [Naiksatam and Figueira 2007], os autores propuseram um algoritmo que inicialmente faz a alocação da banda utilizando a heurística QF. Quando esse modo de alocação falha, permite-se fazer um reescalonamento online nas requisições já admitidas com o intuito de aceitar a nova requisição. O problema desse algoritmo é a escalabilidade, além de ser limitado a uma rede com um único *link*. Nota-se em [Patel et al. 2009] que os autores também utilizam a heurística QF e não permitem o reescalonamento. Porém, a requisição não tem um prazo para concluir a transmissão e não é permitido a coexistência de RA com RI.

Alguns trabalhos também permitem fazer realocações de requisições no modo

*offline* [Schmidt and Kunegis 2007, Shen et al. 2007]. Em [Schmidt and Kunegis 2007], os autores permitem fazer mudanças apenas no caminho da requisição, sendo transparente para o usuário. Porém, esse trabalho não aborda o uso de largura de banda variável nem faz faz uso de RA em conjunto com RI. Em [Schmidt and Kunegis 2007, Shen et al. 2007] o foco é voltado para o problema de roteamento e atribuição de comprimento de onda.

Em [Sharma et al. 2011], as reservas também são maleáveis, porém as mesmas não são acomodadas uma a uma de acordo com a ordem de chegada, mas todas de uma única vez. Diferentemente dessa abordagem, este trabalho utiliza um modelo mais realista no qual não se tem conhecimento das requisições futuras. Em [Patel and Jue 2011] já é permitido o uso da largura de banda variável, porém as requisições não possuem prazo para concluir a transmissão.

## 3. O Algoritmo Balanceamento de Banda Residual (BBR)

A heurística QF não é muito adequada para um ambiente em que coexistam reservas antecipadas e reservas imediatas. Isto acontece, pois esse modo de escalonamento cria variações de banda residual no tempo sobre o *link*, gerando muitos instantes de tempo com nenhuma capacidade de banda disponível e outros com total capacidade de banda disponível. Como consequência, as requisições imediatas acabam tendo dificuldades de encontrar períodos contínuos de tempo com recurso disponível. O algoritmo de Balanceamento de Banda Residual (BBR), por sua vez, consiste numa proposta que procura balancear a banda residual no intervalo de uma dada requisição. Entre seus objetivos, está a redução do impacto das reservas antecipadas na aceitação das reservas imediatas.

Por balanceamento entenda-se deixar os instantes de tempo com a mesma banda residual. Além disso, uma variabilidade muito grande nos recursos ao longo do tempo dificulta a atribuição de banda em períodos contínuos. Por esse motivo, o algoritmo BBR tem o intuito de fazer um esquema de balanceamento que consiste em tentar deixar os diversos instantes de tempo com a mesma banda residual. Para isso, o BBR procura alocar mais recursos no instantes com maior disponibilidade, e menos, no que tiver menor disponibilidade. Através dessa estratégia, o BBR evita que em determinados períodos a banda seja exaurida e, consequentemente, bloquei as requisições imediatas. O funcionamento do BBR é descrito formalmente através dos Algoritmos 2 e 3.

Inicialmente o algoritmo computa os *k*-menores caminhos entre os nodos de origem e de destino (Linha 1 do Algoritmo 2). Em seguida é calculada a máxima capacidade de banda para cada instante dentro do intervalo $[t^I, t^F]$ utilizando a função MaxBandwidth (Linhas 3 à 5 do Algoritmo 2). A partir daí, com os valores de capacidade máxima de largura de banda, verifica-se se é possível aceitar ou não a requisição. Caso sim, faz-se um cálculo para atribuir, de forma proporcional a banda nesses instantes, a demanda da requisição (Linhas 7 à 9 do Algoritmo 2). O objetivo com essa alocação é uniformizar, balancear a banda residual no intervalo $[t^I, t^F]$. Ao final, nem sempre será possível atribuir a demanda da requisição de forma que todos os instantes, dentro do intervalo da requisição, possuam a mesma banda residual. Isso irá depender de como se encontra a banda residual nesses instantes antes da chegada da requisição e dos requerimentos da própria requisição.

**Algoritmo 1:** BBR

**Entrada**: $R$ = ($n^s$, $n^d$, $t^I$, $t^F$, D)
**Saída**: aceita ou rejeita a requisição

```
Computa k menores caminhos entre n^s e n^d;
for path = 0 to path <= K do
    for i = t^I to i <= t^F do
        sum+= MaxBandwidth(path, i);
    end
    if (sum >= D) then
        for i = t^I to i <= t^F do
            Aloca em i de forma proporcional a sua disponibilidade de
            recursos;
        end
        Insere o path em uma lista de soluções L;
    else
        Esse path não possui recursos;
    end
end
if Houver caminho em L then
    SelectPath (Algoritmo 3);
    Atualiza os recursos no path escolhido e aceita a requisição;
else
    rejeita a requisição;
end
```

**Algoritmo 2:** SelectPath

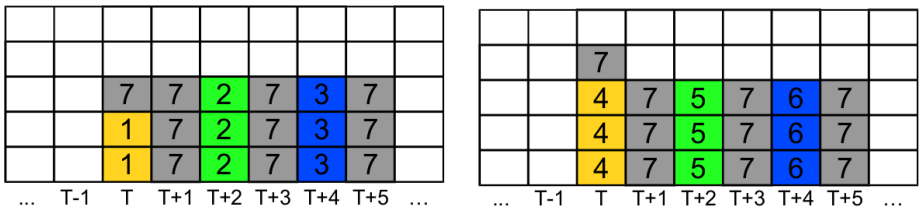
**Entrada**: Um ou mais caminhos entre $n^s$ e $n^d$
**Saída**: Um único caminho para o destino

```
if Houver mais de um caminho para alcançar o destino then
    Escolha o caminho que tiver a menor diferença entre a máxima e o
    mínima banda disponível no intervalo da requisição;
    if Se houver mais de uma opção then
        Escolha o caminho que tiver a maior média baseada na banda
        disponível em cada slot dentro do intervalo da requisição;
    else
        Escolhe o primeiro da lista
    end
else
    Retorna o único caminho obtido;
end
```

Assim, uma vez escalonado a requisição em diferentes caminhos, é provável que cada caminho, dentro do intervalo da requisição, possua banda residual distintas entre eles. Consequentemente, ao final de um escalonamento, teremos uma balanceamento da banda residual diferente para cada caminho. A política de seleção de caminho procura

selecionar o caminho com o melhor balanceamento nesse intervalo, isso é, o que possuir menor variação de disponibilidade de recurso após a alocação da requisição. Quando existe uma mesma variação em mais de um caminho, escolhe-se o que tiver a maior média de banda residual, pois é o caminho que possui mais recursos dentro do intervalo da requisição.

Assumindo o mesmo exemplo de estado da rede e requisições dado no algoritmo 1, a Figura 5(a) mostra o resultado do escalonamento usando o BBR para o caminho 1, e a Figura 5(b) o resultado utilizando o caminho 2.




(a) Caminho 1 (b) Caminho 2

**Figura 5. Requisições aceitas nos caminhos usando BBR.**

Como pode ser visto na Figura 5, o BBR procura dentro do intervalo de tempo da requisição, nesse caso específico [T, T+5], fazer um balanceamento da banda residual. Com isso, os instantes T+2 e T+3, que possuem a menor banda residual, foram os que menos alocações tiveram. Ao passo que os instantes T+1, T+3 e T+5 foram os que sofreram mais alocações por não terem nenhum recurso alocado neles no momento da chegada da requisição.

Essa abordagem tende a adiar o exaurimento da largura de banda nos em cada instante do tempo, além de favorecer uma maior agregação entre as reservas. Ao final da tentativa de alocar em diferentes caminhos, a política que seleciona um caminho, o faz no caminho que ao final da alocação obteve o melhor balanceamento. Nesse exemplo, o caminho 1 foi o selecionado por ter diferença 0 entre o máxima e o mínima banda residual disponível no intervalo da requisição, ao passo que no caminho 2 tem-se uma diferença de 1 *slot de largura de banda* entre a mínima e a máxima banda residual.

## 4. Exemplos Numéricos

Esta seção descreve o modelo de simulação utilizado para avaliar o desempenho dos algoritmos. A métrica de interesse é a probabilidade de aceitação das requisições. Cada experimento foi executado 10 vezes, com diferentes sementes e o intervalo de confiança para o valor médio foi calculado usando um nível de confiança de 95%. O tempo é discretizado em intervalos de tamanhos iguais [Guerin and Orda 2000], denominados de *slots*. O período total de simulação foi de 1000 *slots* e todas as requisições que chegam fora da janela de tempo são bloqueadas. A rede inicia com todos os seus recursos disponíveis.

São utilizados dois algoritmos para alocação de banda de reserva anteciapada: QF e o BBR. Ambos compartilham o mesmo algoritmo para alocação de reserva imediata. As RI são escalonadas no instante em que a requisição chega e a taxa de transferência é fixa durante toda a transmissão. Essas requisições são caracterizadas por demandarem pouca largura de banda comparado as reservas antecipadas, e exigem o uso de *slots* contíguos.

Essa restrição se dá pelo fato das RI serem representadas por aplicações inelásticas. Assim, não é permitido que sejam escalonadas em intervalos não contíguos. Já as RA demandam alta largura de banda e é permitida a alocação em *slots* não contíguos, pois são representadas por aplicações elásticas, que possuem alta flexibilidade para alocação.

A topologia utilizada está representada na Figura 6, sendo sua capacidade abstraída e representada através de *slots de largura de banda* como em [Burchard et al. 2003]. O *link* possui largura de banda de 10 Gbps abstraídos em 10 *slots* de 1 Gbps cada. As requisições foram geradas entres os pares 0 e 3.

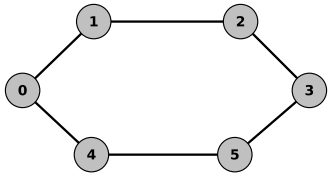



**Figura 6. Topologia**

As redes de pesquisas que fornecem o provisionamento de circuito sob demanda estão em fase de amadurecimento. Diversas pesquisas recentes, feitas inclusive pelos próprios projetistas dessas redes, sinalizam que mudanças devam ocorrer para melhorar o aproveitamento dessa nova metodologia de reserva. Embora a maioria das DCN's ainda destinem recursos exclusivos para RA, na prática, as reservas imediatas e as antecipadas devem coexistir. Portanto, aqui é permitido a coexistência de RA com RI, sendo os mesmos recursos compartilhados entre os dois tipos de requisições. Além disso, tanto a RA como a RI informam a duração da reserva e a largura de banda requerida. Sistemas que permitem a admissão de RI sem o conhecimento da sua duração, tentam através de alguma distribuição modelar a mesma, o que portanto impede o sistema de fornecer um QoS estrito [Wischik and Greenberg 1998, Greenberg et al. 1999].

Assume-se que as reservas imediatas pertencem a uma única classe de tráfego. A chegada das requisições imediatas segue a distribuição de Poisson. O número de *slots de largura de banda* requeridos são 2 e a duração de 3 instantes de tempo (ou 3 *slots*). As reservas antecipadas também seguem uma distribuissão de Poisson com demanda de 20 *slots de largura de banda* e *book-ahead time* uniformemente distribuído entre [10-30] *slots* respectivamente. A flexibilidade utilizada em cada simulação foi de 2, 3, 4 e 5. Os recursos foram distribuídos de forma igualitária entre as reservas antecipadas e as imediatas.

Primeiro foi avaliado o impacto que a carga da rede tem sobre a taxa de aceitação das reservas imediatas. Para isso, foi utilizado a flexibilidade 5. Para carga leve, 0.5 Erlangs, o algoritmo BBR foi superior ao QF em 22%, chegando a alcançar 27% a mais de aceitação com carga de 1 Erlangs, como mostra a Figura 7. Nessas condições de tráfego, a heurística BBR consegue se diferenciar mais do QF, o que faz resulta numa melhor aceitação das RI. Já para carga alta, 1.5 e 2 Erlangs, a maneira como o BBR e o QF escalonam suas requisições tendem a se aproximar, o que consequentemente diminui a diferença entre eles. Porém, ainda assim o BBR é superior em 8% ao utilizara carga 1.5 Erlangs. Para esse mesmo caso, a taxa de aceitação das reservas antecipadas foi a mesma para os dois algoritmos, BBR e o QF.

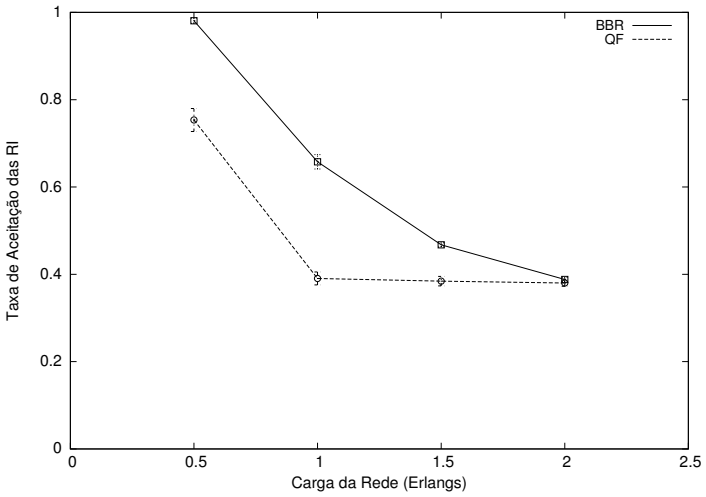



**Figura 7. Taxa de aceitação das RI para o QF e BBR em função da carga da rede.**

Assim, em ambientes que trabalham com uma demanda próximo da capacidade da rede e as requisições possuem uma boa flexibilidade, o algoritmo BBR se mostrou ser mais adequado que o algoritmo QF, pois além de possuir as mesmas taxas de aceitação de reserva antecipada que o algoritmo QF, conseguiu acomodar mais reservas imediatas.

Outro aspecto analisado foi o impacto que a flexibilidade de uma requisição tem sobre a taxa de aceitação. Foi utilizado a carga de 1 Erlangs. Na Figura 8, temos a taxa de aceitação para as reservas antecipadas. Como se pode observar, os dois algoritmos possuem praticamente as mesmas taxas de aceitação para as reservas antecipadas, sendo o QF é superior quando a flexibilidade é 2 em apenas 2%.

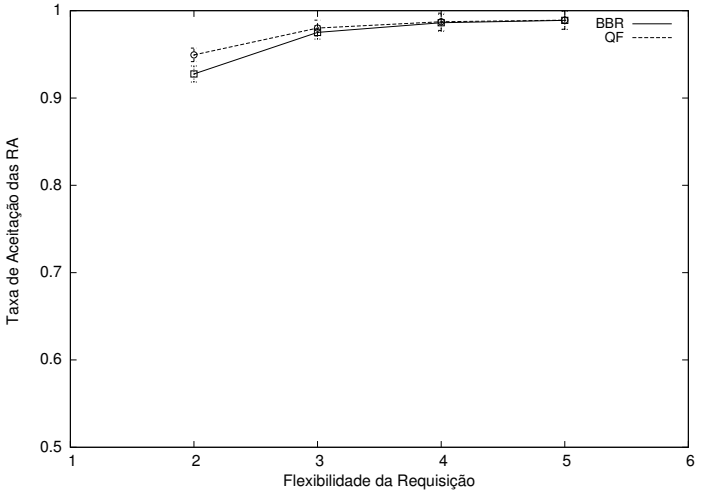



**Figura 8. Taxa de aceitação das RA para o QF e BBR em função da flexibilidade das requisições.**

O fato é que quando se tem uma flexibilidade muito pequena o algoritmo BBR não tem tanta influência no seu modo de alocação, pois o seu espaço para alocação dos recursos de forma proporcional se torna pequeno. Com relação a aceitação das reservas imediatas, ao aumentar a flexibilidade, aumenta a superioridade do algoritmo BBR em

relação ao QF, como mostra a Figura 9.

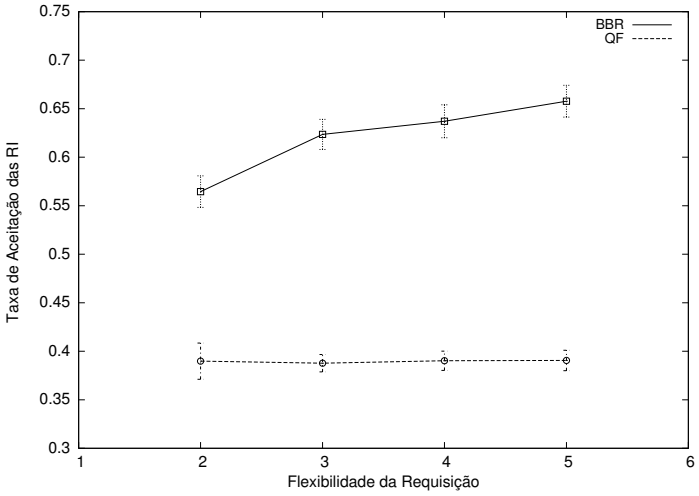



**Figura 9. Taxa de aceitação das RI para o QF e BBR em função da flexibilidade das requisições.**

Esses resultados mostram que quando a flexibilidade é muito pequena, o algoritmo BBR não tem tanta vantagenm sobre o QF. Pois para o caso no qual a flexibilidade é 2, o BBR é superior em 18% na aceitação das RI, porém é inferior à 2% em relação as RA. Já para flexibilidades maiores, não só o BBR tem a mesma taxa de aceitação para reservas antecipadas que o QF, como é também muito superior na taxa de aceitação das RI. Ao aumentar a flexibilidade, aumenta-se os instantes de tempo que o algoritmo BBR tem para tentar balancear a banda residual. O que favorece uma carga mais distribuída ao longo do tempo. Com isso, aumenta-se os efeitos dessa heurística sobre o QF.

Ao final da execução dos algoritmos, foram coletados o estado dos recursos após as alocações terem sido feitas. Três grupos de *slots* foram definidos: *slots* com Recurso Intermediário (SRI), *slots* com Recurso Completo (SRC) e *slots* com Recurso Esgotado (SRE). O recurso intermediário são aqueles *slots* que ao final da execução tinham banda disponível entre 1 e 9. Recurso completo e esgotado possuem 10 e 0 de banda disponível,

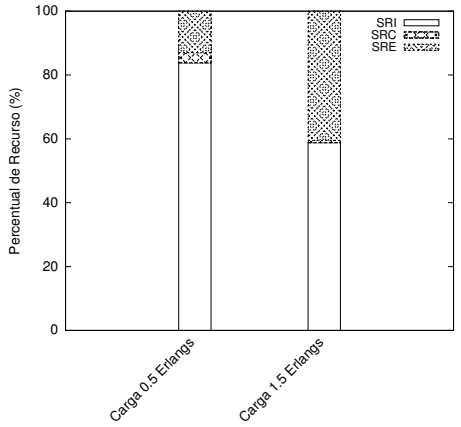



**Figura 10. Estado dos *slots* usando o algoritmo BBR.**

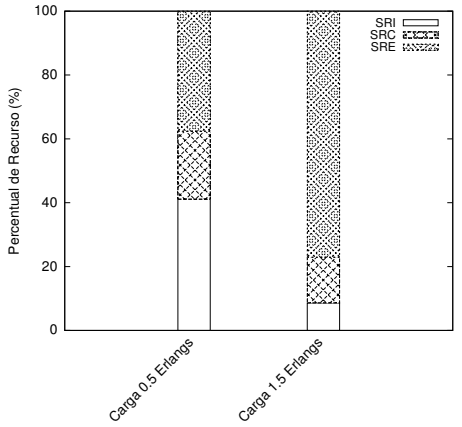



**Figura 11. Estado dos *slots* usando o algoritmo QF.**

respectivamente. Esses dados foram baseados usando a carga 0.5 e 1.5 Erlangs. Conforme mostra a Figura 10, no algoritmo BBR para carga leve, mais de 80% dos *slots* possuíam recursos intermediários. Mesmo para carga alta o algoritmo conseguiu alcançar quase 60% de *slots* com recursos intermediários. Já o algoritmo QF não chega a ter nem 9% de *slots* com recursos intermediários como mostra a Figura 11 para carga alta.

O que o QF faz é deixar muitos *slots* ou com total o nenhum recurso. Acontece que muito dos *slots* com total capacidade dos recursos estão entre *slots* com total capacidade exaurida, o que acaba impedindo o seu uso por reservas imediatas que não podem ser fragmentadas. Por isso, o algoritmo QF acaba bloqueando um maior número de requisições RI em relação ao algoritmo BBR.

## 5. Conclusões e Trabalhos Futuros

As redes de circuito dinâmico é a grande novidade para internet do futuro. Através delas será possível aumentar a rapidez no estabelecimento dos circuitos além de diminuir o custo de gerenciamento das redes. Ainda não se chegou a um consenso de como os *frameworks* que gerenciam as DCN's devem interagir com os usuários de forma a melhorar a utilização dos recursos e ao mesmo tempo atender as restrições dadas pelas aplicações. Um modelo de escalonamento que possa acomodar aplicações heterogêneas é de fundamental importância para um bom desempenho da rede.

Este trabalho mostrou como um modelo de escalonamento baseado no balanceamento da banda residual pode harmonizar a coexistência de aplicações com requerimentos distintos. Simulações mostraram que a nova proposta diminui o impacto das reservas antecipadas sobre a aceitação das reservas imediatas comparado ao método tradiconal de escalonamento. Para trabalhos futuros é necessário fazer um estudo com uma gama maior de topologias e ampliar uma maior número de classes de tráfego tanto para reservas imediatas como para reservas antecipadas.

## Referências

AutoBAHN (Abril, 2013). Available : `http://geant2.archive.geant.net/server/show/conWebDoc.1018.html`.

Balman, M. (2013). Advance resource provisioning in bulk data scheduling. In *Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications*.

Balman, M., Chaniotakisy, E., Shoshani, A., and Sim, A. (2010). A flexible reservation algorithm for advance network provisioning. In *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference*.

Burchard, L.-O., Heiss, H.-U., and De Rose, C. (2003). Performance issues of bandwidth reservations for grid computing. In *Computer Architecture and High Performance Computing, 2003. Proceedings. 15th Symposium on*, pages 82–90.

eXtreme, J. N. (Abril, 2013). Available : `http://www.jgn.nict.go.jp/english/reports/presentation/documents/apii_ws-2012_01.pdf`.

Greenberg, A. G., Srikant, R., and Whitt, W. (1999). Resource sharing for book-ahead and instantaneous-request calls. *IEEE/ACM Trans. Netw.*, 7(1):10–22.

Guerin, R. and Orda, A. (2000). Networks with advance reservations: the routing perspective. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 118–127 vol.1.

Internet2 (Abril, 2013). Available : `http://www.internet2.edu/pubs/DCN-howto.pdf`.

Naiksatam, S. and Figueira, S. (2007). Elastic reservations for efficient bandwidth utilization in lambdagrids. *Future Gener. Comput. Syst.*, 23(1):1–22.

Netto, M. A., Bubendorfer, K., and Buyya, R. (2007). Sla-based advance reservations with flexible and adaptive time qos parameters. In *Proceedings of the 5th international conference on Service-Oriented Computing*, ICSOC '07, pages 119–131, Berlin, Heidelberg. Springer-Verlag.

Oscars (Janeiro, 2014). Available : `http://www.es.net/services/oscars/`.

Patel, A., Hasan, M., Zhu, Y., and Jue, J. (2009). Routing and scheduling for variable bandwidth advance reservation in elastic applications. In *Optical Fiber Communication - incudes post deadline papers, 2009. OFC 2009. Conference on*, pages 1–3.

Patel, A. and Jue, J. (2011). Routing and scheduling for variable bandwidth advance reservation. *Optical Communications and Networking, IEEE/OSA Journal of*, 3(12):912–923.

Rajah, K., Ranka, S., and Xia, Y. (2009). Advance reservations and scheduling for bulk transfers in research networks. *Parallel and Distributed Systems, IEEE Transactions on*, 20(11):1682–1697.

Schmidt, S. and Kunegis, J. (2007). Scalable bandwidth optimization in advance reservation networks. In *Networks, 2007. ICON 2007. 15th IEEE International Conference on*, pages 95–100.

Sharma, S., Katramatos, D., and Yu, D. (2011). End-to-end network qos via scheduling of flexible resource reservation requests. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 68:1–68:10, New York, NY, USA. ACM.

Shen, L., Yang, X., Todimala, A., and Ramamurthy, B. (2007). A two-phase approach for dynamic lightpath scheduling in wdm optical networks. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 2412–2417.

UCLP (Abril, 2013). Available : `http://www.i2cat.net/en`.

Veeraragha, M. and Jukan, A. (Abril, 2010). Available : `http://www.ece.virginia.edu/mv/research/DOE09/documents/deliverables/feb2010/hybrid-arch-final.pdf`.

Verizon (Abril, 2013). Available : `http://www22.verizon.com/`.

Wischik, D. and Greenberg, A. (1998). Admission control for booking ahead shared resources. In *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 873–882 vol.2.

# Criação Automática de Circuitos Dinâmicos em Redes Híbridas Utilizando Regras de Filtragem de Tráfego

**Micael O. M. C. de Mello[1], Cleber de Souza Alcântara[1], Bruno Soares da Silva[1], Mário Augusto da Cruz[1], Sand Luz Corrêa[1], Kleber Vieira Cardoso[1]**

[1]Instituto de Informática (INF) – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brasil

{micaelmello,cleberalcantara,brunosoares,marioaugusto,sand,kleber}@inf.ufg.br

***Abstract.*** *Dynamic circuits can reduce queuing delays and provide alternative routes to network congestions. In general, dynamic circuits are available to users who need to transmit large volumes of data and the circuits are explicitly generated by the users or by custom application systems. In this paper, we present a proposal for automatic generation of dynamic circuits based on rules, similar to a conventional packet filter. We have implemented and evaluated our proposal in laboratory and in the RNP. The results confirm the performance gains achieved by using circuits, which are over 100% in some scenarios. In addition, the results illustrate the effectiveness of our solution to migrate traffic on-going from the conventional network to the circuit. We also present a qualitative evaluation of our solution related to deployment and adoption of a dynamic circuit service.*

***Resumo.*** *Circuitos dinâmicos podem reduzir atrasos de enfileiramento e oferecer rotas alternativas a congestionamentos. Em geral, circuitos dinâmicos visam atender usuários que precisam transmitir grandes volumes de dados, sendo criados através de solicitações explícitas desses usuários ou por meio de aplicações customizadas para esse fim. Neste trabalho, apresentamos uma proposta para criação automática de circuitos dinâmicos baseada em regras, de maneira similar a um filtro de pacotes convencional. Implementamos e avaliamos nossa proposta através de testes em laboratório e na RNP. Os resultados confirmam os ganhos de desempenho que podem ser obtidos com uso de circuitos, os quais podem ultrapassar 100% em determinados cenários. Adicionalmente, os resultados ilustram a eficiência da nossa solução para migrar tráfego em curso na rede convencional para o circuito dinâmico. Apresentamos também uma avaliação qualitativa da nossa solução em termos de implantação e adoção de um serviço de circuitos dinâmicos.*

## 1. Introdução

Redes de pesquisa e educação (*Research and Education Networks* – RENs) como APAN, Internet2, ESnet, GÉANT e RNP possuem uma necessidade crescente de transportar grandes volumes de dados de maneira confiável e eficiente. Grande parte dessa necessidade é justificada pela existência de equipamentos e aplicações em áreas como Física, Meteorologia, Medicina e Computação que, ao evoluírem, demandam maior capacidade de transmissão de dados. Para atender essa demanda crescente, diversas RENs têm adotado um projeto de rede híbrida [Monga et al. 2011] composta de dois núcleos: um deles

IP, baseado no modelo de melhor esforço e que carrega o tráfego convencional ou de produção; e outro núcleo de circuitos dinâmicos, baseado em qualidade de serviço e que transporta o tráfego de aplicações que fazem uso intenso de largura de banda. Redes de circuitos dinâmicos permitem que usuários finais requisitem a criação de circuitos virtuais individuais que atendam à demanda de suas aplicações. Além disso, esses circuitos podem ser usados como atalhos, reduzindo atrasos de enfileiramento, ou ainda, podem criar rotas alternativas que evitam congestionamentos.

Soluções de *middleware* como UCLP [Wu et al. 2006], AutoBAHN [GÉANT 2013] e OSCARS [Guok et al. 2006] têm sido amplamente implantadas em diversas redes híbridas com o intuito de permitir a configuração automática dos dispositivos de rede e o estabelecimento de circuitos dinâmicos interdomínio. Na prática, essas soluções atendem adequadamente administradores de rede e não usuários finais. Além de solicitar informações não triviais para um usuário leigo, como tipo ou ID de VLAN, essas soluções permitem apenas a criação de circuitos em horários específicos. Algumas demandas importantes podem não ser contempladas pelo fato do usuário não saber o horário em que uma determinada aplicação precisa realizar a transferência de dados. Por exemplo, o mecanismo de sincronização de grandes *storages* de dados pode ser ativado de maneira não determinística. Outra dificuldade para o usuário final é a necessidade de estender o circuito dinâmico até a rede de seu laboratório, uma vez que esse circuito pode terminar em outro ponto de sua instituição. Normalmente, essa atividade exige a configuração de múltiplos equipamentos de rede dentro da instituição. Além disso, fica a cargo do usuário definir o que deve fluir através do circuito dinâmico e o que deve ser enviado através da rede convencional. Essa separação é importante pois, tradicionalmente, os usuários precisam que alguns equipamentos ou algumas aplicações troquem tráfego através do circuito, enquanto o restante é trocado pela rede convencional, em geral, com a Internet.

Neste artigo, apresentamos a proposta de um sistema, chamado ATER – Aceleração do Transporte de Dados com o Emprego de Redes de Circuitos Dinâmicos [LABORA 2013], o qual permite aos usuários de uma REN criarem circuitos de forma automatizada através da definição de regras do perfil de tráfego, de maneira similar a um filtro de pacotes convencional. Após a definição de uma regra, o primeiro pacote combinando com a referida regra dá início a uma solicitação automática de criação de circuito e o restante do fluxo de pacotes gera solicitações, também automáticas, de renovação. Além disso, apenas o tráfego que casa com a regra é desviado para o circuito, enquanto o restante é automaticamente encaminhado pela rede convencional. Assim, o ATER torna mais simples o uso de circuitos dinâmicos, especialmente para usuários que não são especialistas em redes. Desenvolvemos a versão inicial do ATER em um GT (Grupo de Trabalho) da RNP e agora estamos avaliando como integrá-lo ao Serviço Experimental de CIrcuitos aPrOvisionados dinamicamente, ou SE-CIPÓ, da RNP.

Este trabalho está organizado da seguinte forma. A Seção 2 discute alguns trabalhos relacionados. Na Seção 3, descrevemos o sistema proposto e a infraestrutura de rede que torna viável sua implantação. Apresentamos a implementação do ATER na Seção 4. Na Seção 5, mostramos e discutimos os resultados da avaliação de desempenho através de experimentos realizados em laboratório e na RNP. Finalmente, a Seção 6 traz as conclusões e as perspectivas futuras para este trabalho.

## 2. Trabalhos Relacionados

Nos últimos anos, diversas RENs adotaram o modelo de rede híbrida em seus *backbones*, implantando tecnologias de *middleware* para circuitos dinâmicos. Por exemplo, na Internet2, o serviço de aprovisionamento de circuitos utiliza duas tecnologias. No nível mais baixo do serviço, localiza-se o DRAGON (*Dynamic Resource Allocation via GMPLS Optical Networks*) [Yang et al. 2006], responsável pelo estabelecimento de circuitos multi-plataforma. Acima dele, encontra-se o OSCARS (*On-demand Secure Circuits and Advance Reservation System*) [Guok et al. 2006], responsável pela reserva de circuitos interdomínio. No *backbone* da rede europeia GÉANT, outra solução de *middleware* denominada AutoBAHN (*Automated Bandwidth Allocation across Heterogeneous Networks*) [GÉANT 2013] foi implantada com o mesmo propósito. No Brasil, a RNP implantou e está testando o aprovisionamento de circuitos na rede experimental CIPÓ, ou RE-CIPÓ[1], a qual é baseada nas tecnologias DRAGON e OSCARS.

A disponibilidade de redes híbridas motivou o desenvolvimento de novos serviços, os quais têm sido desenvolvidos e implantados no topo das soluções de *middleware* para circuitos dinâmicos, com o intuito de agregar novas funcionalidades a essas tecnologias. Um exemplo é o Terapath [Katramatos et al. 2010], um arcabouço de serviços de rede que estende o circuito dinâmico, provido pela REN, até a rede do usuário final, garantindo qualidade de serviço ao longo de todo o caminho virtual. Para assegurar a largura de banda desejada na rede local do usuário, o Terapath faz uso de DiffServ, enquanto no *backbone* da REN essa propriedade é tratada pelo *middleware*. No entanto, para criar o caminho virtual fim-a-fim, o usuário ainda deve fazer sua solicitação explicitamente e também precisa providenciar a separação do tráfego a ser enviado pelo circuito dinâmico.

Uma solução para transferência de grandes volumes de dados entre *storages* distribuídos geograficamente foi apresentada e desenvolvida no projeto StorNet [Gu et al. 2011]. Nessa solução, duas instâncias pares de um *middleware* gerenciador de recursos de armazenamento se coordenam, juntamente com Terapath e OSCARS, para estabelecer, de forma automática, um caminho virtual fim-a-fim entre dois *storages* que necessitam transferir dados com garantia de qualidade de serviço. Contudo, como ocorre no próprio Terapath, o usuário deve indicar explicitamente quando deseja iniciar e encerrar a transferência de dados. Além disso, StorNet é uma solução customizada para *storages* e, portanto, demandaria reimplementação para uso em outras aplicações.

Outra iniciativa que estende as funcionalidades das tecnologias de circuitos dinâmicos foi desenvolvida no SE-CIPÓ e foi denominada MEICAN (*Management Environment for Inter-domain Circuits on Advance Network*) [de Santanna et al. 2012]. O MEICAN é atualmente a aplicação *Web* que o SE-CIPÓ oferece aos usuários para requisitarem a criação de circuitos dinâmicos. Além de oferecer uma interface amigável para o usuário, o MEICAN também implementa um *workflow* de gerenciamento que representa as políticas de operação de cada domínio. No entanto, como nas soluções anteriores, no MEICAN, o usuário deve indicar explicitamente quando deseja iniciar e encerrar a transferência de dados. Também continua sendo responsabilidade do usuário separar o tráfego que deseja enviar através da RE-CIPÓ do tráfego que quer enviar pela rede convencional.

[1] https://wiki.rnp.br/display/secipo/Home

## 3. Sistema Proposto

O propósito do sistema ATER é permitir o transporte de grandes volumes de dados através de circuitos dinâmicos, porém com um enfoque diferente dos anteriores. No ATER, os circuitos são criados automaticamente a partir de regras para identificação do perfil do tráfego. Essas regras são definidas através de uma interface *Web*, onde o usuário fornece apenas informações que descrevem o tráfego. O tráfego pode ser muito específico, exigindo, por exemplo, IPs de origem e destino, protocolo de transporte e portas de origem e destino. Porém, o tráfego pode ser bastante geral, sendo descrito, por exemplo, apenas pelos IPs de origem e destino. As regras podem ser definidas pelos usuários que enviam o tráfego – inicialmente, indivíduos de instituições que tenham necessidade de realizar transporte confiável de grandes fluxos de dados – e também por administradores pertencentes à equipe de redes de alguma instituição ou da própria RNP.

Quando um tráfego combina com uma regra definida no sistema, um circuito é estabelecido e o tráfego é interceptado e desviado para o circuito criado. Como o circuito é ativado sob demanda quando o tráfego começa a fluir, não é necessário indicar quando o circuito deve ser iniciado. Analogamente, também não é necessário indicar quando o circuito termina, pois ele é estendido automaticamente enquanto houver tráfego fluindo pelo circuito. Para que isto ocorra, no ATER, a duração de um circuito é definida em fatias fixas de tempo, cujo tamanho é indicado pelo usuário ou definido pelo administrador. Assim, antes de uma fatia de tempo se encerrar, o ATER verifica se ainda há tráfego fluindo pelo circuito. Caso afirmativo, é solicitada uma extensão do circuito por mais uma fatia de tempo. Caso não seja possível estender o circuito, o ATER volta a encaminhar o tráfego pela rede convencional, de maneira transparente. Quando o usuário não deseja mais utilizar uma regra, pode simplesmente removê-la.

Tipicamente, um usuário final acessa o ATER apenas com um perfil chamado cliente, o qual permite definir uma regra efetiva mas que depende de um usuário com perfil administrador para que essa regra seja aplicada. Alternativamente, um administrador pode criar regras de monitoramento. Nesse modo de operação, uma regra não pode dar origem a um circuito, mas todo tráfego que casar com a regra é contabilizado. As estatísticas obtidas podem ser usadas posteriormente para subsidiar a decisão de autorizar ou não a regra em modo efetivo.

A Figura 1 ilustra como o sistema ATER se acopla à RE-CIPÓ. O ATER possui dois componentes principais: o *Circuit Operation and Rule Establishment* ou CORE, o qual é mantido em um servidor, e *Rule Applier and Circuit Endpoint* ou RACE, o qual é executado em equipamentos intermediários entre a rede do usuário e a rede CIPÓ. No CORE, é hospedado o núcleo do sistema, onde estão: a interface de acesso ao sistema, a gerência de usuários, a gerência de regras, o armazenamento de estatísticas de tráfego, o módulo de comunicação com o SE-CIPÓ, o controle remoto dos RACEs, dentre outros elementos. Nos RACEs, os principais elementos presentes são as regras, a coleta das estatísticas e a configuração para conectar as redes dos usuários aos circuitos dinâmicos. Cada RACE é colocado na borda da RE-CIPÓ, ligando-se ao POD que é um kit de equipamentos usados como nós da rede CIPÓ. Além disso, cada RACE também se conecta à rede de produção da RNP, chamada Ipê.

Atualmente, o RACE é implementado em um computador convencional com algumas interfaces de rede. Inicialmente, duas interfaces do equipamento são colocadas em

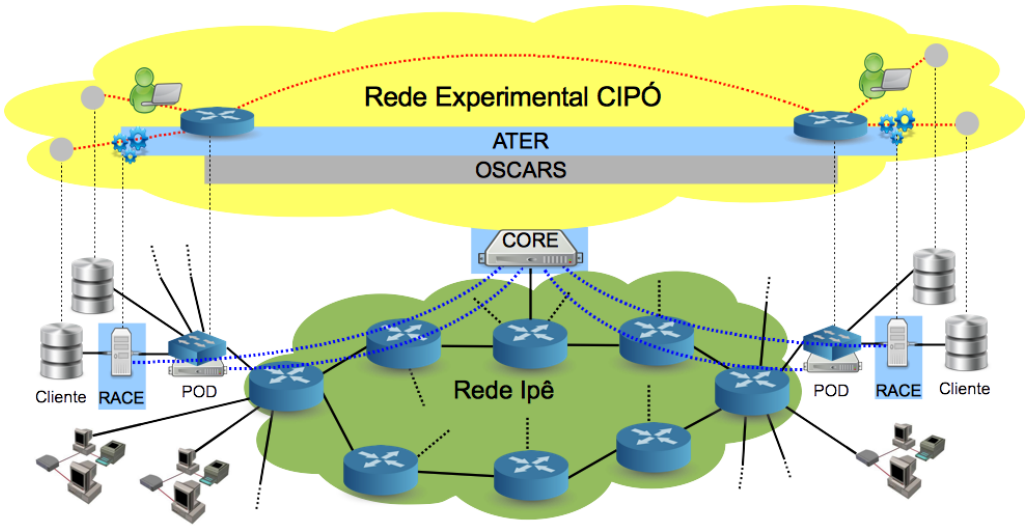



**Figura 1. Visão geral do sistema ATER.**

modo ponte (*bridge*), fazendo com que o encaminhamento seja realizado na camada de enlace. Essa abordagem ajuda a tornar um RACE transparente aos demais equipamentos no mesmo segmento de rede. Os RACEs trabalham aos pares, sendo coordenados pelo CORE para formar as extremidades dos circuitos, como descrito a seguir.

Após a aprovação de uma regra, é tarefa do CORE estabelecer um circuito dinâmico assim que é informado sobre o surgimento de tráfego que atenda à referida regra. Para isso, o CORE entra em contato com a API *Web Service* do OSCARS e solicita a criação do circuito. Quando o circuito se torna disponível, o RACE de origem inicia o processo de encaminhamento do tráfego de acordo com a regra que recebeu do CORE. Em geral, o encaminhamento pelo circuito dinâmico envolve a adição ou alteração de um rótulo de VLAN, para identificar o circuito, e a mudança do endereço MAC dos pacotes, para identificar o RACE que está na outra extremidade. O RACE de destino realiza a tarefa complementar, removendo ou alterando o rótulo de VLAN e substituindo o endereço MAC pelo do equipamento no próximo salto em direção à rede do usuário. Assim, os RACEs funcionam como comutadores de nível 2 (*switches*) que escondem os circuitos das redes que estão nas bordas.

A comunicação entre CORE e RACE é feita através de uma conexão segura, a partir da qual o RACE recebe toda a configuração necessária e pela qual informa sobre o tráfego que casa com alguma regra. CORE e RACE se monitoram mutuamente através de um protocolo simples do tipo *heartbeat*. Caso algum RACE não consiga estabelecer a comunicação com o servidor após algum tempo, todas as regras são desativadas e o equipamento passa a funcionar como um comutador simples, encaminhando de maneira convencional todos os pacotes. Do lado do CORE, a ausência de algum RACE significa que novos circuitos não podem ser iniciados ou terminados naquela localidade.

## 4. Implementação do Sistema

A Figura 2 ilustra os principais módulos do sistema ATER. Apresentamos a seguir informações sobre cada módulo.

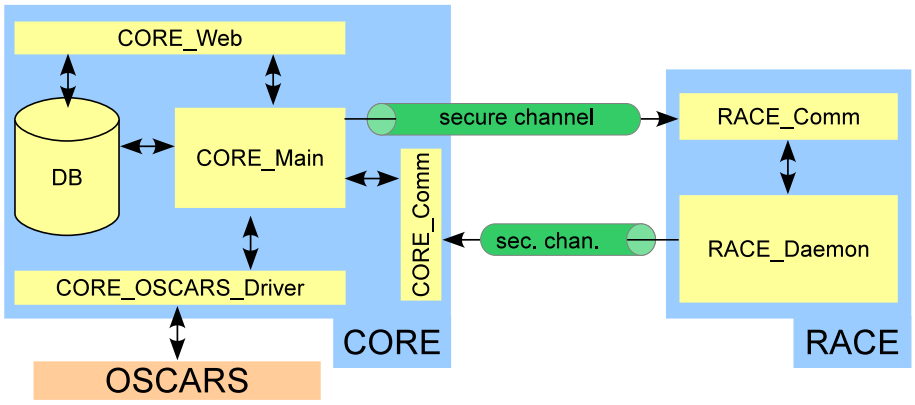



**Figura 2. Módulos de software do ATER.**

**CORE_Web** – consiste em uma aplicação *Web* através da qual os usuários podem criar regras de acordo com o perfil de seu tráfego, visualizar suas regras criadas e consultar estatísticas coletadas sobre essas regras. Esse módulo é também acessado pelos administradores, os quais utilizam a aplicação para gerenciar usuários, manter regras, monitorar estatísticas de tráfego, verificar o estado dos RACES, configurar o sistema e realizar outras tarefas administrativas. Esse módulo foi desenvolvido na linguagem PHP.

**CORE_Main** – consiste no núcleo do sistema de gerenciamento de regras, tendo dois componentes principais. Um deles é o Circuit_Manager, responsável por coordenar os RACES no estabelecimento da conectividade entre as extremidades do circuito. O outro componente é o RACE_Checker que implementa, pelo lado do CORE, o protocolo de *heartbeat* com os RACEs. Esse módulo foi implementado nas linguagens PHP e C.

**CORE_OSCARS_Driver (COD)** – módulo de software responsável pela integração com o OSCARS. Esse módulo foi implementado na linguagem Java e funciona como um *driver* para o OSCARS, criando uma abstração mais simples para o restante do sistema e oferecendo um ponto único de acesso ao SE-CIPÓ.

**RACE_Daemon** – módulo de software, implementado em linguagem C, que é o responsável pela lógica de filtragem e encaminhamento transparente dos pacotes. Para interceptar o tráfego, o RACE_Daemon utilizada uma solução nativa do núcleo do sistema operacional Linux, chamada *Netfilter*, em conjunto com *Linux bridge*. A configuração de regras no *Netfilter* é realizada através do *ebtables*. Para a reinjeção dos pacotes dentro do circuito e sua correspondente remoção na outra extremidade, é usada a API de *raw sockets*. O RACE_Daemon realiza ainda a coleta e envio de estatísticas para o CORE.

**CORE_Comm e RACE_Comm** – módulos de software instanciados no CORE e no RACE, respectivamente, os quais implementam um canal seguro, baseado na *libssh2*, para comunicação entre os componentes do sistema.

Para coordenar a operação do CORE e dos RACEs foram definidos alguns protocolos. A seguir, comentamos brevemente dois protocolos importantes, o que é usado para aplicar uma regra nos RACEs e o responsável por iniciar a criação automática de um circuito dinâmico. Por limitação de espaço, mostraremos apenas o fluxo regular, sem o tratamento de erros. A Figura 3 apresenta como os componentes do ATER interagem para criar uma regra e instalá-la nos respectivos RACEs. Inicialmente, o usuário acessa

o CORE_Web (CORE) e cria sua regra, descrevendo seu tráfego. Através dos endereços IPs fornecidos, o CORE_Web identifica os RACEs que respondem pela origem e o destino informados. O Circuit_Manager (CORE) envia uma mensagem `setRule(new)` para o RACE destino, para que esse equipamento instale a nova regra enviada. Se essa mensagem tem sucesso, o Circuit_Manager prossegue com a instalação da regra no RACE origem. Em caso de sucesso, a regra é registrada no CORE como aplicada.

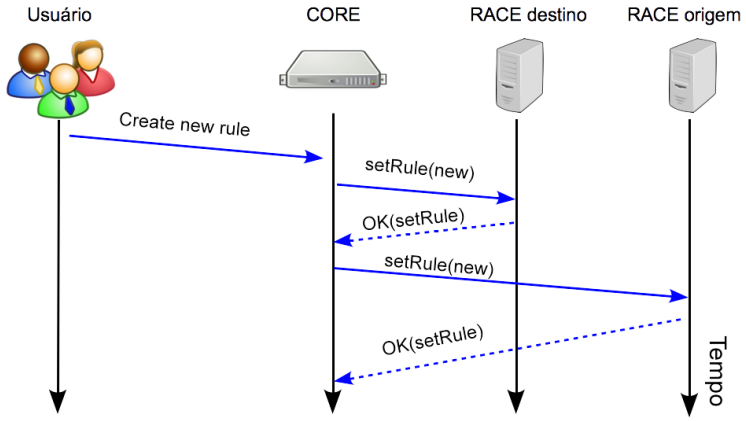



**Figura 3. Sinalização para instalação de regra nos RACEs.**

Quando ocorre o primeiro casamento de tráfego com a regra instalada no RACE origem, esse equipamento envia uma mensagem `setRuleMatch(new)` para o Circuit_Manager (CORE) iniciar a criação do circuito, como mostramos na Figura 4. Enquanto aguarda pela sinalização do CORE, o RACE origem continua encaminhando o tráfego pela rede convencional. O Circuit_Manager solicita a criação de um circuito ao COD, enviando a mensagem `setCircuit(new)`, o qual repassa a solicitação para o OSCARS. Após a confirmação de criação do circuito pelo COD, o Circuit_Manager notifica primeiramente o RACE destino sobre a existência do circuito através de uma mensagem `setCircuitEndPoint(new)`. Se essa mensagem for processada com sucesso, o Circuit_Manager prossegue notificando o RACE origem. Ao receber a mensagem `setCircuitEndPoint(new)`, o RACE origem reconhece que o circuito já está estabelecido e que o RACE destino está pronto para receber os dados. A partir de então, todo pacote que combinar com a regra será encaminhado pelo circuito.

É importante destacar que o acoplamento entre o sistema ATER e a rede da RNP existe apenas em um único ponto: o COD, o qual trata a mensagem `setCircuit`. Para que o ATER funcione com outras tecnologias como AutoBAHN ou UCLP é suficiente reescrever esse componente. CORE e RACEs não precisam ser alterados.

## 5. Avaliação

A avaliação do sistema ATER foi realizada em dois ambientes: nosso laboratório de pesquisa no Instituto de Informática (INF) da UFG e a rede experimental CIPÓ da RNP. Os testes no laboratório foram importantes para avaliar o sistema sob condições variadas e controladas de perda de pacotes e atraso fim-a-fim. No ambiente de laboratório, também pudemos analisar o cenário ideal, no qual a banda do circuito dinâmico é reservada. Quando os testes foram realizados, os circuitos dinâmicos da RE-CIPÓ ainda não possuíam reserva de banda. Os testes na RE-CIPÓ envolveram dois PoPs e permitiram

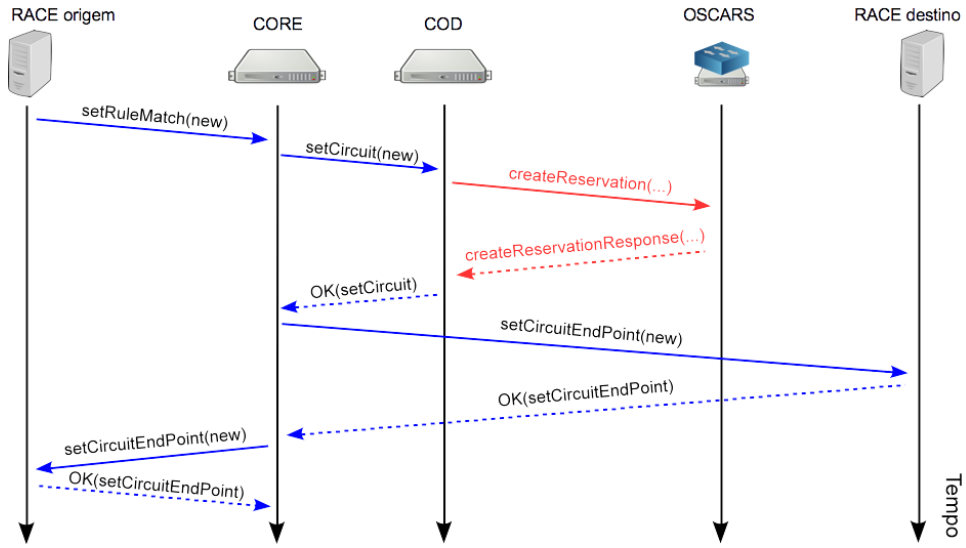



**Figura 4. Sinalização para criação automática do circuito dinâmico.**

ilustrar o potencial benefício dos circuitos dinâmicos, como a redução do atraso fim-a-fim. Adicionalmente, os testes no ambiente real nos permitiram mensurar o desempenho do ATER ao interagir com os demais componentes da RE-CIPÓ, em especial com o OSCARS. Para cada configuração avaliada, realizamos 30 testes e apresentamos os valores médios com nível de confiança de 95% no cálculo do intervalo de confiança.

### 5.1. Testes em Laboratório

A Figura 5 ilustra o ambiente de testes implantado no laboratório para avaliação do sistema ATER. O ambiente é composto por 6 computadores com arquitetura PC convencional e 1 comutador GbE de 24 portas, o qual não é mostrado na figura para melhor legibilidade. Os computadores que implementam os papéis de RACEs têm processador i5 de 3,2 GHz e 4 GB de RAM. O VCCOR possui processador i7 de 3,4 GHz e 8 GB de RAM. O VCCOR executa as máquinas virtuais do CORE, COD, OSCARS e um roteador, no qual são adicionados atrasos e perdas artificiais para avaliação. Os demais computadores possuem processadores menos potentes e menos memória, mas seus recursos de hardware são suficientes para cumprir os papéis que realizam, a saber: cliente A, cliente B e roteador. Os clientes utilizam o SO Ubuntu 12.04 e os demais equipamentos executam o SO Debian Linux 7.1, sendo que o VCCOR utiliza o Xen 4.1 como hipervisor.

Verificamos que o ambiente virtual utilizado apresentou um impacto negligenciável no desempenho fim-a-fim das transferências. Com os dois clientes ligados diretamente, sem nenhum equipamento intermediário, a vazão útil máxima obtida pelo TCP foi de 942 Mbps. Na configuração em que o fluxo TCP passa pelos RACEs e pelos dois roteadores, sem atrasos ou perdas artificiais, a vazão útil máxima alcançada foi de 940 Mbps. Para emular os atrasos e as perdas artificiais, empregamos a ferramenta `netem`[2]. Utilizamos o TCP CUBIC [Ha et al. 2008] no controle de congestionamento, pois esse é o módulo padrão da distribuição Linux dos clientes A e B. Utilizamos as recomendações apresentadas em [Augusto et al. 2008] para configuração básica de *buf-*

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

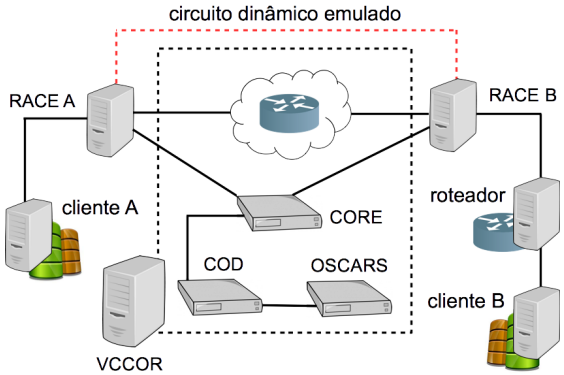



**Figura 5. Ambiente de testes implementado no laboratório.**

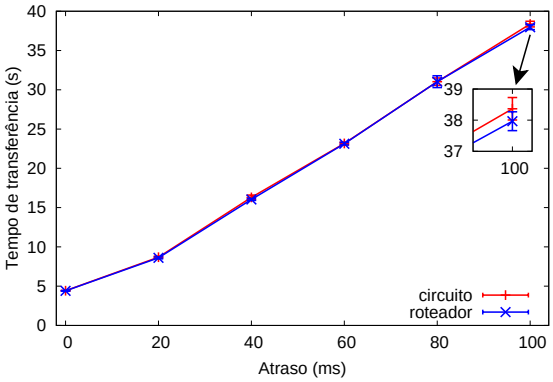



**Figura 6. Tempo de transferência em função do atraso fim-a-fim.**

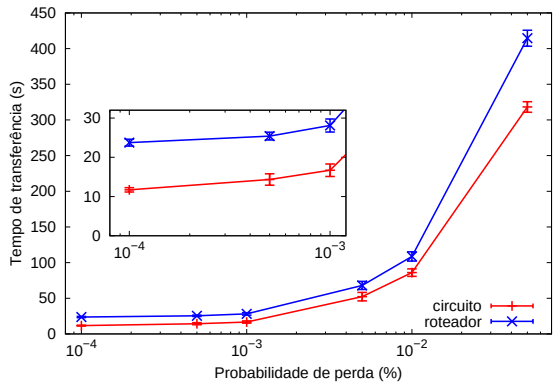



**Figura 7. Tempo de transferência em função da probabilidade de perda de pacotes.**

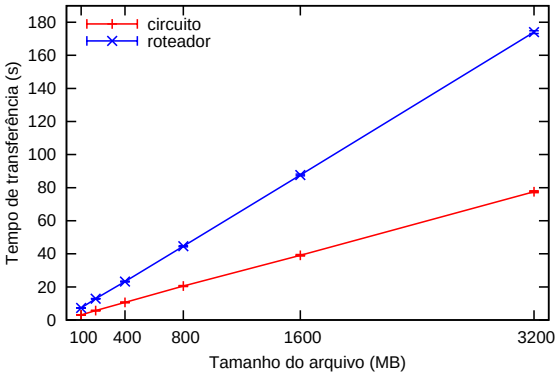



**Figura 8. Tempo de transferência em função do tamanho do arquivo a ser transferido.**

*fers* do TCP em redes com alto produto banda passante e atraso. No entanto, não realizamos um ajuste fino dos parâmetros do TCP e nem realizamos uma inspeção rigorosa em todos os componentes físicos envolvidos nos testes, o que justifica a diferença de aproximadamente 4% da taxa útil ideal, a qual se aproxima de 980 Mbps [da Silva 2006].

Inicialmente, avaliamos o desempenho do sistema ATER sob condições variadas de atraso. O intuito dessa avaliação é mensurar o impacto dos RACEs como elementos intermediários na comunicação. Há também uma pequena sobrecarga por conta do rótulo de VLAN adicionado a cada pacote enviado através do circuito, porém assumimos que o impacto dos 4 *bytes* adicionados é negligenciável. A Figura 6 mostra o tempo de transferência em função do atraso fim-a-fim entre os clientes A e B. Podemos observar que o impacto dos RACEs no desempenho é muito baixo, chegando a apenas 1% em seu máximo, conforme destacado na figura. Além disso, os equipamentos utilizados como RACE na RE-CIPÓ utilizam processadores mais rápidos que os empregados no laboratório e, portanto, o impacto no desempenho tende a ser ainda menor. Veremos na próxima seção, sobre testes na RNP, que é possível obter atrasos fim-a-fim nos circuitos dinâmicos menores que os da rede convencional de pacotes. Nessas condições, o ganho de desempenho ao utilizar circuitos se torna representativo.

Em seguida, avaliamos como a probabilidade de perda influencia o tempo de trans-

ferência em um cenário no qual o circuito dinâmico possui um atraso fim-a-fim diferente do caminho da rede convencional de pacotes. Utilizamos uma configuração para atrasos fim-a-fim, descrita na Seção 5.2, similar a que foi observada no ambiente real, sendo 25 ms para o circuito dinâmico e 60 ms para a rede convencional. A Figura 7 mostra como o tempo de transferência varia em função da probabilidade de perda de pacotes. Naturalmente, a transferência através do circuito dinâmico ocorre sempre em um tempo menor que por meio da rede convencional devido à diferença entre os atrasos fim-a-fim. À medida que a probabilidade de perda aumenta, a vantagem no uso do circuito tende a se tornar mais perceptível. No entanto, percentualmente, os maiores ganhos de desempenho são obtidos com baixas probabilidades de perda. Em geral, taxas de perda de pacotes altamente variáveis podem ser observadas em enlaces de acesso, devido a congestionamentos. Porém, alguns enlaces da RNP, como os que ligam São Paulo a Rio de Janeiro e a Minas Gerais, também já se aproximam da saturação em determinados períodos[3].

Para encerrar a avaliação em laboratório, apresentamos os resultados dos testes com diferentes tamanhos de arquivo. Novamente, utilizamos o atraso de 25 ms para o circuito e 60 ms para a rede convencional, porém não introduzimos perdas artificiais em nenhum dos dois caminhos. A Figura 8 apresenta o tempo de transferência em função do tamanho do arquivo a ser transferido tanto pelo circuito quanto pela rede convencional. Conforme esperado, os arquivos são transferidos mais rapidamente pelo caminho com menor atraso, ou seja, pelo circuito. À medida que o tamanho do arquivo aumenta, a vantagem do circuito se torna mais expressiva.

### 5.2. Testes na RNP

Os testes realizados na RE-CIPÓ envolveram dois PoPs da RNP, o de Goiás e o do Rio de Janeiro, como mostrado na Figura 9. No PoP-RJ, além de um RACE, foi utilizado um computador adicional para atuar como cliente do sistema ATER. O cliente do lado do PoP-GO foi implementado em uma máquina virtual de um servidor, localizado no INF/UFG. Esse servidor possui 2 processadores Intel Xeon de 2.20 GHz, com 4 núcleos cada, e utiliza SO Debian Linux 7.1 e hipervisor Xen 4.1. A comunicação entre a máquina virtual e o RACE do PoP-GO foi estabelecida através de uma VLAN, sem reserva de banda, a qual atravessa a rede da UFG e parte da rede metropolitana MetroGyn[4].

Durante o período em que realizamos os testes, a rota entre o PoP-GO e o PoP-RJ na rede Ipê era por MT, MS, PR e SP. No entanto, os circuitos dinâmicos eram criados por outra rota que tinha apenas DF como intermediário entre GO e RJ. Esse caminho além de ter menos saltos, apresenta menor congestionamento e menor atraso fim-a-fim em relação ao anterior. Em média, o atraso pela rede Ipê foi de aproximadamente 60 ms e pela RE-CIPÓ de 25 ms. Embora essa configuração não tenha sido estabelecida pelo sistema ATER, ela ilustra alguns benefícios potenciais de redes de circuitos dinâmicos. De fato, além de uma rota padrão pré-configurada, o SE-CIPÓ oferece a possibilidade de definir por quais domínios do OSCARS se deseja estabelecer cada circuito dinâmico.

Inicialmente, comparamos o desempenho de transferências de dados realizadas através de circuitos dinâmicos criados a partir do sistema ATER com transferências feitas na rede convencional. Para representar as transferências de dados, utilizamos a ferra-

[3]http://www.rnp.br/ceo/trafego/panorama.php
[4]http://www.redecomep.rnp.br/?consorcio=12

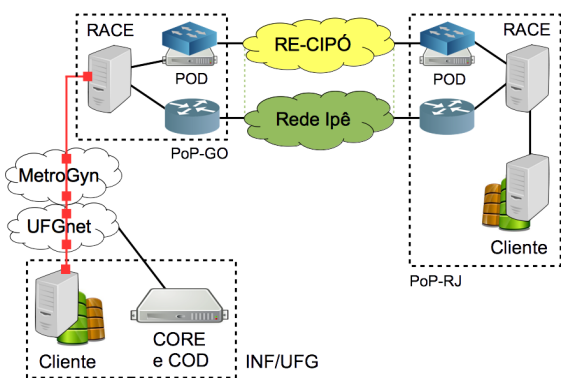



**Figura 9. Infraestrutura envolvida nos testes realizados na RE-CIPÓ.**

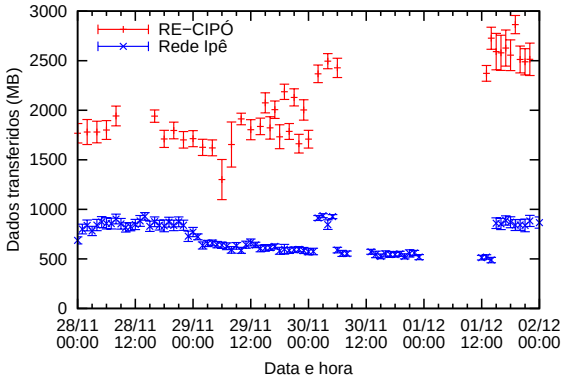



**Figura 10. Volume de dados transferidos ao longo do tempo.**

menta `Iperf` com protocolo TCP, pois assim podemos controlar com precisão o tempo de duração dos testes. Semelhante ao que foi feito em laboratório, aplicamos apenas as recomendações apresentadas em [Augusto et al. 2008] para configuração básica de *buffers* do TCP em redes com alto produto banda passante e atraso.

A Figura 10 apresenta a quantidade média de *bytes* transferidos pelo `Iperf` em 30 segundos. A média e o intervalo de confiança são calculados sobre 30 testes, os quais agendamos para rodar a cada hora, entre os dias 28 de novembro e 02 de dezembro de 2013. Conforme comentamos previamente, quando os testes foram realizados, a RE-CIPÓ ainda não oferecia reserva de banda para os circuitos. No entanto, cada circuito dinâmico está sujeito a policiamento de tráfego, ou seja, a vazão dos dados enviados pelo circuito não consegue exceder a capacidade indicada durante sua criação. Durante os testes, criamos circuitos dinâmicos de 700 Mbps a 1 Gbps conforme a disponibilidade, pois havia competição pela capacidade do enlace do PoP-RJ. Nesse contexto, o ganho de desempenho no uso de circuitos dinâmicos, o qual é próximo ou ultrapassa 100% na maior parte dos testes, se deve ao caminho com menor congestionamento e menor atraso fim-a-fim. Ou seja, o desempenho pode ser ainda melhor quando o recurso de banda reservada estiver disponível para os circuitos dinâmicos.

No período em que realizamos esses testes, a RE-CIPÓ estava acessível apenas em caráter experimental e, portanto, não havia compromisso ainda com disponibilidade da infraestrutura. Assim, já esperávamos por períodos de indisponibilidade em parte da rede, como os que ocorreram nos dias 28 e 30 de novembro de 2013. No dia 01 de dezembro, um problema de fornecimento de energia para um dos equipamentos do GT interrompeu temporariamente todos os testes.

Na RE-CIPÓ, tivemos a oportunidade de avaliar também aspectos operacionais do sistema ATER que são difíceis de emular com acurácia em laboratório. Assim, mensuramos a sobrecarga de controle adicionada pelo sistema ATER na criação de cada circuito dinâmico e a quantidade de pacotes perdidos e reordenados durante a comutação automática do tráfego da rede convencional para circuito dinâmico e vice-versa. A Tabela 1 apresenta os resultados da avaliação do ATER com relação a alguns aspectos operacionais relevantes. A média e o desvio padrão exibidos na tabela se referem a 160 testes.

O tempo total de criação de circuito dinâmico envolve todas as operações reali-

**Tabela 1. Aspectos operacionais do sistema ATER.**

| | Média | Desvio padrão |
|---|---|---|
| Tempo total para criação de um circuito | 70,93274 | 1,24841 |
| Tempo consumido pelo sistema ATER | 1,69507 | 0,70795 |
| Perdas ao comutar para circuito | 40,38125 | 63,65877 |
| Reordenações ao comutar para circuito | 283,10625 | 51,05301 |
| Perdas ao comutar para rede convencional | 34,70625 | 123,42128 |
| Reordenações ao comutar para rede convencional | 27,28125 | 9,95093 |

zadas pelo sistema ATER e pelo SE-CIPÓ, sobretudo pelo OSCARS, até que os pacotes possam começar a fluir pelo circuito. Dentre essas operações, algumas das que são realizadas pelo OSCARS consomem a maior parte do tempo, tais como computar a rota a partir de sua base de dados de topologia e a troca de mensagens entre todos os domínios envolvidos. O tempo consumido pelo sistema ATER corresponde à sua sobrecarga de controle, a qual envolve todas as operações desde que o primeiro pacote que casa com uma regra é detectado até o último RACE ser adequadamente configurado para enviar o tráfego através do circuito. No entanto, desse valor é descontado o tempo que o COD informa ter sido consumido até o retorno da API *Web Service* do OSCARS. Assim, podemos observar que o tempo consumido com atividades de controle do sistema ATER é negligenciável em relação ao tempo total de criação de um circuito dinâmico.

Para medir as perdas e reordenações de pacotes apresentadas na Tabela 1, utilizamos a ferramenta `Iperf` para enviar pacotes UDP a uma taxa de 100 Mbps, ou aproximadamente 8.500 pps, desde antes da criação do circuito, em ambos os sentidos. Adicionamos uma regra para que esses pacotes disparassem a criação de um circuito dinâmico. Após a criação do circuito, o sistema ATER realiza a migração do tráfego que está fluindo pela rede Ipê para o circuito dinâmico criado na RE-CIPÓ. Em condições normais, o circuito dinâmico é sempre renovado e não há a migração da RE-CIPÓ para a rede Ipê. No entanto, emulamos esse comportamento, modificando o sistema ATER para não renovar automaticamente o circuito. Assim, podemos avaliar o sistema quando um circuito não pode ser estendido por falta de recursos ou devido a alguma falha na RE-CIPÓ.

Como não é possível garantir sincronismo perfeito na configuração das tabelas de encaminhamento dos dois RACEs que terminam um circuito, o tráfego está sujeito a perdas e reordenações de pacotes durante a migração do tráfego entre a rede Ipê e a RE-CIPÓ, e vice-versa. Na Tabela 1, podemos observar que a quantidade média de perdas em ambos os sentidos é similar, com uma diferença inferior a 6 pacotes, ou seja, menos de 1 ms, dada a taxa de transmissão utilizada. Embora existam algumas ocorrências com quantidade elevada de perdas, como sugere o desvio padrão, verificamos que na maior parte das ocorrências a quantidade de perdas é menor que a média. Naturalmente, a quantidade de perdas é função do tempo consumido para concluir a migração do fluxo bidirecional de pacotes entre as redes. Dada a taxa de transmissão e a média de perdas, podemos estimar que esse tempo de comutação do tráfego é inferior 10 ms em média. As reordenações estão associadas à memória da rede, sobretudo por conta dos *buffers* dos roteadores intermediários. Portanto, já esperávamos que a quantidade de reordenações ao migrar da rede Ipê para a RE-CIPÓ fosse sensivelmente mais alta que o número de reordenações na migração inversa.

### 5.3. Avaliação Qualitativa

Atualmente, o usuário do SE-CIPÓ deve informar quando deseja criar e encerrar os seus circuitos dinâmicos. Além disso, o usuário é responsável por separar o tráfego que deseja enviar através da RE-CIPÓ do tráfego que quer enviar pela rede convencional. É importante lembrar que o acesso à Internet continua sendo feito pela rede de produção Ipê, enquanto comunicações entre pares específicos, geralmente dentro da própria RNP, podem ser estabelecidas através da RE-CIPÓ.

No contexto apresentado, o sistema ATER facilita o uso dos circuitos dinâmicos porque não exige que o usuário informe quando deseja enviar ou receber o tráfego e nem estimar quanto tempo deve durar a transmissão. O usuário precisa apenas informar, através de regras de filtragem, o perfil do tráfego que deseja enviar no circuito dinâmico. Caso o perfil do tráfego não mude, o usuário pode definir a regra uma única vez e utilizá-la, de maneira transparente, quantas vezes desejar.

Cada RACE colocado na borda da RE-CIPÓ também se conecta à rede de produção Ipê. Através das regras definidas pelo usuário, o RACE é capaz de separar o tráfego que deve ser enviado através de circuitos dinâmicos do tráfego que deve fluir através da Internet convencional. Isso significa que o usuário não precisa realizar separação de tráfego em sua rede de origem, basta conectá-la a um RACE.

Por outro lado, na configuração atual da RE-CIPÓ, uma equipe de redes da instituição do usuário, coordenada com a equipe de um PoP, precisa identificar quais equipamentos dentro de uma unidade, por exemplo, um laboratório, deverão ter acesso a circuitos dinâmicos. A equipe da instituição deve então estender o serviço oferecido pela RE-CIPÓ, geralmente por meio de VLANs, até os equipamentos do usuário. Adicionalmente, o usuário deve providenciar para que apenas tráfego específico seja enviado através dos circuitos dinâmicos. Caso o usuário tenha demanda por uso da Internet nos mesmos equipamentos que estão sendo utilizados os circuitos dinâmicos, precisará tomar providências adicionais, como acréscimo de interface de rede e/ou configuração adequada dos equipamentos.

## 6. Conclusão e Trabalhos Futuros

Neste trabalho, apresentamos o sistema ATER. Descrevemos sua arquitetura, seus principais componentes e o avaliamos em laboratório e na RNP. Mostramos que o uso de circuitos dinâmicos pode trazer ganhos de desempenho expressivos para usuários que precisam enviar grande volumes de dados. Avaliamos também aspectos operacionais do sistema ATER, como o tempo consumido com atividades de controle e a perda de pacotes ao realizar a migração do tráfego entre redes de circuitos e pacotes. Verificamos que os resultados do ATER são satisfatórios, pois são baixas a sobrecarga de controle e a perda de pacotes durante a migração. Por fim, analisamos o impacto do sistema ATER na RE-CIPÓ, uma vez que nosso sistema facilita o acesso aos circuitos dinâmicos e oferece transparência na criação desses circuitos.

Como trabalhos futuros, estão planejados a implementação de uma versão OpenFlow do RACE e testes com usuários, como parte da avaliação do piloto de um potencial serviço. A versão OpenFlow do RACE visa ampliar escalabilidade e robustez para o sistema, substituindo o computador com múltiplas interfaces por um *switch* OpenFlow.

## Agradecimentos

Este trabalho recebeu apoio financeiro e tecnológico da Rede Nacional de Ensino e Pesquisa. Este trabalho também foi parcialmente financiado por CAPES, CNPq e FAPEG. Agradecemos às contribuições do restante da equipe que participou do GT-ATER: Douglas V. Santana, Marcelo A. Inuzuka, Vinícius B. da S. Lima e Vívian L. Barreto.

## Referências

Augusto, C. H. P., da Silva, M. W. R., Cardoso, K. V., Mendes, A., Guedes, R. M., and de REZENDE, J. F. (2008). Segmentação de Conexões TCP para Transferência Fim-a-Fim em Alta Velocidade. In *Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance)*, pages 141–160.

da Silva, L. A. F. (2006). Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade. Master's thesis, Programa de Pós-Graduação de Engenharia Elétrica – COPPE/UFRJ.

de Santanna, J., Wickboldt, J., and Granville, L. (2012). A BPM-based solution for interdomain circuit management. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 385–392.

GÉANT (2013). GÉANT Bandwidth on Demand. `http://www.geant.net/Services/ConnectivityServices/Pages/Bandwidth_on_Demand.aspx`. [Último acesso: 16-Novembro-2013].

Gu, J., Katramatos, D., Liu, X., Natarajan, V., Shoshani, A., Sim, A., Yu, D., Bradley, S., and McKee, S. (2011). StorNet: Co-scheduling of end-to-end bandwidth reservation on storage and network systems for high-performance data transfers. In *Computer Communications Workshops, 2011 IEEE Conference on*, pages 121–126.

Guok, C., Robertson, D., Thompson, M., Lee, J., Tierney, B., and Johnston, W. (2006). Intra and Interdomain Circuit Provisioning Using the OSCARS Reservation System. In *Broadband Communications, Networks and Systems, 2006.*, pages 1–8.

Ha, S., Rhee, I., and Xu, L. (2008). CUBIC: a new TCP-friendly high-speed TCP variant. *SIGOPS Operating Systems Review*, 42(5):64–74.

Katramatos, D., Liu, X., Shroff, K., Yu, D., McKee, S., and Robertazzi, T. (2010). TeraPath: End-to-end network resource scheduling in high-impact network domains. *International Journal on Advances in Internet Technology*, 3(1-2):104–117.

LABORA (2013). GT-ATER: Aceleração do Transporte de Dados com o Emprego de Redes de Circuitos Dinâmicos. `https://labora.inf.ufg.br/gt-ater/`. [Último acesso: 03-Dezembro-2013].

Monga, I., Guok, C., Johnston, W., and Tierney, B. (2011). Hybrid networks: lessons learned and future challenges based on esnet4 experience. *Communications Magazine, IEEE*, 49(5):114–121.

Wu, J., Savoie, M., Zhang, H., and Campbell, S. (2006). A User-Controlled Lightpath Provisioning System for Grid Optical Networks. In *IEEE Singapore International Conference on Communication systems (ICCS)*, pages 1–5.

Yang, X., Tracy, C., Sobieski, J., and Lehman, T. (2006). GMPLS-based dynamic provisioning and traffic engineering of high-capacity Ethernet circuits in hybrid optical/packet networks. In *IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1–5.

# AuthFlow: Um Mecanismo de Autenticação e Controle de Acesso para Redes Definidas por *Software*

**Diogo Menezes Ferrazani Mattos e Otto Carlos Muniz Bandeira Duarte**

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

***Resumo.*** *As Redes Definidas por Software (Software Defined Network – SDN) desacoplam o controle do encaminhamento de dados, oferecendo alta programabilidade e uma visão global da rede. A adoção dessa abordagem é crescente em redes empresariais, centros de dados e infraestruturas críticas como as redes elétricas inteligentes. No entanto, prover segurança nessas redes de nova geração é um desafio. Este artigo apresenta as principais ameaças de segurança às redes definidas por software e propõe o AuthFlow, um mecanismo de autenticação e controle de acesso de estações finais baseado na credencial da estação. O mecanismo proposto apresenta duas contribuições principais: i) autentica a estação diretamente na camada de enlace em uma rede OpenFlow, o que introduz uma baixa sobrecarga de controle e assegura um controle de acesso refinado; ii) usa a credencial de autenticação para realizar o controle de acesso de acordo com o nível de privilégio de cada estação, através da associação da credencial ao conjunto de fluxos pertencentes à estação. Um protótipo do mecanismo proposto foi implementado sobre o POX, controlador OpenFlow. Os resultados mostram que a proposta bloqueia o acesso de estações não autorizadas, mesmo no cenário em que uma estação tem a sua permissão de acesso revogada. Por fim, comprova-se que cada estação pode ter um nível diferente de acesso aos recursos da rede em função da sua credencial.*

***Abstract.*** *Software Defined Networks are been widely adopted by enterprise networks. Providing security features in these next generation networks, however, is a challenge. In this paper, we present the main security threats in Software Defined Networks and we propose AuthFlow, an authentication and access control mechanism based on host credentials. The main contributions of the proposed mechanism are twofold: i) AuthFlow authenticates hosts directly at the data link layer in an OpenFlow network, which introduces a low overhead and ensures a fine-grained access control, ii) AuthFlow uses the authentication credential to perform access control according to the privilege level of each host, through the association of the host credential with the set of flows that belongs to the host. A prototype of the proposed mechanism was implemented over POX, an OpenFlow controller. The results show that the proposed mechanism blocks access from unauthorized hosts, even in the scenario where a host has its access authorization revoked. Finally, we show that each host can have different levels of access to network resources according to their authentication credential.*

## 1. Introdução

Prover segurança em redes é uma necessidade crescente em redes empresariais, em redes de centro de dados para computação em nuvem e em redes que constituem infraestruturas críticas como as redes elétricas inteligentes. As principais dificuldades para

Este trabalho foi realizado com recursos da CNPq, CAPES, FINEP, FUNTTEL e FAPERJ.

garantir um alto nível de segurança nas redes [Nayak et al., 2009, Kreutz et al., 2013] são a variedade de equipamentos de redes, como comutadores, roteadores, *middleboxes*, entre outros; e o fato de as estações finais que se conectam à rede nem sempre serem confiáveis e, até mesmo, poderem apresentar diversas vulnerabilidades. Assim, a implantação de políticas de segurança requer do operador da rede configurações manuais dos equipamentos de acordo com o padrão e as funcionalidades de cada um. Além disso, as estações finais também devem ser autenticadas para garantir o acesso à rede somente às estações que apresentam credenciais válidas e autorizadas.

O paradigma de Redes Definidas por *Software* (*Software Defined Networks* – SDN) desacopla o controle do encaminhamento de dados, oferecendo alta programabilidade do controle e uma visão global da rede. A adoção desta tecnologia permite desenvolver, de forma logicamente centralizada, políticas integradas de segurança [Levin et al., 2012] e, assim, facilita a solução de problemas complexos de segurança em rede. A Interface de Programação de Aplicação (API) OpenFlow [McKeown et al., 2008] é a implementação de maior sucesso de uma rede definida por *software*. O controlador OpenFlow, implementando em *software*, centraliza o plano de controle. O plano de encaminhamento é executado por comutadores de alto desempenho compatíveis com o OpenFlow. Contudo, esse novo paradigma apresenta algumas limitações quanto à segurança da rede, pois um componente com comportamento malicioso pode comprometer o funcionamento de toda a rede, por exemplo, realizando um ataque de negação de serviço no controlador. Dessa forma, o controle de acesso é necessário para garantir a segurança. Tanto a autenticação das estações que têm acesso à rede, quanto o nível de privilégio atribuído a cada estação são essenciais para garantir a segurança da rede definida por *software*.

Este artigo propõe o AuthFlow, um mecanismo de autenticação e controle de acesso para redes definidas por *software*. O mecanismo AuthFlow apresenta duas contribuições principais: (i) a autenticação das estações finais diretamente na camada de enlace; e (ii) a associação das credenciais de acesso de uma estação aos fluxos pertencentes a essa estação. A autenticação da estação final na camada de enlace é realizada através do padrão IEEE 801.X que garante que as informações de autenticação sejam trocadas de forma padronizada entre a estação e o autenticador e, portanto, não requer qualquer alteração nas estações finais. O mecanismo de autenticação encapsula as mensagens no formato *Extensible Authentication Protocol* (EAP), o que permite a adoção de diferentes métodos de autenticação. A autenticação do mecanismo AuthFlow é direto na camada de enlace e, portanto, tem a vantagem prover uma baixa sobrecarga de controle quando comparada a uma autenticação na camada rede, ou na camada de aplicação, que dependem da atribuição de um IP à estação que está se autenticando e dependem da troca de informações da aplicação para a autenticação. Outra vantagem do mecanismo proposto é o provimento de um controle de acesso refinado, já que o AuthFlow permite definir políticas de acesso por fluxo para cada estação de acordo com sua credencial. Assim, o controle de quais serviços uma estação pode acessar passa a ser realizado de acordo com as suas credenciais e não mais com seus endereços IP ou MAC. O mecanismo AuthFlow é composto por uma aplicação que executa sobre o POX, o controlador OpenFlow, e possui mais dois componentes principais: o autenticador e o servidor RADIUS. O autenticador recebe as mensagens no padrão IEEE 802.1X e valida as credenciais da estação com o servidor RADIUS.

As principais propostas para prover segurança às redes definidas por *software* buscam desenvolver módulos de segurança no controlador que facilitam o desenvol-

vimento de novas aplicações seguras [Porras et al., 2012, Shin et al., 2013]. Por outro lado, outras propostas de autenticação de estações finais em SDN consideram que a autenticação deve ser feita somente após a estação receber um endereço IP temporário e ser redirecionada a um sítio *Web*, onde deve apresentar suas credenciais [Nayak et al., 2009, Casado et al., 2007]. Contudo, essas propostas estão sujeitas a ataques de falsificação de endereço, além de introduzirem uma maior sobrecarga de controle quando comparadas ao AuthFlow. Outras propostas descrevem algumas ameaças de segurança das redes definidas por *software* e indicam possíveis direções para solucionar essas ameaças [Kreutz et al., 2013, Heller et al., 2012]. Considerando as principais ameaças às redes definidas por *software*, um protótipo do mecanismo AuthFlow foi desenvolvido e avaliado no ambiente de experimentação *Future Internet Testbed with Security* (FITS) [Guimarães et al., 2013][1]. A eficiência do mecanismo de controle de acesso proposto é evidenciada nos experimentos que mostram o bloqueio de estações ao tentar usar a rede, tanto no caso em que a estação não está autenticada, quanto no caso de a estação ter sua autenticação revogada. Os resultados da avaliação do protótipo mostram ainda que as estações finais têm visões diferentes da rede, liberando ou bloqueando acesso a determinados serviços, dependendo do nível de privilégio que cada estação tem de acordo com a sua credencial de acesso.

O restante do artigo está organizado da seguinte forma. A Seção 2 discute os principais conceitos do paradigma de Redes Definidas por *Software* e suas limitações de segurança. O mecanismo AuthFlow proposto é apresentado na Seção 3. A Seção 4 apresenta os resultados experimentais da avaliação do mecanismo proposto. Os trabalhos relacionados são discutidos na Seção 5. A Seção 6 conclui o artigo.

## 2. O Paradigma Redes Definidas por *Software*

O paradigma de Redes Definidas por Software (*Software Defined Networking* - SDN) [Casado et al., 2012] se baseia na separação das funções de controle, plano de controle, das funções de encaminhamento de quadros, no plano de encaminhamento. A ideia chave da separação é prover maior flexibilidade às funções de controle enquanto o *hardware* especializado para comutar quadros a alta velocidade permanece inalterado. Logo, a SDN oferece uma alta programabilidade das funções de controle da rede em um comutador com alto desempenho de encaminhamento de quadros. O operador pode definir de maneira simples os fluxos e as ações sobre os fluxos através de uma interface de programação de aplicação [Guedes et al., 2012].

A Figura 1 ilustra uma rede definida por *software* com controle centralizado, separado dos elementos comutadores responsáveis pelo encaminhamento de pacotes. O controle da rede é executado por um *software* de propósito geral, denominado controlador de rede, sobre o qual se desenvolvem aplicações com propósitos específicos para o controle da rede. O controlador se comunica com os comutadores e, então, possui uma visão unificada de todo o estado da rede. Assim, uma das principais vantagens da abordagem SDN é a formação de uma visão global, unificada, do controle da rede facilitando a tomada de decisões sobre sua operação. A visão global centralizada torna a programação da rede mais fácil e simplifica a representação de problemas [Guedes et al., 2012]. O OpenFlow define que os elementos de encaminhamento ofereçam uma interface de programação de aplicação (*Application Programming Interface* - API) que permita um nó controlador centralizado estender as ações de controle e de acesso sobre a tabela utilizada pelos

[1]O FITS é uma rede de testes interuniversitária desenvolvida a partir da parceria de instituições brasileiras e europeias. Maiores informações em http://www.gta.ufrj.br/fits/.

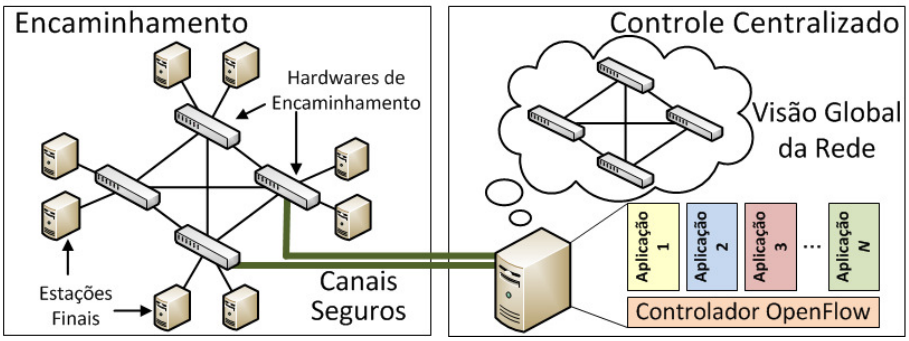



**Figura 1. Rede Definida por *Software* separa as funções de encaminhamento e controle. O *hardware* de encaminhamento é controlado por aplicações centralizadas que têm uma visão global da rede.**

componentes de encaminhamento para determinar o próximo destino de cada pacote encaminhado.

### 2.1. Ameaças de Segurança em Redes Definidas por *Software*

A visão global centralizada das redes definidas por *software* permite que a lógica de controle das aplicações de segurança seja mais completa e integrada do que as atuais [Shin et al., 2013] e, portanto, simplifica o tratamento de problemas complexos de segurança em rede. As aplicações de segurança em rede se valem do controlador centralizado para implementarem lógicas de definição de fluxos baseadas em estados e, também, segurança baseada em fluxos como, por exemplo, algoritmos de detecção de intrusão ou de anomalias. Contudo, a criação de aplicações de segurança em SDN é um desafio, pois a própria segurança de uma rede definida por *software* ainda é questionável [Kreutz et al., 2013]. Os desafios de segurança de uma rede definida por *software* se dividem em três categorias: negação de serviço, ausência de confiança entre componentes e vulnerabilidades de componentes.

**Negação de Serviço** pode ocorrer tanto no plano de dados quanto no plano de controle. No plano de dados, uma estação maliciosa que gere fluxos falsos pode exaurir tanto os recursos de banda, quanto os recursos de memória, ou tabela de fluxos, dos comutadores da rede. A negação de serviço no plano de controle pode ser alcançada em dois pontos distintos da rede: no controlador e na comunicação do controlador com os comutadores. É possível exaurir a capacidade de processamento do controlador de rede ao se enviar uma grande quantidade de pacotes com diferentes cabeçalhos. Isto acontece, pois todo pacote é analisado e um pacote com cabeçalho que não corresponde a nenhum fluxo já definido deve ser enviado ao controlador de rede. Assim, em um cenário em que um comutador envia uma quantidade atípica de novos cabeçalhos de pacotes para o controlador, este pode ter seus recursos de processamento exauridos e não ser capaz de responder a pedidos de novos fluxos em tempo hábil. Da mesma forma, a negação de serviço pode ser obtida quando o enlace de conexão entre o controlador e os comutadores na rede é intencionalmente congestionado. Caso não haja redundância ou banda suficiente no enlace que conecta os comutadores ao controlador, um comutador malicioso pode gerar tráfego suficiente para sobrecarregar esse enlace e, consequentemente, impedir a comunicação do controlador com os demais comutadores. A autenticação de estações e dos comutadores, usando *Secure Socket Layer* (SSL) e infraestrutura de chaves públicas (*Public Key Infrastructure* – PKI), é capaz de evitar esse tipo de ameaça, pois somente nós autorizados têm

acesso à rede e, no caso de identificação de um comportamento malicioso, a autenticação do nó pode ser revogada e o nó, expulso da rede.

**Ausência de Confiança entre Componentes da Rede** compromete uma Rede Definida por *Software*, pois as aplicações executadas sobre o controlador podem ter comportamentos maliciosos. Nesse caso, o controlador deve ser capaz de identificar quais são as aplicações confiáveis e quais são maliciosas. Assim, uma possível medida para aumentar a segurança nas aplicações executadas em uma SDN é o uso de mecanismos de atestação de aplicações e o estabelecimento de cadeias de confiança por atestação. Algumas propostas para prover segurança em SDN consideram o uso de um núcleo de segurança no próprio controlador para garantir a execução segura de aplicações, sem que uma aplicação interfira em outras ou execute ações proibidas na rede [Shin et al., 2013, Porras et al., 2012]. Por outro lado, a ausência de confiança também afeta o registro de ações que ocorreram na rede (*log*), já que o registro de ações, quando é feito, não apresenta nenhuma garantia de que a ação ocorreu e de que a aplicação que o registrou não tinha comportamento malicioso. Uma possível solução é a adoção de registro de ações por aplicação, assinados por cada aplicação, que por sua vez sejam atestadas e assinadas por desenvolvedores.

**Vulnerabilidade de Componentes** não é um desafio de segurança exclusivo desse novo paradigma de rede, mas torna-se mais crítico, pois uma vulnerabilidade em um nó controlador torna toda a rede vulnerável. Assim, são três as possíveis fontes de vulnerabilidades: comutadores, controlador e estações de gerenciamento. Uma vulnerabilidade em um comutador pode permitir que um atacante, que ganhe acesso a um comutador, exerça um ataque contra o plano de controle, a exemplo da falsificação de mensagens de outros comutadores para exaurir os recursos do controlador. Uma vulnerabilidade no controlador permite que um atacante altere o plano de controle ou, até mesmo, execute uma nova aplicação de controle da rede. Uma vulnerabilidade em uma estação de gerenciamento permite que o atacante exerça configurações no plano de controle diferentes das corretas. Medidas de prevenção a este tipo de ataque são a atestação das aplicações de controle, o uso de protocolos de certificação dupla entre estações de controle e aplicações e, por fim, a replicação das aplicações de controle para a tolerância a falhas e a intrusão.

Entre alguns dos principais desafios em segurança para as redes definidas por *software*, é possível destacar três características necessárias a essas redes: escalável, responsiva e disponível [Nayak et al., 2009]. Para prover tais características há o desafio da localização de controladores [Heller et al., 2012]. Nesse sentido, a localização e a quantidade de controladores replicados necessários a uma rede devem respeitar os requisitos de segurança, escala, disponibilidade e tempo de resposta da rede.

A autenticação, autorização e controle de acesso são primitivas essenciais em uma Rede Definida por *Software*. Essas primitivas, somadas à atestação e replicação de controladores, são a base de uma rede segura em que componentes maliciosos, sejam por vulnerabilidades em *software*, sejam pelo comportamento nocivo à rede, podem ser identificados e isolados do funcionamento normal da rede [Nagahama et al., 2012, Nayak et al., 2009, Shin et al., 2013].

## 3. O Mecanismo AuthFlow

A ideia principal do mecanismo AuthFlow é realizar a autenticação usando protocolos da camada de enlace, fazendo o mapeamento da identidade usada na autenticação em fluxos criados por uma dada estação autenticada. Para tanto, o mecanismo proposto usa o padrão IEEE 802.1X e o *Extensible Authentication Protocol* (EAP). O EAP encap-

sula as trocas de mensagens de autenticação entre a estação suplicante[2] e um servidor de autenticação RADIUS. O autenticador empregado no mecanismo AuthFlow é um processo que se comunica como uma aplicação OpenFlow, que executa sobre o controlador POX. A aplicação aceita ou bloqueia o tráfego de rede da estação suplicante, dependendo do resultado da autenticação entre o suplicante e o autenticador.

O mecanismo AuthFlow adota o padrão IEEE 802.1X, pois esse padrão especifica a autenticação diretamente na camada de enlace e, por ser um padrão bastante adotado, não requer modificações nas estações finais para a autenticação na rede. Assim, quando uma estação compatível com o padrão IEEE 802.1X inicia, ela também inicia a fase de autenticação através do envio da mensagem de `início` do IEEE 802.1X para o endereço reservado MAC *multicast* (`01:80:C2:00:00:03`), com o tipo Ethernet definido em `0x888E`. Dessa forma, o procedimento de autenticação de uma estação não depende de nenhum conhecimento prévio acerca da rede, nem mesmo da tradução de um endereço IP em um endereço MAC. Esse procedimento evita que a estação receba um IP temporário para só recebrer um IP definitivo dependendo do resultado da autenticação. O uso do padrão IEEE 802.1X facilita muito o processo de autenticação.

A seguir, discute-se a arquitetura e o funcionamento do mecanismo AuthFlow. O estudo de caso considerado é usar o mecanismo proposto para a autenticação de roteadores virtuais em uma infraestrutura de rede virtual híbrida Xen e OpenFlow. Contudo, a proposta não se limita a esse estudo de caso e pode ser usada, sem qualquer alteração, na autenticação de estações finais em uma rede OpenFlow. No estudo de caso considerado, as estações componentes da rede OpenFlow são máquinas virtuais que se comportam tanto como estações finais quanto como roteadores.

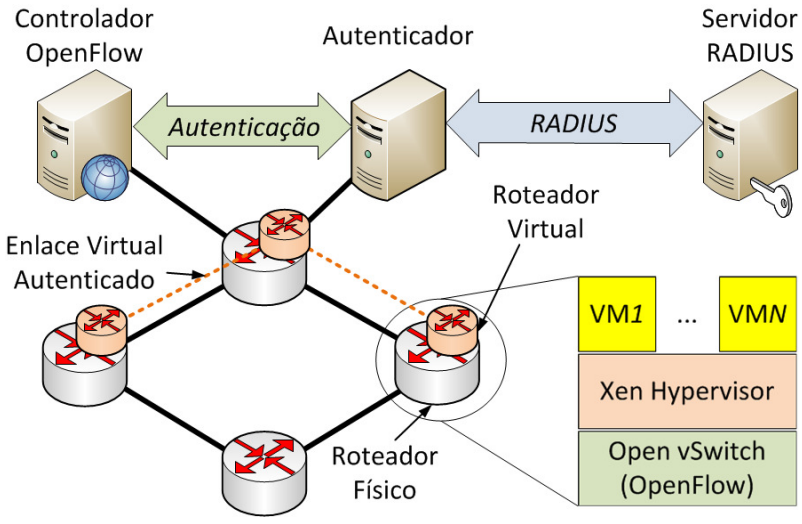



**Figura 2. A arquitetura do mecanismo AuthFlow é composta por três nós essenciais: o controlador OpenFlow, o Autenticador e o Servidor RADIUS. Uma máquina virtual ao se autenticar na rede, envia um pacote IEEE 802.1X com a autenticação EAP. O Autenticador usa o conteúdo EAP do pacote IEEE 802.1X para autenticar o roteador virtual com o servidor RADIUS e comunica o resultado da operação ao controlador OpenFlow.**

A arquitetura do mecanismo AuthFlow é composta de máquinas físicas hospedando máquinas virtuais, comutadores OpenFlow por *software*, um controlador POX, um

[2]Os nomes suplicante, autenticador e servidor de autenticação são definidos pelo padrão IEEE 802.1X.

autenticador e um servidor de autenticação RADIUS, conforme mostrado na Figura 2. As máquinas físicas e virtuais agem como roteadores e, então, são denominadas respectivamente roteadores físicos e roteadores virtuais. Os roteadores físicos são os nós com o sistema de virtualização Xen que hospedam os roteadores virtuais. O encaminhamento dos pacotes entre os roteadores físicos e virtuais é realizado por um comutador por *software* compatível com a API OpenFlow. No caso do mecanismo proposto, o comutador OpenFlow é um Open vSwitch[3] instanciado em cada roteador físico. O modelo de virtualização adotado no mecanismo proposto é o modelo híbrido Xen e OpenFlow usado no sistema XenFlow [Mattos et al., 2011, Mattos et al., 2013]. O controlador POX executa uma aplicação para manipular o encaminhamento de todos os pacotes, em especial, os pacotes[4] do padrão IEEE 802.1X. Os pacotes IEEE 802.1X são encaminhados diretamente para o autenticador. Autenticador é um cliente RADIUS que implementa o padrão IEEE 802.1X e repassa o conteúdo EAP para o RADIUS. O autenticador foi desenvolvido como uma versão adaptada do `hostapd`[5], que é um autenticador usado em redes sem fio. O `hostapd` foi modificado para informar à aplicação POX sobre a autenticação das redes virtuais. Assim, ao realizar a autenticação de uma rede, o `hostapd` envia uma confirmação de autenticação para o POX através de um canal seguro, criptografado e autenticado usando o esquema de distribuição de chaves públicas (*Public Key Infrastructure* – PKI) e o padrão SSL 3.0 (*Secure Socket Layer*). O servidor de autenticação é um servidor RADIUS que extrai as informações de autenticação do encapsuladas pelo EAP e valida as credenciais apresentadas pelos roteadores virtuais. Como o EAP permite o uso de diversos métodos de autenticação diferentes, o método adotado foi o MS-CHAP v2 [Zorn, 2000] que autentica o roteador virtual em uma base de dados usando como credenciais nome do usuário e senha. Para tanto, foi usada uma base de dados *Lightweight Directory Access Protocol* (LDAP). Contudo, o mecanismo de autenticação e a base de dados a serem usados não são essenciais para a descrição do mecanismo proposto, pois não interagem diretamente com a rede OpenFlow.

O mecanismo de autenticação AuthFlow funciona da seguinte maneira. Um roteador virtual envia um pedido de autenticação, no padrão IEEE 802.1X, e o controlador POX redireciona para o Autenticador. Em seguida, o Autenticador responde e a estação suplicante envia suas credenciais. O Autenticador verifica as credenciais de estação suplicante com o servidor RADIUS, executando o método de autenticação definido no EAP. Se as credenciais estão corretas, o Autenticador envia uma mensagem de `sucesso` para a estação suplicante e envia uma mensagem de autorização e confirmação de autenticação para o POX, através do canal seguro SSL, identificando a estação suplicante. Após o estágio de autenticação, o controlador POX permite que a estação suplicante acesse aos recursos da rede. Em caso de revogação das credenciais da estação suplicante, o Autenticador comunica ao POX, que imediatamente suspende o acesso da estação à rede.

O controle de acesso do mecanismo proposto funciona através da liberação e bloqueio de enlaces. Assim, ao iniciar uma rede OpenFlow que empregue o AuthFlow, todos os enlaces da rede estão bloqueados para a comunicação, inclusive os enlaces que interconectam os nós comutadores OpenFlow, ou seja, os enlaces pertencente ao núcleo da rede. Nesse caso, esses enlaces não necessitam realizar o processo de autenticação para ter o seu tráfego liberado. Para tanto, o mecanismo AuthFlow realiza a descoberta da topologia

[3]http://www.openvswitch.org/.

[4]A nomenclatura de pacote foi usada, pois é mais genérica e a API OpenFlow tem acesso é às camada de enlace Ethernet até a camada de transporte.

[5] http://hostap.epitest.fi/hostapd/.

do núcleo da rede através de pacotes *Link Layer Discovery Protocol* (LLDP) encaminhados enlace a enlace. Como o controlador gera e verifica cada pacote LLDP transmitido no núcleo da rede, o controlador é capaz de identificar quais enlaces estão entre comutadores OpenFlow e quais são enlaces finais ligados a estações finais. Os pacotes LLDP gerados pelo controlador são marcados unicamente para evitar ataques de repetição (*replay*) ou de falsificação (*spoofing*).

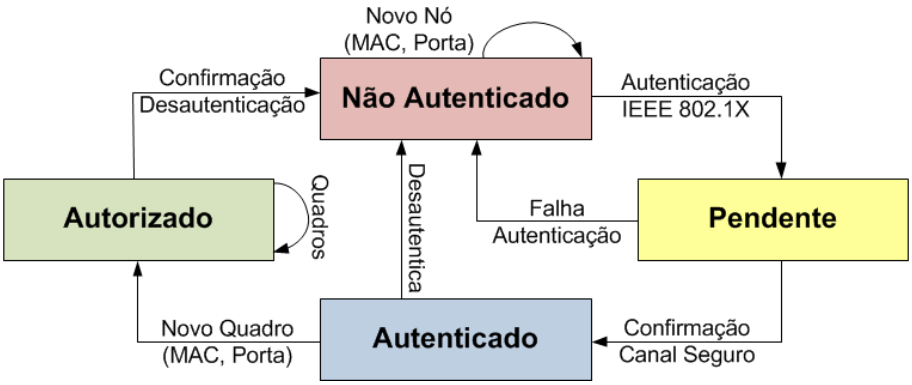



**Figura 3. O controle de acesso do mecanismo AuthFlow ocorre em quatro estados. i) Não autenticado, quando a estação não iniciou o processo de autenticação; ii) Pendente, enquanto o processo de autenticação ocorre; iii) Autenticado, quando a estação já finalizou a autenticação com sucesso; iv) Autorizado, quando a estação está autorizada a usar a rede.**

O controle de acesso no mecanismo AuthFlow é executado de acordo com o diagrama de estados apresentado na Figura 3. Na figura, um nó da rede é sempre representado por uma tupla `(MAC, porta)`. Essa tupla identifica que uma estação com um dado endereço `MAC` está conectada na `porta` do comutador. A figura sintetiza o processo de autenticação, mostrando que a autenticação ocorre no sentido de controlar o acesso de um MAC através de uma porta do comutador. Dessa forma, um nó, ao ingressar na rede, está inicialmente no estado `não autenticado` e, assim, todo tráfego gerado ou destinado a esse nó é bloqueado, exceto pelo tráfego com tipo Ethernet `0x888E` (IEEE 802.1X). O controle de acesso é pela definição de dois fluxos padrão: i) encaminhar todo o tráfego de autenticação para o Autenticador, através de fluxo *multicast*; ii) descartar pacotes não autenticados. Esses fluxos são criados ao chegar um novo pacote de um endereço MAC não autenticado. Essa estratégia evita o DDoS. No sentido contrário, é encaminhado em fluxos *unicast*, já que o Autenticador aprende o endereço MAC da estação suplicante após o recebimento do primeiro pacote IEEE 802.1X. Assim que a estação inicia o procedimento de autenticação, enviando a mensagem de `início`, a estação é movida para o estado de `pendente`, estado em que todo o tráfego da estação continua bloqueado, mas a estação está no aguardo de uma confirmação do Autenticador para o POX de que sua autenticação foi bem sucedida e quais as credenciais foram usadas na autenticação. Assim que há a confirmação de que a autenticação foi bem sucedida, o POX move a estação para o estado `autenticado`. Nesse estado, o POX confere as permissões de acesso a recursos da rede que a estação possui, de acordo com suas credenciais, e libera o acesso da estação à rede. No entanto, quando há tráfego para a estação, o POX confere se o trafego para a estação está de acordo com as políticas referentes às credenciais usadas pela estação para acessar a rede. Caso as políticas estejam de acordo com o uso da rede, a estação é movida para estado `autorizado` e acessa os recursos da rede de acordo com seus privilégios.

Tendo em vista o controle de acesso empregado, a liberação ou bloqueio do tráfego de uma estação final é realizado de acordo com as credenciais apresentadas pela estação. A autenticação da tupla `(MAC, porta)` está relacionada com a identidade da estação. Dessa forma, é possível fazer o mapeamento dos fluxos de uma dada estação para a sua identidade. O mapeamento ocorre no seguinte sentido. Se um fluxo OpenFlow apresenta entre suas características o endereço MAC de origem (`dl_src`) e a porta de entrada no comutador (`in_port`) iguais aos que estão na tupla de autenticação, a credencial de autenticação da estação é atribuída ao fluxo. Assim, a decisão de encaminhamento desse fluxo pode tomar como parâmetro, também, a credencial da estação e, portanto, o controle de acesso à rede pode ser mais refinado. De forma análoga, se um fluxo OpenFlow apresenta entre suas características o endereço MAC de destino (`dl_dst`) e a porta de saída do comutador (`output`) iguais aos que estão na tupla de autenticação, a credencial de autenticação da estação é também atribuída ao fluxo. Assim, o mecanismo AuthFlow também controla os fluxos destinados a uma estação de acordo com a sua identidade. Portanto, no AuthFlow, as políticas de controle de acesso podem definir regras tanto de saída quanto de entrada de pacotes para as estações finais de acordo com sua identidade.

## 4. Os Resultados Experimentais

O protótipo do mecanismo AuthFlow foi implementado em uma ilha do *Future Internet Testbed with Security* (FITS) [Guimarães et al., 2013]. O protótipo utiliza o hipervisor Xen 4.1.4 para prover os domínios virtuais que agem como estações finais acessando uma rede OpenFlow que, por sua vez, é implementada através do comutador programável Open vSwitch 1.2.2. O Open vSwitch [Pfaff et al., 2009] é configurado para ser controlado pelo POX[6], o controlador OpenFlow utilizado. A aplicação que realiza o controle de acesso das estações finais à rede e o controle do encaminhamento de pacotes na rede OpenFlow foi desenvolvida em Python. O autenticador usado no protótipo é uma versão modificada do `hostapd`, para criar o canal seguro e informar ao controlador POX quando há uma autenticação ou perda da autenticação de uma estação final. O servidor RADIUS empregado no protótipo é o FreeRADIUS v2.1.12[7]. Como prova de conceito, o método de autenticação testado no protótipo foi o EAP-MSCHAP v2 [Zorn, 2000].

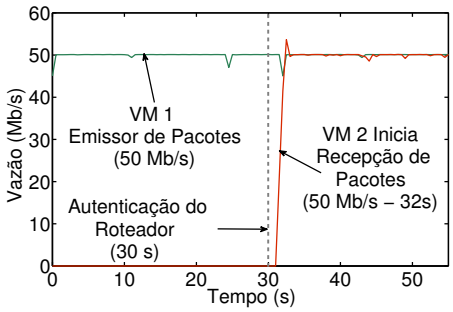
As ferramentas `Iperf`, `Nmap` e `Tcpdump`[8] foram usadas para realizar as medidas de avaliação de desempenho do protótipo. Quatro computadores pessoais compõem o cenário dos experimentos. Todos executam o protótipo do mecanismo AuthFlow. Nos computadores pessoais foram instanciadas quatro máquinas virtuais que agem como roteadores, enviam e recebem pacotes, dependendo de cada experimento. Todos os computadores possuem processadores Intel Core 2 Quad 2.4 GHz, 3 GB de memória RAM e executam o Debian Linux 3.2.0-4-amd64. Cada computador possui, no mínimo, 2 interfaces de rede sendo que todas são configuradas para funcionarem a 100 Mb/s, para garantir homogeneidade, uma vez que havia também interfaces de 1 Gb/s. As máquinas virtuais são configuradas com uma CPU virtual, 128 MB de memória RAM e executa o Debian Linux 3.2.0-4-amd64. As máquinas virtuais executam os protocolos de roteamento através da plataforma XORP [Handley et al., 2003].

O primeiro experimento avalia a eficácia do mecanismo AuthFlow em bloquear tráfegos não autorizados. O encaminhamento de pacotes de um fluxo só é liberado após a

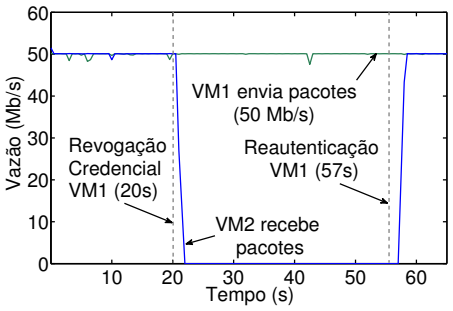
[6]O controlador POX utilizado nos experimentos é uma adaptação do controlador usado na rede de testes FITS para dar suporte ao mecanismo AuthFlow.

[7]http://freeradius.org/.

[8]http://iperf.sourceforge.com/, http://www.nmap.org/ e http://www.tcpdump.org/.




(a) Liberação de tráfego pelo roteador situado entre as máquinas virtuais 1 e 2 (VM1 e VM2) após a sua autenticação que ocorre em 30 s.



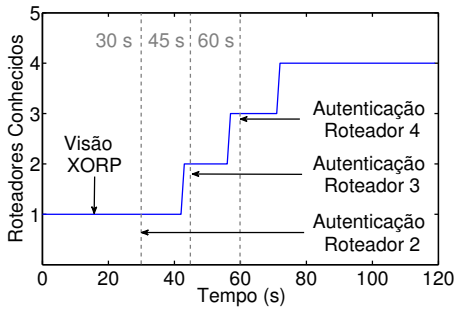

(b) Bloqueio do tráfego quando em 20 s a autenticação da máquina virtual 2 é revogada e liberação do tráfego quando a autenticação é restabelecida ao fim do teste.

**Figura 4. Bloqueio da função encaminhamento de tráfego por falta de autenticação e por revogação de credencial. Máquina virtual 1 (VM1) envia pacotes para a máquina virtual 2 (VM2). a) Autenticação do roteador entre VM1 e VM2. b)Revogação da autenticação da VM1.**

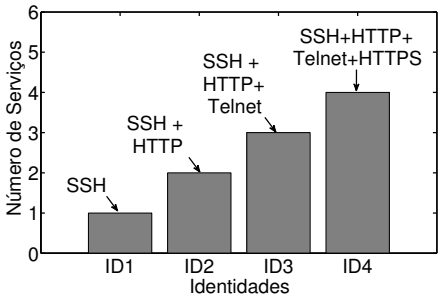
autenticação, caso contrário, os pacotes são descartados. O cenário é simples, a máquina virtual 1 (VM1) envia pacotes destinados à máquina virtual 2 (VM2) através de um roteador virtual. Assume-se que as máquinas virtuais 1 e 2 foram previamente autenticadas na rede e o roteador virtual que as interconecta não está autenticado. A VM1 gera um fluxo de pacotes UDP de tamanho 1472 B de conteúdo a uma taxa constante de 50 Mb/s. Como o roteador não está autenticado, o fluxo não chega à VM2. Após 30 s, o roteador se autentica na rede, como mostrado na Figura 4(a), e o fluxo UDP é recebido pela VM2. A Figura 4(a) evidencia que há um atraso, da ordem de 2 s a 2,5 s entre o início do processo de autenticação do roteador e a efetiva liberação do acesso à rede. Esse atraso é devido ao processo de autenticação do padrão IEEE 802.1X somado ao tempo de instanciação do fluxo OpenFlow. Esse atraso ocorre somente no momento em que o roteador, entre VM1 e VM2, ingressa na rede.

O segundo experimento evidencia a eficácia do mecanismo de revogação de credencial do AuthFlow, mostrado na Figura 4(b). O cenário consiste de uma máquina virtual, VM1, que se comunica diretamente com outra máquina virtual, VM2, não há roteadores entre elas. Mais uma vez assume-se que inicialmente ambas as máquinas virtuais estão autenticadas. Após 20 s, a autenticação da VM2 é revogada e, então, o acesso à rede da VM 2 é bloqueado, tanto para o envio quanto para a recepção de pacotes. Observa-se que o atraso para o bloqueio da atividade da VM2 na rede é menor que 1 s. Após 57 s, a autenticação da VM2 é restabelecida e a VM2 volta a receber os pacotes. O procedimento de restabelecimento da autenticação ocorre com um atraso de aproximadamente 2 s, assim como a autenticação de uma nova estação. Deve ser ressaltada a efetividade do mecanismo proposto AuthFlow correspondente à ação de liberação e de bloqueio de tráfego através do encaminhamento e do descarte de pacotes, respectivamente, associada ao procedimento de autenticação e revogação de credenciais. Assim, o AuthFlow se constitui em um forte aliado na defesa contra ataques de negação de serviço devido a sua efetividade na ação de liberar e bloquear fluxos em redes definidas por *software*, condicionadas ao processo de autenticação.

Os experimentos seguintes demonstram a visão da rede para as estações autenticadas, ou seja, quais estações da rede uma estação autenticada alcança e quais os serviços da rede essa estação acessa. A Figura 5(a) mostra a visão da rede segundo um roteador




(a) Quantidade de vizinhos um roteador virtual percebe na rede de acordo com o número de outros roteadores autenticados.




(b) Número de serviços providos pela rede, de acordo com a credencial usada pela estação final ao se autenticar na rede.

**Figura 5. Visão da rede a partir de um roteador virtual (a) e de uma estação final (b). Cada identidade usada só tem acesso a um determinado conjunto de serviços na rede.**

virtual executando o protocolo de roteamento de estado de enlace OSPF (*Open Shortest Path First*). A topologia considerada é um anel, conectando quatro roteadores virtuais. O experimento consiste em verificar a base de dados do OSPF ao longo do tempo e, então, identificar quantos roteadores vizinhos o roteador observado conhece. No início do experimento, somente o roteador observado está autenticado na rede. Após 30 s, autentica-se o segundo roteador. Em 45 s, autentica-se o terceiro e, finalmente em 60 s, autentica-se o quarto roteador. Vale ressaltar que o atraso entre a autenticação e a descoberta de cada novo roteador, indicado na Figura 5(a), é devido ao tratamento de pacotes de *brodcast*/*multicast* adotado na rede OpenFlow. Para evitar a sobrecarga na rede, a cada inundação de pacotes, é inserida uma regra nas tabelas de fluxos para realizar o descarte de pacotes com a mesma característica do pacote inundado por 5 s.

O quarto experimento procura demonstrar uma das principais vantagens oferecidas pelo mecanismo AuthFlow, que consiste no uso da credencial de autenticação como forma de realizar o encaminhamento de fluxos. A ideia chave é a autenticação prover uma "identificação" dos fluxos correspondentes aos serviços que são autorizado para a estação. Assim, a autenticação pelo mecanismo AuthFlow possibilita a liberação dos fluxos correspondentes aos serviços que foram liberados, bloqueando todos os demais fluxos. O experimento consiste em uma estação solicitante, autenticada com uma das quatro identidades possíveis (ID1, ID2, ID3 ou ID4), acessar outra estação fornecedora de serviços na rede. Cada identidade permite o acesso a um determinado número de serviços na rede (um, dois, três ou quatro serviços, respectivamente). Para tanto, a estação solicitante executa uma varredura de portas (`nmap`) na estação fornecedora de serviços. No cenário de testes, a estação solicitante é a mesma, para as quatro identidades, mantendo o mesmo endereço IP e MAC durante todo o experimento. A única modificação no cenário de testes é a autenticação da estação solicitante com outra identidade a cada teste. A Figura 5(b) mostra os serviços que a estação solicitante consegue acessar na estação fornecedora de serviços na rede. Assim, é possível observar que, ao estar autenticada com uma identidade, a estação só consegue acessar os serviços liberados para aquela identidade. Vale ressaltar que a varredura de portas retorna que as portas que não têm serviços liberados são filtradas, o que mostra que o bloqueio dos demais serviços é realizado pelo descarte dos pacotes `SYN`, o que, de fato, ocorre pelas regras instaladas pelo controlador POX ao verificar que uma estação não tem o devido nível de privilégio para acessar um serviço.

## 5. Os Trabahos Relacionados

A segurança de redes definidas por *software*, em especial a segurança de redes OpenFlow, é um tema bastante discutido atualmente. Há propostas para o desenvolvimento de aplicações de segurança sobre a infraestrutura de rede OpenFlow, como também há outras que visam garantir a segurança da infraestrutura. Contudo, garantir a autenticação, o controle de acesso, a escalabilidade, o baixo tempo de resposta, a confidencialidade e a disponibilidade em SDN continua a ser um desafio [Kreutz et al., 2013].

Kreutz *et al.* apresentam uma classificação dos principais vetores de ataque a uma rede definida por *software* e possíveis contramedidas para se proteger desses ataques [Kreutz et al., 2013]. Kreutz *et al.* restringem-se a ataques contra a resiliência e confiabilidade da rede. Visando garantir a confidencialidade e a disponibilidade de redes definidas por *software*, o sistema QFlow [Mattos e Duarte, 2012, Mattos et al., 2013] se baseia em um sistema híbrido Xen e OpenFlow para prover o isolamento de recursos e de comunicação entre redes virtuais sobre uma infraestrutura SDN. O sistema adota o encaminhamento de pacotes por filas para garantir a reserva de banda para cada rede virtual e marca os pacotes de cada rede com um marcador de VLAN, para multiplexar a rede virtual que um pacote pertence. No QFlow, no entanto, não há mecanismos de autenticação ou controle de acesso entre máquinas virtuais e a infraestrutura de rede. Assim, o AuthFlow é complementar ao QFlow, pois garante ao controle de acesso.

A rede UPV/EHU [Matias et al., 2011], uma rede europeia de testes OpenFlow, também adota uma proposta de autenticação baseada no padrão IEEE 802.1X. Contudo, tal proposta não considera o uso das credenciais de autenticação do nó na rede para a definição de novos fluxos. O diferencial da proposta AuthFlow é realizar o mapeamento das credencias de autenticação para os fluxos, assim, a qualquer tempo, é possível identificar os nós que estão gerando ou recebendo um tráfego em um dado comutador e, se necessário, revogar sua autenticação. A proposta AuthFlow fornece ainda a primitiva de se definir regras de encaminhamento no controlador da rede de acordo com a identidade das estações, o que é um diferencial em relação a outras propostas.

Guenane *et al.* propõem um mecanismo de autenticação de redes virtuais usando EAP-TLS implementado em cartões inteligentes (*smart cards*) [Guenane et al., 2012]. A proposta consiste em garantir o acesso de máquinas virtuais e de clientes das redes virtuais a cartões inteligentes, que implementam o protocolo TLS e encapsulam as mensagens em EAP. As mensagens encapsuladas EAP são enviadas para um servidor RADIUS que autentica os componentes da rede virtual, assim como os clientes da rede virtual, através da autenticação mútua provida pelos certificados, assinados por uma Autoridade Certificadora, apresentados durante a negociação TLS. No entanto, essa proposta não define como seria o mecanismo de controle de acesso dos nós à rede e como a autenticação é usada para autorizar o acesso do cliente aos recursos da rede. O mecanismo AuthFlow proposto nesse artigo pode ser usado conjuntamente com esta proposta de autenticação com cartões inteligentes, uma vez que os dados de autorização são encapsulados em EAP. Assim, o AuthFlow controlaria o acesso dos roteadores virtuais aos recursos da rede.

Resonance [Nayak et al., 2009] e Ethane [Casado et al., 2007] são outras propostas que visam a autenticação de nós em uma rede definida por *software*. Ambas defendem que a autenticação do nó na rede deve ser feita por através de portal *Web* em que o usuário deve apresentar as suas credenciais. Essa abordagem apresenta uma restrição básica que é a necessidade de o nó ter um navegador *Web* instalado. Esse requisito é bem limitante, quando se consideram ambientes formados por redes virtuais compostas por máquinas virtuais extremamente leves que não possuem nem interface gráfica. Outra

desvantagem desse método de autenticação é a limitação ao modelo de autenticação por usuário e senha, enquanto o modelo de autenticação adotado pelo AuthFlow baseia-se no encapsulamento EAP, assim, qualquer que seja o método de autenticação escolhido, se for compatível com EAP, é trivialmente suportado pelo AuthFlow. Como citado anteriormente, até mesmo métodos de autenticação robustos baseados em microcontroladores seguros são possíveis com o mecanismo AuthFlow. Outra vantagem do AuthFlow em relação a essas propostas é a autenticação diretamente na Camada 2, assim não há a necessidade de um nó adquirir um endereço IP para depois se autenticar, como ocorre no Resonance ou no Ethane. No AuthFlow, assim que uma estação entra na rede, ela inicia sua autenticação de acordo com o padrão IEEE 802.1X diretamente na camada de enlace, autenticando o seu endereço MAC na porta do comutador em que está conectado. Esse procedimento impede que um nó use um endereço MAC falsificado, ao contrário de outras propostas que não visam impedir a falsificação do endereço MAC.

As propostas FRESCO [Shin et al., 2013] e FortNOX [Porras et al., 2012] definem um conjunto de primitivas de segurança para redes OpenFlow. A proposta FortNOX defende a criação de um núcleo seguro de execução de aplicações sobre um controlador da rede OpenFlow. Esse núcleo seguro impede que uma aplicação execute ações que interfiram nas políticas de controle de outra aplicação. A proposta FortNOX defende o fatiamento da rede entre aplicações sobre um mesmo controlador, o que gera um controle mais fino dos privilégios e do domínio de controle de cada aplicação do que o previsto pelo FlowVisor [Sherwood et al., 2009]. A proposta do FlowVisor, por sua vez, fatia a rede entre diversos controladores, contudo, não prevê uma política de segurança entre controladores para que as ações de um controlador não afete aos demais. Seguindo a ideia do núcleo seguro de execução de aplicações, a proposta FRESCO define um conjunto de primitivas e uma linguagem modular para o desenvolvimento de aplicações de segurança para a rede OpenFlow.

## 6. Conclusão

A segurança de redes empresariais depende de mecanismos de controle de acesso e de autenticação eficientes. Com a crescente adoção de redes definidas por *software* (SDN) por redes empresarias, o desafio de prover segurança às SDN tornou-se ainda mais fundamental. Esse artigo propõe o AuthFlow, um mecanismo de autenticação e controle o acesso à infraestrutura de um rede definida por *software* OpenFlow, baseado no padrão IEEE 802.1X e no servidor de autenticação RADIUS. O mecanismo AuthFlow proposto implementa a autenticação através de uma base dados LDAP com RADIUS. A proposta, no entanto, é extensível a outros métodos de autenticação, como o EAP-TLS, que autentica os nós com base em certificados. Os resultados mostram que o mecanismo de autenticação proposto impede que estações não autorizadas acessem recursos da rede, mesmo quando já autenticadas e, após, perdem seus privilégios. Os resultados mostram ainda que o mecanismo proposto é mais eficiente que as demais propostas, já que introduz menor sobrecarga de controle, e permite a definição de políticas de controle de acesso por fluxo de acordo com as credenciais de acesso de cada estação.

Como trabalhos futuros, pretende-se implantar o mecanismo AuthFlow no *Future Internet Testbed* (FITS), como seu mecanismo de autenticação e controle de acesso padrão, e estender o AuthFlow para novos métodos de autenticação, tal como o EAP-TLS que permite o uso de certificados assinados como credencial de acesso.

## 7. Referências


[Casado et al., 2007] Casado, M., Freedman, M., Pettit, J., Luo, J., McKeown, N. e Shenker, S. (2007). Ethane: Taking control of the enterprise. *ACM SIGCOMM Computer Communication Review*, 37(4):1–12.

[Casado et al., 2012] Casado, M., Koponen, T., Shenker, S. e Tootoonchian, A. (2012). Fabric: a retrospective on evolving SDN. Em *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, p. 85–90, New York, NY, USA. ACM.

[Guedes et al., 2012] Guedes, D., Vieira, L., Vieira, M., Rodrigues, H. e 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*, p. 160–210.

[Guenane et al., 2012] Guenane, F., Samet, N., Pujolle, G. e Urien, P. (2012). A strong authentication for virtual networks using eap-tls smart cards. Em *Global Information Infrastructure and Networking Symposium (GIIS), 2012*, p. 1–6.

[Guimarães et al., 2013] Guimarães, P. H., Ferraz, L., Torres, J. V., Mattos, D., Murillo, A., Lopez, M. A., Alvarenga, I., Rodrigues, C. e Duarte, O. C. M. B. (2013). Experimenting Content-Centric networks in the future internet testbed environment. Em *IEEE International Conference on Communications 2013: IEEE ICC'13 - Workshop on Cloud Convergence: challenges for future infrastructures and services (WCC 2013) (ICC'13 - IEEE ICC'13 - Workshop WCC)*, p. 1398–1402, Budapest, Hungary.

[Handley et al., 2003] Handley, M., Hodson, O. e Kohler, E. (2003). XORP: An open platform for network research. *ACM SIGCOMM Computer Communication Review*, 33(1):53–57.

[Heller et al., 2012] Heller, B., Sherwood, R. e McKeown, N. (2012). The controller placement problem. Em *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, p. 7–12, New York, NY, USA. ACM.

[Kreutz et al., 2013] Kreutz, D., Ramos, F. M. e Verissimo, P. (2013). Towards secure and dependable software-defined networks. Em *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, p. 55–60, New York, NY, USA. ACM.

[Levin et al., 2012] Levin, D., Wundsam, A., Heller, B., Handigol, N. e Feldmann, A. (2012). Logically centralized?: state distribution trade-offs in software defined networks. Em *Proceedings of the First workshop on Hot topics in software defined networks*, HotSDN '12, Helsinki, Finland. ACM.

[Matias et al., 2011] Matias, J., Jacob, E., Toledo, N. e Astorga, J. (2011). Towards neutrality in access networks: A nando deployment with openflow. Em *ACCESS 2011, The Second International Conference on Access Networks*, p. 7–12, Luxembourg City, Luxembourg.

[Mattos et al., 2011] Mattos, D., Fernandes, N. C. e Duarte, O. C. M. B. (2011). XenFlow: Um sistema de processamento de fluxos robusto e eficiente para migração em redes virtuais. Em *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2011*.

[Mattos et al., 2013] Mattos, D., Ferraz, L. e Duarte, O. C. M. B. (2013). Um mecanismo para isolamento seguro de redes virtuais usando a abordagem híbrida xen e openflow. Em *SBSeg 2013 - XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, Manaus - Brazil.

[Mattos e Duarte, 2012] Mattos, D. M. F. e Duarte, O. C. M. B. (2012). QFlow: Um sistema com garantia de isolamento e oferta de qualidade de serviço para redes virtualizadas. Em *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2012*.

[McKeown et al., 2008] 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., 2008*.

[Nagahama et al., 2012] Nagahama, F. Y., Granville, L., Farias, F., Cerqueira, E., Aguiar, E., Gaspary, L. e Abelém, A. (2012). IPSFlow – uma proposta de sistema de prevenção de intrusão baseado no framework openflow.

[Nayak et al., 2009] Nayak, A. K., Reimers, A., Feamster, N. e Clark, R. (2009). Resonance: Dynamic access control for enterprise networks. Em *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, p. 11–18, New York, NY, USA. ACM.

[Pfaff et al., 2009] Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M. e Shenker, S. (2009). Extending networking into the virtualization layer. *Proc. HotNets*.

[Porras et al., 2012] Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M. e Gu, G. (2012). A security enforcement kernel for openflow networks. Em *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, p. 121–126, New York, NY, USA. ACM.

[Sherwood et al., 2009] Sherwood, R., Gibb, G., Yap, K., Appenzeller, G., Casado, M., McKeown, N. e Parulkar, G. (2009). Flowvisor: A network virtualization layer. Relatório técnico, Tech. Rep. OPENFLOW-TR-2009-01, OpenFlow Consortium.

[Shin et al., 2013] Shin, S., Porras, P., Yegneswaran, V., Fong, M., Gu, G. e Tyson, M. (2013). Fresco: Modular composable security services for software-defined networks. Em *Proceedings of Network and Distributed Security Symposium*.

[Zorn, 2000] Zorn, G. (2000). Microsoft PPP CHAP Extensions, Version 2. RFC 2759 (Informational).


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 16*

## *Redes e Ambientes Virtualizados*

# Sistema Automatizado de Gerência de Recursos para Ambientes Virtualizados

**Govinda Mohini Gonzalez Bezerra, Diogo Menezes Ferrazani Mattos, Lyno Henrique Gonçalves Ferraz e Otto Carlos Muniz Bandeira Duarte**

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

***Resumo.*** *A gerência de recursos em ambientes virtualizados é uma tarefa complexa e desafiadora, pois deve se adaptar dinamicamente às oscilações de carga de trabalho e garantir o nível de serviço das diferentes aplicações em execução. Este artigo propõe o sistema AMAS (Automatic Migration and Allocation System for Virtual Resource Management) que monitora máquinas físicas e virtuais e constrói perfis de uso de diferentes recursos computacionais, como processador, memória e rede. O sistema proposto é capaz de detectar, com base nos perfis elaborados, a escassez dos recursos monitorados e automaticamente redistribuir as cargas de trabalho, evitando a sobrecarga dos nós físicos e violações de acordos de níveis de serviços. O sistema proposto automatiza migrações ao vivo de máquinas virtuais, que permitem transferir uma máquina virtual de um nó físico origem para um nó físico destino sem a interrupção dos serviços em execução. Um protótipo do sistema AMAS foi desenvolvido e o seu desempenho foi avaliado no Future Internet Testbed with Security (FITS). Os resultados obtidos mostram a eficácia do sistema em distribuir as cargas de trabalho e eliminar sobrecargas de processamento e memória dos nós físicos.*

***Abstract.*** *One of the major challenges in virtualized environments is dynamically managing available resources, adapting them to changes in workload and ensuring the level of service of the running applications. In this paper, we propose AMAS, Automatic Migration and Allocation System for Virtual Resource Management, a system for monitoring physical and virtual machines, and it profiles the use of different computing resources, such as processor, memory and network usage. Based on profiles, the system is able to detect the lack of a monitored resource and automatically redistribute workloads, avoiding the overhead of physical nodes and violations of Service Level Agreements (SLA). The proposed system performs load distribution through live migration primitive of virtual machines that can transfer a virtual machine from a physical source node to another without interruption of running services. The proposed system was developed and we evaluated its performance with the Future Internet Security Testbed (FITS). The results show the effectiveness of the system to distribute workloads and to eliminate overloads of physical nodes.*

## 1. Introdução

A tecnologia de virtualização provê o compartilhamento de recursos computacionais entre diferentes ambientes virtuais de execução, chamados de máquinas virtuais. As

Este trabalho foi realizado com recursos da CNPq, CAPES, FINEP, FUNTTEL e FAPERJ.

máquinas virtuais acessam os recursos da máquina física de forma isolada com a ilusão de acessarem os recursos computacionais como se lhes fossem dedicados, para a execução de aplicações e sistema operacional próprios. A virtualização está sendo adotada nas empresas para consolidação de servidores, que consiste em reunir diversos servidores virtuais, lógicos, sobre uma mesma máquina física. Além das vantagens econômicas, a virtualização proporciona uma infraestrutura ágil e dinâmica, pois permite realocar facilmente os recursos computacionais, de forma a responder eficazmente às variações de demanda por recursos e melhor atender as necessidades dos diferentes serviços.

O gerenciamento e a alocação dinâmica de recursos em ambientes virtualizados são tarefas bastante complexas. A funcionalidade de migração de máquinas virtuais ao vivo é uma abordagem eficiente para realizar a gerência de recursos. Através da migração é possível reorganizar os domínios nos nós físicos de acordo com a disponibilidade dos recursos. Assim, se houver um aumento significativo na carga de uma máquina virtual, de forma a sobrecarregar um nó físico, esse domínio virtual poderá ser migrado para outro nó que possua recursos disponíveis. Essa abordagem permite realizar a distribuição de cargas de uma forma eficiente, porém também traz alguns desafios. O primeiro consiste em prever o comportamento futuro das máquinas virtuais para estimar a demanda futura de recursos. O segundo consiste em encontrar a melhor combinação de máquinas físicas e virtuais que é um problema de otimização do tipo *bin-packing*, cuja complexidade é *NP-difícil*. Logo, dada a complexidade do problema, os trabalhos desenvolvidos nesta área são baseados em heurísticas e modelos estatísticos.

Este artigo propõe o *Automatic Migration and Allocation System for Virtual Resource Management* (AMAS), um sistema automático de gerência de recursos que é capaz de realocar dinamicamente as máquinas virtuais de acordo com o nível de utilização das máquinas físicas monitoradas. O sistema proposto monitora os recursos computacionais em ambientes virtualizados, tais como capacidade de processamento, memória e banda passante, detecta situações de sobrecarga e toma decisões que eliminam a sobrecarga. O sistema distribui automaticamente as cargas de trabalho entre as máquinas físicas disponíveis, realocando máquinas virtuais críticas em máquinas físicas menos sobrecarregadas. Assim, as principais contribuições deste artigo são: i) o monitoramento e a criação de perfis do uso de recursos das máquinas físicas e virtuais; ii) a proposta de um algoritmo simples capaz de tomar decisões de realocação de máquinas virtuais , baseado nos perfis de uso, que elimine situações de sobrecarga; iii) a implementação de um painel de controle que permita a visualização, em tempo real, da alocação e utilização dos recursos computacionais monitorados. O sistema AMAS é compatível com diversas tecnologias de virtualização, mas é focado no Xen, pois esta é a tecnologia utilizada no testbed *Future Internet Testbed with Security* (FITS)[1].

As principais propostas para prover o balanceamento e a distribuição de cargas em ambientes virtualizados se diferenciam em analisar o estado de cada nó individualmente [Wood et al., 2009a, Carvalho e Duarte, 2012, Khanna et al., 2006] ou o estado de global do conjunto de todos os nós físicos [Arzuaga e Kaeli, 2010]. Algumas propostas focam na análise de afinidade entre máquinas físicas e virtuais [Sonnek et al., 2010, Wood et al., 2009b], ou ainda, consideram a topologia da rede e a banda disponível para realizar a migração de máquinas virutais [Singh et al., 2008]. A proposta deste artigo, no

[1]O FITS é uma rede de testes interuniversitária desenvolvida a partir da parceria de instituições brasileiras e europeias. Maiores informações em http://www.gta.ufrj.br/fits/.

entanto, foca em estabelecer um algoritmo simples e rápido para tomar decisões de migrar máquinas virtuais em tempo real para garantir o nível de serviço prestado por essas máquinas. Um protótipo do sistema foi implementado e avaliado. Os resultados mostram que o sistema AMAS é capaz de identificar gargalos no uso de recursos computacionais e reagir em tempo hábil para evitar a degradação do nível de serviço prestado pelas máquinas virtuais.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 discute a arquitetura do sistema AMAS. O protótipo do sistema e os resultados da avaliação de desempenho do protótipo são discutidos na Seção 4. A Seção 5 conclui o artigo.

## 2. Trabalhos Relacionados

A alocação de máquinas virtuais em servidores físicos é um problema complexo. Há propostas de heurísticas [Fajjari et al., 2011] e modelagens baseadas em otimização de sistemas [Alkmim et al., 2011, Rodriguez et al., 2013] que buscam o melhor aproveitamento dos recursos físicos. Um fator importante a ser considerado no uso de algoritmos de otimização é o tempo de convergência do algoritmo, que vai influir diretamente na dinamicidade do sistema. Propostas de otimização de uso de recursos físicos são complementares ao sistema proposto, pois podem ser usadas no o algoritmo de migração. Este trabalho não foca na otimização e opta por uma proposta simples para uma rápida tomada de decisão.

Sandpiper [Wood et al., 2009a] é uma ferramenta de gerenciamento de recursos para centros de dados (*data centers*), formado basicamente por três componentes: um mecanismo de criação de perfis, um detector de sobrecarga e um gerenciador de migração. A criação de perfis é feita através do monitoramento e da coleta de estatísticas do uso de recursos das máquinas físicas e virtuais. O detector de sobrecarga monitora estes perfis em busca de nós sobrecarregados que consistem em máquinas físicas cuja utilização de pelo menos um dos recursos ultrapassou um limiar ou se ocorreu a violação do acordo de nível de serviço (*Service Level Agreement* - SLA). O detector de sobrecarga determina quando é necessário realizar uma realocação de recursos, ou seja, quando o gerenciador deverá atuar para eliminar esta sobrecarga. Cabe ao gerenciador decidir qual máquina virtual migrar e qual servidor irá recebê-la. Diferentemente da proposta deste artigo, o Sandpiper utiliza o estado local de cada máquina física e não leva em conta a afinidade entre as máquinas e topologia da rede para realizar as migrações.

Voltaic [Carvalho e Duarte, 2012] é um sistema de gerência voltado para computação em nuvem que visa garantir o cumprimento de acordos de níveis de serviços (SLAs) e otimizar o uso dos recursos computacionais. Assim como o Sandpiper, o Voltaic também é uma solução baseada em perfis de uso. O sistema pode ser dividido em três módulos principais: o primeiro, chamado de coletor de estatísticas, é responsável por coletar informações de uso dos recursos computacionais; o segundo consiste no analisador de perfil que utiliza os dados provenientes do coletor de estatística para gerar perfis de uso; e o terceiro é o orquestrador, módulo central do Voltaic, responsável por gerenciar as máquinas, detectar escassez de recursos e orquestrar a migração. O orquestrador monitora a carga de todas as máquinas físicas e caso a média das últimas amostras de carga de alguma das máquinas ultrapassarem um limiar de segurança, o algoritmo começa o procedimento de realocação de recursos. As máquinas físicas são ordenadas de acordo com a carga do sistema e pelo número de máquinas virtuais críticas que ela possui. Dada a so-

brecarga de algum dos recursos monitorados, a máquina virtual escolhida para a migração será a máquina mais crítica do servidor físico mais sobrecarregado. O servidor de destino será aquele que possuir recursos disponíveis compatíveis com o perfil da máquina virtual. O comportamento do Voltaic foi simulado e o sistema não foi implementado. Quando comparado ao Voltaic, o sistema proposto neste artigo usa uma heurística mais simples que permite reagir às mudanças de cargas de trabalho em tempo real.

Yanagisawa *et al.* propõem um algoritmo de alocação de máquinas virtuais baseado em programação inteira, que considera *a priori* padrões de flutuações da demanda de recursos para alocação ótima e confiável de máquinas virtuais em uma infraestrutura de nuvem [Yanagisawa et al., 2013]. A proposta usa a migração de máquinas virtuais entre servidores físicos somente quando há a necessidade de manutenção ou falhas no servidor. A ideia principal é minimizar as possibilidades de falha e de degradação de desempenho de máquinas virtuais devido às migrações. Na proposta, o requisito mais importante de confiabilidade a ser considerado no planejamento de capacidade é que os recursos computacionais sejam suficientes e estejam disponíveis a todo o momento, mesmo quando houver falhas no servidor. Contudo, essa proposta considera que os usuários sejam altamente conservadores. Por sua vez, o sistema AMAS considera que os usuários podem ter comportamentos bem variáveis e o sistema deve ser capaz de realocar as máquinas virtuais e atender às mudanças de demanda por recursos em tempo real. O sistema AMAS usa a mesma abordagem de Yanagisawa *et al.* ao limitar o número de migrações possíveis, para evitar a degradação de desempenho das máquinas virtuais.

Guo *et al.*, por sua vez, propõem um modelo de alocação de banda entre máquinas virtuais em centros de dados para nuvens [Guo et al., 2013]. A ideia central da proposta é aplicar a teoria de jogos e modelar a disputa por banda entre os diferentes inquilinos da nuvem como um problema de negociação de Nash (*Nash bargaining problem*). Assim, a proposta desenvolve um jogo cooperativo para alocação de banda entre as máquinas virtuais. A proposta alcança uma banda mínima garantida entre pares de máquinas virtuais e garante a justiça na divisão da banda entre todos os pares de máquinas virtuais. No entanto, essa proposta modela somente a alocação de banda entre máquinas virtuais e não considera a alocação de outros recursos, tais como processamento e memória. O modelo de alocação de banda pode ser usado em paralelo ao sistema AMAS para garantir a divisão justa de banda entre máquinas virtuais sobre uma mesma máquina física.

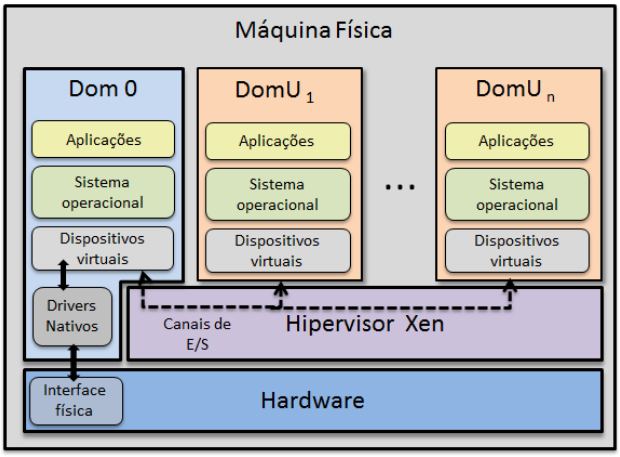



**Figura 1. A arquitetura Xen e a centralização das atividade de E/S no Domínio 0.**

Este trabalho propõe um sistema de gerenciamento de recursos baseado em perfis e na funcionalidade de migração ao vivo de máquinas virtuais. O sistema proposto usa a biblioteca libvirt para aquisição de dados e um protótipo do sistema foi implementado na plataforma de virtualização Xen. O Xen [Barham et al., 2003, Xen Project, 2009] é uma plataforma de virtualização de código aberto composta por um hipervisor, um Domínio 0 e diversos Domínios Us. O hipervisor Xen é a camada de abstração básica de software localizada logo acima da camada de hardware. É responsável pelo escalonamento de CPU e pelo fatiamento de memória. O Domínio 0 é uma máquina virtual que possui privilégios especiais para acessar os recursos físicos de E/S e para interagir com os outros domínios em execução no sistema. Os Domínios Us são domínios não privilegiados que não têm acesso direto ao hardware da máquina física.

A libvirt [Red Hat, 2013] é uma API para gerenciar, de forma segura, máquinas em ambientes virtualizados. A libvirt permite o gerenciamento completo das máquinas virtuais, tornando possível criar, modificar, controlar, migrar e destruir as máquinas através de sua interface. A ferramenta permite o acesso a hipervisores utilizando conexões autenticadas e criptografadas, através do protocolo TLS (*Transport Layer Security*). A criptografia e a autenticação são feitas usando a abordagem de chave pública (PKI), logo, é necessário que todos os computadores monitorados possuam certificados válidos, além das chaves pública e privada.

## 3. O Sistema de Gerência e Distribuição de Cargas Proposto

O sistema AMAS (*Automatic Migration and Allocation System for Virtual Resource Management*) monitora as máquinas, físicas e virtuais, com objetivo de gerenciar a alocação e melhorar a utilização dos recursos físicos, além de garantir o cumprimento dos acordos de níveis de serviço (SLAs) dos servidores virtuais. A ideia central da proposta é criar perfis de uso de diferentes recursos para todas as máquinas físicas e virtuais de um aglomerado (*cluster*) e, com base nestes perfis, detectar nós físicos sobrecarregados e distribuir a carga de trabalho através da migração ao vivo.

A Figura 2 mostra a arquitetura do AMAS, que possui três módulos principais: o Gerador de Perfil, responsável por monitorar o uso dos recursos físicos; o Detector de sobrecarga, capaz de detectar escassez de recursos monitorados; e o Orquestrador de migração, responsável por tomar decisões visando distribuir as cargas de trabalho em caso de sobrecarga. Toda a comunicação entre o gerenciador e as máquinas físicas é realizada através da libvirt, o que garante uma maior portabilidade do sistema para outras plataformas.

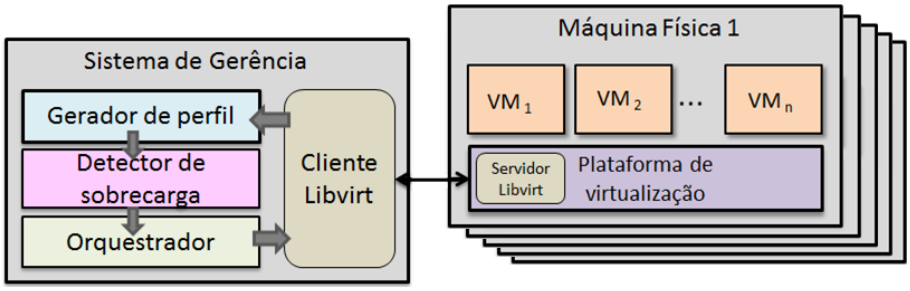



**Figura 2. Arquitetura do sistema de gerência AMAS com o gerente e módulos servidores libvirt em cada máquina física.**

O mecanismo de monitoramento foi desenvolvido utilizando uma abordagem do tipo caixa preta. Na solução adotada, todas as informações relativas ao uso de recursos das máquinas virtuais são obtidas através da libvirt, sem a instalação de nenhum programa específico nas máquinas virtuais. Essa abordagem foi escolhida por garantir uma menor interferência no funcionamento dos domínios, assegurando maior autonomia e privacidade aos proprietários das máquinas virtuais. Essa propriedade é importante em ambientes virtualizados, pois em muitas soluções comerciais baseadas em virtualização, as máquinas virtuais não pertencem aos proprietários do hardware, não sendo, portanto, viável a instalação de softwares de monitoramento nos domínios de clientes.

Como pode ser observado na Figura 2, o Gerador de Perfil interage com as máquinas físicas através da libvirt. Essa conexão é feita utilizando o protocolo *Transport Layer Security* (TLS), que consiste em uma conexão TCP/IP autenticada e criptografada, garantindo a segurança das informações. A coleta de dados é realizada através da amostragem periódica do uso de CPU, da quantidade de memória alocada e da quantidade de dados de rede enviados e recebidos pela máquina. Com as amostras obtidas, são criados três perfis de uso, representando cada uma das três métricas, que consistem em séries temporais representadas por uma janela deslizante contendo $N$ amostras.

A coleta das amostras de uso dos recursos é feita de modo paralelo através da utilização de processos leves, chamados *threads*. Dessa forma, é possível construir simultaneamente o perfil das diferentes máquinas monitoradas, aproveitando melhor os recursos computacionais da máquina gerenciadora e permitindo maior coerência temporal das amostras obtidas. Cada módulo do sistema possui um *thread* responsável pela coleta de dados e construção dos seus perfis de uso. Os perfis criados são apresentados em um painel de controle através de gráficos que indicam a participação de cada máquina virtual, inclusive do Domínio-0, na utilização dos recursos das máquinas físicas monitoradas.

O monitoramento de CPU das máquinas virtuais é realizado através da amostragem da medida $tempo\ de\ CPU$ obtida diretamente da libvirt, que corresponde ao tempo de processamento utilizado pela máquina virtual desde a sua criação. O perfil de uso do processador dos nós físicos é calculado indiretamente, através da soma dos perfis de todos os domínios que nele executam, pois não é possível obter a medida $tempo\ de\ CPU$ das máquinas físicas através da libvirt. Esta medida possui uma imprecisão associada, pois as amostras das máquinas virtuais podem não corresponder a exatamente o mesmo instante de tempo. Os perfis de memória das máquinas virtuais são construídos a partir da amostragem da quantidade de memória alocada ao domínio. Para os perfis das máquinas físicas, é utilizada a quantidade de memória alocada. Ambas as medidas são obtidas diretamente da libvirt. No Xen, o Domínio-0 realiza as operações de rede pelos outros domínios não privilegiados. Logo, uma forma simples de obter informações que representem a atividade de rede das máquinas virtuais é medindo a quantidade de dados enviados e recebidos através de suas interfaces virtuais, o que também é facilmente obtido através da libvirt. Essa métrica é importante para identificar a contribuição de cada máquina virtual no tráfego total gerenciado pelo Domínio-0. Então, é possível identificar se a causa desse consumo provém da utilização intensiva de rede e qual a participação de cada máquina virtual no tráfego total gerenciado pelo Domínio-0.

No sistema AMAS são considerados dois tipos de sobrecarga, a de processamento (CPU) e a de memória. A sobrecarga de CPU ocorre quando a utilização do processador ultrapassa um determinado limiar durante as $K$ últimas amostras. O conceito de sobre-

carga de memória é similar e ocorre quando a quantidade de memória alocada de um nó físico ultrapassa o limiar definido para memória durante as últimas $M$ amostras. Apesar de o sistema monitorar a intensidade do tráfego de rede, não há uma sobrecarga associada a esse perfil, uma vez que a métrica de rede implementada na versão atual não leva em conta os parâmetros nem a topologia da rede de computadores. A medida é utilizada, então, para verificar o impacto que o tráfego de rede dos domínios não privilegiados causa no uso de processamento do Domínio-0.

Um nó físico é considerado sobrecarregado se pelo menos um dos recursos monitorados estiver escasso naquele nó. A cada intervalo de tempo $T$, o algoritmo percorre as máquinas físicas verificando se há sobrecarga no sistema. O algoritmo constrói duas listas, uma contendo as máquinas físicas sobrecarregadas e outra contendo as máquinas com recursos disponíveis e ordena-as de acordo com o nível de utilização de cada um dos recursos monitorados. Detectada a escassez de algum dos recursos, o Orquestrador de migração é acionado e deve tomar decisões para eliminar a sobrecarga e distribuir as cargas de trabalho. O algoritmo realiza uma migração a cada iteração, então é preciso selecionar: i) uma máquina física sobrecarregada que terá sua carga diminuída, ii) um domínio virtual deste nó físico que será migrado e iii) uma máquina física com recursos disponíveis para receber esse domínio.

A escolha dos elementos envolvidos na migração é feita da seguinte forma:

**Seleção da máquina física sobrecarregada**: As máquinas físicas são classificadas de acordo com o nível de utilização de CPU e alocação de memória. Assim, se existirem diversos nós sobrecarregados com os dois tipos de sobrecarga, a máquina física com maior nível de utilização de processamento é selecionada. Caso só haja sobrecarga de memória, a máquina com menos memória disponível é selecionada.

**Seleção da máquina física de destino**: A máquina física que receberá a máquina virtual migrada é aquela que possuir a maior quantidade de recursos disponíveis, suficientes para alocar a máquina virtual sem ficar sobrecarregada.

**Seleção da máquina virtual**: Caso haja uma sobrecarga de CPU, é escolhida, dentre os domínios da máquina física sobrecarregada, a máquina virtual mais crítica de acordo com a equação

$$criticidade = \frac{cpu + net \cdot cpuDomain0}{memória}, \tag{1}$$

onde $cpu$ representa o uso de CPU, $net$ indica o tráfego de rede, $cpuDomain0$ representa o uso do processador pelo Domínio-0 e memória a quantidade de memória alocada ao domínio. O termo $net \cdot cpuDomain0$ visa incluir na equação de criticidade a sobrecarga causada no Domínio-0 pelo uso da rede [Cherkasova e Gardner, 2005].

Para realizar a migração ao vivo de uma máquina virtual, é preciso transmitir todas as suas páginas de memória à máquina física de destino, logo o tempo de migração é diretamente influenciado pela quantidade de memória utilizada e pela taxa de atualização dos dados na memória. O tempo de migração também varia de acordo com o nível de utilização dos recursos nas máquinas físicas de origem e destino, pois o processo de migração consome recursos em ambas as máquinas físicas [Wu e Zhao, 2011]. Assim, optou-se pela relação $\frac{cpu}{memória}$ na equação de criticidade.

Caso a sobrecarga seja de memória, é migrada a máquina virtual que possuir a menor quantidade de memória alocada do nó físico sobrecarregado. Com esta solução pode

ser necessário à migração de mais de uma máquina virtual para eliminar a sobrecarga, entretanto ela evita migrações longas e, com isso, tende a diminuir o processamento total a ser realizado pelo Domínio-0 das máquinas físicas envolvidas na migração.

O algoritmo do Orquestrador percorre a lista de máquinas virtuais do nó físico mais sobrecarregado buscando a máquina virtual a ser migrada. É escolhida a máquina virtual mais crítica que consiga ser migrada para a máquina física com mais recursos disponíveis. É possível notar que a sobrecarga de CPU é tratada com prioridade maior do que a sobrecarga de memória. Esta escolha foi feita porque o uso de CPU é mais dinâmico, já que o monitoramento de memória é realizado através da quantidade de memória alocada e não pela quantidade de memória usada. Nota-se também que se todas as máquinas físicas estiverem saturadas, ou se a migração puder causar sobrecarga na máquina de destino, a migração não é realizada.

O sistema AMAS é baseado na funcionalidade de migração ao vivo do Xen, logo, é necessário que as imagens dos discos de todos os domínios virtuais estejam disponíveis para todas as máquinas do sistema. Além disso, é necessário que todas as máquinas físicas possuam a libvirt e os certificados, necessários para autenticar as conexões, instalados. É desejável que as máquinas físicas monitoradas possuam configurações homogêneas de hardware e software, pois, em caso de migração, configurações distintas podem implicar em perda de desempenho das aplicações em execução. Na versão atual do sistema, por simplicidade, a modelagem não inclui verificação e adequação de determinadas variáveis como, por exemplo, modelos e frequências de processadores diferentes. Assim, em um ambiente onde os servidores possuem alto nível de utilização, se uma máquina virtual intensiva em rede for migrada para uma máquina física com configurações de escalonamento diferente da máquina física de origem, as aplicações sensíveis a latência podem ter o seu funcionamento fortemente alterado [Govindan et al., 2007, Lee et al., 2010].

## 4. Avaliação de Desempenho do Protótipo Desenvolvido

O sistema proposto foi implementado no ambiente de testes FITS do Grupo de Teleinformática e Automação da UFRJ/COPPE/PEE. Foram utilizados computadores padrão de mercado com placas de rede gigabit Ethernet. A implementação de cada um

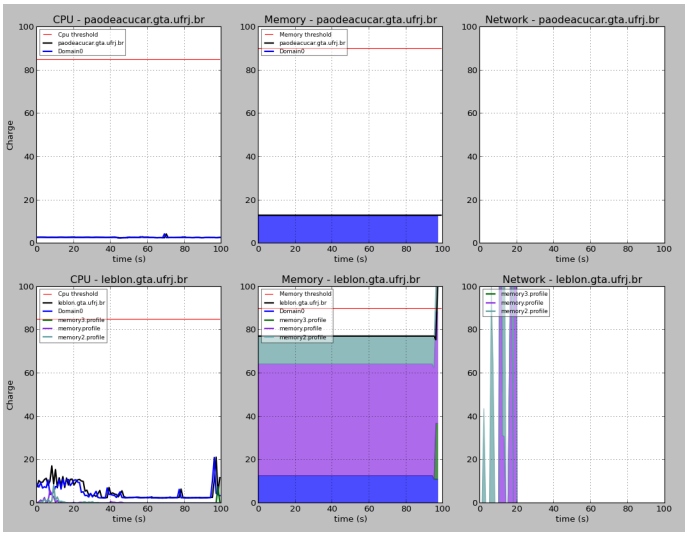
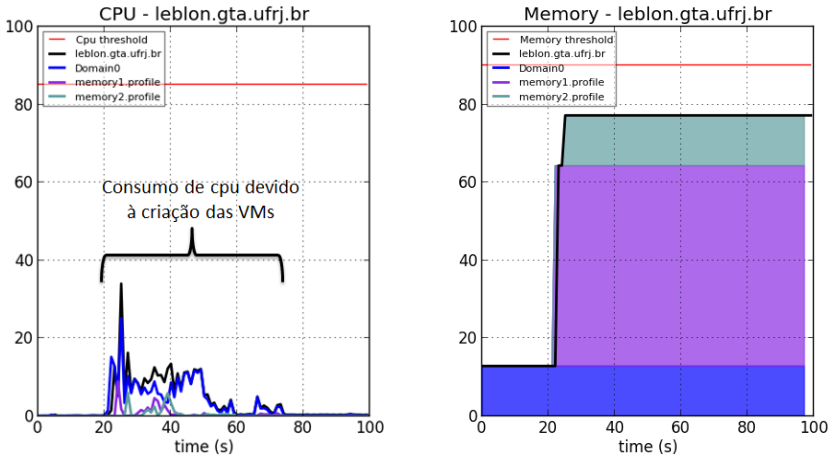



**Figura 3. Painel de controle do sistema de gerenciamento AMAS.**



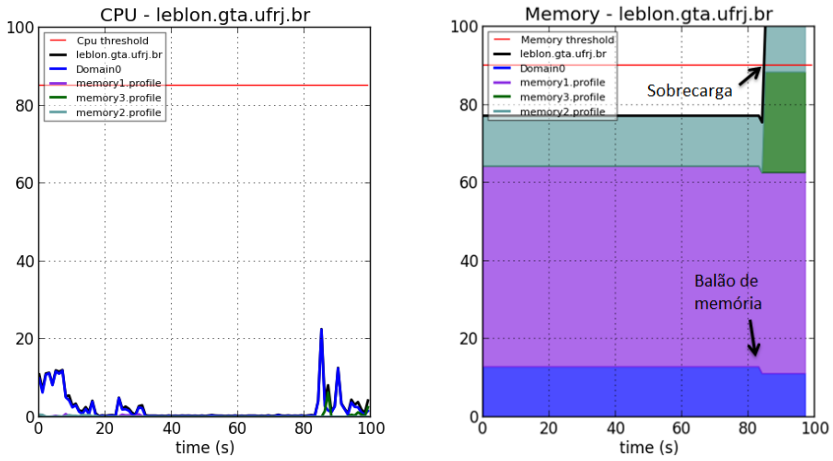

(a) Uso de processador. (b) Memória alocada.

**Figura 4. Passo 1 do cenário de testes. Configuração da máquina leblon.gta.ufrj.br após a criação das máquinas virtuais memory1 e memory2. a) A criação das máquinas virtuais consome processamento do Domínio 0. b)A alocação de memória após a criação das máquinas virtuais revela o uso de quase 80% da memória física disponível.**

dos módulos desenvolvidos segue as diretrizes da orientação a objetos, na linguagem de programação Python. O escalonador utilizado no Xen foi o Credit Scheduler e o *timeslice* de 30 ms, ambos são configurações padrão do Xen. A memória de inicialização do Domínio-0 foi definida como 2048 MB, a opção de balão foi habilitada e a memória mínima para este domínio foi configurada como 1024 MB. A funcionalidade de balão de memória foi ativada, pois apesar de o sistema operacional `Debian Wheezy` necessitar de mais de 1GB de memória para iniciar o sistema, o total de memória gasto pelo Domínio-0 e pelo hipervisor para gerenciar as máquinas virtuais está na ordem de 1GB. Então, visto que uma parte da memória alocada exclusivamente para o Domínio-0 ficaria ociosa durante todo o funcionamento do nó físico, optou-se por habilitar o balão de memória visando melhor aproveitar a memória disponível. Alguns parâmetros do sistema são configurados de acordo com a necessidade e as características das aplicações e do conjunto de máquinas monitoradas. São eles: o tamanho da janela de observação, ou seja, o número de amostras contidas no perfil das máquinas; os limiares que definem a sobrecarga de CPU e de memória; o período de amostragem; a quantidade de amostras acima do limiar que devem ser consideradas antes de realizar a migração. Foi desenvolvido um painel de controle para permitir a visualização do uso de recursos das máquinas físicas e virtuais em tempo real. O painel é composto de diferentes gráficos, representando cada um dos recursos monitorados de cada uma das máquinas físicas. A Figura 3 mostra o painel de controle em funcionamento com algumas máquinas virtuais de teste, utilizando duas máquinas do testbed FITS.

Para validar o funcionamento do sistema, foi desenvolvido um cenário de teste que permite testar o algoritmo nos diferentes casos de sobrecarga. Através deste cenário de teste também foi possível observar o comportamento do Domínio 0 durante a fase de migração e durante as atividades de rede dos domínios não privilegiados. A carga de CPU foi simulada utilizando o programa Stress [Waterland, 2013], de licença livre, que permite gerar cargas de trabalho de forma controlada.

Dentre os computadores disponíveis na rede de testes, foram selecionadas duas




(a) Uso de processador. (b) Memória alocada.

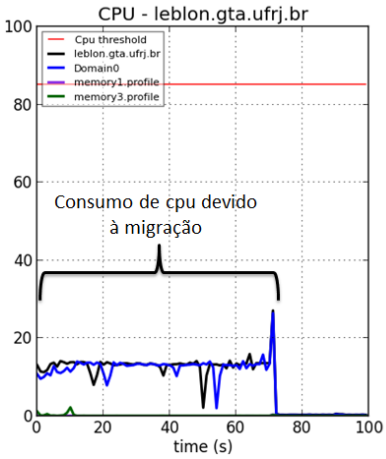
**Figura 5. Passo 2 do cenário de testes. Configuração da máquina leblon.gta.ufrj.br após a criação da máquina virtual memory3, gerando uma situação de sobrecarga de memória. a) A criação da máquina virtual consome processamento do Domínio 0. b) A alocação de memória após a criação das máquinas virtuais atinge 100%, ultrapassando o limiar de segurança e levando o domínio0 a ceder memória para o domínio memory3 recém criado.**

máquinas equipadas com processador Intel I7 de 3.2 GHz e 16 GB de memória RAM. O sistema operacional instalado nas máquinas é o `Debian Wheezy` e o hipervisor Xen utilizado é a versão 4.1.3. As imagens dos discos das máquinas virtuais e os arquivos de configuração encontram-se em um nó central da arquitetura do FITS e são acessíveis por qualquer máquina da rede de teste. Para criar diferentes cenários de sobrecarga, foram utilizadas máquinas virtuais com diferentes configurações, de modo a consumir cada um dos recursos monitorados. Os domínios virtuais denominados cpu1, cpu2 e cpu3 foram configurados cada um com 8 processadores virtuais e 512 MB de memória RAM. Os domínios denominados memory1, memory2 e memory3 foram configurados cada um com 1 processador virtual e 1024 MB, 2048 MB e 4096 MB de memória RAM, respectivamente.

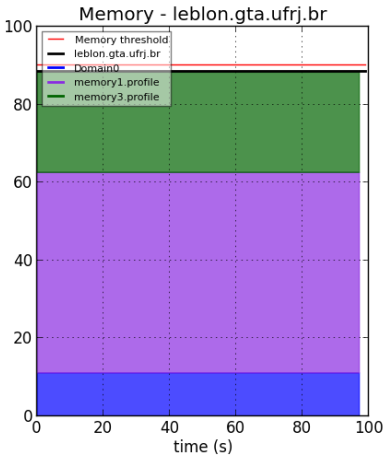
O cenário de teste é dividido nas seguintes passos:

**Passo 1** - criação das máquinas memory1, memory2 na máquina física *leblon.gta.ufrj.br*
**Passo 2** - criação da máquina memory3 na máquina *leblon.gta.ufrj.br*
**Passo 3** - criação das máquinas cpu1 e cpu2 na máquina física *paodeacucar.gta.ufrj.br* e da máquina cpu3 na *leblon.gta.ufrj.br*
**Passo 4** - geração de carga em 5 vCPUs da máquina cpu2 e em 2 vCPUs da cpu3

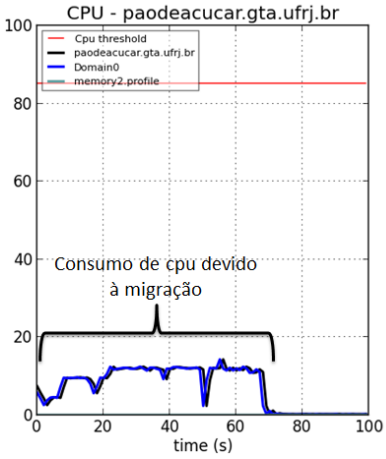
O uso de memória e processador pelas duas máquinas físicas, sem nenhuma máquina virtual em execução além do Domínio 0 é muito pequeno e a sua memória alocada está na ordem de 12% da quantidade de memória total. O primeiro item do cenário de testes visa criar um conjunto inicial de máquinas virtuais na máquina física. A Figura 4 ilustra esta etapa, onde é possível notar o consumo de CPU pelo Domínio-0 para realizar a criação das duas máquinas virtuais. O passo 2 cria a máquina memory3 na máquina *leblon.ufrj.br*, ilustrado na Figura 5, onde é possível notar que a quantidade de memória alocada atinge 100%, levando o Domínio 0 a realizar o balão de memória e ceder uma



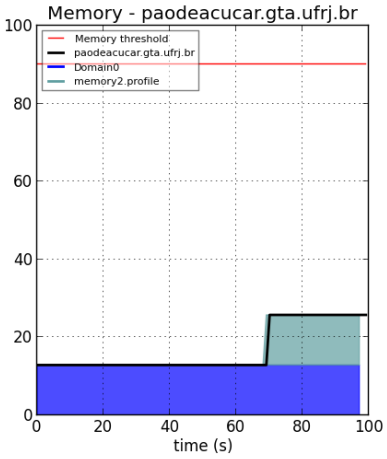

(a) Uso do processador da máquina leblon.gta.ufrj.br.




(b) Memória alocada da máquina leblon.gta.ufrj.br.




(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.

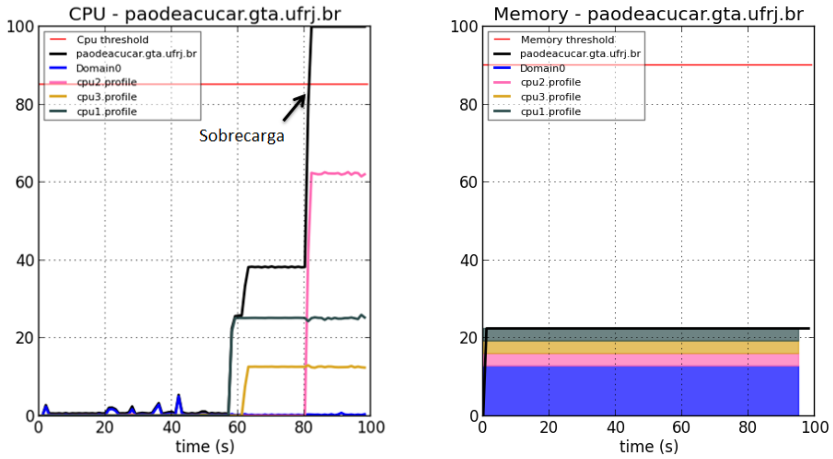



(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

**Figura 6. Configuração do cenário de testes após o Passo 2. O sistema AMAS migrou a máquina memory2 para eliminar a sobrecarga de memória na máquina leblon.gta.ufrj.br. a) A migração da máquina virtual consome processamento do Domínio 0 na máquina de origem. b) A migração da máquina virtual memory2 libera memória alocada, eliminando a sobrecarga de memória. c) A migração da máquina virtual consome processamento do Domínio 0 na máquina de destino. d) Enquanto uma máquina virtual não é completamente migrada, a libvirt não contabiliza sua memória como alocada.**

parte de sua memória ao domínio memory3, recém criado. Com a criação desse domínio, o limiar de segurança é ultrapassado e o Orquestrador de migração é acionado para distribuir as cargas. A Figura 6 ilustra o resultado da ação do Orquestrador que consistiu na migração da máquina virtual memory2 da máquina *leblon.ufrj.br* para a máquina *paodeacucar.ufrj.br*. Conforme o esperado, a máquina memory2 foi selecionada por possuir a menor quantidade de memória alocada dentre as 3 máquinas virtuais candidatas. Através da Figura 6 também é possível observar o impacto que a migração de máquinas virtuais causa na utilização do processador pelo Domínio 0 das duas máquinas físicas envolvidas.

No Passo 4 do cenário de testes, é simulada uma carga de trabalho nas máquinas




(a) Uso de processador. (b) Memória alocada.

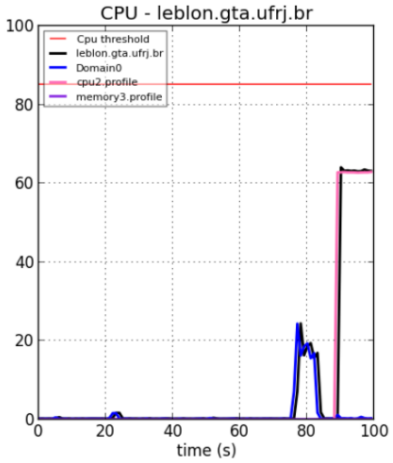
**Figura 7. Passo 4 do cenário de testes. Configuração da máquina paodeacucar.gta.ufrj.br durante a execução do `Stress` nas máquinas virtuais cpu1, cpu2 e cpu3. a) A soma do consumo de CPU de todas as máquinas virtuais gera uma situação de sobrecarga de processamento. b) A alocação de memória se mantém abaixo do limiar já que não há a criação de nenhuma nova máquina virtual.**

cpu1, cpu2 e cpu3, através do programa Stress. Esse passo é ilustrado na Figura 7, onde é possível observar que a utilização do processador da máquina *paodeacucar.gta.ufrj.br* ultrapassa o limiar de segurança de processamento, acionando o algoritmo de migração. De acordo com o algoritmo de decisão, a máquina selecionada para a migração é a cpu2, uma vez que apresenta a maior relação $\frac{CPU}{\text{memória}}$ dentre as máquinas virtuais candidatas. O resultado da migração pode ser visto na figura 8, onde é possível verificar a diminuição da memória alocada e do processamento da máquina *paodeacucar.gta.ufrj.br* e um aumento na quantidade de memória alocada e do processamento da máquina *leblon.gta.ufrj.br*.

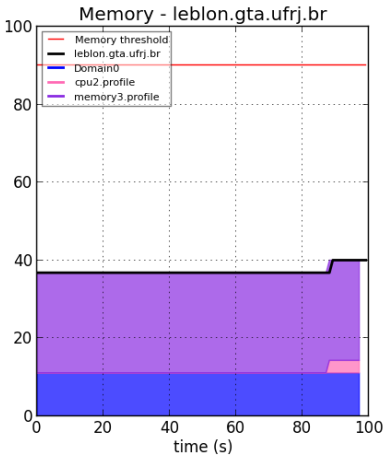
Os resultados obtidos nos testes demonstram que o sistema proposto é capaz de realizar o monitoramento de CPU, memória e rede, detectar a escassez destes recursos e distribuir as cargas de trabalho entre as máquinas físicas monitoradas. O sistema também permite a observação do comportamento do Domínio 0 nas diferentes situações, como criação, migração e destruição de máquinas virtuais. Isto é muito importante para a gerência dos acordos de níveis de serviço para verificação de alguma tentativa maliciosa de ataque negação de serviço.

## 5. Conclusão

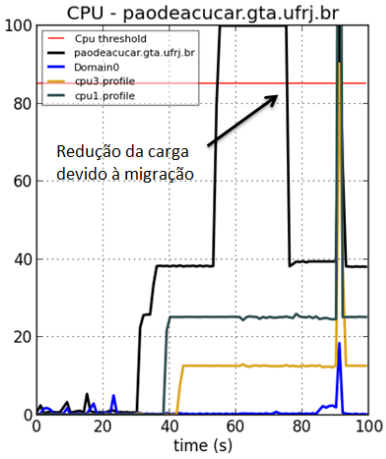
Esse artigo propõe o *Automatic Migration and Allocation System for Virtual Resource Management* (AMAS), um sistema de gerência automatizado de recursos para ambientes virtualizados, no qual é possível realizar o monitoramento, a detecção de sobrecargas e a distribuição de cargas de trabalho. O sistema é capaz de monitorar a utilização do processador, a quantidade de memória alocada e o tráfego de rede de um conjunto de máquinas físicas e virtuais, detectando escassez desses recursos e distribuindo cargas, de forma a evitar perda de desempenho nas aplicações em execução. O mecanismo de distribuição de cargas é baseado na técnica de migração ao vivo fornecida pelo Xen, que permite a migração de máquinas virtuais entre nós físicos, sem a interrupção dos serviços em execução. O sistema foi testado utilizando máquinas da plataforma de testes *Future Internet Testbed with Security* (FITS), permitindo a validação e uma avaliação



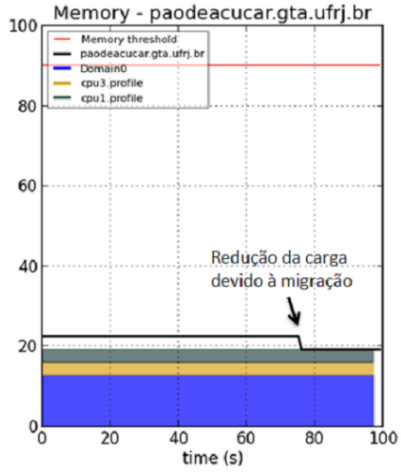

(a) Uso do processador da máquina leblon.gta.ufrj.br.




(b) Memória alocada da máquina leblon.gta.ufrj.br.




(c) Uso do processador da máquina paodeacucar.gta.ufrj.br.




(d) Memória alocada da máquina paodeacucar.gta.ufrj.br.

**Figura 8. Configuração do cenário de testes após o Passo 4. O sistema AMAS identifica a sobrecarga de processamento e migra a máquina virtual cpu2 para balancear a carga. a) Após a migração, a máquina virtual cpu2 executa na máquina física leblon.gta.ufrj.br, cuja carga continua abaixo do limiar de segurança definido pelo sistema AMAS. b) Enquanto uma máquina virtual não é completamente migrada, a libvirt não contabiliza sua memória como alocada. c) A migração da máquina cpu2 libera recursos de processamento, eliminando a sobrecarga de cpu da máquina física paodeacucar. d) A alocação de memória após a migração da máquina virtual cpu2 libera memória alocada.**

do desempenho da proposta. Diferentes cenários de sobrecarga foram testados e os resultados obtidos mostram que o sistema proposto é capaz de manter o nível de serviço das máquinas virtuais, mesmo em cenários de sobrecarga das máquinas físicas. Também foi implementado um painel de controle que permite a visualização do uso dos recursos computacionais em tempo real.

Como trabalho futuro, pretende-se integrar os mecanismos de monitoramento e distribuição de carga nas máquinas do FITS e disponibilizá-los como serviço na interface

Web da plataforma de teste. O sistema AMAS pode ser adaptado para computação verde através do uso métricas e critérios de migração que minimizem o consumo de energia.

## 6. Referências


[Alkmim et al., 2011] Alkmim, G., Batista, D. e Fonseca, N. (2011). Mapeamento de redes virtuais em substratos de rede. Em *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2011*, Campo Grande, MS.

[Arzuaga e Kaeli, 2010] Arzuaga, E. e Kaeli, D. R. (2010). Quantifying load imbalance on virtualized enterprise servers. Em *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, WOSP/SIPEW '10, p. 235–242, New York, NY, USA. ACM.

[Barham et al., 2003] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T. L., Ho, A., Neugebauer, R., Pratt, I. e Warfield, A. (2003). Xen and the art of virtualization. Em *SOSP*, p. 164–177. ACM.

[Carvalho e Duarte, 2012] Carvalho, H. E. T. e Duarte, O. C. M. B. (2012). Voltaic: volume optimization layer to assign cloud resources. Em *Proceedings of the 3rd International Conference on Information and Communication Systems*, ICICS '12, p. 3:1–3:7.

[Cherkasova e Gardner, 2005] Cherkasova, L. e Gardner, R. (2005). Measuring cpu overhead for i/o processing in the xen virtual machine monitor. Em *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, p. 24–24, Berkeley, CA, USA. USENIX Association.

[Fajjari et al., 2011] Fajjari, I., Aitsaadi, N., Pujolle, G. e Zimmermann, H. (2011). Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic. Em *Communications (ICC), 2011 IEEE International Conference on*, p. 1–6.

[Govindan et al., 2007] Govindan, S., Nath, A. R., Das, A., Urgaonkar, B. e Sivasubramaniam, A. (2007). Xen and co.: Communication-aware cpu scheduling for consolidated xen-based hosting platforms.

[Guo et al., 2013] Guo, J., Liu, F., Zeng, D., Lui, J. e Jin, H. (2013). A cooperative game based allocation for sharing data center networks. Em *INFOCOM, 2013 Proceedings IEEE*, p. 2139–2147.

[Khanna et al., 2006] Khanna, G., Beaty, K., Kar, G. e Kochut, A. (2006). Application Performance Management in Virtualized Server Environments. Em *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, p. 373–381.

[Lee et al., 2010] Lee, M., Krishnakumar, A. S., Krishnan, P., Singh, N. e Yajnik, S. (2010). Xentune: Detecting xen scheduling bottlenecks for media applications. Em *GLOBECOM*, p. 1–6. IEEE.

[Red Hat, 2013] Red Hat (2013). libvirt: The virtualization API. `http://libvirt.org/index.html`. Acessado em dezembro de 2013.

[Rodriguez et al., 2013] Rodriguez, E., Alkmim, G., Batista, D. e da Fonseca, N. (2013). Live migration in green virtualized networks. Em *Communications (ICC), 2013 IEEE International Conference on*, p. 2262–2266.

[Singh et al., 2008] Singh, A., Korupolu, M. e Mohapatra, D. (2008). Server-storage virtualization: Integration and load balancing in data centers. Em *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, p. 1–12.

[Sonnek et al., 2010] Sonnek, J., Greensky, J., Reutiman, R. e Chandra, A. (2010). Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration. Em *Parallel Processing (ICPP), 2010 39th International Conference on*, p. 228–237.

[Waterland, 2013] Waterland, A. (2013). stress project page. `http://people.seas.harvard.edu/~apw/stress/`. Acessado em dezembro de 2013.

[Wood et al., 2009a] Wood, T., Shenoy, P., Venkataramani, A. e Yousif, M. (2009a). Sandpiper: Black-box and gray-box resource management for virtual machines. *Comput. Netw.*, 53(17):2923–2938.

[Wood et al., 2009b] Wood, T., Tarasuk-Levin, G., Shenoy, P., Desnoyers, P., Cecchet, E. e Corner, M. D. (2009b). Memory buddies: Exploiting page sharing for smart colocation in virtualized data centers. Em *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, p. 31–40, Washington, DC, USA. ACM.

[Wu e Zhao, 2011] Wu, Y. e Zhao, M. (2011). Performance modeling of virtual machine live migration. *2012 IEEE Fifth International Conference on Cloud Computing*, 0:492–499.

[Xen Project, 2009] Xen Project (2009). How Does Xen Work? `http://www-archive.xenproject.org/files/Marketing/HowDoesXenWork.pdf`. Acessado em dezembro de 2013.

[Yanagisawa et al., 2013] Yanagisawa, H., Osogami, T. e Raymond, R. (2013). Dependable virtual machine allocation. Em *INFOCOM, 2013 Proceedings IEEE*, p. 629–637.

# Reconectando Partições de Infraestruturas Físicas: Rumo a uma Estratégia de Expansão para o Mapeamento Eficiente de Redes Virtuais

**Marcelo Caggiani Luizelli[1], Leonardo Richter Bays[1], Luciana Salete Buriol[1], Marinho Pilla Barcellos[1], Luciano Paschoal Gaspary[1]**

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{mcluizelli,lrbays,buriol,marinho,paschoal}@inf.ufrgs.br

***Abstract.*** *One of the research challenges approached recently in the literature is the efficient mapping of virtual networks on top of physical infrastructures. Although there have been efforts to solve it, we observe that a number of virtual network requests are rejected due to the exhaustion of resources only in key points of the infrastructure. In this paper, we propose an expansion strategy based on the reconnection of strongly connected components (partitions) of the infrastructure in order to suggest adjustments that lead to higher virtual network acceptance and, in consequence, to improved physical resource utilization. The obtained results evidence that an expansion of 10% to 20% of the infrastructure resources using the proposed strategy leads to a sustained increase of up to 30% in the number of accepted virtual networks and of up to 45% in resource usage compared to the original network.*

***Resumo.*** *Um dos desafios de pesquisa abordados recentemente na literatura é o mapeamento eficiente de redes virtuais em infraestruturas físicas. Embora tenha-se empreendido esforços para resolvê-lo, observa-se que parte das requisições de redes virtuais é rejeitada devido ao esgotamento de recursos apenas em pontos-chave da infraestrutura. Neste artigo, propõe-se uma estratégia de expansão baseada na reconexão de componentes fortemente conexos (partições) da infraestrutura para sugerir ajustes que levem a uma maior aceitação de redes virtuais e, em consequência, a um melhor aproveitamento da infraestrutura física. Os resultados obtidos evidenciam que a expansão de 10% a 20% dos recursos da infraestrutura utilizando a estratégia proposta contribui para um aumento sustentado de até 30% no número de redes virtuais aceitas e de até 45% no aproveitamento dos recursos em comparação com a rede original.*

## 1. Introdução

A virtualização de redes é um mecanismo que permite a coexistência de múltiplas redes virtuais (VNs – *Virtual Networks*) compartilhando recursos de um mesmo substrato físico. Essas VNs podem apresentar arquiteturas, protocolos e topologias independentes das do substrato de rede na qual serão instanciadas. Provedores de Infraestruturas (InPs – *Infrastructure Providers*), lançando mão das facilidades de alocação e desalocação de redes virtuais e do isolamento de recursos que as tecnologias de virtualização provêm, passam, assim, a poder oferecer suporte à criação, sob demanda, de redes personalizadas, atendendo a diferentes requisitos impostos pelos contratantes.

Um dos maiores desafios de pesquisa em virtualização de redes é a alocação eficiente de recursos de infraestruturas físicas para requisições de redes virtuais (VNE – *Virtual Network Embedding*). Essa alocação de recursos físicos deve ser ciente das

capacidades dos equipamentos de rede, bem como das demandas requeridas pelas redes virtuais (por exemplo, largura de banda dos enlaces virtuais e capacidade de processamento dos roteadores virtuais). Apesar de haver um número considerável de trabalhos recentes que exploram o problema *online* de mapeamento de redes virtuais [Yu et al. 2008, Chowdhury et al. 2009, Alkmim et al. 2011, Cheng et al. 2012, Bays et al. 2012, Alkmim et al. 2013], constata-se que as taxas de rejeição para o conjunto de requisições entrantes são, normalmente, altas. Em uma avaliação previamente realizada [Luizelli et al. 2013], verificou-se que um subconjunto dessas rejeições é causado por insuficiências temporárias de recursos, ou seja, períodos em que os recursos disponíveis na infraestrutura como um todo não são capazes de suprir a demanda. Observou-se, contudo, que grande parte das rejeições ocorre em situações em que há grande disponibilidade de recursos, mas alguns poucos já saturados acabam inviabilizando, em função de características de conectividade do substrato, o atendimento de novas requisições.

Como mencionado, embora tenha-se empreendido esforços para resolver o problema do mapeamento de redes virtuais, desconhecem-se trabalhos que investiguem como a rede física de um provedor de infraestrutura pode ser expandida para acomodar uma maior quantidade de redes virtuais. Ao mesmo tempo, abordagens clássicas para o planejamento da expansão de infraestruturas físicas [Mukherjee et al. 1996, Ramaswami and Sivarajan 1996, Krishnaswamy and Sivarajan 2001] não são adequadas para ambientes de virtualização de redes. No caso dessas abordagens tradicionais (e consolidadas), conhece-se *a priori* quais pares de dispositivos requerem uma maior quantidade de recursos, e a expansão é realizada com base em uma matriz de demanda (que indica onde se encontram os gargalos da rede). No entanto, a matriz de demanda observada como resultado da criação de *slices* em redes de InPs[1] tende a apresentar uma distribuição comparativamente mais homogênea de recursos entre pares de dispositivos físicos. Tal advém do fato de que elementos de redes virtuais, em geral, podem ser hospedados em (praticamente) qualquer dispositivo físico com recursos suficientes disponíveis. Visto que a localidade de dispositivos físicos possui pouca influência no processo de mapeamento de redes virtuais, não é possível identificar gargalos de forma tão clara. Essas características tornam um desafio a identificação de regiões do substrato que precisam ser replanejadas para aumentar a probabilidade de acomodar com sucesso novas requisições de redes virtuais.

Neste artigo, propõe-se uma estratégia embasada na reconexão de componentes fortemente conexos (partições) para sugerir ajustes na infraestrutura que levem a um aumento nas taxas de aceitação. Tal estratégia tem como objetivo promover a reconexão de partições recorrentes, favorecendo a acomodação de um maior número de redes virtuais a longo prazo. Para provar o conceito e a viabilidade técnica da estratégia proposta, conduziu-se um conjunto extensivo de experimentos a fim de observar aspectos como ganhos em taxa de aceitação e no aproveitamento dos recursos físicos. As principais contribuições deste artigo se desdobram em duas: *(i)* definição pioneira do problema de expansão de infraestruturas físicas no contexto da virtualização de redes; e *(ii)* avaliação detalhada da estratégia proposta, demonstrando como o fortalecimento de partes-chave de uma infraestrutura pode levar a sua ocupação de forma muito mais satisfatória.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta uma discussão dos trabalhos relacionados ao problema de expansão de redes. Na Seção 3 é formalizado o problema de expansão de infraestruturas de InPs e apresentada uma solução algorítmica para o referido problema. Na Seção 4 são apresentados e avaliados os resultados obtidos. Por fim, a Seção 5 conclui o artigo com considerações finais e

[1]Neste texto as expressões *substrato físico*, *rede física*, *infraestrutura física* e *rede de InP* são usadas como sinônimos.

perspectivas de trabalhos futuros.

## 2. Trabalhos Relacionados

Nesta seção apresenta-se alguns dos principais trabalhos relacionados ao problema de planejamento e expansão de redes de computadores. Faz-se um breve resumo dos principais trabalhos, destacando os contextos em que foram aplicados e as metodologias adotadas. Inicialmente, são apresentados trabalhos clássicos de planejamento e expansão de redes e, posteriormente, trabalhos mais recentes.

Mukherjee et al. [Mukherjee et al. 1996] propõem um modelo de expansão para redes ópticas cujos objetivos são minimizar o atraso médio entre os nós da infraestrutura e maximizar a capacidade dos enlaces, visando a favorecer o atendimento de demandas futuras. A solução proposta é baseada na metaheurística *Simulated Annealing* e no algoritmo de desvio de fluxos. As expansões são realizadas tendo como base uma matriz de tráfego estática, a qual representa a média de tráfego entre os pares origem-destino da infraestrutura.

Ramaswami et al. [Ramaswami and Sivarajan 1996] apresentam uma formulação para o problema de expansão considerando a minimização global do congestionamento da infraestrutura. As expansões são realizadas considerando limites máximos de atraso permitidos e demandas existentes entre pares origem-destino. Os autores apresentam inicialmente um modelo de otimização e, posteriormente, por questões de escalabilidade, uma abordagem heurística que realiza a expansão da capacidade de caminhos físicos entre pares de roteadores com alta demanda, priorizando caminhos mais curtos.

Krishnaswamy et al. [Krishnaswamy and Sivarajan 2001] também abordam o problema de expansão de redes ópticas com o objetivo de minimizar o congestionamento entre nós da infraestrutura. No entanto, consideram que não é permitida a adição de repetidores na topologia física e, por isso, a inserção de novas fibras é restrita a um comprimento máximo. Os autores apresentam resultados numéricos para o modelo proposto aplicando-o em redes pequenas. Para instâncias que representam redes maiores, os autores realizam a relaxação de variáveis inteiras do modelo e, posteriormente, aplicam uma técnica de integralização nas mesmas.

Passa-se, agora, a uma breve revisão dos trabalhos recentes da área. Curtis et al. [Curtis and Lopez-Ortiz 2009] apresentam um modelo que possui como objetivo otimizar a utilização da capacidade dos elementos da infraestrutura considerando aspectos de balanceamento de carga e resiliência. O modelo visa a garantir que o planejamento da infraestrutura é capaz de satisfazer um conjunto de requisitos referente à qualidade de serviço. De forma similar aos trabalhos clássicos apresentados anteriormente, os autores empregam uma estimativa baseada em matrizes de tráfego para realizar o planejamento da capacidade dos enlaces da infraestrutura.

Johnston et al. [Johnston et al. 2011] apresentam um esquema no qual uma rede de *backup* é planejada para prover proteção contra múltiplas falhas em enlaces. Os autores propõem um modelo de expansão que determina quais elementos precisam ser adicionados à infraestrutura física, além de definir a capacidade adequada de cada um. Por razões de escalabilidade, os autores propõem, ainda, uma metaheurística baseada em *Simulated Annealing* para viabilizar a resolução do problema considerando redes de grande porte.

Gangxiang et al. [Shen and Tucker 2009], por sua vez, propõem um modelo de planejamento de infraestruturas que busca a redução do consumo de energia de redes de *backbone*. O modelo considera o consumo energético dos componentes da rede para definir em quais regiões da infraestrutura novos recursos devem ser adicionados. Assim como os demais trabalhos da área, os autores utilizam matrizes de demanda para realizar

a expansão dos recursos. Os autores também desenvolvem heurísticas, visando a garantir a escalabilidade da solução proposta.

O ponto comum dos trabalhos descritos até então reside no emprego de matrizes de demanda para planejar a expansão de redes físicas. No entanto, no contexto de virtualização de redes, um estudo preliminar realizado em antecipação a esse trabalho revelou que as matrizes de demanda observadas em redes de InPs tendem a apresentar uma utilização comparativamente mais homogênea de recursos entre pares de dispositivos físicos. Como consequência de tal homogeneidade, as características topológicas da infraestrutura, desconsideradas nos trabalhos anteriores, passam a apresentar uma importância consideravelmente superior no processo de expansão do que as matrizes de demanda observadas.

Além dos trabalhos mencionados anteriormente, cujo foco é a expansão da capacidade de enlaces em redes de *backbone*, há também soluções recentes que focam no planejamento de redes de *datacenters*. Dentre esses trabalhos, destacam-se as publicações de Curtis et. al. [Curtis et al. 2012] e Gao et al. [Gao et al. 2012], que introduzem modelos de otimização e heurísticas aplicados à expansão de tais redes. De forma similar aos trabalhos que focam na expansão de redes de *backbone*, consideram-se a inclusão de equipamentos e/ou o aumento da capacidade de equipamentos já existentes de forma a satisfazer a demanda requerida nas infraestruturas.

Os trabalhos focados no planejamento/expansão de redes de *datacenters* objetivam, em geral, maximizar a capacidade de transmissão de dados simultânea sem interferência entre pares de dispositivos da infraestrutura (largura de banda bisseccional). Os algoritmos desenvolvidos para a expansão de tais redes são, em geral, projetados para topologias específicas (como, por exemplo, topologias baseadas em árvores), o que dificulta (ou mesmo, inviabiliza) a adaptação destes para redes de *backbone*. Ainda que a adaptação fosse possível, a aplicação de tais soluções no contexto de redes de *backbone* traria ineficiência (desperdício) na alocação de recursos, haja visto que existem diferenças entre os objetivos dos modelos de expansão. Diante dessa nova realidade, torna-se um desafio a identificação de regiões do substrato que precisam ser replanejadas para aumentar a probabilidade de acomodar com sucesso novas requisições de redes virtuais, sendo esse alvo de estudo deste trabalho.

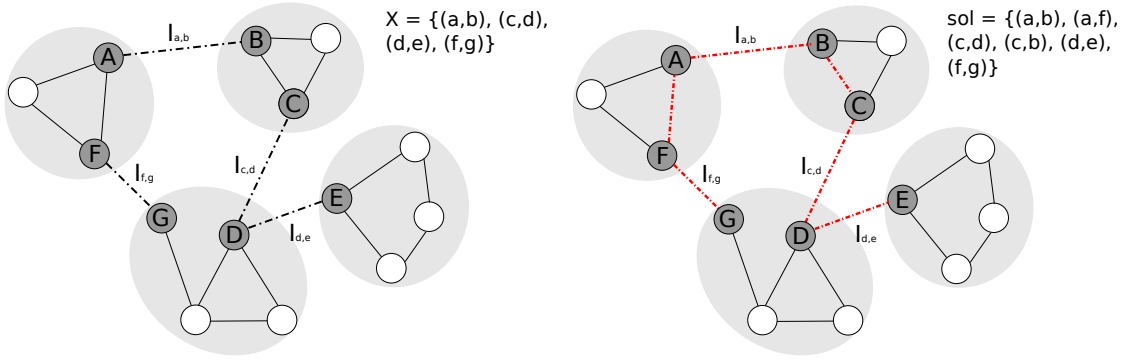
## 3. Expansão de Redes de InP para o Mapeamento de Redes Virtuais

Nesta seção, descreve-se o problema de expansão de redes de InP e formaliza-se um modelo baseado em Programação Linear Inteira. Após, apresenta-se uma solução algorítmica para o modelo.

### 3.1. Visão Geral do Problema

Uma das principais causas de rejeições de redes virtuais no contexto do problema de mapeamento de redes virtuais é a ausência de uma partição adequada na infraestrutura física [Luizelli et al. 2013]. As partições podem ser entendidas como conjuntos isolados entre si de roteadores. A sua ocorrência está diretamente relacionada ao esgotamento total (ou quase total) dos recursos disponíveis em dispositivos específicos (*e.g*, pontes ou *hubs*) da infraestrutura no processo de mapeamento. Com a variação na utilização (alocação/reserva) dos recursos físicos, há também variações no conjunto de partições e, consequentemente, nos dispositivos que pertencem a cada partição. Dessa forma, faz-se necessário identificar os elementos que mais impactam nos particionamentos, isto é, elementos que em situação de consumo total passam a particionar a infraestrutura. A Figura 1(a) ilustra o estado de uma infraestrutura física com, momentaneamente, quatro partições, representadas pelos círculos em tom claro. Na Figura, enlaces que foram

identificados como elementos particionadores ao longo de consecutivos mapeamentos de redes virtuais – $(a, b)$, $(c, d)$, $(d, e)$ e $(f, g)$ – são representados por linhas hachuradas.




(a) Identificação dos enlaces de corte, armazenados em $X$.

(b) Solução baseada no conceito de reconexão dos enlaces de corte. Em destaque, os elementos a serem expandidos.

**Figura 1. Aplicação da estratégia proposta na infraestrutura física.**

Partindo-se da constatação de que o particionamento é uma das grandes causas de rejeições de redes virtuais e observando que a infraestrutura física, por uma perspectiva global, apresenta uma grande quantidade de recursos disponíveis, propomos, como estratégia de expansão de redes de InPs, a reconexão de partições recorrentes. Dessa forma, os elementos antes considerados para a expansão de infraestruturas (matrizes de demanda) deixam de ser considerados e os fatores topológicos passam a ser empregados no processo de decisão. O processo de expansão de redes é sujeito a algumas restrições dos provedores como, por exemplo, o número de elementos a serem expandidos e o capital disponível. Nesse contexto, busca-se investigar quais elementos físicos de redes de InPs (priorizados de acordo com características topológicas relevantes como, por exemplo, enlaces de corte) precisam ser replanejados objetivando a reconexão das partições mais recorrentes. A Figura 1(b) ilustra a solução baseada na reconexão de partições, na qual é possível perceber o fortalecimento estrutural (*i.e.*, onde os investimentos de expansão serão feitos) do caminho composto por múltiplos enlaces – $(g, f)$, $(f, a)$, $(a, b)$, $(b, c)$, $(c, d)$ e $(d, e)$ – e roteadores – $(a, b, c, d, e, f, g)$ – interconectando as principais regiões da infraestrutura. Espera-se que a estratégia proposta seja utilizada com uma certa periodicidade (*e.g.*, a cada seis meses ou a cada ano), dependendo da política de investimentos dos InPs, e que a cada vez possam ser feitas expansões em porções diferentes da rede física, fortalecendo-a de forma orgânica.

### 3.2. Definições e Modelagem

A seguir, são detalhadas as entradas, as variáveis e as restrições desse modelo. Letras sobrescritas são usadas para representar se um conjunto ou variável refere-se a recursos virtuais ($V$) ou físicos ($P$), ou se está associado a roteadores ($R$) ou enlaces ($L$).

**Substrato e redes virtuais.** A topologia da rede física, bem como a de cada rede virtual requisitada, é representada por um grafo direcionado $N = (R, L)$. Os vértices $R$ representam roteadores, enquanto que cada aresta $L$ representa um enlace unidirecional. Enlaces bidirecionais são representados como um par de arestas em direções opostas (por exemplo, $(a, b)$ e $(b, a)$). Dessa forma, o modelo permite a representação de quaisquer tipos de topologias físicas e virtuais.

Em ambientes reais, roteadores físicos tem capacidade limitada de recursos, além de um limite físico para a expansão de suas capacidades. No modelo, as capacidades de

CPU e memória são representadas, respectivamente, por $C_i^P$ e $M_i^P$, enquanto que o limite máximo de expansão, por $EC_i^P$ e $EM_i^P$. Da mesma forma, enlaces físicos possuem uma capacidade, representada por $B_{i,j}^P$, e um limite físico de expansão representado por $EB_{i,j}^P$.

Para considerar o consumo de recursos físicos da infraestrutura, faz-se necessário considerar as redes virtuais previamente alocadas no substrato. Os roteadores virtuais previamente alocados são representados pelos conjunto $A_{i,r,j}^R$, o qual indica se o roteador virtual $j$ da rede virtual $r$ está hospedado no roteador físico $i$. Já os enlaces virtuais previamente alocados são representados pelo conjunto $A_{i,j,r,k,l}^L$, o qual indica se o enlace virtual $(k, l)$ da rede virtual $r$ está hospedado no enlace físico $(i, j)$. Os requisitos de CPU e memória de cada roteador $i$ da rede virtual $r$ são representados por $C_{r,i}^V$ e $M_{r,i}^V$, enquanto que $B_{r,k,l}^V$ representa os requisitos de largura de banda para cada enlace virtual.

**Particionamento do Substrato.** Infraestruturas físicas podem ter diferentes graus de conectividade e topologias. Como consequência do esgotamento dos recursos físicos em regiões específicas do substrato (*e.g*, pontes ou enlaces conectados a *hubs*), o particionamento da infraestrutura ocorre em diferentes níveis. Nesse contexto, uma partição é definida como um Componente Fortemente Conexo (CFC), cujos enlaces possuem largura de banda livre igual ou superior à média requisitada pelas redes virtuais. Os enlaces de corte identificados na infraestrutura são representados pelo conjunto $X$ e, para cada $(i, j) \in X$, existe um valor correspondente no conjunto $I$, o qual representa a importância de cada enlace de corte no contexto do mapeamento de redes virtuais. A forma de valorar esses enlaces será explicada na Subseção 3.3.

**Custos da Expansão.** A expansão das capacidades dos dispositivos incorre em custos para os provedores de redes (InPs). Além disso, os custos monetários para aumentar as capacidades (*e.g,* memória ou largura de banda) dos dispositivos físicos – roteadores e enlaces – são heterogêneos. Tais custos são modelados como $costCpu_i^P$ e $costMem_i^P$, relacionados à expansão de CPU e memória do roteador $i$, e $costBw_{i,j}^P$ relacionado à expansão da largura de banda do enlace físico $(i, j)$. Neste artigo, trabalha-se com a ideia de que a unidade mínima acrescida é modelada em megabits por segundo para enlaces, megabytes para memória e módulos de processamento para CPU. Dessa forma, é possível generalizar o modelo de expansão. A Equação 1 apresenta uma estimativa dos custos totais para expandir a infraestrutura. As variáveis $xb_e, xm_i$ e $xc_i$ representam, respectivamente, a quantidade acrescida de recursos de largura de banda, memória e CPU nos dispositivos da infraestrutura.

$$\begin{aligned} CustoExp = & \sum_{e \in L^P} xb_e \cdot costBw^P e \\ & + \sum_{i \in R^P} (xc_i \cdot costCPU^P i + xm_i \cdot costMem^P i) \end{aligned} \tag{1}$$

**Variáveis.** As variáveis do modelo representam a solução ótima do problema de expansão. As variáveis $z_e$ indicam quais são os enlaces de corte (armazenados em $X$) selecionados para compor um novo *core* para a infraestrutura. Além disso, as variáveis $x_e$ indicam se o enlace $e$ faz parte do caminho construído entre os enlaces de corte selecionados por $z_e$. As variáveis auxiliares $q_u$ indicam se o roteador $u$ do enlace de corte $e$ pertence ao novo *core*. Todas as variáveis pertencem ao domínio binário.

**Restrições.** Além dos custos de expansão, trabalha-se com uma quantidade limitada de recursos. Por esse motivo, o modelo possui um conjunto de restrições, descrito a seguir.

As Equações 2, 3 e 4 garantem que a capacidade expandida nos enlaces (largura

de banda) e nos roteadores físicos (CPU e memória) não excederá o limite máximo de expansão dos recursos da infraestrutura. Ressalta-se que as Equações 3 e 4 são válidas para ambos os índices ($i$ e $j$) da variável $x$. Porém, por questões de espaço, apenas as equações relativas ao índice $i$ são mostradas.

$$x_e \cdot \left(B^P_{i,j} + \frac{CapitalDisp \cdot Cobertura}{costBw^P_{i,j}}\right) \leq EB^P_{i,j} \qquad \forall e \in L^P \quad (2)$$

$$x_{i,j} \cdot \left(C^P_i + \frac{CapitalDisp \cdot Cobertura}{costCpu^P_i}\right) \leq EC^P_i \qquad \forall (i,j) \in L^P \quad (3)$$

$$x_{i,j} \cdot \left(M^P_i + \frac{CapitalDisp \cdot Cobertura}{costMem^P_i}\right) \leq EM^P_i \qquad \forall (i,j) \in L^P \quad (4)$$

A Equação 5 garante que o montante a ser investido na infraestrutura física pelo InP não exceda o montante de capital de que o mesmo dispõe.

$$CustoExp \leq CapitalDisp \quad (5)$$

As Equações 6 e 7 definem um subconjunto de dispositivos físicos para compor um novo *core*, que atua como um reforço estrutural para a rede. O novo *core* é definido a partir de um subconjunto de roteadores físicos ($R^C$), o qual é definido por todos os roteadores que tem um ou mais enlaces em $X$. Constrói-se um caminho entre os enlaces de corte mais relevantes selecionados pela função objetivo (Equação 9). O símbolo $\delta^+(u)$ representa os enlaces de saída do roteador $u$, enquanto que $\delta^-(u)$ representa os enlaces de entrada do roteador $u$.

$$\sum_{a \in \delta^+(u)} x^k_a - \sum_{a \in \delta^-(u)} x^k_a = \begin{cases} 1 \cdot q_u & \text{se } u = s \\ -1 \cdot q_u & \text{se } u = k \\ 0 & u \in R^P / \{s,k\} \end{cases} \qquad \forall u \in R^P, \forall k \in R^C \quad (6)$$

$$q_u \geq \frac{\sum_{a \in \delta^+(u)} z_a}{|\delta^+(u)|} \qquad \forall u \in R^P \quad (7)$$

Na prática, um InP dificilmente desejará expandir a capacidade de todos os dispositivos da rede física. Portanto, a Equação 8 garante que somente um subconjunto dos dispositivos físicos da infraestrutura será afetado pelo procedimento de expansão. Em outras palavras, essa equação define os limites de cobertura para a expansão dos enlaces. O lado esquerdo da equação define o percentual de enlaces afetados pela expansão.

$$\frac{\sum_{\forall e \in L^P} x_e}{|L^P|} \leq Cobertura \quad (8)$$

**Objetivo.** O objetivo do modelo, exibido na Equação 9, é maximizar a reconexão entre os enlaces de corte considerados mais importantes. Como consequência, a taxa de aceitação deve ser maximizada e a taxa de utilização dos recursos ociosos deve ser aprimorada.

$$Maximize \sum_{e \in X} I_e \cdot z_e \quad (9)$$

### 3.3. Estratégia de Expansão Proposta

Nesta subseção é apresentada a estratégia proposta para o problema de expansão de redes de InPs. É detalhado cada procedimento específico utilizado para construir uma solução factível, e apresentada uma visão geral do procedimento algorítmico. A estratégia proposta é composta por duas etapas. A primeira etapa consiste em identificar quais elementos serão replanejados, enquanto que a segunda define uma estratégia para a distribuição de recursos entre os dispositivos já selecionados. Ressalta-se que uma estratégia heurística foi adotada devido à similaridade do problema sendo tratado com o problema da árvore de Steiner mínima [Goemans and Myung 1993], o qual é conhecidamente NP-Difícil.

O Algoritmo 1 apresenta uma versão simplificada da estratégia proposta de reconexão de partições recorrentes em pseudocódigo, e seus detalhes são explicados a seguir. Como descrito anteriormente, uma partição é definida em termos da utilização dos recursos físicos. Assim, em um determinado intervalo de tempo (o que pode ser, por exemplo, um dia ou uma semana), aplica-se um procedimento para identificar o conjunto de partições da infraestrutura, bem como os enlaces de corte (isto é, enlaces que causam particionamentos devido ao esgotamento dos seus recursos). Nas linhas 5 e 6, calcula-se os particionamentos e os enlaces de corte observados em cada um desses intervalos. Existem algoritmos polinomiais para encontrar Componentes Fortemente Conexos (CFC) em grafos (e, por conseguinte, os enlaces de corte) como o algoritmo de Tarjan [Sleator and Tarjan 1983], o qual pode ser computado com complexidade linear de $O(|R| + |L|)$.

Um histórico dos enlaces de corte (computados anteriormente) é armazenado no conjunto $X$ como uma informação para guiar o algoritmo na seleção dos principais dispositivos físicos. Como os enlaces de corte variam ao longo do tempo (o que ocorre devido à variação da utilização dos recursos físicos), mantém-se um conjunto $I$ que contém, para cada enlace $(i, j)$ pertencente a $X$, informações sobre a relevância de tal enlace. O valor armazenado para cada enlace de corte é uma combinação da frequência e da cobertura, os quais vão sendo acumulados a cada intervalo em que os particionamentos e os enlaces de corte são determinados (linhas 7-8). Para a frequência, realiza-se a contagem de quantas vezes o enlace $(i, j)$ foi a causa de um particionamento da infraestrutura. O valor é então normalizado em relação ao montante total. A cobertura, por sua vez, mede a percentagem de roteadores que ficaram desconectados da infraestrutura a partir de tal particionamento.

No momento em que é realizada a expansão da infraestrutura física (o que pode acontecer, por exemplo, semestralmente ou anualmente), analisa-se o histórico de enlaces de corte armazenados até o momento em $X$ e a importância $I$ de cada um. Quanto maiores os valores para $I_{i,j}$, maiores serão as chances de que o enlace em questão esteja frequentemente causando o particionamento de um grande conjunto de roteadores físicos na infraestrutura. O valor de $I_{i,j}$ é calculado pela multiplicação dos valores acumulados para a freqüência e a cobertura. O procedimento é realizado na linha 11 do Algoritmo 1. A estratégia proposta leva em consideração a importância dos elementos topológicos da infraestrutura física, ao contrário das estratégias aplicadas em trabalhos clássicos, as quais são fundamentadas unicamente na utilização de matrizes de demanda para selecionar os dispositivos a serem expandidos. Primeiramente, ordena-se os enlaces físicos $(i, j)$ armazenados em $X$ em relação ao valor correspondente em $I_{i,j}$ (linha 14). Então, um subgrafo $N^c = (R^c, L^c)$ é construído, onde $R^c$ é um subconjunto de roteadores, (cujo tamanho é igual ao percentual de cobertura) contido no conjunto $X$ e $L^c$ é representado por todos os enlaces que compõem os menores caminhos (ou caminhos de menor custo) entre os roteadores de $R^c$ (linhas 15-18). Aplica-se no subgrafo $N^c$ um algoritmo de *Minimum Spanning Tree* (*MST*) para construir um novo *core* para a infraestrutura, o qual irá atuar como um reforço estrutural evitando o particionamento de regiões críticas (linha

**Input**: Capital disponível, Percentual de cobertura, Periodicidade da expansão, Infraestrutura física do InP $N$, Mapeamento das redes virtuais $A^R_{i,r,j}$ and $A^L_{i,j,r,k,l}$
**Output**: Conjunto de dispositivos físicos a serem expandidos

1 $X \leftarrow \emptyset$
2 $F \leftarrow \emptyset$
3 $C \leftarrow \emptyset$
4 **foreach** $UnidadeTempo$ **do**
5 $particoes \leftarrow obterConjuntoParticoes(N, E^R, E^L)$
6 $X \leftarrow atualizarEnlacesCorte(particoes)$
7 $F \leftarrow atualizarFrequencia(X)$
8 $C \leftarrow atualizarCobertura(X, particoes)$
9 **if** $PeriodicidadeDaExpansao$ **then**
10 **foreach** $(i,j) \in X$ **do**
11 $I_{i,j} \leftarrow \frac{F_{i,j}}{\sum_{\forall(i,j)} F_{i,j}} \cdot C_{i,j}$
12 **end**
13 **while** $true$ **do**
14 $ordenaDesc(X, I)$
15 $list \leftarrow$ seleciona % dos enlaces de $X$ (% igual ao percentual de cobertura)
16 $R^c \leftarrow$ roteadores de $list$
17 $L^c \leftarrow$ enlaces dos caminhos de custo mínimo entre cada par de roteador $(a,b) \in R^c$
18 $N^c = (R^c, L^c)$
19 $sol^c \leftarrow mst(N^c)$
20 **if** $numeroRoteadores(sol^c) > Cobertura$ *ou* $numeroEnlaces(sol^c) > Cobertura$ **then**
21 $X.removeItem()$
22 **else**
23 $parar$
24 **end**
25 **end**
26 $sugiraExpansaoEnlaces(CapitalDisponivel)$
27 $sugiraExpansaoRoteadores(CapitalDisponivel)$
28 $X \leftarrow \emptyset$
29 **end**

**Algoritmo 1:** Visão geral da solução proposta para o problema de expansão de redes de InPs.

19). Como existem algoritmos polinomiais para realizar esse procedimento, tal como o algoritmo de Prim, o procedimento pode ser calculado em $O(|L^c| + |R^c|log|R^c|)$ passos. É importante mencionar que a solução resultante da $mst(N^c)$ pode conter mais elementos que a cobertura mínima (restrição 7). Isso acontece porque $L^c$ é composto por um conjunto de caminhos que interligam todos os roteadores de $R^c$. Por esse motivo, podem existir roteadores intermediários na solução que não pertençam inicialmente a $R^c$. Quando isso acontece, repete-se iterativamente o processo de recontrução do grafo $N^c$, eliminando o enlace de corte menos relevante (linhas 20-24).

Por fim, após a criação da estrutura de reforço, é sugerida a expansão da capacidade dos enlaces e roteadores (linhas 22-23). A expansão dos dispositivos físicos pode seguir diferentes estratégias. Por exemplo, todo o capital disponível para a expansão pode ser investido uniformemente entre os roteadores e enlaces selecionados ou definir uma distribuição probabilística baseada na importância de cada elemento. Por questões de espaço, neste artigo a solução de expansão não utiliza uma estratégia de priorização para a distribuição do investimento entre os dispositivos selecionados na etapa anterior (ou seja, os recursos são distribuídos uniformemente).

## 4. Avaliação do Impacto da Expansão no Mapeamento de Redes Virtuais

Para mensurar o impacto de expansões sugeridas pela estratégia proposta no processo de mapeamento de redes virtuais, a mesma foi implementada e submetida a um processo sistemático de avaliação. Os experimentos foram realizados em uma máquina com quatro processadores AMD Opteron 6276 e 64 GB de memória RAM, usando o sistema operacional Ubuntu GNU/Linux Server 11.10 x86_64.

### 4.1. Carga de Trabalho e Modelo de Mapeamento

Para realizar os experimentos, desenvolveu-se um gerador de requisições de redes virtuais. Esse gerador é executado por um período de 360 unidades de tempo (nesta avaliação trabalhamos com a ideia de que cada unidade de tempo corresponde a um dia, perfazendo o período de 360 dias). Em cada unidade de tempo, três requisições de redes virtuais são geradas. Cada requisição gerada possui uma duração limitada, ou seja, após um determinado número de unidades de tempo, a mesma é removida. Ressalta-se que essa forma de instanciação, baseada no emprego de unidades de tempo e com uso de parâmetros fixos para a chegada de requisições, tem como objetivo proporcionar um maior controle sobre os experimentos, evitando que os resultados sejam afetados de forma imprevisível pela variação simultânea de múltiplos fatores.

As redes utilizadas como substrato físico foram geradas por meio da ferramenta IGen[2]. A topologia dessas redes segue o padrão *hub & spoke*, conforme caracterização realizada por Luizelli et al. [Luizelli et al. 2013]. Essa classe de topologia foi escolhida como uma aproximação de redes encontradas em ambientes reais. As redes físicas instanciadas possuem 50 roteadores, cada um com capacidade total de CPU definida como 100% e 256 MB de memória, enquanto que a largura de banda dos enlaces físicos é de 10 Gbps. Além disso, as capacidades de expansão de roteadores e enlaces físicos são definidas como o dobro de suas capacidades iniciais.

A topologia de cada rede virtual segue a forma de anel. Acredita-se que tal topologia é capaz de representar requisições de redes virtuais de forma adequada, devido a características como a resiliência a falhas individuais em enlaces. Além disso, ressalta-se que o emprego de classes de topologias típicas de provedores (como, por exemplo, *hub & spoke*), nesse caso, poderia levar a uma redução artificial nas taxas de rejeição devido à similaridade entre as redes virtuais e a rede física. As redes virtuais possuem 5 roteadores cada. Roteadores virtuais requerem 20% de CPU e 48 MB de memória, enquanto que a largura de banda requisitada é de 2.5 Gbps. Ressalta-se que tal escolha de parâmetros tem como objetivo reproduzir, por meio de uma carga de trabalho homogênea, os mesmos fenômenos observados em avaliações prévias [Luizelli et al. 2013]. Tais fenômenos incluem uma alta taxa de rejeição de redes virtuais e uma quantidade significativa de recursos disponíveis na infraestrutura de maneira global. Na avaliação recém referida, observou-se que a carga de trabalho descrita leva a uma taxa de aceitação de 66,60% e a uma utilização global de recursos de 60,03% para largura de banda, 52,43% para CPU e 55,93% para memória. Ou seja, apesar do consumo de recursos ficar abaixo de 60%, ainda se observa em [Luizelli et al. 2013] taxas de rejeição da ordem de 30%. Vale frisar que realidade semelhante é observada em outros trabalhos relacionados a VNE, tais como os citados na Introdução. Com a expansão criteriosa da infraestrutura física, cujos resultados são apresentados em Subseção a seguir, espera-se alcançar patamares bem superiores de aceitação e, mais importante, a um melhor aproveitamento sustentado dessa grande quantidade de recursos ociosos.

Durante a avaliação, realiza-se uma única expansão, na 180ª unidade de tempo (*i.e.*, passado um semestre). Os custos de expansão também são definidos de maneira homogênea, isto é, considera-se que os dispositivos são iguais e, portanto, os custos de expansão são os mesmos. Os experimentos possuem, ainda, dois parâmetros variáveis: a cobertura de expansão e o percentual de crescimento. Foram executadas 30 repetições de cada experimento, considerando como base diferentes instâncias de substrato físico.

Como o modelo de expansão proposto está intrinsecamente relacionado com o problema de mapeamento de redes virtuais, considera-se o problema *online* proposto por

[2]http://igen.sourceforge.net/

Luizelli et al. [Luizelli et al. 2013]. Naquele modelo, os autores consideram as principais restrições relatadas na literatura recente. Em contraste, neste trabalho considera-se uma versão simplificada na qual não são consideradas restrições de localidade, uma vez que o objetivo é avaliar, inicialmente, o impacto da aplicação do modelo de expansão em uma versão genérica do problema de mapeamento de redes virtuais.

### 4.2. Resultados

Primeiramente, analisa-se o ganho médio em termos de requisições de redes virtuais aceitas após a expansão dos recursos da infraestrutura física. Ressalta-se que requisições de redes virtuais somente são aceitas caso seja possível mapear todos os seus roteadores e enlaces virtuais no substrato. A Figura 2 ilustra os ganhos médios de redes virtuais *adicionais* aceitas considerando variações no percentual de recursos físicos expandidos (expansão) e na quantidade de dispositivos afetados pela expansão (cobertura). Quanto maior é o percentual da cobertura, mais o investimento é multiplexado entre os dispositivos da infraestrutura. O gráfico revela que o ideal é haver um equilíbrio entre o investimento e a cobertura da expansão. Se a cobertura da expansão for demasiadamente alta ou baixa, os benefícios obtidos são menos significativos. As médias observadas apontam que, para maximizar a aceitação adicional de redes virtuais, a cobertura da expansão deve ser entre 20% e 30%. Considerando esses percentuais de cobertura, observa-se, respectivamente, um aumento de 31,06% e 31,76% na aceitação adicional de redes virtuais para um aumento (investimento) de 20% de novos recursos em relação à rede original. Para um investimento de 10%, os ganhos observados são de, respectivamente, 14,48% e 12,87%.

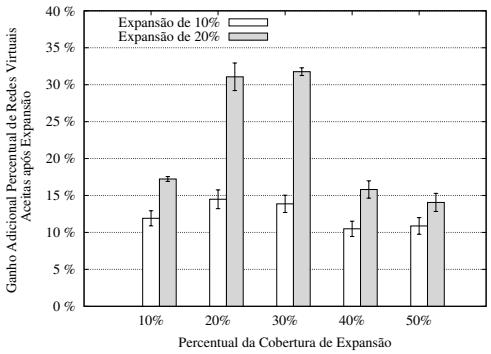
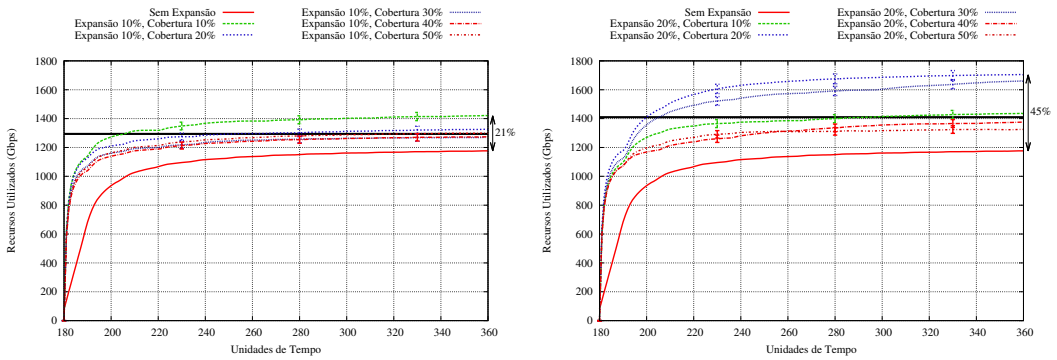



**Figura 2. Média adicional de redes virtuais aceitas após a aplicação do procedimento de expansão.**

A Figura 3 ilustra a utilização média dos enlaces físicos após a expansão. Cada ponto no gráfico representa a utilização média alcançada desde o início do experimento até a unidade de tempo em questão. A linha horizontal em preto em cada gráfico representa a expansão de recursos realizada. Valores acima dessa linha indicam que o provedor está utilizando, além dos recursos expandidos (10% na Figura 3(a) e 20% na Figura 3(b), como explicado a seguir), uma parcela de recursos não utilizados (ou pouco utilizados) antes do processo de expansão. Nos casos em que a média é inferior à linha horizontal, ocorre a subutilização dos recursos expandidos. A Figura 3(a) apresenta a média de utilização dos recursos para uma expansão de 10% dos recursos da rede física. Nesse caso, nota-se que a utilização dos recursos é melhor aproveitada quando a cobertura é menor. Para uma cobertura de até 20%, é possível obter um ganho sustentado de utilização de recursos de até 21% em relação ao cenário sem expansão. Nos demais casos, com cobertura igual ou superior a 30%, o benefício obtido em termos de utilização de recursos mantém-se abaixo dos 10% investidos. Isso está diretamente relacionado à multiplexação de poucos recursos em muitos dispositivos, o que leva a um aumento bem pouco significativo das

capacidades dos recursos expandidos. A Figura 3(b) apresenta a média de utilização de largura de banda para uma expansão de 20% dos recursos. Nota-se que, para coberturas de 20% e 30%, tem-se um ganho de utilização de recursos de até 45% em relação ao cenário sem expansão. Os ganhos observados em termos de utilização de recursos devem-se, principalmente, ao reforço estrutural realizado em certas regiões da infraestrutura, o que leva a um menor nível de particionamento e favorece diretamente a uma menor fragmentação dos recursos ociosos. Ressalta-se que tais benefícios podem levar a custos mais baixos para os solicitantes, uma vez que o provedor consegue melhor aproveitar recursos antes não utilizados. Assim como na avaliação anterior, o gráfico revela que existe uma relação entre o percentual investido, a cobertura da expansão e o percentual de ganho na utilização dos recursos físicos.




(a) Expansão de 10%. (b) Expansão de 20%.

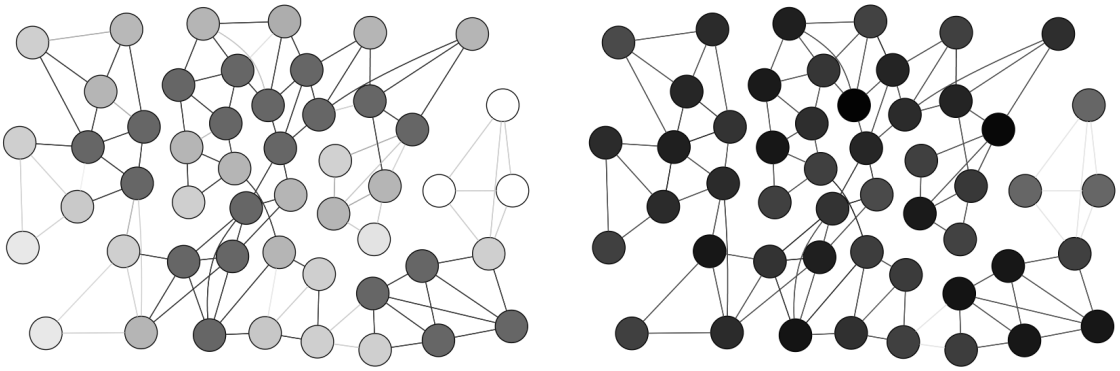
**Figura 3. Utilização global de largura de banda dos enlaces físicos após a expansão.**

A seguir, a Figura 4 apresenta uma visão qualitativa da estratégia proposta por meio de uma representação gráfica da utilização média dos recursos de uma infraestrutra física antes e depois da aplicação do procedimento de expansão. Nota-se, primeiramente, que a utilização dos recursos dos enlaces físicos e roteadores antes da expansão está concentrada em certas regiões da infraestrutura. Devido à superutilização dos recursos de certos dispositivos físicos (principalmente de nós *hubs* e enlaces de corte), os recursos da infraestrutra, de modo geral, acabam sendo subutilizados. Após a aplicação da estratégia de expansão, a qual cria uma estrutura de reforço baseada na reconexão das partições recorrentes, nota-se que a distribuição do consumo de recursos físicos é, de maneira geral, significativamente mais homogênea. Ademais, a estratégia de expansão proporciona uma maior utilização de recursos previamente subutilizados na infraestrutura.

Por fim, discute-se o tempo médio necessário para encontrar uma solução factível para o problema de expansão. Em todos os experimentos, o tempo médio necessário para a resolução do algoritmo proposto permanece abaixo de 1 segundo, já que a estratégia de expansão proposta apresenta complexidade polinomial. Tais resultados indicam que a solução pode ser aplicada em infraestruturas com escalas maiores, ainda assim sendo capaz de gerar soluções em um tempo hábil.

## 5. Conclusão

A virtualização de redes é um tema que vem recebendo considerável atenção da comunidade científica e da indústria, resultando em uma série de trabalhos que envolvem principalmente questões de mapeamento de redes virtuais. Embora tenha-se empreendido esforços para resolver tal problema, observa-se que grande parte das requisições de redes virtuais são rejeitadas devido ao esgotamento de recursos apenas em pontos-chave da



(a) Consumo de recursos antes da expansão. (b) Consumo de recursos após a expansão.

**Figura 4. Representação gráfica do consumo médio de recursos da infraestrutura antes (a) e depois (b) de uma expansão de 20% dos recursos com cobertura de 20%. Tons mais escuros representam maior utilização de recursos físicos.**

mesma. Além disso, desconhecia-se trabalhos que investigassem como a rede física de um provedor de infraestrutura pode ser expandida para acomodar uma fração significativamente maior de redes virtuais em comparação ao alcançado com a rede física original. Neste artigo propôs-se uma estratégia de expansão baseada na reconexão de partições recorrentes para suprir as necessidades que a virtualização impõe em redes de InPs. A principal inovação da estratégia consiste em considerar fatores topológicos no processo de expansão ao invés de matrizes de demanda.

Após formalizar uma estratégia de expansão para infraestruturas físicas de InPs no contexto do problema de mapeamento de redes virtuais e aplicá-la sobre substratos com topologias observadas em redes de provedores, avaliou-se a mesma no que diz respeito a melhorias em relação à taxa de aceitação e utilização de recursos da infraestrutura. Os resultados obtidos evidenciam que a expansão dos recursos da infraestrutura utilizando a estratégia proposta contribui significativamente para um aumento sustentado de até 30% no número de redes virtuais aceitas e de até 45% no aproveitamento dos recursos em comparação com a rede original.

Como perspectivas de trabalhos futuros, pretende-se estender a avaliação da estratégia de expansão aplicando-a em outras topologias de *backbone*, bem como aprofundar a compreensão do inter-relacionamento dos parâmetros do modelo. Além disso, pretende-se propor e avaliar novas estratégias de distribuição de recursos entre os dispositivos da infraestrutura a serem expandidos.

## Referências

Alkmim, G. P., Batista, D. M., and Fonseca, N. L. S. (2011). Mapeamento de redes virtuais em substratos de rede. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 45–58, Campo Grande, MS, Brazil.

Alkmim, G. P., Batista, D. M., and Fonseca, N. L. S. (2013). Mapping virtual networks onto substrate networks. *Journal of Internet Services and Applications*, 3(4):1–15.

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 *Proceedings of the 8th International Conference on Network and Service Management*, pages 378–384, Las Vegas, USA.

Cheng, X., Su, S., Zhang, Z., Shuang, K., Yang, F., Luo, Y., and Wang, J. (2012). Virtual network embedding through topology awareness and optimization. *Computer Networks*, 56(6):1797 – 1813.

Chowdhury, N., Rahman, M., and Boutaba, R. (2009). Virtual network embedding with coordinated node and link mapping. In *Proceedings of the 28th IEEE Conference on Computer Communications*, pages 783 –791, Rio de Janeiro, Brasil.

Curtis, A., Carpenter, T., Elsheikh, M., Lopez-Ortiz, A., and Keshav, S. (2012). Rewire: An optimization-based framework for unstructured data center network design. In *Proceedings of the 31st IEEE Conference on Computer Communications*, pages 1116–1124, Orlando, USA.

Curtis, A. and Lopez-Ortiz, A. (2009). Capacity provisioning a valiant load-balanced network. In *Proceedings of the 28th IEEE Conference on Computer Communications*, pages 3006–3010, Rio de Janeiro, Brazil.

Gao, P. X., Curtis, A. R., Wong, B., and Keshav, S. (2012). It's not easy being green. *SIGCOMM Computer Communication Review*, 42(4):211–222.

Goemans, M. X. and Myung, Y.-S. (1993). A catalog of steiner tree formulations. *Networks*, 23(1):19–28.

Johnston, M., Lee, H.-W., and Modiano, E. (2011). A robust optimization approach to backup network design with random failures. In *Proceedings of the 30th IEEE Conference on Computer Communications*, pages 1512–1520, Shanghai, China.

Krishnaswamy, R. M. and Sivarajan, K. N. (2001). Design of logical topologies: a linear formulation for wavelength-routed optical networks with no wavelength changers. *IEEE/ACM Transactions on Networking*, 9(2):186–198.

Luizelli, M., Bays, L., Buriol, L., Barcellos, M., and Gaspary, L. (2013). Characterizing the impact of network substrate topologies on virtual network embedding. In *9th International Conference on Network and Service Management*, pages 42–50, Zurich, Switzerland.

Mukherjee, B., Banerjee, D., Ramamurthy, S., and Mukherjee, B. (1996). Some principles for designing a wide-area wdm optical network. *IEEE/ACM Transactions on Networking*, 4(5):684–696.

Ramaswami, R. and Sivarajan, K. (1996). Design of logical topologies for wavelength-routed optical networks. *IEEE Journal on Selected Areas in Communications*, 14(5):840–851.

Shen, G. and Tucker, R. (2009). Energy-minimized design for ip over wdm networks. *IEEE/OSA Journal of Optical Communications and Networking*, 1(1):176–186.

Sleator, D. D. and Tarjan, R. E. (1983). A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362 – 391.

Yu, M., Yi, Y., Rexford, J., and Chiang, M. (2008). Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Computer Communication Review*, 38(2):17–29.

# A study on substrate network synchrony demands to support hybrid synchrony virtual networks

**Rasha Ghassan Hasan**[1], **Odorico Machado Mendizabal**[1], **Rômulo Reis De Oliveira**[1], **Fernando Luís Dotti**[1]

[1]Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681 Prédio 32 sala 505, Porto Alegre, Brazil

rasha.hasan@acad.pucrs.br, odoricomendizabal@furg.br,

romulo.reis@acad.pucrs.br, fernando.dotti@pucrs.br

***Abstract.*** *Network virtualization has been proposed in the last years, and it received special attention from both networking and distributed systems communities. By offering a flexible and economic alternative for the deployment of customized networks, a wide set of applications becomes eligible to run on top of such infrastructures. However, specific applications' requirements, such as topology, security, and resilience, pose different challenges to the network embedding problem. Among these requirements lays the one of synchrony, that some applications demand time bounds for processing and communication. In this sense, Hybrid Synchrony Virtual Networks (HSVN) has been proposed to fulfil specific synchrony requirements and better support a considerable class of distributed systems. Considering HSVNs, one of the main challenges is to minimize the need for synchronous components in the substrate network, due to their high cost. In this paper, we propose a mathematical model that aims at defining an economic synchrony configuration of the substrate network, subject to the virtual networks demands, and we analyse the physical synchronous resources regarding their properties and topology.*

## 1. Introduction

Network virtualization has been proposed as a flexible and economic approach to build multiple virtual networks over a shared substrate [Chowdhury et al. 2012]. As can be seen in the literature, the diversity of applications pose different challenges for the VNs embedding process, *i.e.* how resources of a physical network (a.k.a. Substrate Network – SN) are used to support VNs. For example, topology [Zhu and Ammar 2006], security [Bays et al. 2012], resilience [Yu et al. 2011], and synchrony [Hasan et al. 2013] requirements.

On a recent work [Hasan et al. 2013], we proposed a new architecture based on VNs for distributed applications that require hybrid synchrony. Those distributed applications may demand a subset of resources to behave synchronously. Thus, the SN has to support such subsets of resources that ensure time bounds on communication and processing. The resulting architecture that combines *Hybrid Synchrony (HS)* with *Virtual Networks (VN)* we name *HSVN*. The SN is composed by two classes of nodes: (i) *synchronous nodes*, which allow timely CPU execution, achieved through implementing periodical real-time tasks, and (ii) *asynchronous nodes*, that have no timely guarantees. Analogously, two classes of the SN physical links are available: (i) *synchronous links*, which

ensure time-bounded messages transmission delay, achieved through implementing QoS communication guarantees, and (ii) *asynchronous links* that have no timely guarantees.

From the system developer standpoint, delegating the resource allocation to the VN mapping process is very advantageous. As well as CPU capacity or bandwidth, the synchrony level demanded by applications is also considered by the mapping process in searching for an economic sharing of resources. From the network virtualization standpoint, the main advantage of HSVN, compared to classical VNs architecture, is the coexistence of synchronous and asynchronous components. By providing this setup, distributed applications running on top of HSVNs can rely on the hybrid synchronous assumption [Cristian and Fetzer 1999, Veríssimo 2006]. Therefore, stronger properties provided by synchronous components can be enjoyed by the application as a whole, even though only some (possibly the majority) of HSVN's components are asynchronous. As discussed in Section 2, important classes of applications can benefit from hybrid architectures.

Although virtualization drops considerably the cost with synchronous components, the HSVN embedding process presented in [Hasan et al. 2013] does not take into account the amount of synchrony demanded by HSVNs. Rather, a fixed number of synchronous components in the SN need to be booked in advance. Since the synchronous components building cost is remarkably higher than the asynchronous ones, a key challenge is to efficiently handle HSVN's synchronous requirements.

In this paper, we consider the synchrony demands on the HSVNs as an input for the embedding process. With this information, the embedding process determines a sufficient number of synchronous components, and their location on the SN, so that the VNs requirements are satisfied. The mapping model aims at minimizing the synchronous resources made available by the SN. With such a model, it is possible to define exactly which physical resources have to be synchronous, and then configure them to preserve timely guarantees (*e.g.* by scheduling real-time tasks for nodes, and applying QoS policies for links). Differently from results presented in [Hasan et al. 2013], the embedding process proposed in this work avoids reservation of unused synchronous resources at the SN. Moreover, this evaluation shows levels of synchrony that will be demanded to a SN.

The rest of this paper is organised as follows. Section 2 motivates the use of hybrid synchronous models and virtual networks, Section 3 presents our proposed mathematical model, that aims at configuring the SN in terms of synchrony, and maps the HSVN on it. Section 4 details about the performance evaluation and gives important insights about the topology and features of physical resources chosen to be synchronous on the SN. Finally, Section 5 describes related work, and Section 6 concludes this paper.

## 2. Hybrid Synchronous Models and Virtual Networks

The development of distributed systems is strongly dependent on the assumptions about the environment. By understanding how system components and the surrounded environment behaves, system developers are capable to design algorithms that are correct under certain assumptions.

Generally, stronger assumptions about system's components and the environment allow the development of simpler solutions. The downside is that strong assumptions do not represent with sufficient accuracy a wide set of real environments. Conversely,

weaker assumptions require complex solutions and, in some cases, some problems are even impossible to solve under weaker assumptions. An advantage, though, is that weaker assumptions usually describe realistic environments and solutions proved to be correct under weaker assumptions are also correct under stronger assumptions.

In practical terms, components of a given infrastructure can have different properties in various aspects such as synchrony, security, or performance. For instance, while some infrastructure channels would be secure (*i.e.*, all information passing through these channels are encrypted), others would just send a stream of unmodified data to the destination. Therefore, it is important that a system architect chooses the most suitable components of the underlying infrastructure to fulfil system's requirements, and then develops applications in accordance with infrastructure's guarantees. If a distributed application requires confidentiality of the data being transferred, then secure channels would be preferred. Otherwise, by using insecure channels, the confidentiality of data would have to explicitly be provided by the application before sending the data.

In this paper the synchrony aspect is addressed, *i.e.*, processes evolve synchronously through rounds of computation delimited by a global clock, or such timely guarantees are relaxed or even non-existent. In an asynchronous system, no assumption about process execution speed and/or message delivery delays is made. Conversely, in a synchronous system, relative processing speed and the message delays are bounded [Schneider 1993]. Therefore, different assumptions about process execution speeds and message delivery delays would require specific design decisions.

Assuming that the underlying infrastructures behaves asynchronously showed to be realistic to a wide range of applications. Furthermore, it is the weakest model in terms of synchrony. That means an algorithm that works in an asynchronous model also works in other models with stronger synchrony assumptions. The opposite is not valid, *i.e.*, an algorithm that works in a synchronous model is prone to incorrect behaviour if timing constraints are violated. Although it is very attractive, in the asynchronous model it is impossible to distinguish a crashed process from an arbitrarily slow process [Chandra and Toueg 1996]. As a consequence, many key problems of fault-tolerant computing are not solvable under the asynchronous assumption. For example, the FLP impossibility result proved that consensus cannot be solved deterministically in asynchronous systems where, at least, one process may crash [Fischer et al. 1985].

By asserting that a system is synchronous, system developers can rely on the timely behaviour of the components. This, in turn, enables one to employ simpler algorithms than those required to solve the same problem in an asynchronous system [Schneider 1993]. For instance, processes can perfectly distinguish faulty from slow processes. Then, the FLP impossibility result is not applicable to the synchronous model. However, building synchronous systems requires infrastructures composed exclusively by timely components, which could be very expensive or even infeasible.

Hybrid models assume intermediate levels of synchrony. Cristian and Fetzer proposed the timed-asynchronous model [Cristian and Fetzer 1999], where the system alternates between synchronous and asynchronous behaviour. In that model the degree of synchrony varies over time. In [Veríssimo 2006], Veríssimo presented the wormhole model, that also exploits the space dimension to provide hybrid synchrony. This means

that timely guarantees of system components may be different (*i.e.*, some components are more predictable than others). For instance, one part of a system would behave synchronously, while other part would be fully asynchronous. Due to the hybrid behaviour, both models are stronger than asynchronous model and weaker than synchronous one.

Hybrid models become a good alternative to the development of distributed systems with timeliness requirements. By enforcing small parts of the system to behave synchronously while other parts are asynchronous, stronger properties provided by synchronous parts can be enjoyed by the system as a whole. For this reason, hybrid systems overcome limitations of the homogeneous systems (fully synchronous/asynchronous).

In [Hasan et al. 2013], we introduced virtualized infrastructure based on the hybrid synchronous model. The Hybrid Synchrony Virtual Networks (HSVN) combines the advantages of hybrid synchronous model with the advantages of virtualization. Sharing of physical resources among VNs becomes an attractive solution to reduce costs with infrastructure, especially the synchronous resources which are more expensive than the asynchronous ones. Therefore, even a small set of synchronous components could be enough to support synchrony requirements from different VNs.

In [Hasan et al. 2013], we illustrated a perfect failure detector $\mathcal{P}$ being implemented in a HSVN. An interesting aspect of implementing $\mathcal{P}$ in a hybrid synchronous system is that, application workload is totally independent of the failure detector modules. Thus, application processes can communicate through asynchronous channels and still enjoy stronger properties provided by the failure detector service. Failure detectors have attracted interest in the development of reliable distributed systems, since consensus and related problems (*e.g.* atomic broadcast [Chandra and Toueg 1996]) can also be solved with it. Further, the failure detection approach can be adapted to solve other relevant problems, such as predicate detection [Gartner and Kloppenburg 2000] and election [Matsui et al. 2000].

Notice that other classes of applications would also benefit from HSVN. In [Mostéfaoui et al. 2006] Mostefaoui *et al.* present an eventual leader election protocol. The proposed protocol benefits from the best of both worlds (synchronous and asynchronous), such that it converges as soon as some synchrony assumption is satisfied. In fact, general round-based protocols for synchronous systems would be adapted to execute in hybrid synchronous models. The design of such hybrid protocols would require a suitable programming model. In this sense, Gorender *et al.* [Gorender et al. 2007] proposed an adaptive programming model for distributed computing, which provides upper-layer applications with process state information according to the current system synchrony. The underlying system model is hybrid, composed by a synchronous part and an asynchronous part.

Even though synchronous components can be shared by multiple HSVNs, the synchronous components are considerably more expensive than the asynchronous ones. Thus, an efficient mechanism for HSVNs mapping that prevents allocation of synchronous nodes unnecessarily is needed.

## 3. A New Embedding Model for HSVN

In this section we propose an allocation model for HSVN in the shape of a mixed integer program (MIP) [Sierksma 2002]. Differently from our previous work [Hasan et al. 2013],

this model decides the physical resources that need to be synchronous, in an economic manner, and maps the VNs nodes and links based on it. The model constraints (eq. 2-9) are the same in [Hasan et al. 2013] because they impose soundness conditions to the HSVN model that cannot be dropped, while the objective function and variables are different. For instance, certain parameters in the previous paper became open variables in the current work.

**Variables definition**- The substrate network is represented by an undirected graph $G(N, L)$, composed of a set of physical nodes $N$ connected through a set of physical links $L$. Each virtual network $VN^k$ belonging to the set of virtual networks $VN$ will be presented by an undirected graph $G^k(N^k, L^k)$, where $N^k$ is given by $N^k = N^k_s \cup N^k_a$, where $N^k_s$ and $N^k_a$ contain the synchronous and asynchronous VN nodes respectively. Similarly, $L^k = L^k_s \cup L^k_a$. A binary function $sync(i^k)$ expresses the VN nodes synchrony demand: $sync(i^k) = 1$ if $i^k \in N^k_s$, otherwise $sync(i^k) = 0$ (*i.e.*, $i^k \in N^k_a$). Similarly, $sync(i^k, j^k)$ expresses the VN links synchrony demand. Besides synchrony, two other attributes are considered for the SN and VNs elements: nodes $CPU$, and links bandwidth $(BW)$. The syntax for those attributes on the SN and VN respectively are: $cpu(i)$, $bw(i, j)$, $cpu(i^k)$, and $bw(i^k, j^k)$. Finally, we define the output variables for our mathematical model, they are four binary functions: 1) $sync(i)$ indicates the synchrony status of the SN node, 2) $sync(i, j)$ indicates the synchrony of the SN link, 3) $\sigma(i^k, i)$ expresses whether node $i \in N$ maps node $i^k \in N^k$, and 4) $\rho(i^k, j^k, i, j)$ expresses whether the physical link $(i, j) \in L$ is part of the path that maps the virtual link $(i^k, j^k) \in L^k$.

**The mathematical model**- The objective function (O.F.) aims at minimizing the synchronous resources (nodes and links) on the SN, beside minimizing the BW consumption.

**Objective: minimize**

$$\sum_{\forall VN^k \in VN} \sum_{\forall (i^k, j^k) \in L^k} \sum_{\forall (i,j) \in L} (\rho(i^k, j^k, i, j) \cdot bw(i^k, j^k)) + \sum_{\forall i \in N} sync(i) + \sum_{\forall (i,j) \in L} sync(i, j) \tag{1}$$

**Subject to**

- *Capacity constraints:*

for every $(i, j) \in L$

$$\sum_{\forall VN^k \in VN} \sum_{\forall (i^k, j^k) \in L^k} \rho(i^k, j^k, i, j) \cdot bw(i^k, j^k) \leq bw(i, j) \tag{2}$$

for every $i \in N$

$$\sum_{\forall VN^k \in VN} \sum_{\forall i^k \in N^k} \sigma(i^k, i) \cdot cpu(i^k) \leq cpu(i) \tag{3}$$

- *Nodes mapping constraints:*

for every $VN^k \in VN$, $i^k \in N^k$

$$\sum_{\forall i \in N} \sigma(i^k, i) = 1 \tag{4}$$

for every $VN^k \in VN$, $i \in N$

$$\sum_{\forall i^k \in N^k} \sigma(i^k, i) \leq 1 \tag{5}$$

- *Links mapping constraint:*
for every $VN^k \in VN$, $(i^k, j^k) \in L^k$, $i \in N$

$$\sum_{\forall j \in N} \rho(i^k, j^k, i, j) - \sum_{\forall j \in N} \rho(i^k, j^k, j, i) = \sigma(i^k, i) - \sigma(j^k, i) \tag{6}$$

- *Nodes synchrony constraints:*
for every $VN^k \in VN$, $i^k \in N^k$, $i \in N$

$$sync(i^k) \cdot \sigma(i^k, i) \leq sync(i) \tag{7}$$

- *Links synchrony constraints:*
for every $VN^k \in VN$, $(i^k, j^k) \in L^k$, $(i, j) \in L$

$$sync(i^k, j^k) \cdot \rho(i^k, j^k, i, j) \leq sync(i, j) \tag{8}$$

for every $VN^k \in VN$, $(i^k, j^k) \in L^k$, $(i, j) \in L$

$$sync(i^k, j^k) \cdot \rho(i^k, j^k, i, j) \leq sync(i) \cdot sync(j) \tag{9}$$

The capacity constraint (2) assures that the total bandwidth of the virtual links, mapped on paths that include a certain physical link, does not exceed the bandwidth capacity of this physical link. Similarly, constraint (3) represents the equivalent restriction regarding nodes $CPU$. The node mapping constraint (4) assures that each virtual node is mapped on a physical one. Constraint (5) assures that virtual nodes belonging to the same $VN$ are not mapped on the same physical node. This is to achieve load balancing, besides improving the reliability, since the unavailability of a SN node will impact, at most, one node on a given VN. For any virtual link $(a, b)$, the links mapping constraint (6), adopted by [Bays et al. 2012] and [Infuhr and Raidl 2011], assures the creation of a valid physical path. Because the right side of the equation will be 1 and -1 for $a$ and $b$ respectively, so, $a$ will have an outgoing arc and $b$ an ingoing one. For all other nodes on the SN, the right side of the equation will be zero, thus the concatenation of arcs will form a valid path. The nodes synchrony constraint (7) assures that synchronous virtual nodes are mapped only on synchronous SN nodes, whereas asynchronous virtual nodes are allowed to be mapped on synchronous or asynchronous SN nodes. This is acceptable because, the synchronous SN nodes supply what the asynchronous ones do, but the reverse is not valid. Similarly, the links synchrony constraint is presented in Equation 8. Finally, Equation 9 guarantees that the intermediate physical nodes on the synchronous physical path should be also synchronous. This is because these nodes play role in the routing process, thus impacting the source-destination delay.

After solving the mathematical model, each node and link on the SN is defined to be synchronous or asynchronous, each virtual node is mapped to one physical node, and each virtual link is mapped to one physical path at maximum, where a physical path can be a unique physical link or a concatenation of physical links.

## 4. Performance Evaluation

This section evaluates the performance of our model for allocation of HSVN. The aspects considered during the analysis of our model are: i) economy in the configuration of the SN synchronous resources; ii) privilege of physical resources chosen to be synchronous, and iii) the topology of the subnetwork composed of synchronous resources on the SN.

Like in related works (e.g., [Yu et al. 2008, Bays et al. 2012]), physical and virtual networks were randomly generated. For this we used BRITE [Medina ] tool with Waxman [Waxman 1988] model. We implemented the mathematical model with ZIMPL language [Koch 2004] and used CPLEX [IBM ] to solve the MIP, running on a computer with 6 cores Intel Xeon processor, 2x2.66 GHz, 32GB of RAM memory, and MAC operating system. Most of the experiments took a considerable time to reach the optimal solution, the reason is that the MIP we propose has four output variables, which leads to considerable set of variables based on the problem size under analysis. Besides, two of those variables (the mapping variables) are based on the value of the other two variables (the synchrony variables) found by the solver, this makes the optimization process long and consumes exponentially the machine memory. For the VNs with small size, the solver reached optimal solution, whereas for bigger size (e.g. group C), the solver took too long (in the order of 24h), or the process was stopped due to fully memory allocation. We decided to stop the solver after finding a solution, even if the solution is not optimal. This might match realistic scenarios, in which the client might prefer a semi-optimal solution in an acceptable computational time, rather than an optimal solution in too long computational time. Thus, during the discussion of results the reader should consider that an optimal solution would perform even better in terms of synchronous resource sharing.

In all the following experiments, the SN size was fixed in 25 nodes. Initially all CPUs of SN nodes are free, and links BW is uniformly distributed between 1-3 Gbps. We start reporting twelve experiments divided into three groups, A, B and C, with VNs total size of 10, 20, and 30 nodes respectively. We refer to these scenarios together as set $X$. The VNs were generated with 3, 4, or 5 nodes each, and CPU demands 10%, 15%, or 25% of the SN nodes CPU capacity, and BW demands uniformly distributed between 100 Mbps and 1 Gbps. In scenarios 1, 2, 3 and 4 of each group, the VNs synchrony demand varies between 0%, 30%, 60%, and 100%. The parameters for each experiment in the set $X$ are described in Table 1. These parameters match the ones in our previous work [Hasan et al. 2013], since one of the goals in this paper is to compare our current to the previous mapping model. We will summarize our previous work before the comparison.

**Table 1. Experiments parameters in set $X$**

| Expe. | A1 | A2 | A3 | A4 | B1 | B2 | B3 | B4 | C1 | C2 | C3 | C4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VN size | 10 nodes | | | | 20 nodes | | | | 30 nodes | | | |
| VNs sync. | 0%. | 30% | 60% | 100% | 0% | 30% | 60% | 100% | 0% | 30% | 60% | 100% |
| SN size | 25 nodes | | | | | | | | | | | |
| SN BW | uniformly distributed: 1 Gbps-3 Gbps | | | | | | | | | | | |
| SN CPU | nodes fully free initially | | | | | | | | | | | |
| each VN size | 3,4,5 nodes | | | | | | | | | | | |
| VNs BW | uniformly distributed: 100Mbps-1Gbps | | | | | | | | | | | |
| VNs CPU | 10,15,25 % of SN nodes CPU | | | | | | | | | | | |

In brief, in our previous work [Hasan et al. 2013], we argued that for mapping virtual networks with hybrid synchrony demands, it is a waste of resources to use a fully synchronous SN, like the SN used in [Zhang et al. 2010, Infuhr and Raidl 2011]. Rather,

using a hybrid synchronous SN, together with an economic mapping process, reduces the cost. The mapping model proposed was aware of sparing the physical synchronous resources, and using them only when needed. In [Hasan et al. 2013], we used the same SN of 25 nodes, with the same attributes illustrated in Table 1. Respectively, 33% and 34% of the physical nodes and links were fixed to be synchronous on the SN, then we started mapping VNs with increasing size on the configured SN (scenarios in set $X$). We showed that this approach let the SN, described above, adequate to handle fully synchronous VNs, with total size up to 30 nodes (*scenario C4*).

In the current paper, we have a new approach for mapping HSVNs, with the goal of minimizing the mapping cost even more. We do not reserve, in advance, the synchronous resources on the SN. Rather, the SN synchrony is defined as an output of the MIP. Thus, the physical resources chosen to be synchronous are subject to the VNs demands, and they change if the VNs demands change (*i.e.,* physical synchronous resources are different in each scenario in set $X$). The model proposed minimizes the number of physical synchronous resources to the limit enough for a given set of VNs. Figure 1 compares SN synchronous resources allocation in our previous and current work, both for scenarios in set $X$. We notice a clear reduction in the amount of SN synchronous resources needed to map the virtual components. For example, in scenario A4, 16% of synchronous nodes and 6% of synchronous links were enough to map the demand. In comparison with our previous work, this means economizing 52% and 83% of SN synchronous nodes and links, respectively. Obviously, higher gains are observed in scenarios A1, B1, and C1, where no synchronous resources are required. Since in the previous approach some SN components must be synchronous independently of VN demands, the improvement on synchronous resources reservation is of 100%. The reduction rate in the SN synchronous resources for the complete set of scenarios can be found in Table 2.

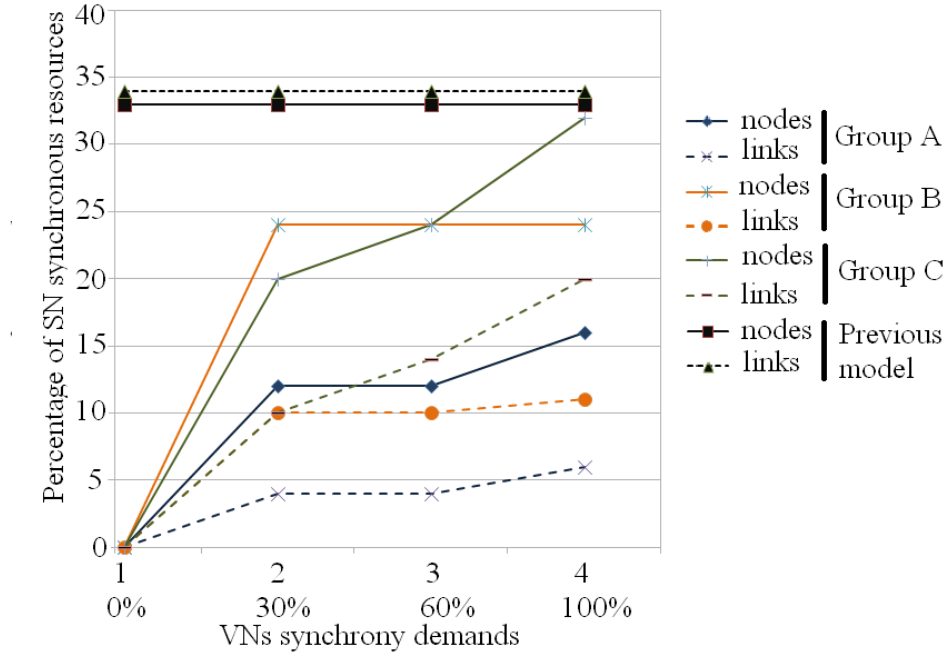



**Figure 1. Percentage of sync. resources in SN for scenarios in the set $X$**

**Table 2. Economy in SN sync resources between the current model and our previous one [Hasan et al. 2013] - performed for scenarios set X**

| SN sync resources | economy in SN sync nodes (%) : economy in SN sync links (%) | | | |
|---|---|---|---|---|
| Scenario | 1 | 2 | 3 | 4 |
| Group A | 100 : 100 | 64 : 88 | 64 : 88 | 52 : 83 |
| Group B | 100 : 100 | 27 : 71 | 27 : 71 | 27 : 68 |
| Group C | 100 : 100 | 39 : 71 | 27 : 59 | 3 : 41 |

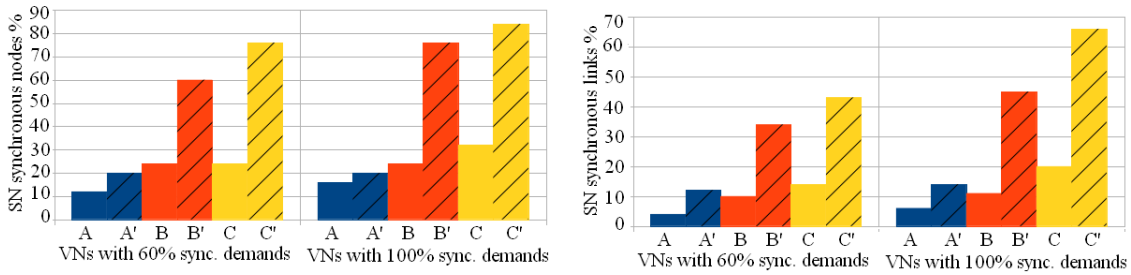
In Figure 1, we note some observations: *i)* for each scenario, synchrony demands on links and nodes grow compatibly (see solid and hashed lines for nodes and links in the graph). *ii)* in scenario C4, the synchronous nodes defined with our new model reached 32%, very near to the fixed portion in the previous work (33% synchronous nodes), which attests the correctness of the previous work, where the fixed synchronous resources could map the VNs in scenario C4. *iii)* in some scenarios, even when VNs synchrony demands increases, the number of SN synchronous resources does not necessarily increases (*e.g.* scenarios B2, B3, and B4). This can be explained by resources sharing provided by VNs mapping. Several virtual components can be mapped on the same physical resources, as long as this does not violate the constraints of CPU and BW capacities for nodes and links, constraints 2 and 3, respectively.

One important feature of the mapping model proposed in this paper is that it lets observe the physical resources chosen to be synchronous and the topology of the subnetworks composed of physical synchronous resources. Next, we discuss these aspects. To perform this kind of evaluation accordingly, we increased the problem size under analysis. We increase: *a)* the size of each virtual network, *b)* the CPU demands, and *c)* the total size of VNs (adding Group D'). The new set of scenarios are named $X'$ (with prim), and their parameters are shown in Table 3. Figures 2(a) and 2(b) depict a comparison in the SN synchronous resources needed in the counterparts scenarios in the $X$ and $X'$ sets, at similar VNs synchrony demands, *i.e.* 60% and 100%. The figures illustrate that, the increment in the problem size has pushed the model to define larger set of synchronous physical nodes and links.

**Table 3. Experiments parameters in set $X'$**

| **Expe.** | **A'1** | **A'2** | **A'3** | **A'4** | **A'5** | **A'6** | **B'1** | **B'2** | **B'3** | **B'4** | **B'5** | **B'6** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **VN size** | 10 nodes | | | | | | 20 nodes | | | | | |
| **Expe.** | **C'1** | **C'2** | **C'3** | **C'4** | **C'5** | **C'6** | **D'1** | **D'2** | **D'3** | **D'4** | **D'5** | **D'6** |
| **VN size** | 30 nodes | | | | | | 40 nodes | | | | | |
| **VNs sync.** | 0%. | 20% | 40% | 60% | 80% | 100% | 0%. | 20% | 40% | 60% | 80% | 100% |
| **SN size** | 25 nodes | | | | | | | | | | | |
| **SN BW** | uniformly distributed: 1 Gbps-3 Gbps | | | | | | | | | | | |
| **SN CPU** | nodes fully free initially | | | | | | | | | | | |
| **each VN size** | fixed to 5 nodes | | | | | | | | | | | |
| **VNs BW** | uniformly distributed: 100Mbps-1Gbps | | | | | | | | | | | |
| **VNs CPU** | 10,20,30,40,50% of SN nodes CPU | | | | | | | | | | | |

The SN described in Table 3 has nodes that vary in their connectivity degree between [2-8]. See Figure 3, its vertical axis indicates the connectivity degree, and its horizontal axis starts with the SN (as base of comparison, as we will see), and continues with each scenario performed in set $X'$. On the leftmost side of Figure 3, we can see the number of nodes available on the SN at each connectivity degree in the range [2-8]. For example, on the figure we can read that the SN has 1 node with connectivity 8, 1 node with connectivity 7, 3 nodes with connectivity 6, and so on. The rest of the figure illus-




(a) SN sync nodes %

(b) SN sync links %

**Figure 2. Percentage of SN sync. resources: Comparing groups $X$ and $X'$**

trates the total number of physical synchronous nodes for each scenario in set $X'$, at each connectivity degree [2-8]. For example, in scenario B6, we note that the model has allocated 19 synchronous nodes, in a way that 1 was with connectivity 8, 1 with connectivity 7, 3 with connectivity 6, and so on. In general, we notice that the model tends toward choosing the physical nodes with high connectivity degree to be synchronous. For example, in scenario C5, the model chose all the physical nodes with connectivity [6-8] on SN to be synchronous (compare the numbers at scenario C5 with the base numbers at SN). A possible interpretation for this behaviour is that, nodes with high connectivity degree allow multi-use of the same node, since on one hand, it is connected to a high number of neighbour nodes which fulfils topology constraints, and on the other hand, nodes with high connectivity have high bandwidth sum (*i.e.* the sum of BW capacity of all the physical links connected to it) which fulfils BW constraint. Once the model aims at minimizing the number of the synchronous resources, then such nodes choice is chased.

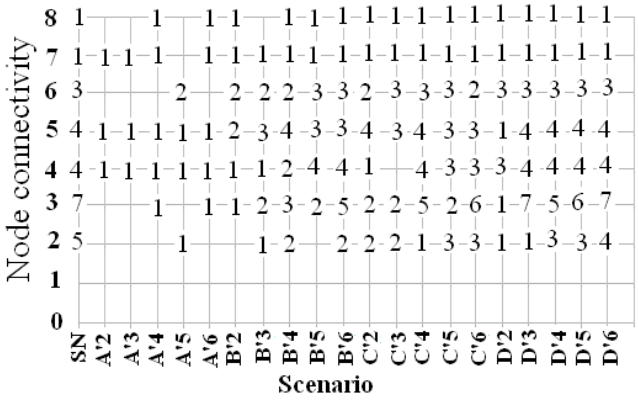



**Figure 3. Frequency of synchronous nodes vs. nodes connectivity**

Next, we evaluate the topology of synchronous resources in each scenario. We noticed that the topology of the synchronous subnetwork starts by a ring topology for small problem size, and tends towards becoming mesh topology with the increment in the problem size. This observation confirms the previous one regarding the synchronous nodes chosen, being the ones with high connectivity degree. Figure 4 depicts the topology of the synchronous resources in scenario A'3, A'4, A'6, and B'4 as an example. Synchronous nodes are plotted as red circles, and synchronous links as hashed lines. We notice that the synchronous resources gather on one subnetwork, no synchronous islands are observed. Such a gathering allows multi-use of same synchronous nodes and links to answer given

VNs, which fits with the model goal in minimizing the number of synchronous resources.

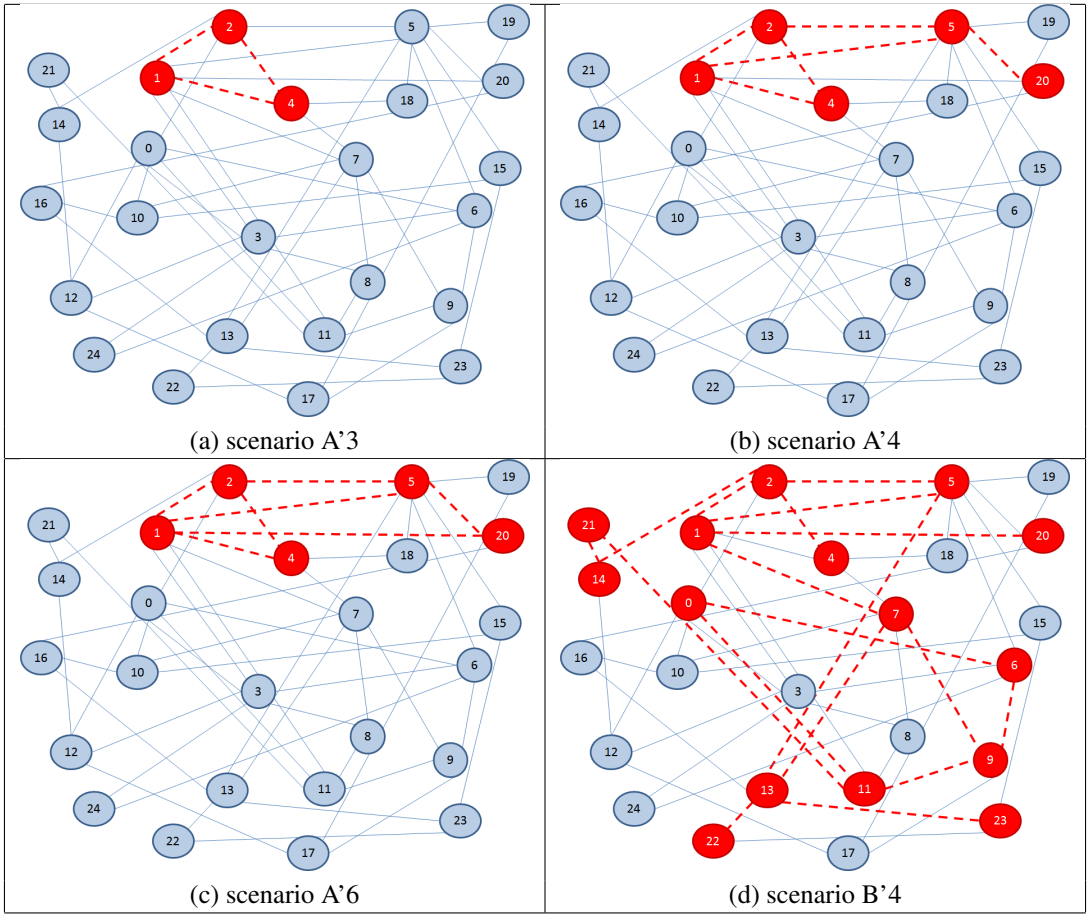



**Figure 4. Topology divergence of synchronous resources**

## 5. Related Work

Revising the literature, we found several works treating the VNs mapping problem through two main approaches: optimization models and heuristics. Chawdhury *et al.* propose a MIP model, where the objective function is a weighted sum of node and link mapping, with the goal of increasing the acceptance ratio and decrease cost. Bay *et al.* [Bays et al. 2012] propose a security-aware mapping model where three levels of security are discussed. In [Yu et al. 2011], authors propose an algorithm that combines VN mapping with substrate link backup to improve VNs resilience. Unlike the previous works, Hsu *et al.* [Hsu et al. 2012] map virtual links through path splitting technique, in addition, path migration is used to maximize the number of coexisting VNs. Botero *et al.* [Botero et al. 2013] were the first to propose a heuristic algorithm that considers the CPU of the physical paths intermediate nodes. For wider collection of VNs mapping, see [Belbekkouche et al. 2012] for survey.

In the tropic of VNs mapping, we find that our work is nearer to those who were concerned with delay constraints. For example, Zhang et al. [Zhang et al. 2010] propose a heuristic algorithm for mapping virtual multicast service-oriented networks subject to delay and delay variation. They consider SNs composed of links with maximum delay.

Their work benefits real-time and interactive applications, where packets are supposed to be received at the destination within specific time bounds, and the delay difference of packets reception at multiple destinations should be minimal.

Another work [Infuhr and Raidl 2011] addressed the VNs mapping problem with delay constraints besides routing and location constraints. The SN considered is composed of links with maximum delay, and nodes that have maximum routing capacity and location constraints. Four different categories were used to represent cases in which VNs have different sets of requirements regarding BW, delay, and nodes CPU: *(1) web slice* for low BW requirements, short delays, and no specific CPU requirements, *(2) stream slice* for medium to high BW requirements, no delay bounds, and 3 processing units per routed bandwidth, *(3) P2P slice* for medium BW and CPU requirements and no delay bounds, and *(4) VoIP slice* for medium BW and delay requirements, and high CPU requirements.

In our previous work [Hasan et al. 2013], we proposed the abstraction of Hybrid Synchrony Virtual Networks (HSVN), *i.e.,* virtual networks that have a subset of nodes and links that obey time bounds for processing and communication. We argued that important classes of distributed systems may benefit from hybrid synchrony, and presented an example of perfect failure detector implemented on top of a HSVN. The embedding of several virtual networks in a substrate network allows resource sharing, which is important since synchronous resources are expensive. Economizing the cost, which we aimed at, was achieved through: 1) the usage of a hybrid synchronous SN instead of a fully synchronous one, and 2) sparing the use of synchronous resources, in other words, mapping asynchronous virtual demands on top of synchronous physical resources is considered the last resource invested only before rejecting the demand. Comparing our previous work with others, we achieved less mapping cost for VNs type that has different synchrony demands. For example, some VNs slices adopted in [Infuhr and Raidl 2011] had no delay requirements, yet the SN considered had no distinction in kind of resources, which results in an unneeded cost.

In the current paper, we achieved a better cost economy for the HSVN mapping, by proposing a model that is able to define the physical synchronous resources needed to map given HSVNs. This eliminates the existence of any unused synchronous resources on the SN. Moreover, we issued a study on the SN that maps the HSVNs, in the sense of resources type chosen to be synchronous, and the topology divergence of the physical subnetwork composed of synchronous resources, when the HSVN size increases. To the best of the authors knowledge, this type of study has not yet been presented in the literature.

## 6. Conclusion

This paper revisits our former work [Hasan et al. 2013] addressing Hybrid Synchrony Virtual Networks (HSVN), (*i.e.,* virtual networks with subsets of nodes and links that require time bounds for processing and communication). Since the cost of synchronous resources is higher than asynchronous ones, in [Hasan et al. 2013] we proposed a mapping model that aims at carefully using the synchronous resources of a SN that supports hybrid synchrony. In this paper we proposed an enhanced version of the HSVN mapping model that allows configuring the substrate network synchrony in an economic manner.

By considering the HSVN synchrony requirements, the new mapping approach al-

lows to set certain nodes and links on a given SN to be synchronous, aware of minimizing the amount of synchronous physical resources to map the demands. A performance evaluation of our approach indicates a substantial economy of synchronous resources compared to previous results [Hasan et al. 2013]. In addition, by exploring a set of scenarios with different synchrony requirements, the analysis conducted in this work gives some important insights about the privilege of physical resources chosen to be synchronous and their topology on the SN. We observed that the model tends towards choosing the nodes with high connectivity degree to be synchronous. A possible interpretation for this is that, such nodes choice allows their multi-use when mapping, since each node is connected to high number of neighbouring nodes and is provided with high bandwidth sum. Such a choice minimizes the number of synchronous resources on the SN, which is the aim of the mapping model. Moreover, we noticed that the synchronous resources configured on the SN tend towards gathering in a mesh topology. Which agrees with the first observation about the nodes type chosen.

Our future work goes in the direction of online mapping for the HSVNs, when the SN resources, or/and the VNs demands are time variant. The mapping approach proposed in this paper, allows only static resource allocation, and thus, does not answer the online mapping. For this reason, we are developing a heuristic algorithm, which is supposed to allow resource allocation for the HSVNs in a dynamic manner.

## References

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. of CNSM*.

Belbekkouche, A., Hasan, M. M., and Karmouch, A. (2012). Resource discovery and allocation in network virtualization. *IEEE Communication Surveys and Tutorials*, 14(4).

Botero, J. F., Hesselbach, X., Fischer, A., and De Meer, H. (2013). Optimal mapping of virtual networks with hidden hops. *Telecommunication System*, 52(3).

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

Chowdhury, M., Rahman, M. R., and Boutaba, R. (2012). Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. on Networking*, 20(1).

Cristian, F. and Fetzer, C. (1999). The timed asynchronous distributed system model. *IEEE Trans. on Parallel and Distributed Systems*, 10(6).

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2).

Gartner, F. C. and Kloppenburg, S. (2000). Consistent detection of global predicates under a weak fault assumption. In *The 19th IEEE Symposium on Reliable Distributed Systems*. IEEE.

Gorender, S., de Araujo Macedo, R. J., and Raynal, M. (2007). An adaptive programming model for fault-tolerant distributed computing. *Dependable and Secure Computing, IEEE Transactions on*, 4(1):18–31.

Hasan, R., Mendizabal, O. M., and Dotti, F. L. (2013). Hybrid synchrony virtual networks: Definition and embedding. In *The Thirteenth International Conference on Networks (ICN)*, pages 104–110.

Hsu, W. H., Shieh, Y. P., Wang, C. H., and Yeh, S. C. (2012). Virtual network mapping through path splitting and migration. In *Proc. of The 26th International Conference on Advanced Information Networking and Applications Workshops*.

IBM. Cplex. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer.

Infuhr, J. and Raidl, G. R. (2011). Introducing the virtual network mapping problem with delay, routing and location constraints. In *Proc. of 5th International Networking Optimization Conference (INOC)*.

Koch, T. (2004). *Rapid mathematical programming*. PhD thesis, Tichnische Universität Berlin.

Matsui, H., Inoue, M., Masuzawa, T., and Fujiwara, H. (2000). Fault-tolerant and self-stabilizing protocols using an unreliable failure detector. *IEICE Trans. on Information and Systems*, 83(10).

Medina, A.; Lakhina, A. M. I. B. J. Brite: Boston university representative internet topology generator. http://www.cs.bu.edu/brite.

Mostéfaoui, A., Mourgaya, E., Raynal, M., and Travers, C. (2006). A time-free assumption to implement eventual leadership. *Parallel Processing Letters*, 16(02):189–207.

Schneider, F. B. (1993). Distributed systems (2nd ed.). chapter What good are models and what models are good? ACM Press/Addison-Wesley Publishing Co.

Sierksma, G. (2002). *Linear and Integer Programming: Theory and Practice, Second Edition*. Marcel Dekker publisher.

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, M., Yi, Y., Rexford, J., and Chiang, M. (2008). Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM-SIGCOMM*, 38(2):17–29.

Yu, Y., zhi, C. S., Xin, L., and Yan, W. (2011). Rmap: An algorithm of virtual networks resilience mapping. In *Proc. of the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*.

Zhang, M., Wu, C., Jiang, M., and Yang, Q. (2010). Mapping multicast service-oriented virtual networks with delay and delay variation constraints. In *IEEE GLOBECOM*. IEEE Communication Society.

Zhu, Y. and Ammar, M. (2006). Algorithms for assigning substrate network resources to virtual network components. In *IEEE INFOCOM*, Barcelona, Spain.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 17*

## *Redes Ópticas I*

# Alocação Preventiva de Regeneradores em Redes Ópticas Translúcidas

**Alexandre Fontinele[1], Iallen Santos[1], Gilvan Durães[2], José Maranhão[3], André Soares[1]**

[1]Programa de Pós-Graduação em Ciência da Computação
Universidade Federal do Piauí (UFPI) Teresina – PI – Brasil

[2]Laboratório de Computação Aplicada – Instituto Federal Baiano (IFBaiano)
Catu – BA – Brasil

[3]Departamento de Comunicações (DECOM) – Universidade de Campinas (Unicamp)
Campinas – SP – Brasil.

andre.soares@ufpi.edu.br

***Abstract.*** *In this paper we propose the APR algorithm to assignment regenerator (RA) in translucent optical networks. The aim of our propose to reduce the blocking probability cause by the degradation made by the establishment of a new lightpath in the others already established in the network. The APR is compared to other strategy that searches assign the minimal number of regenerator per lightpath (AMR). For the considered scenarios in this paper, the APR achieved a superior performance than AMR blocking probability terms.*

***Resumo.*** *Neste trabalho o algoritmo APR é proposto para alocação de regenerador (AR) em redes ópticas translúcidas. O objetivo desta proposta é reduzir a probabilidade de bloqueio causada pela degradação gerada com estabelecimento de um novo circuito óptico nos outros circuitos já estabelecidos na rede. O algoritmo APR é comparado com outra estratégia que busca atribuir o número mínimo de regeneradores por circuito (AMR). Para os cenários considerados neste artigo, o algoritmo APR alcançou um desempenho superior ao AMR em termos de probabilidade de bloqueio.*

## 1. Introdução

As redes ópticas *Wavelength Division Multiplexing* (WDM) podem ser classificadas em três tipos: redes ópticas opacas, redes ópticas transparentes e redes ópticas translúcidas [Yang and Ramamurthy 2005].

As redes ópticas opacas são caracterizadas por realizar conversão óptico-eletrônico-óptico (OEO) em cada nó intermediário entre o nó de origem e destino. Nesse tipo de rede, o sinal óptico possui uma alta qualidade uma vez que em cada nó intermediário a conversão OEO permite uma regeneração completa do sinal. As redes ópticas transparentes são caracterizadas por não realizar conversões OEO o que impede a regeneração do sinal óptico ao longo dos nós intermediários. Isso pode tornar a comunicação inviável por baixa qualidade do sinal óptico [Yang and Ramamurthy 2005].

As redes ópticas translúcidas são apontadas como uma solução em que um conjunto de nós da rede possuem capacidade de regenerar através de conversões OEO. Assim, o sinal óptico pode ser regenerado em alguns pontos estratégicos o que viabiliza

uma melhor qualidade do sinal óptico. Vale ressaltar que nesse tipo de rede, a quantidade de conversores OEO é consideravelmente inferior comparada às redes ópticas opacas [Yang and Ramamurthy 2005], [Shen and Tucker 2007].

Durante o estabelecimento de um circuito em uma rede óptica translúcida devem ser definidos uma rota e um comprimento de onda, este é o problema conhecido como *Routing and Wavelength Assignment* (RWA). Além disso, devem ser definidos os regeneradores que serão utilizados ao longo do circuito, este problema é conhecido como *Regenerator Assignment* (RA) [Yang and Ramamurthy 2005].

Um circuito óptico é bloqueado quando a qualidade do sinal é inferior ao limiar definido para a rede. Um novo circuito também deve ser bloqueado se degradar o sinal dos circuitos já estabelecidos de forma a inviabilizá-los [Maranhão et al. 2010], [Pachnicke et al. 2009], [Manousakis et al. 2009], [Deng et al. 2004].

Este artigo apresenta um novo algoritmo de alocação de regeneradores chamado Alocação Preventiva de Regeneradores (APR). O algoritmo proposto busca manter a qualidade do sinal óptico (QoT – *Quality of Transmition*) em um nível acima do limiar considerando uma margem de segurança que é definida em uma fase de planejamento da rede. Neste trabalho o nível do QoT é medido pela relação sinal-ruído óptica (OSNR – *Optical Signal to Noise Ratio*) [Pereira et al. 2009]. É também realizado um estudo de avaliação de desempenho comparando o APR com a proposta que tenta alocar o mínimo de regeneradores, utilizada em [Yang and Ramamurthy 2005], [Chaves et al. 2012], [Ye et al. 2003], [Kim and Seo 2001]. Para os cenários analisados, o algoritmo APR obteve um ganho de no mínimo 45% em termos de probabilidade de bloqueio, quando comparado à estratégia de minimização do uso de regeneradores.

As demais seções deste artigo estão organizadas da seguinte forma. A Seção 2 apresenta as redes ópticas translúcidas e o problema de alocação de regeneradores. A Seção 3 discute os trabalhos relacionados e apresenta as contribuições deste artigo. O modelo de camada física utilizado neste artigo é descrito na Seção 4. O algoritmo proposto é apresentado na Seção 5. A Seção 6 apresenta um estudo de avaliação de desempenho considerando as métricas de probabilidade de bloqueio e utilização de regeneradores. Por fim, as conclusões do trabalho são destacadas na Seção 7.

## 2. Redes Ópticas Translúcidas

O sinal óptico sofre degradações ao se propagar através de enlaces de fibra óptica, matrizes de comutação, amplificadores ópticos e outros elementos. Devido a essas degradações, o sinal óptico pode chegar no destino sem a qualidade mínima exigida. Em uma rede óptica translúcida o sinal pode ser regenerado em determinados nós da rota para reamplificá-lo, remodelá-lo e retemporizá-lo (3R) . As redes ópticas translúcidas usam um conjunto de regeneradores posicionados estrategicamente com o propósito de regenerar o sinal [Shen and Tucker 2007]. Para realizar a regeneração, o sinal é convertido para o domínio elétrico. Após a regeneração o sinal volta a ser convertido para o domínio óptico.

A rede óptica translúcida tem como objetivo unir a agilidade das redes ópticas transparentes à qualidade do sinal óptico provido com a conversão OEO das redes ópticas opacas [Shen and Tucker 2007]. Diferentes arquiteturas de nós translúcidos foram propostas na literatura. Esses diferentes tipos de nós podem ser combinados gerando diferentes classes de redes translúcidas. Os trabalhos [Shen and Tucker 2007],

[Iyer and Singh 2013] e [Yang and Ramamurthy 2005] apresentam de forma detalhada algumas das principais arquiteturas de nós translúcidos e classes de redes ópticas translúcidas.

Neste artigo será considerada a classe de redes ópticas translúcidas com posicionamento esparso de regeneradores. O posicionamento esparso pode alcançar um melhor custo efetivo em termos de regeneradores [Chaves et al. 2012]. Em uma classe de redes ópticas translúcidas com posicionamento esparso, somente um subconjunto de nós na rede terá capacidade de regeneração. Depois de decidir quais nós da rede serão translúcidos, é feito o posicionamento dos regeneradores entre os nós translúcidos. Assim, o número de regeneradores em cada nó translúcido pode variar para cada rede. Isto é, um nó translúcido pode ter mais regeneradores do que um outro nó da mesma rede. Um nó translúcido com $x$ regeneradores tem a capacidade de regenerar simultaneamente $x$ circuitos que passam por este nó.

Para o estabelecimento de um circuito óptico entre um nó origem e um nó destino, é necessário definir uma rota e alocar um comprimento de onda em cada enlace dessa rota. O problema de roteamento e alocação de comprimento de onda é conhecido como *Routing and Wavelength Assignment* (RWA). Quando os algoritmos RWA utilizam informações de camada física para tomar decisões sobre a alocação dos recursos da rede eles são chamados de *Impairment Aware - Routing and Wavelength Assignment* (IA-RWA). Além do problema IA-RWA, outros dois problemas típicos em redes ópticas translúcidas são o posicionamento de regeneradores (RP – *Regenerator Placement*) e a alocação de regeneradores (RA – *Regenerator Assignment*).

O Problema RP é um problema de otimização que deve ser resolvido durante o estágio de planejamento da rede. O objetivo do Problema RP é posicionar um número limitado de regeneradores de forma a minimizar os impactos de camada física. Isso deve ser feito considerando também os custos introduzidos com os regeneradores. Como o problema RP é NP-Completo, utiliza-se soluções heurísticas que consideram a topologia, o tráfego e a qualidade do sinal [Tordera et al. 2009], [Yang and Ramamurthy 2005], [Chaves et al. 2012].

Alguns trabalhos na literatura utilizam uma modelagem de rede ópticas translúcida na qual a alocação de regeneradores é obrigatória [Shen and Tucker 2007]. Nessa modelagem, qualquer circuito que atravessa um nó com capacidade de regeneração deve ser regenerado. Isto é, um regenerador será alocado obrigatoriamente em cada nó intermediário com capacidade de regeneração. Dessa forma, o problema de alocação de regenerador é relaxado. Por outro lado, os algoritmos de roteamento devem selecionar rotas viáveis em termos de qualidade do sinal.

Um outro tipo de modelagem aplicada a redes ópticas translúcidas introduz o problema RA. Nessa modelagem, quando o sinal óptico passa por um nó com capacidade de regeneração, o sinal pode ou não ser regenerado. O Problema RA deve ser resolvido durante a fase de estabelecimento dos circuitos ópticos. A decisão de regenerar ou não um sinal óptico é tomada pelo algoritmo RA de acordo com a qualidade do sinal óptico e a disponibilidade de regeneradores no nó em questão [Shen and Tucker 2007]. Um algoritmo RA tem como objetivo alocar regeneradores minimizando os bloqueios por baixa qualidade de sinal óptico.

O fluxograma apresentado na Figura 1 descreve as etapas seguidas para o estabelecimento de circuitos ópticos em uma rede óptica translúcida.

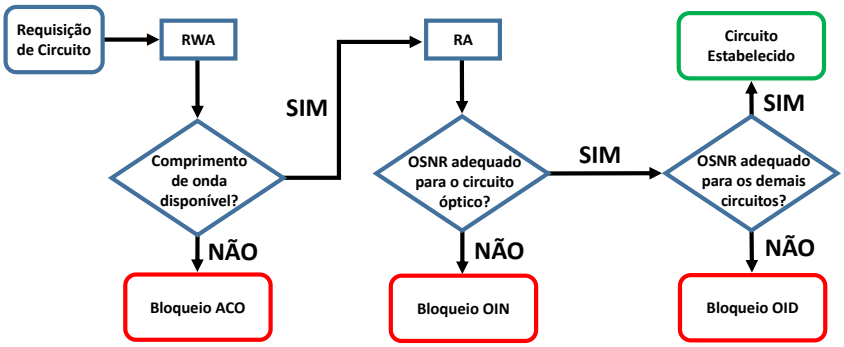



**Figura 1. Processo de alocação de recursos para estabelecimento de um novo circuito óptico em uma rede translúcida.**

Após a requisição de um novo circuito qualquer, são realizados os seguintes passos. Inicialmente o algoritmo RWA seleciona uma rota e posteriormente tenta alocar um comprimento de onda disponível. Se não existir comprimento de onda disponível o circuito é bloqueado por **A**usência de **C**omprimento de **O**nda (**ACO**). Caso contrário, escolhe-se o comprimento de onda e o algoritmo RA verifica a necessidade de regeneração. Se for o caso, o algoritmo RA seleciona regeneradores disponíveis nos nós da rota definida. Em seguida é analisado o OSNR do novo circuito, o circuito que está em processo de estabelecimento. Se o OSNR ficar abaixo do limiar mínimo de QoT, o novo circuito será bloqueado por **O**SNR **I**nadequado para o **N**ovo circuito óptico (**OIN**). Se não houver bloqueio OIN, é verificado se o estabelecimento do novo circuito impacta significativamente nos demais circuitos já estabelecidos. Se o impacto do novo circuito resultar em um OSNR inadequado para os circuitos já estabelecidos, o novo circuito será bloqueado por **O**SNR **I**nadequado para os **D**emais circuitos ópticos já ativos na rede (**OID**).

Os passos apresentados na Figura 1 foram adaptados de [Maranhão et al. 2010] para o cenários de rede óptica translúcidas.

## 3. Trabalhos Relacionados

De maneira geral, na literatura, observa-se dois modelos de redes ópticas com limitações de camada física. A primeira modelagem (adodata pelos trabalhos [Yang and Ramamurthy 2005], [Chaves et al. 2012], [Huang et al. 2005], [Cardillo et al. 2005]) não considera o impacto de um novo circuito óptico na QoT dos outros circuitos já estabelecidos. Isto é, um circuito óptico é bloqueado somente se não for possível estabelecê-lo mantendo a QoT acima do limiar determinado.

Na segunda modelagem (adodata pelos trabalhos [Maranhão et al. 2010], [Deng et al. 2004], [Pachnicke et al. 2009], [Manousakis et al. 2009], um novo circuito óptico é estabelecido se, e somente se, todas as seguintes exigências forem atendidas: i) Haver disponibilidade de comprimento de onda na rota escolhida; ii) Obter QoT aceitável para o circuito que está sendo estabelecido e iii) Assumindo que o novo circuito óptico seja atendido, manter aceitável a QoT para todos os demais circuitos já ativos na rede. Por ser mais realística, este trabalho considera a segunda modelagem de redes ópticas com limitação de camada física.

Seguindo a modelagem mais realística [Maranhão et al. 2010], pode-se decompor a probabilidade de bloqueio geral da rede em três componentes: i) Bloqueio por Ausência de Comprimento de Onda disponível (ACO); ii) OSNR Inadequado para o Novo circuito óptico (OIN) e iii) OSNR Inadequado para os Demais circuitos ópticos já ativos na rede (OID).

Em [Pachnicke et al. 2009], os autores apresentam um algoritmo de roteamento ciente de camada física para redes ópticas translúcidas que procura rotas alternativas para evitar o bloqueio OID, chamado *Constraint-Based Routing* (CBR). O algoritmo utiliza um grafo onde o custo de suas arestas expressa as penalidades de camada física calculadas previamente. A partir deste grafo, o CBR tenta encontrar rotas com menor impacto de camada física.

Em [Deng et al. 2004], os autores propõem quatro algoritmos diferentes para alocação de comprimento de onda que seleciona o comprimento de onda disponível com a menor degradação por *crosstalk*. São eles: *Crosstalk-aware Random Pick* (C-RP), *Crosstalk-aware First Fit* (C-FF), *Crosstalk-aware Most-Used* (C-MU) e *Crosstalk-aware Least-Used* (C-LU). O objetivo desses algoritmos é reduzir bloqueios OIN e OID em redes ópticas transparentes.

Em [Tordera et al. 2009] é tratado o problema de alocação de regeneradores em redes ópticas translúcidas. Os autores destacam que os algoritmos IA-RWA sob tráfego dinâmico apresentam melhor desempenho quando a rede utiliza uma inteligência para alocação de regeneradores. Os autores em [Tordera et al. 2009] consideram um tráfego estático apenas para o posicionamento de regeneradores. Entretanto, a alocação de regeneradores em uma dada rota é feita apenas se necessária, isto é, quando o QoT está abaixo do limiar exigido.

Em [Manousakis et al. 2009], os autores estudam o problema IA-RWA sob tráfego estático para redes ópticas translúcidas que levam em conta os bloqueios OID e propõem um algoritmo para posicionamento e alocação de regeneradores. Por se tratar do problema RWA estático, a alocação de regeneradores nesse trabalho busca minimizar o número de regeneradores usados na rede para o estabelecimento de circuitos estáticos.

Em [Chaves et al. 2012] os autores propõem duas estratégias para o posicionamento esparso de regeneradores em redes ópticas translúcidas, chamadas de *Most Used Regenerator Placement* (MU-RP) and *Maximum Simultaneous Used Regenerator Placement* (MSU-RP). Ambos os algoritmos são heurísticas baseadas no tráfego da rede. Assim como em [Yang and Ramamurthy 2005], [Ye et al. 2003], [Kim and Seo 2001], os autores em [Chaves et al. 2012] apresentam um algoritmo RA que busca alocar o mínimo possível de regeneradores. Vale destacar que nesses algoritmos RA o bloqueio por OID não é verificado na fase de alocação de regeneradores, isto é, durante o estabelecimento do circuito.

A principal contribuição deste artigo é a proposta de um novo algoritmo de alocação de regeneradores para redes ópticas translúcidas sob tráfego dinâmico. Diferentemente das outras propostas de algoritmos RA da literatura, o algoritmo APR visa alocar regeneradores para prevenir contra futuras reduções no seu OSNR. Resumidamente, isso reduz o bloqueio do tipo OID nos futuros circuitos a serem alocados. O APR utiliza um próprio limiar de OSNR, chamado APR-OSNR que é mais rigoroso do que o OSNR

exigido pela rede. O algoritmo APR é detalhado na Seção 5. O algoritmo APR obteve menor probabilidade de bloqueio quando comparado com uma política de alocação de regeneradores utilizada em [Chaves et al. 2012].

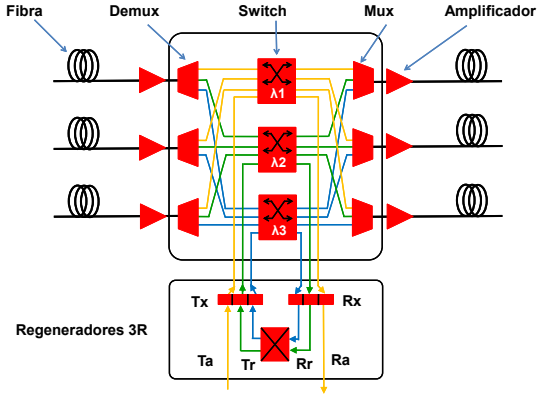
## 4. Modelo da Camada Física

Este artigo considera o impacto da camada física em função do *Polarization Mode Dispersion* (PMD), *Amplified Spontaneous Emission* (ASE), *crosstalks*, perdas de inserção dos dispositivos ópticos e a atenuação da fibra óptica. A modelagem de camada física utilizada está de acordo com a degradação de OSNR apresentada em [Pereira et al. 2009]. Os receptores ópticos possuem uma curva de desempenho que relaciona o OSNR diretamente com uma taxa de erro de bit, portanto, o OSNR pode ser usado como critério de Qualidade de Serviço (QoS) de camada física de um circuito óptico.

Em função da limitação do número de páginas as equações que modelam os efeitos de camada física não serão detalhadas, mas sim referenciadas.

Um circuito óptico é originado em um transmissor do nó de origem e finalizado em um receptor no nó destino. Matrizes de comutação são empregadas para encaminhar os circuitos ópticos ao destino correto. Elementos MUX/DEMUX são utilizados para multiplexar e demultiplexar os comprimentos de onda transportados nas fibras ópticas. Por fim, amplificadores *Erbium Doped Fiber Amplifier* (EDFA) foram utilizados para ajudar a compensar as atenuações inseridas pelos demais elementos da rota.

No caso do sinal óptico ser regenerado em um nó translúcido, o sinal óptico é completamente recuperado. Assim o OSNR na saída de um regenerador é igual ao $OSNR_{in}$, que corresponde ao OSNR na saída do laser de transmissão. E o alargamento temporal percentual do pulso óptico será igual a zero. A arquitetura do nó translúcido utilizada neste artigo está de acordo com a descrita em [Yang and Ramamurthy 2005] e é ilustrada na Figura 2a, adaptada de [Yang and Ramamurthy 2005]. Considerando essa arquitetura, os circuitos ópticos que passam por um nó translúcido podem ser ou não regenerados. Quando houver regeneração é usado um dos regeneradores do nó.




a)

| Parâmetro | Valor | Descrição |
|---|---|---|
| $OSNR_{in}$ | 40 dB | OSNR no transmissor. |
| $OSNR_{QoT}$ | 23 dB | Tolerância de OSNR no receptor. |
| $\Delta t_{QoT}$ | 0.10 | Máximo alargamento do pulso óptico. |
| $\lambda_1$ | 1528,77 nm | Primeiro comprimento de onda no *grid*. |
| W | 24 / 36 | Número de canais para a Pacific Bell e para a NSFnet. |
| B | 40 Gbps | Taxa de transmissão de bits. |
| $\epsilon$ | 38 dB | Fator de isolamento do comutador óptico. |
| $\beta$ | 30 dB | Fator de isolamento em relação à porta vizinha. |
| $L_{Sw}$ | 2 dB | Perda de inserção no comutador óptico. |
| $L_{Mx}$ | 2 dB | Perda de inserção no multiplexador. |
| $L_{Dm}$ | 2 dB | Perda de inserção no demultiplexador. |
| $P_{in}$ | 3 dBm | Potência de saída do transmissor. |
| $\alpha$ | 0.2 dB/km | Coeficiente de atenuação da fibra |
| $\Delta f$ | 100 GHz | Espaçamento entre canais |
| $P_{Sat}$ | 20 dBm | Potência de saturação na saída dos amplificadores. |
| NF | 5.5 dB | Figura de ruído dos amplificadores. |
| $D_{PMD}$ | 0.04 ps/(km^(1/2)) | Coeficiente de PMD da fibra. |

b)

**Figura 2. a) Modelo de nó translúcido. b) Parâmetros de camada física utilizados nas simulações.**

No *switch* e MUX/DEMUX além de considerar a atenuação do dispositivo são

considerados os três tipos de crosstalk: *co-wavelength crosstalk*, *self-wavelength* e *neighbor wavelength crosstalks*, descritos detalhadamente em [Deng et al. 2004]. A potência de ruído óptico gerada pelo *co-wavelength crosstalk* está de acordo com [Pereira et al. 2009].

A potência de ruído óptico gerada por *self-wavelength* e *neighbor wavelength crosstalks* é dada pela equação 1.

$$N_{Mux} = \left( \frac{\beta \sum_{j=1}^{M} P_{Demux_j}(\lambda)}{L_{Sw_1} L_{Mx}} \right) \tag{1}$$

$P_{Demux_j}(\lambda)$ é a potência óptica do sinal no comprimento de onda $\lambda$ que vaza em um comprimento de onda vizinho no momento da demultiplexação. Essa potência é transportada até o MUX onde é adicionada como ruído de *crosstalk*. $\beta$ representa o fator de isolamento em relação à porta vizinha do DEMUX. $M$ é a quantidade de comprimentos de onda vizinhos de $\lambda$ no DEMUX e no MUX.

Nos amplificadores é considerado o ruído adicionado pelo ASE. O efeito de saturação de ganho dos amplificadores não é considerado porque o modelo de amplificador utilizado é com controle automático de ganho modelado de acordo com [Oliveira 2007]. A potência do ruído ASE é descrita em [Oliveira 2007].

Na entrada do receptor, pode-se calcular as potências ópticas do sinal ($P_{out}$) e do ruído ($N_{out}$) na saída do enlace. As equações recursivas 2 e 3 são utilizadas para calcular $P_{out}$ e $N_{out}$ para uma rota com *I* enlaces.

$$P_{out_i} = P_{out_{i-1}} \left( \frac{G_{amp_{b,i}} G_{amp_{p,i}}}{L_{Sw_i} L_{Mx_i} L_{f_i} L_{Dm_i}} \right) \tag{2}$$

$$\begin{aligned} N_{out_i} = N_{out_{i-1}} \left( \frac{G_{amp_{b,i}} G_{amp_{p,i}}}{L_{Sw_i} L_{Mx_i} L_{f_i} L_{Dm_i}} \right) + N_{Sw_i} \left( \frac{G_{amp_{b,i}} G_{amp_{p,i}}}{L_{Mx_i} L_{f_i} L_{Dm_i}} \right) + \\ N_{Mx_i} \left( \frac{G_{amp_{b,i}} G_{amp_{p,i}}}{L_{f_i} L_{Dm_i}} \right) + N_{amp_{b,i}} \left( \frac{G_{amp_{p,i}}}{L_{f_i} L_{Dm_i}} \right) + N_{amp_{p,i}} / (L_{Dm_i}) \end{aligned} \tag{3}$$

Com $P_{out_0} = P_{in}$ e $N_{out_0} = N_{in}$. $G_{amp_{b,i}}$ e $G_{amp_{p,i}}$ representam os ganhos do *booster* e do pré-amplificador, respectivamente. $L_{Sw_i}$, $L_{Mx_i}$ e $L_{Dm_i}$ são as perdas de inserção causadas pelos *switches*, multiplexadores e demultiplexadores, respectivamente. $L_{f_i}$ é a atenuação da fibra. $N_{Sw_i}$ representa *co-wavelength crosstalk* no *switch* e $N_{Mx_i}$ representa os *crosstalks* dos tipos *self-wavelength* e *neighbor wavelength* relacionados com o multiplexador. $Sw_i$ é o *switch* do nó de origem do enlace $i$. $N_{amp_{b,i}}$ e $N_{amp_{p,i}}$ são os ruídos ASE gerados nos amplificadores *booster* e pré-amplificador, respectivamente.

Os valores de $P_{out}$ e $N_{out}$ no receptor são determinados por $P_{out} = P_{out_i}/L_{Sw_{i+1}}$ e $N_{out} = N_{out_i}/L_{Sw_{i+1}} + N_{Sw_{i+1}}$, onde $Sw_{i+1}$ é o *switch* do nó de destino da rota. Dividindo $P_{out}$ por $N_{out}$, no nó destino, obtém-se o OSNR de saída ($OSNR_{out}$) que é avaliado para

verificar se o limiar $OSNR_{QoT}$ foi garantido para as requisições na rede. No caso do limiar não ser atingido, o sinal está com QoT inaceitável e a requisição é bloqueada.

O alargamento do pulso óptico, devido ao efeito de PMD, ao final de um circuito óptico é calculado de acordo com a descrição em [Pereira et al. 2009]. Quanto maior for o OSNR e menor for o alargamento do pulso óptico, menos penalizado está o sinal do circuito óptico.

## 5. Algoritmo Proposto

Na estratégia de alocação de regeneradores apresentada em [Yang and Ramamurthy 2005], [Chaves et al. 2012], [Ye et al. 2003], [Kim and Seo 2001], um regenerador é alocado somente se for essencial para manter o circuito com QoT acima do limiar estabelecido. De forma geral, esta política tenta utilizar um número mínimo de regeneradores por circuito. Por isso, neste artigo, essa política será referenciada por **Alocação Mínima de Regeneradores (AMR)**.

Este artigo propõe o algoritmo **Alocação Preventiva de Regeneradores (APR)**. O APR busca alocar regeneradores para alcançar um nível de QoT mais rigoroso do que o limiar aceitável definido pela rede. Esse nível mais exigente em termos de QoT é chamado de APR-OSNR. A Figura 3 apresenta funcionamento do APR durante a fase de operação da rede e também ilustra o APR-OSNR. O $n$ representa a quantidade de nós para uma dada rota.

**Algoritmo 1** APR

```
i = 1
while i ≤ n do
    o = OSNR no i-ésimo Nó
    if o < (OSNR_QoT + σ) then
        Voltar ao último nó com capacidade
        de regeneração e alocar um regenerador.
        Não alocar mais de um regenerador no
        mesmo Nó para essa requisição.
    end if
    i = i + 1
end while
```

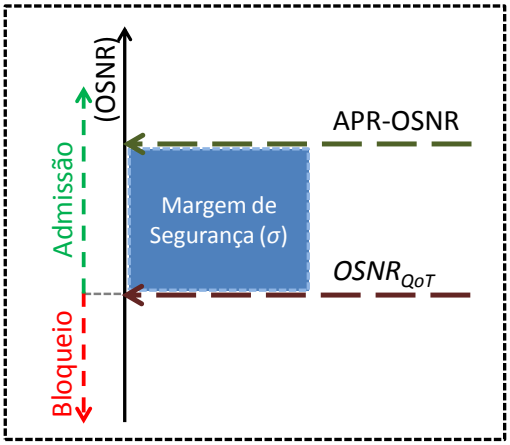



**Figura 3. Algoritmo de operação do APR e nível APR-OSNR.**

O limiar mínimo de OSNR definido para rede como QoT aceitável é dado por $OSNR_{QoT}$. Entretanto, o APR tenta alocar regeneradores tentando obter para cada novo circuito um OSNR maior do que o APR-OSNR. O objetivo dessa estratégia é garantir uma margem de segurança em termos de OSNR, definida por $\sigma$. Desta forma, os circuitos estabelecidos alcançarão um nível de OSNR mais resistentes às interferências causadas pela admissão de futuros circuitos. Essa política tende a reduzir os bloqueios do tipo OID. Vale destacar que, de fato, o circuito somente será bloqueado se o seu OSNR ou o OSNR dos demais circuitos já estabelecidos forem inferiores a $OSNR_{QoT}$.

A escolha da margem de segurança ($\sigma$) deve ser feita em uma fase de planejamento da rede ou durante uma alteração na quantidade de nós, enlaces ou regeneradores.

A Figura 4a apresenta a probabilidade de bloqueio de acordo com diferentes valores de $\sigma$ para a topologia da rede Pacific Bell (Figura 7). A Figura 4b apresenta a composição desses três tipos de probabilidade de bloqueio em função do valor de $sigma$. A carga na rede é fixa em 100 Erlangs.

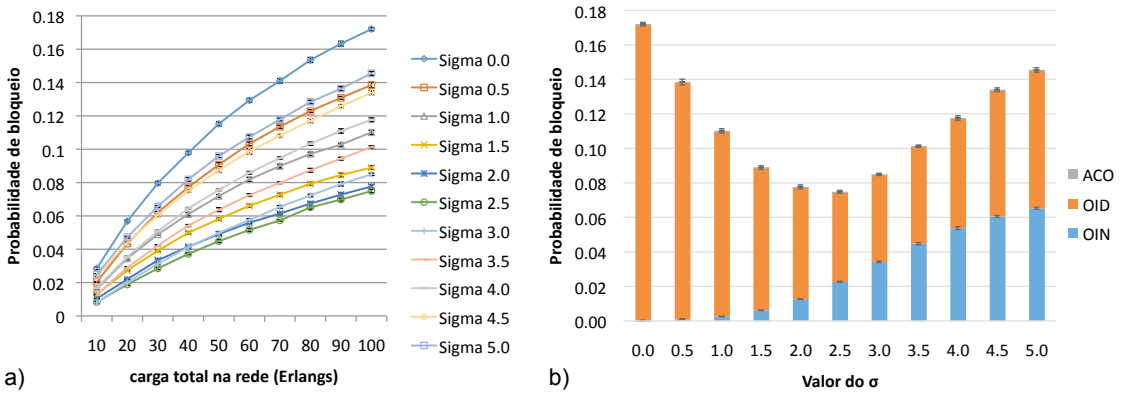



**Figura 4. a) Curvas de probabilidade de bloqueio em função da carga na topologia Pacific Bell utilizando diferentes valores para $\sigma$. b) Composição da probabilidade de bloqueio geral de acordo com o valor de $\sigma$ na topologia Pacific Bell sob uma carga de 100 Erlangs.**

Para este cenário, o valor $2,5$ é o mais apropriado para $\sigma$, uma vez que a curva de probabilidade de bloqueio para $\sigma = 2,5$ apresenta menor probabilidade de bloqueio em todos os pontos de carga avaliados.

Na Figura 4b, analisando o primeiro momento ($0,0 \leq \sigma < 3,0$), a utilização de um maior valor de $\sigma$ provoca uma alocação de mais regeneradores por circuito estabelecido. Desta forma, esses circuitos se tornam mais resistentes, o que reduz a probabilidade de bloqueio do tipo OID. Em contrapartida, os bloqueios do tipo OIN tendem a aumentar uma vez que a quantidade de regeneradores disponíveis na rede se torna menor. No segundo momento ($\sigma \geq 3,0$), além do aumento de bloqueios do tipo OIN ocorre também um aumento no bloqueio do tipo OID. Isso porque a margem de segurança excessiva provoca um rápido consumo de regeneradores pelos primeiros circuitos. Com isso, os circuitos estabelecidos posteriormente acabam sendo viabilizados com uma baixa margem de segurança em termos de OSNR. No cenário abordado os bloqueios do tipo ACO foram próximos a zero.

A escolha do $\sigma$ deve ser feita de forma a minimizar a probabilidade de bloqueio geral, que é composta pelas probabilidades de bloqueio dos tipos OIN, OID e ACO.

A Figura 5 apresenta a probabilidade de bloqueio geral obtida para a topologia Pacific Bell em função do valor de $\sigma$ com a carga na rede fixa em 30, 60 e 100 Erlangs. Os valores de 30, 60 e 100 Erlangs foram escolhidos empiricamente. Para a topologia da rede Pacific Bell, independente da carga na rede, observa-se que o comportamento da curva de probabilidade de bloqueio em função do $\sigma$ utilizado é semelhante. Além disso, observa-se também um melhor desempenho para $\sigma = 2,5$. Outros detalhes desse cenário simulado são apresentados na Seção 6.

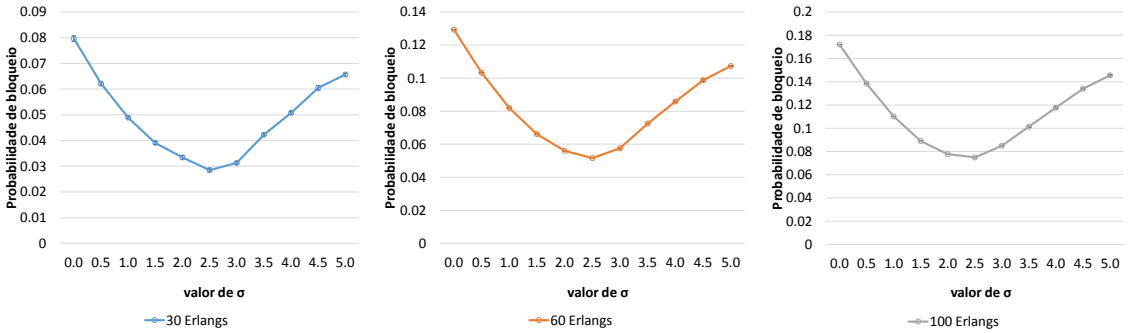



**Figura 5. Probabilidade de bloqueio para a topologia Pacific Bell em função de $\sigma$ considerando a carga total na rede fixa em 30, 60 e 100 Erlangs.**

O algoritmo proposto neste trabalho consiste em duas etapas. A primeira etapa é a escolha do $\sigma$ ideal. Tal escolha deve ser feita durante a fase de planejamento da rede, ou durante uma alteração da configuração da rede. A segunda etapa consiste da aplicação do algoritmo APR considerando o $\sigma$ obtido.

A Figura 6 apresenta o algoritmo utilizado pelo APR durante a fase de planejamento para a escolha do valor ideal do $\sigma$. O valor $p$ é a granularidade utilizada na busca do valor ideal de $\sigma$. $Pb_{(m)}$ representa a probabilidade de bloqueio obtida através de simulação da rede considerando o valor de $\sigma$ igual a $m$.

**Algoritmo 2** APR – planejamento

```
σ = 0.0
p = 0.5
buscando = TRUE
while buscando do
   if Pb(σ+p) < Pb(σ) then
      σ = σ + p
   else
      buscando = FALSE
   end if
end while
```

**Figura 6. Algoritmo para escolha do valor de $\sigma$.**

Quanto menor o valor de $p$, mais refinada será a busca do $\sigma$ ideal, entretanto, o custo computacional do algoritmo também será maior, uma vez que serão feitas mais simulações da rede.

Na definição do $\sigma$ ideal, o algoritmo APR simula o comportamento da rede considerando diferentes valores de $\sigma$ em ordem crescente. No primeiro momento, a medida que o valor de $\sigma$ é incrementado o valor de $Pb_{(m)}$ é reduzido. Entretanto, para um dado valor de $\sigma$ esse comportamento é invertido, conforme ilustrado na Figura 5. Assim, o APR procura um valor ideal de $\sigma$ até a inversão do comportamento da curva de $Pb_{(m)}$.

## 6. Avaliação de Desempenho

O algoritmo Alocação Preventiva de Regeneradores (APR) foi comparado ao algoritmo Alocação Mínima de Regeneradores (AMR) sob os aspectos de probabilidade

de bloqueio e utilização de regeneradores. O AMR foi implementado de acordo com [Chaves et al. 2012].

Foram realizados experimentos considerando as topologias Pacifc Bell e NSFnet, ilustradas na Figura 7. O valor apresentado para cada enlace das topologias indica a sua distância em Km. Os algoritmos de menor caminho de Dijkstra e First-Fit foram utilizados respectivamente para o roteamento e a alocação de comprimento de onda. A distância foi considerada como custo na definição das rotas de menor caminho.

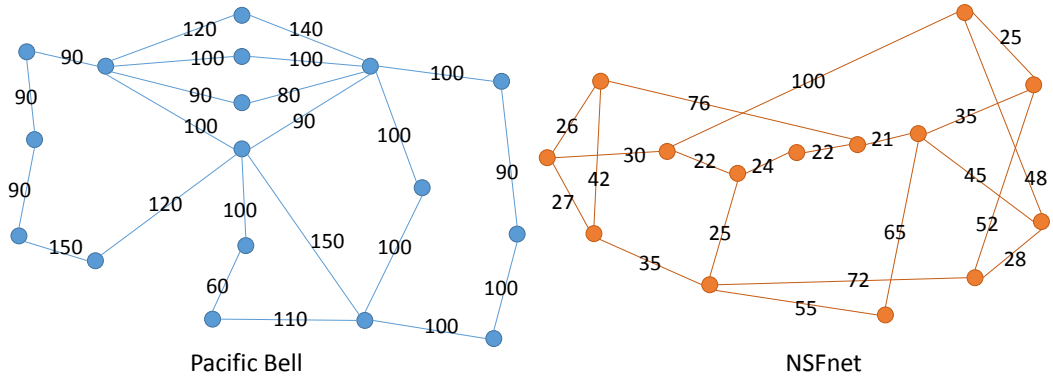



**Figura 7. Topologias utilizadas no estudo de avaliação de desempenho.**

O estudo de avaliação de desempenho apresentado nesta seção foi realizado com o auxílio da ferramenta de simulação TONetS [Soares et al. 2008]. Foi desenvolvido um módulo que implementa a modelagem de camada física apresentada na Seção 4. Além disso, também foi implementado um Controle de Admissão de Chamadas (CAC) Translúcido de acordo com o descrito em [Chaves et al. 2012]. O CAC Translúcido é capaz de identificar se existe ou não a necessidade de regenerar o sinal óptico. O uso de regeneradores pode ser por OSNR abaixo do limiar ou simplesmente para viabilizar uma conversão de comprimento de onda. Os parâmetros usados nas simulações estão listados na Figura 2b. Os ganhos dos amplificadores foram dimensionados para compensar as perdas.

A carga de tráfego é distribuída uniformemente entre todos os pares do nós origem destino. A geração de requisições é um processo poissoniano de taxa de chegada média $\lambda$ e o tempo médio de retenção dos circuitos é distribuído exponencialmente com média $1/\mu$. A intensidade de tráfego na rede em Erlangs é dada por $\rho = \lambda/\mu$. Todos os enlaces da rede são bidirecionais e possuem 24 comprimentos de onda para a Pacific Bell [Chaves et al. 2012] e 36 comprimentos de onda para a NSFnet [Pereira et al. 2009]. Para cada simulação são realizadas 10 replicações com diferentes sementes de geração de variável aleatória. São geradas 100.000 requisições para cada replicação. Em todos os resultados apresentados neste artigo foi considerando um nível de confiança de 95%.

O algoritmo MSU-RP [Chaves et al. 2012] foi utilizado para posicionar 120 regeneradores nas duas topologias para cada valor do $\sigma$. O MSU-RP considera o tráfego e a quantidade de regeneradores disponíveis na decisão de como distribuir os regeneradores entre os nós da rede. Uma carga alvo para a realização dos posicionamentos foi escolhida para cada topologia. Foi utilizada a carga 60 Erlangs para o posicionamento de regeneradores na topologia Pacific Bell para replicar o cenário estudado em [Chaves et al. 2012].

Para a topologia NSFNet a carga 100 Erlangs foi escolhida de forma empírica, com probabilidade de bloqueio abaixo de 10%.

A aplicação do algoritmo APR encontrou o $\sigma = 2,5$ com o melhor desempenho para a topologia Pacific Bell, como foi mostrado na Seção 5. O valor do $\sigma = 3,0$ mostrou o melhor desempenho para a topologia NSFnet. Esses são os valores para o $\sigma$ utilizado pelo APR nos resultados apresentados nesta Seção.

A Figura 8 apresenta a probabilidade de bloqueio e a utilização média de regeneradores obtidas com os algoritmos APR e AMR em função da carga na rede na topologia Pacific Bell. Para a topologia Pacific Bell sob todos os valores de carga analisados, o APR apresentou melhor desempenho do que o AMR. Sob uma carga de 100 erlangs na topologia Pacific Bell, APR e AMR alcançaram respectivamente $0,08$ e $0,18$ de probabilidade de bloqueio. Isso representa um ganho de aproximadamente 51%.

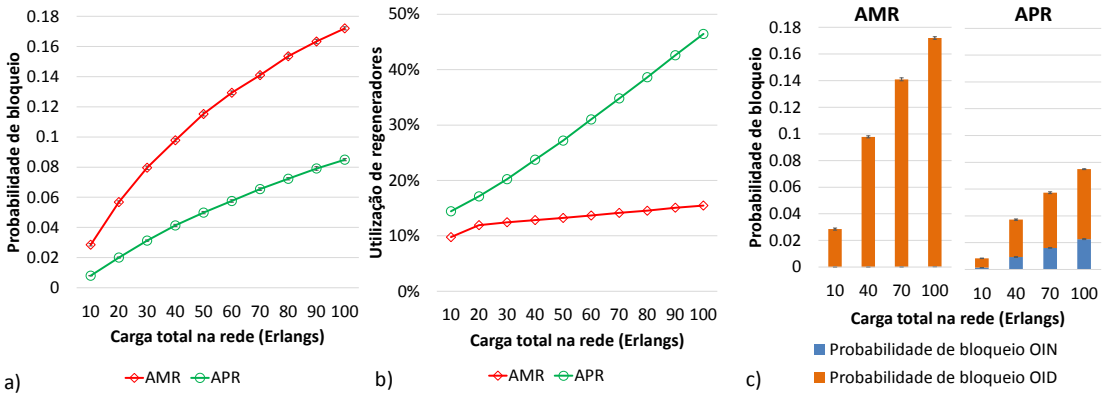



**Figura 8. Resultados na topologia Pacific Bell: a)Probabilidade de bloqueio geral da rede. b) Utilização de regeneradores. c) Composição de bloqueio OIN e OID.**

A minimização do uso de regeneradores obtida com o uso do algoritmo AMR provoca uma maior disponibilidade de regeneradores (Figura 8b). Entretanto, de maneira geral, o OSNR dos circuitos estabelecidos atinge um valor de OSNR ligeiramente superior a $OSNR_{QoT}$. Por sua vez, essa característica do AMR torna o OSNR dos circuitos estabelecidos mais vulneráveis ao estabelecimento de futuros circuitos. Isto é, aumenta a probabilidade de bloqueio OID de circuitos futuros (Figura 8c). Conforme ilustra a Figura 8c, o melhor desempenho obtido pelo algoritmo APR deve-se a maior utilização de regeneradores por circuito. Essa maior utilização de regeneradores reduz os bloqueios OID e aumenta bloqueio OIN, mas de forma geral, reduz a probabilidade de bloqueio da rede. Vale destacar que a ocorrência de bloqueio do tipo ACO foi praticamente zero.

A Figura 9 apresenta os resultados de probabilidade de bloqueio e utilização média de regeneradores para a topologia NSFnet com o uso dos algoritmos APR e AMR. Na topologia NSFnet o algoritmo AMR também apresentou probabilidade de bloqueio superior ao algoritmo APR para todos os valores de carga analisados. O ganho relativo do APR sobre o AMR em termos de probabilidade de bloqueio foi de pelo menos $45\%$ em todos os pontos de carga observados. Quando a carga na rede variou entre 140 e 160 erlangs o ganho do APR foi de 64%. Assim como observado nos estudos para a topologia Pacific Bell, para a topologia NSFnet o melhor desempenho do APR foi devido a maior utilização de regeneradores o que acarretou na redução de bloqueios OID.

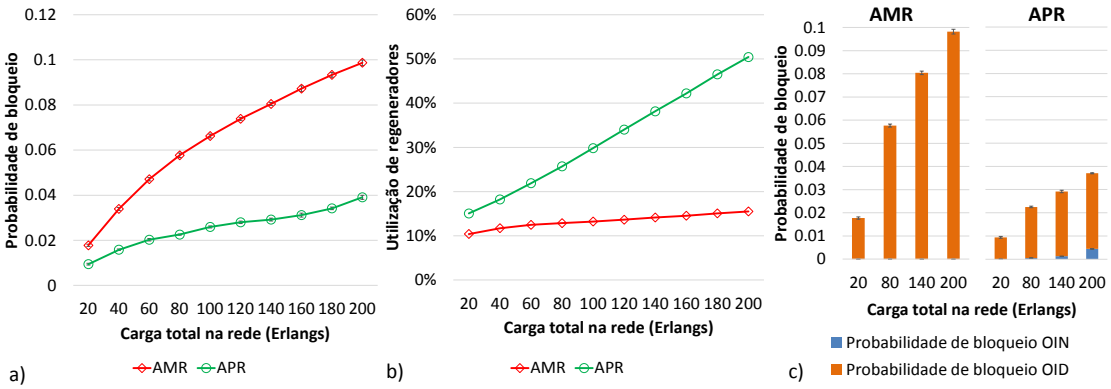



**Figura 9. Resultados na topologia NSFnet: a)Probabilidade de bloqueio geral da rede. b) Utilização de regeneradores. c) Composição de bloqueio OIN e OID.**

O algoritmo APR busca estabelecer os circuitos com uma qualidade do sinal acima do limiar de OSNR definido para a rede. Desta forma, é esperado que a taxa de utilização de regeneradores obtida com o uso do APR seja superior à taxa obtida com o AMR. Esse comportamento foi observado para as duas topologias de rede estudadas.

A utilização média dos regeneradores com o uso do algoritmo AMR apresenta valores inferiores aos obtidos com o algoritmo APR. Para ambas as topologias, o algoritmo AMR obteve uma taxa de utilização média de regeneradores inferior a 20%. Entretanto, para os cenários avaliados, tal economia de regeneradores do AMR em relação ao APR não é vantajosa. Isso porque a economia de regeneradores resultou em uma pior probabilidade de bloqueio.

## 7. Conclusões

Este artigo propôs o algoritmo APR que realiza a alocação de regeneradores em redes ópticas translúcidas. O objetivo do APR é alocar regeneradores de forma a diminuir os bloqueios OID. O algoritmo APR apresentou resultados superiores à estratégia de alocação de regeneradores que busca minimizar o uso de regeneradores, referenciada neste artigo por AMR. Em termos de probabilidade de bloqueio, para os cenários avaliados neste artigo, o APR obteve ganhos de pelo menos 50% e 45% em relação ao AMR quando aplicados as topologias Pacific Bell e NSFnet.

Como trabalho futuro pretende-se avaliar o desempenho de algoritmos IA-RWA com o APR.

## Referências

Cardillo, R., Curri, V., and Mellia, M. (2005). Considering transmission impairments in wavelength routed networks. In *Optical Network Design and Modeling, 2005. Conference on*, pages 421–429.

Chaves, D., Carvalho, R., Pereira, H., Bastos-Filho, C., and Martins-Filho, J. (2012). Novel strategies for sparse regenerator placement in translucent optical networks. *Photonic Network Communications*, 24(3):237–251.

Deng, T., Subramaniam, S., and Xu, J. (2004). Crosstalk-aware wavelength assignment in dynamic wavelength-routed optical networks. In *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on*, pages 140–149.

Huang, Y., Heritage, J., and Mukherjee, B. (2005). Connection provisioning with transmission impairment consideration in optical wdm networks with high-speed channels. *Lightwave Technology, Journal of*, 23(3):982–993.

Iyer, S. and Singh, S. P. (2013). A novel optical control plane for switching a limited range wavelength converter based electro-optical hybrid node in translucent wdm optical networks. In *Communications (NCC), 2013 National Conference on*, pages 1–5.

Kim, S.-W. and Seo, S.-W. (2001). Regenerator placement algorithms for connection establishment in all-optical networks. *Communications, IEE Proceedings-*, 148(1):25–30.

Manousakis, K., Christodoulopoulos, K., Kamitsas, E., Tomkos, I., and Varvarigos, E. (2009). Offline impairment-aware routing and wavelength assignment algorithms in translucent wdm optical networks. *Lightwave Technology, Journal of*, 27(12):1866–1877.

Maranhão, J., Soares, A., and Waldman, H. (2010). Alocação de comprimento de onda em redes Ópticas considerando as degradações de camada física. In *SBRC 2010*, pages 1–14.

Oliveira, J. C. R. (2007). *Amplificadores Ópticos com Controle Automático de Ganho para Aplicação em Redes Ópticas Reconfiguráveis*. PhD thesis, Unicamp.

Pachnicke, S., Luck, N., and Krummrich, P. (2009). Online physical-layer impairment-aware routing with quality of transmission constraints in translucent optical networks. In *Transparent Optical Networks, 2009. ICTON '09. 11th International Conference on*, pages 1–4.

Pereira, H., Chaves, D., Bastos-Filho, C., and Martins-Filho, J. (2009). Osnr model to consider physical layer impairments in transparent optical networks. *Photonic Network Communications*, 18(2):137–149.

Shen, G. and Tucker, R. (2007). Translucent optical networks: the way forward [topics in optical communications]. *Communications Magazine, IEEE*, 45(2):48–54.

Soares, A., Durães, G., Giozza, W., and Cunha, P. (2008). Tonets: Ferramenta para avaliação de desempenho de redes Ópticas transparentes. In *IV Salão de Ferramentas do SBRC*, pages 1–8.

Tordera, E., Martinez, R., Munoz, R., Casellas, R., and Solé-Pareta, J. (2009). Improving ia-rwa algorithms in translucent networks by regenerator allocation. In *Transparent Optical Networks, 2009. ICTON '09. 11th International Conference on*, pages 1–4.

Yang, X. and Ramamurthy, B. (2005). Sparse regeneration in translucent wavelength-routed optical networks: Architecture, network design and wavelength routing. *Photonic Network Communications*, 10(1):39–53.

Ye, Y., Cheng, T., and Lu, C. (2003). Novel algorithm for upgrading of translucent optical networks. *Opt. Express*, 11(23):3022–3033.

# Algoritmo Justo e Eficiente para Provisão de Recursos e Proteção Parcial em Redes Ópticas WDM com Diferenciação de Serviço

**Paulo J. S. Júnior**[1], **André C. Drummond**[2]

[1] Departamento de Engenharia Elétrica/FT
Universidade de Brasília

[2]Departamento de Ciência da Computação/IE
Universidade de Brasília

{paulojunior@cic.unb.br,andred@unb.br}

***Abstract.*** *The WDM optical networks have a high transmission capacity, but still present challenges in the efficient provision of resources. With the increasing deployment of WDM technology in WAN and MAN networks, the demand for resources has become increasingly more heterogeneous, with demands for best-effort traffic, going along with traffic demands of high capacity and* ***QoS*** *requirements, of emerging applications (e.g. video 4K, 8K and e-science). This article proposes a solution that combines several techniques for the efficient provisioning of resources in WDM networks, such as dynamic traffic grooming, multipath routing and preventive dynamic blocking, survivability through partial protection, and a wavelength reservation mechanism. The proposed algorithm is more efficient as a whole, and is also able to meet the demands of each class efficiently, compared with the literature.*

***Resumo.*** *As redes ópticas WDM possuem uma alta capacidade de transmissão, mas ainda apresentam desafios na provisão eficiente de recursos. Com o aumento da implantação da tecnologia WDM em redes WAN e MAN, a demanda por recursos tem se tornado cada vez mais heterogênea, com demandas de tráfego best-effort, ocorrendo juntamente com demandas de tráfego de alta capacidade e requisitos de QoS restritos, como a de aplicações emergentes (e.g. vídeo 4K, 8K e e-ciência). Este artigo propõe uma solução que agrega várias técnicas para a provisão eficiente de recursos em redes WDM, como a agregação dinâmica de tráfego, o roteamento multicaminho, o bloqueio preventivo e a provisão de sobrevivência através da proteção parcial, além de um mecanismo de reserva de comprimentos de onda por classe de serviço. O algoritmo proposto é mais eficiente de forma global, e também capaz de atender as demandas de tráfego de cada classe de forma mais eficiente, e justa em comparação com a literatura.*

## 1. Introdução

As redes ópticas com multiplexação por comprimentos de onda (WDM) possuem uma alta capacidade de transmissão o que abriu caminho para novas aplicações de alta capacidade como as aplicações de processamento de imagens [Deelman et al. 2005], e-Ciência [Taylor et al. 2007], ou de vídeo de fluxo contínuo de super alta definição

[Simeonidou et al. 2008]. Como a tecnologia WDM é amplamente utilizada em redes WAN, o risco de falhas, seja por causa de cortes acidentais de suas fibras, falhas em equipamentos, ou mesmo devido a ataques à sua infraestrutura, é bastante significativo. Devido a esse risco, é também necessário implementar mecanismos de sobrevivência utilizando técnicas de proteção e/ou restauração de caminhos ópticos [Ramamurthy et al. 2003] [Yao 2005].

Nas redes de alta capacidade, alguns serviços e aplicações demandam estabelecimentos de conexões com requisito de banda passante superior ao de um comprimento de onda, tornando necessário rotear o tráfego por múltiplos caminhos ópticos. Para estas aplicações, mecanismos de proteção total são inviáveis, pois a demanda de banda necessária para se prover redundância de caminho impactaria significativamente a alocação de recursos na rede. Como alternativa, a proteção parcial reduz o impacto sobre a rede e, também, permite que aplicações com tolerância à perda de dados possam operar sem que haja prejuízo na qualidade experimentada (QoE) pelos usuários [Kim and Choi 2010] [Padmanabhan et al. 2003]. A proteção parcial garante que o fluxo de dados da conexão afetada sofra uma perda de no máximo um determinado percentual garantido em contrato (SLA). Uma forma eficiente de se implementar a proteção parcial em redes WDM é através do uso da técnica de roteamento multicaminhos [Das et al. 2009], que permite a obtenção de soluções que apresentam melhor desempenho e menor custo, mitigando o impacto das aplicações de alta capacidade sobre a rede [Drummond et al. 2012].

As redes WDM suportam a implantação de múltiplos canais de transmissão paralelos por fibra óptica utilizando diferentes comprimentos de onda. Para o estabelecimento de um circuito, ou caminho óptico, é necessário resolver o problema de roteamento e alocação de comprimentos de onda (RWA)[Banerjee and Mukherjee 1996], que define uma rota e um comprimento de onda que será utilizado pelo caminho óptico. A técnica conhecida como agregação de tráfego (*traffic grooming*) [Cerutti et al. 2004] define o problema de rotear uma demanda de tráfego sobre uma topologia virtual, i.e., sobre a rede de caminhos ópticos estabelecidos na rede.

Este artigo introduz um algoritmo de roteamento adaptativo que trata do problema de agregação e proteção de tráfego em redes ópticas WDM transparentes para aplicações de alta capacidade e de *Best-effort* com proteção parcial. O algoritmo proposto utiliza a técnica de roteamento multicaminhos [Chen et al. 2009], e evidencia a potencialidade da mesma na construção de mecanismos de proteção em um cenário com demandas heterogêneas. O algoritmo proposto se destaca pela aplicação de técnicas que objetivam diminuir o bloqueio das chamadas, provendo proteção parcial de forma mais justa e com uma menor utilização de recursos. A solução proposta utiliza uma base de rotas *off-line*, um mecanismo de bloqueio preventivo, uma função de custo abrangente e adaptativa, com um mecanismo de reserva de comprimentos de onda por classe de serviço.

O artigo está organizado da seguinte forma: na Seção 2 são apresentados trabalhos relacionados; a Seção 3 introduz o algoritmo *Adaptative Routing Partial Protection* - ARPP; na Seção 4 são derivados resultados através de simulações para avaliar a eficiência das soluções apresentadas, e finalmente, na Seção 5 são desenhadas as conclusões.

## 2. Trabalhos relacionados

Existem vários trabalhos na literatura propondo técnicas de sobrevivência para redes ópticas WDM. Em [Ramamurthy et al. 2003, Wang et al. 2002, Xue et al. 2005, Rai et al. 2007], os autores analisam técnicas de proteção e restauração. As propostas são baseadas em dois paradigmas, proteção por caminho e por enlace, e uma comparação entre eficiência e complexidade é feita entre estes paradigmas.

Em [Das et al. 2009], os autores exploram a técnica de roteamento multicaminho, comparam o custo e o uso dos recursos de rede entre a proteção total e parcial. Eles demonstram que a proteção parcial é uma solução de menor custo, tão eficiente e confiável quanto a total. O estudo também propõe um algoritmo que objetiva atingir a máxima proteção parcial possível, o algoritmo *Min-Cut Disjoint Paths*, MDP. No mesmo artigo, os autores também propuseram outro algoritmo chamado SMART-MDP, modificando o algoritmo MDP original de tal forma que ele preserve a banda passante dos enlaces mais utilizados.

Em [Huang et al. 2011], é proposto um algoritmo chamado SPLIT-DDCKDP que utiliza o algoritmo de K menores caminhos para encontrar um conjunto de conjuntos de caminhos de baixo custo.

Os autores [Tian et al. 2012] apresentam um algoritmo de roteamento com base em prioridades. O algoritmo proposto destina-se a maximizar o número de pedidos de comunicação aceitos submetidos a um número limitado de canais de comprimento de onda. A nova abordagem considera a relação entre o tipo de caminho e volume de tráfego para resolver a prioridade de solicitações de conexão, e também incorpora o estado da rede com base na relação (roteamento/comprimento de onda). Os resultados das simulações mostram que a proposta pode alcançar uma taxa menor de bloqueio e uma melhor utilização dos recursos da rede óptica.

Os autores deste artigo criaram três versões de algoritmos adaptativos para proteção parcial, o SmartReal-MDP [Drummond et al. 2012] , o BCPP1 [de Souza Júnior and Drummond 2013a], o BCPP2 [de Souza Júnior and Drummond 2013b]. Os algoritmo propostos foram desenvolvidos para o cenário de tráfego dinâmico e tem por objetivo prover protecão parcial para aplicações de alta capacidade utilizando-se de técnica de roteamento multicaminho e agregação de tráfego.

O SmartReal-MDP [Drummond et al. 2012] é uma adaptação do arquivo Smart-MDP[Das et al. 2009] para redes WDM transparentes. Para uma melhor avaliação em redes ópticas WDM transparentes, o SmartReal-MDP passa a considerar detalhes que o Smart-MDP não considera como caminhos ópticos já estabelecidos (agregação de tráfego) e restrição de continuidade. A característica que torna estes algoritmos inteligentes (*smart*), segundo os autores [Das et al. 2009], é o fato dos pesos dos enlaces da rede serem inicializados com uma unidade e na medida que as requisições entram ou partem da rede, os pesos dos enlaces envolvidos são incrementados ou decrementados, respectivamente, sendo utilizados pela rotina de menores caminhos.

O algoritmo proposto, Best-Cost with Partial Protection, BCPP1 [de Souza Júnior and Drummond 2013a], além de utilizar as técnicas do algoritmo SmartReal-MDP, adota uma estratégia inicial de gerar um número máximo de caminhos

ópticos possíveis e depois aplicar uma funcão objetivo que prioriza aqueles de menor custo. O número máximo de caminhos adota uma estratégia de obter de maneira off-line todas as rotas para todos os pares origem-destino da rede e sem limitação de numero máximo de saltos.

O Algoritmo BCPP2 [de Souza Júnior and Drummond 2013b] é uma evolução do BCPP1. O BCPP2 implementa um mecanismo de bloqueio preventivo, criando duas listas de enlaces de interesse, a *Blacklist* e a *Graylist*, que limitam o conjunto de rotas que serão efetivamente consideradas na busca por caminhos. A *Blacklist* contém os enlaces congestionados e com pouca capacidade livre, ou seja, gargalos, e a *Graylist* possue enlaces também congestionados, mas que não entraram na *Blacklist*. Estes contém poucos comprimentos de onda livres e poucos fluxos agregados, portanto possuem uma maior probabilidade de liberarem recursos.

## 3. Algoritmo Proposto

O algoritmo proposto, *Adaptative Routing with Partial Protection* - ARPP, é inspirado no algoritmo BCPP2 [de Souza Júnior and Drummond 2013b]. O ARPP, assim como o BCPP2, também foi construido para o cenário de tráfego dinâmico e tem por objetivo prover proteção parcial para aplicações de alta capacidade utilizando as técnicas de roteamento multicaminho e agregação de tráfego. Além disso, foi adicionado um tratamento diferenciado de classes de tráfego e uma nova técnica de seleção de caminhos (*off-line*). A diferenciação de tráfego é feita através da definição de duas classes de serviço, (i) melhor esforço (*Best-Effort*) e (ii) alta capacidade (demandas equivalentes a capacidade de um comprimento de onda). Para o tratamento diferenciado, tanto a função custo como a reserva de comprimentos de ondas se adaptam as diferentes classes de tráfego.

A restrição de continuidade é priorizada como forma de diminuir os custo de conversão óptica-elétro-óptica (OEO). A agregação de tráfego também diminui o custo e o esforço computacional de se criar novo caminho óptico, além de diminuir o consumo de mais recursos da rede ao aproveitar os recursos já estabelecidos.

Para o roteamento multicaminho, o ARPP adota a estratégia de obter inicialmente, de maneira *off-line*, todas as rotas para todos os pares origem-destino da rede, porém utiliza um limite máximo de saltos para restringir o tamanho das rotas geradas. Com isso, limita-se o número de soluções e obtêm-se um resultado mais eficiente, pois evita-se a seleção de rotas muito extensas que consumiriam muitos recursos de rede.

O bloqueio preventivo é uma etapa que filtra os caminhos candidatos que serão posteriormente avaliados pela função custo adaptativa. O bloqueio preventivo segue o mesmo princípio utilizado no BCPP2 [de Souza Júnior and Drummond 2013b], com modificações que priorizam as aplicações de alta capacidade. O mecanismo de bloqueio preventivo se baseia em duas listas de enlaces de interesse, a *Blacklist* que trabalha no contexto das aplicações de alta capacidade e a *Graylist* que trabalha no contexto das aplicações *Best-Effort*. A *Blacklist* é composta por todos os enlaces que possuam menos banda do que o percentual de proteção parcial contratada pela aplicação de maior capacidade e, portanto, contém os enlaces que estão bastante congestionados. A *Graylist*, por sua vez, é composta por enlaces que não entraram na *Blacklist*, mas que apresentam certas características que indicam que os recursos alocados nos mesmos podem ser liberados rapidamente e consequentemente disponibilizar mais recursos. Os enlaces que

se enquadram nesta lista são os enlaces com apenas um comprimento de onda livre ou menos comprimentos de ondas livres e que atendem uma pequena quantidade de fluxos, na prática, 90% do número máximo de comprimentos de ondas por enlace ocupados e no máximo 10% de capacidade consumida por enlace.

A função custo utilizada se adapta priorizando métricas diferentes de acordo com a classe considerada. Como as demandas da classe *Best-Effort* consomem muito rápido os comprimentos de ondas dos enlaces, o número de comprimentos de ondas livres passa a ter maior relevância na composição do custo. Já as demandas de alta capacidade consomem mais banda passante, fazendo com que a capacidade disponível seja priorizada.

A técnica de reserva de comprimentos de ondas ou tratamento diferenciado de classes de tráfego, adota estratégias diferentes de alocação de comprimento de onda de acordo com a classe de cada aplicação. Para a classe *Best-Effort*, devido a sua baixa demanda por banda passante, a quantidade de comprimentos de onda reservados na rede é de apenas 20% do total. Já para as aplicações de alta capacidade, o restante dos comprimentos de ondas (80%) fica reservado.

Formalizando o problema, considera-se um grafo $G = (V, E)$, onde $V$ é o conjunto de nós da rede e $E$ é o conjunto de enlaces de fibra que conectam os nós na topologia física. Cada requisição $r$, é representada por uma tupla, $r =< s, d, b, f >$, onde $s$ é o nó origem, $d$ é o nó destino, $b$ é a banda passante requisitada, e $f$ é o fator de proteção desejado, sendo $f = [0, 1]$, pois $f = 1$ representaria um pedido de proteção total. Para o cálculo da quantidade máxima de tráfego que pode ser transportado em cada caminho óptico e, equivalentemente, para a definição do número de caminhos necessários para a provisão da proteção requerida, deve-se considerar:

$$\left\lceil \frac{b}{m} \right\rceil \leq b(1 - f) \tag{1}$$

Onde $b(1-f)$ é a perda aceitável contratada, e $m$ é o numero de caminhos ópticos utilizados. Para se definir o numero mínimo de caminhos disjuntos necessários para o requisito de proteção parcial, utiliza-se a equação $|P| \geq \left\lceil \frac{b}{b(1-f)} \right\rceil$.

Na Figura 1 pode-se ver o pseudocódigo do algoritmo ARPP. O algoritmo recebe como entradas as topologias física, $PT(V_p, E_p)$, virtual da rede, $VT(V_v, E_v)$, e um conjunto de rotas *off-line*, $getAllPaths(PT)$. Este conjunto de rotas *off-line*, $P$, é adaptado a uma técnica de avaliação de caminho de comprimento máximo, $pairPaths(Rotas, s, d, diam)$, na linha 2. Após estes dados de entrada, o algoritmo forma um conjunto $P$ de soluções que recebe estas rotas e remove deste conjunto todas as rotas que contenham pelo menos um dos enlaces listados na *Blacklist* ou na *Graylist*. Desta forma, todos os enlaces considerados congestionados e/ou candidatos a serem liberados mais rapidamente são bloqueados, sendo todos os caminhos que os consideram removidos do conjunto $P$. Com este conjunto, é gerado mais dois subconjuntos, um com as rotas agregáveis, $GLP(VT, s, d)$, na Linha 3, e outro com as rotas viáveis, $VLP(P, P_{glp}, s, d)$, na Linha 4. A função $GLP$ recebe como parâmetros a topologia virtual da rede ($VT$), o par de nós origem-destino, e retorna um conjunto de caminhos ópticos disjuntos disponíveis. Com isso, o algoritmo passa a considerar a possibilidade de realizar agregação de tráfego nos caminhos ópticos existentes. Este conjunto de caminhos ópticos formam a lista de caminhos agregáveis. A função $VLP$ recebe como parâmetro

**ARPP - Adaptative Routing with Partial Protection**

---

**Entrada:** Topologia física $PT(V_p, E_p)$.
**Entrada:** Topologia virtual $VT(V_v, E_v)$.
**Entrada:** $Rotas = getAllPaths(PT)$

1: **Para** cada chamada $R(s, d, b, f)$ **faça**
2: $P = pairPaths(Rotas, s, d, diam) - (Blacklist() + Graylist())$
3: $P_{glp} = GLP(VT, s, d)$
4: $P_{vlp} = VLP(P, P_{glp}, s, d)$
5: $P' = sortCost(P_{vlp}, P_{glp})$
6: **Se** banda disponível em $P' \geq b$ **então**
7: **Se** $|P'| \geq \left\lceil \frac{b}{b(1-f)} \right\rceil$ **então**
8: $i = 0$
9: **Enquanto** banda alocada $< b$ **faça**
10: **Se** $Class(b) == 1$ **então**
11: Aloca W $\leq W_{Class_1}$
12: **Senão**
13: Aloca W $\geq W_{Class_2}$
14: **Fim Se**
15: $P^* \leftarrow P'[i]$
16: Aloca o máximo de banda até $b(1-f)$
17: $i$++;
18: **Fim Enquanto**
19: **Fim Se**
20: **Fim Se**
21: **Se** Existir $P^*$ **então**
22: Aceita a requisição
23: **Senão**
24: Rejeita a requisição
25: **Fim Se**
26: **Fim Para**

**Figura 1. Algoritmo ARPP**

a topologia física da rede ($PT$) e retorna um conjunto de caminhos ópticos viáveis $P_{vlp}$ para a mesma origem e destino da requisição. Para tal, $VLP$ executa um algoritmo que avalia cada caminho em potencial buscando a ocorrência de pelo menos um comprimento de onda livre ao longo de todos os enlaces deste caminho. Em caso positivo o caminho é mantido, caso contrário o caminho é removido do conjunto. Em seguida, os conjuntos $P_{glp}$ e $P_{vlp}$ são passados para a função $sortCost$. Em seguida, na Linha 5, a função custo adaptativa, $sortCost()$, é executada para a ordenação dos caminhos e agrupamento dos caminhos disjuntos de menor custo em um único conjunto $P'$. Neste ponto, o algoritmo tem um conjunto de caminhos candidatos para a solução final de alocação, e na Linha 7, a capacidade desta solução é avaliada. Na sequência, das Linhas 7 a 19, cada caminho é avaliado quanto a sua capacidade de banda disponível. Na Linha 9, é conferido se cada caminho de $P'$ possui uma quantidade de banda passante disponível suficiente para acomodar a chamada, e se o número de caminhos em $P'$ atende o nível de proteção parcial requisitado. A partir desse ponto, entre as Linhas 11 e 14, a técnica de reserva de comprimentos de ondas é executada, onde a faixa utilizada depende do tipo de classe. Neste caso, as variáveis $W_{Class_1}$ e $W_{Class_2}$ representam respectivamente o percentual de 30% e 70% da capacidade de comprimentos de onda dos enlaces da rede. Na Linha 15, é definido o conjunto $P^*$ composto do mínimo de caminhos ópticos provenientes de $P'$ necessários para atender a demanda de banda passante da chamada. Nesse ponto, se exis-

tir uma solução, a chamada será aceita (Linha 22), e seu fluxo de dados será distribuído entre os caminhos ópticos agregáveis e/ou viáveis de $P^*$. Caso contrário, a requisição será rejeitada.

### 3.1. Função Custo Adaptativa

A função custo adaptativa procura representar o consumo de recursos de cada caminho óptico. O custo $F_c$ de um caminho é calculado por uma função custo adaptativa, $sortCost$, onde são utilizadas 5 métricas distintas para a composição do custo total de um caminho. Tal custo pode sofrer um ajuste fino de acordo com o tipo de classe que está sendo atendido. Isto é feito através da manipulação dos pesos $\alpha_x$, o que permite a adaptação da sensibilidade do algoritmo a cenários específicos, e da modificação da precedência das variáveis na composição da fórmula. Os pesos da função senguem uma relação, onde $\alpha_1 > \alpha_2 > \alpha_3 > \alpha_4 > \alpha_5$. Para a utilização dos pesos, as variáveis devem ter suas precedências avaliadas quanto ao que cada representa com relação a utilização dos recursos de uma rede óptica.

A função custo $F_{c_1}$ da classe *best-effort* é calculado segundo a Equação 2.

$$F_{c_1} = \alpha_1.C_H + \alpha_2.C_D + \alpha_3.C_W + \alpha_4.C_F + \alpha_5.C_{BW} \tag{2}$$

A função custo $F_{c_2}$ da classe de alta capacidade é calculado segundo a Equação 3.

$$F_{c_2} = \alpha_1.C_{BW} + \alpha_2.C_H + \alpha_3.C_D + \alpha_4.C_W + \alpha_5.C_F \tag{3}$$

A equações a seguir apresentam as principais variáveis de uma rede óptica:

$$C_H = \frac{numHops}{maxHops} \tag{4}$$

$$C_D = \frac{Delay}{maxDelay} \tag{5}$$

$$C_W = 1 - \frac{availableW}{maxW} \tag{6}$$

$$C_F = 1 - \frac{numFlow}{maxFlows} \tag{7}$$

$$C_{BW} = 1 - \frac{availableBW}{maxBW} \tag{8}$$

A variável $C_H$, custo em saltos, é a razão entre o numero de saltos do caminho e o numero de saltos do caminho relativo ao diâmetro da rede. A variável $C_D$, custo de retardo, é a razão entre o retardo de transmissão do caminho e o retardo do caminho relativo ao diâmetro da rede. A variável $C_W$, custo de comprimento de ondas disponíveis, é o complemento da razão entre o número de comprimentos de ondas disponíveis fim-a-fim no caminho, e a capacidade de comprimentos de onda de um enlace. A variável $C_F$, custo de fluxos, é o complemento da razão entre o numero de fluxos utilizados no caminho e a quantidade máxima de fluxos que podem ser agregados no mesmo. Finalmente, a variável $C_{BW}$, custo de banda passante, é o complemento da razão entre a quantidade de banda passante disponível no caminho e a capacidade do mesmo. $C_H$ e $C_D$ são métricas genéricas pois valem para qualquer caminho, e as demais métricas são mais específicas,

pois são relevantes apenas nos casos de caminhos ópticos já estabelecidos. Os caminhos de menor custo são aqueles que consomem menos recursos da rede. Na prática, a união entre métricas genéricas e específicas, e a correta manipulação de seus pesos permite, por exemplo, que um caminho óptico já estabelecido que esteja atendendo a uma grande quantidade de fluxos e que possua banda passante disponível, possa receber um custo baixo pelo fato de representar um recurso já alocado na rede que provavelmente não será liberado no curto prazo.

## 4. Avaliação Numérica

Para avaliar o desempenho dos algoritmos propostos foram realizadas simulações utilizando o simulador de redes ópticas WDMSim [Drummond 2010]. Foram implementados os algoritmos SmartReal-MDP (evolução do Smart-MDP [Huang et al. 2011]), BCPP1 e BCPP2 para serem comparados com o algoritmo ARPP.

Cada simulação foi realizada 10 vezes utilizando o método de replicações independentes. Para os resultados apresentados foram calculados intervalos de confiança com 95% de confiabilidade. Em cada simulação foram geradas 100.000 requisições de conexão para diferentes níveis de carga na rede. A carga, medida em *Erlang*, é definida como a taxa média de chegada $\times$ a duração da chamada $\times$ a banda passante da chamada normalizada pela capacidade do enlace (10Gbps). As métricas utilizadas foram a taxa média de bloqueio de banda (MBBR) por chamada aceita na rede. Para cada evento de chegada simulado, os algoritmos foram executados levando em consideração a origem, destino, demanda de banda passante e fator de proteção da chamada, além do estado da rede. Caso o algoritmo encontre uma solução, os recursos eram alocados na rede e a chamada aceita, caso contrário, a chamada era bloqueada.

As topologias consideradas nas simulações foram a USANet com 24 nós e 43 enlaces bidirecionais, e a topologia PanEuro com 27 nós e 81 enlaces bidirecional, Figura 2. Os nós da rede são comutadores ópticos, OXCs, configurados com portas de agregação suficientes para que não haja contenção. Cada enlace de fibra carrega 16 comprimentos de onda, cada um com capacidade de 10 Gbps. A arquitetura utilizada é do tipo transparente e não há conversão de comprimentos de onda.

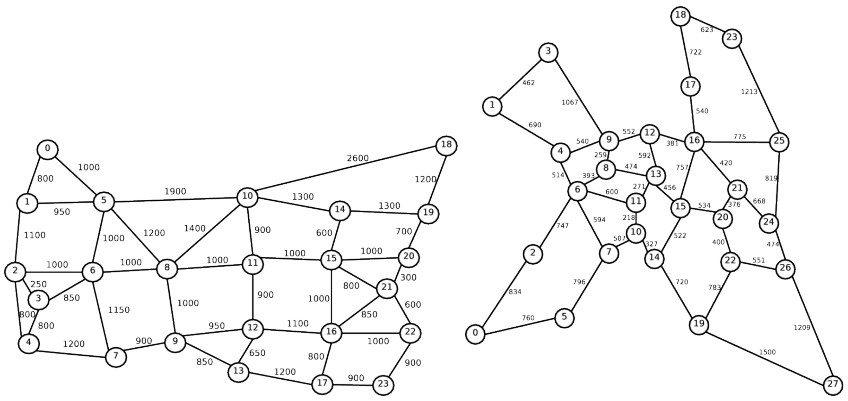


**Figura 2. Topologias USANet e PanEuro**

O processo de chegada das requisições de tráfego segue a distribuição de Poisson, com as granularidades baseadas em aplicações de vídeo de alta definição (2k, 4k e 8k) compactado e não compactado [Simeonidou et al. 2008], *best-effort* (classe 1) e de alta

capacidade (classe 2). As chamadas foram distribuídas uniformemente para todos os pares de comunicação da rede, o número de requisições de granularidade foi inversamente distribuído de acordo com sua demanda por largura de banda apresentados na Tabela 1. De forma que chamadas com menor granularidade ocorram com maior frequência, e vice e versa. O fator de proteção considerado para todas as chamadas foi de 50% ($f = 0,5$), o mesmo utilizado em [Das et al. 2009].

**Tabela 1. Fluxos Best-Effort e de Alta Capacidade**

| **Classe 1** | Banda (Mbps) | Peso |
|---|---|---|
| 2k /Compactado | 192 | 100 |
| 4k /Compactado | 382 | 50 |
| **Classe 2** | Banda (Mbps) | Peso |
| 8k /Compactado | 1200 | 16 |
| 2k /Não Compactado | 3056 | 6 |
| 4k /Não Compactado | 6112 | 3 |
| 8k /Não Compactado | 19200 | 1 |

### 4.1. Avaliação da Função Custo

A Figura 3 mostra um estudo comparativo de desempenho da MBBR resultante considerando separadamente cada variável, para que se possa definir a precedência das variáveis na composição da função custo. Esta avaliação foi feita sobre uma carga de 22 *Erlang*. A rede avaliada foi a USANet, pois apresenta um grau maior de conectividade que a PanEURO. Esta figura demonstra que a função custo $F_c$ justifica sua aplicação em comparação com qualquer variável de forma isolada. O gráfico destaca dois grupos de variáveis, as variáveis $C_H$ e $C_D$ que são métricas genéricas, pois valem para qualquer caminho, seja agregável ou viável, e as demais métricas, $C_W$, $C_F$ e $C_{BW}$, que são mais específicas, pois são relevantes apenas nos casos de caminhos ópticos já estabelecidos ou agregáveis. O primeiro grupo de variáveis tem um resultado muito melhor que o segundo grupo. Como os resultados das variáveis $C_H$ e $C_D$ estão muito próximos, a maior precedência da variável $C_H$ com relação a $C_D$ pode ser justificada devido ao fato de que, um caminho com mais saltos consume mais recursos que outro com menos saltos. A variável $C_D$, por outro lado, não garante que um caminho de maior distância utilize mais recurso que outro. Podem existir casos em que caminhos mais distantes possam ter menos saltos que um outro e consequentemente consumir menos recursos. No caso da função custo, a variável $C_D$ tem menor precedência que a variável $C_H$ e é fundamentalmente utilizada para diferenciar dois caminhos disjuntos com a mesma quantidade de saltos. O segundo grupo de variáveis, tem melhores resultados que o primeiro grupo. A variável $C_w$ é uma métrica que pode representar um enlace bastante congestionado, pois mostra o número de comprimentos de ondas livres. Já as variáveis $C_f$, número de fluxos, e $C_{BW}$, capacidade de banda disponível, têm resultados muitos próximos, mas não garantem que um enlace esteja sem recursos livres, simplesmente por apresentar pouca capacidade de banda livre ou com muitos fluxos sendo transportados. O resultado das variáveis demonstram que isoladas não conseguem escolher uma melhor solução, mas que combinadas, conforme a função custo adaptativa $F_c$, podem apresentar um resultado mais eficiente.

### 4.2. Avaliação do Comprimento Máximo dos Caminhos

O ARPP utiliza uma técnica de roteamento *off-line* que calcula todos os caminhos possíveis entre os pares origem-destino da rede, o que possibilita a aplicação mais efi-

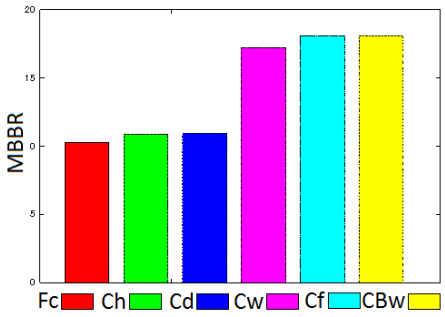



**Figura 3. Estudo Comparativo entre Variáveis para a rede USANet para 22 *Erlang*.**

ciente da função custo sem vícios de roteamento. Todavia, uma análise empírica para definir um limite para o número de caminhos considerados faz-se necessária pois, ao se considerar todos os caminhos sem restrição, além de demandar um tempo de processamento muito alto, pode-se apresentar uma solução com um número de saltos muito grande, o que consumirá muitos recursos da rede. Na Figura 4, foi avaliado o limite no tamanho de caminhos considerados, analisando-se a MBBR e a complexidade de tempo para a topologia USANet. A topologia USANet tem como diâmetro 6 saltos, e a topologia PanEURO 8 saltos. O Diâmetro é definido como o mais longo caminho de todos os caminhos mais curtos calculados numa rede. Na topologia USANet, a partir do tamanho 10, a MBBR começa a piorar e o tempo de processamento a subir. O mesmo ocorre na topologia PanEURO, a partir de 14 saltos, a solução piora. Portanto, para ambas as topologias, foi considerado o tamanho máximo dos caminhos candidatos como sendo de 1,7 vezes o diâmetro da rede.

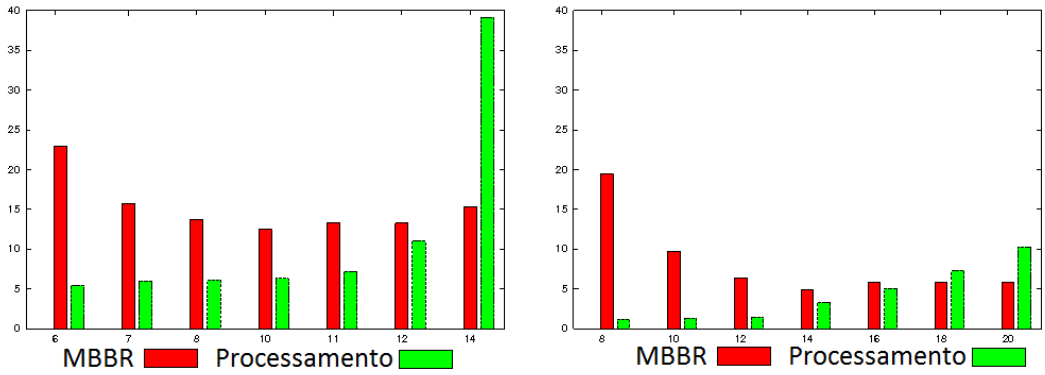



**Figura 4. Comprimento Máximo para a rede USANet (22 *Erlang*) e PanEURO (16 *Erlang*).**

## 4.3. Resultados

Para melhor avaliação, as Tabelas 2 e 6 apresentam as MBBR por carga para os quatro algoritmos: SmartReal-MDP (evolução do Smart-MDP [Huang et al. 2011]), BCPP1 [de Souza Júnior and Drummond 2013a] e BCPP2 [de Souza Júnior and Drummond 2013b] e o ARPP. Na tabela 2, o ARPP é diferenciado em ARPP (sem classes) e ARPP para destacar a importância da função custo adaptativa. O ARPP (sem classes) utiliza a mesma função para as duas classes de tráfego. A vantagem do uso da função adaptativa junto com o mecanismo de reserva de comprimentos de onda por classe de serviço é destacada nas duas tabelas e gráficos.

Na Figura 5, foi avaliada a MBBR, para a topologia USANet. Nesta figura, o ganho do algoritmo ARPP com relação aos outros algoritmos é de uma ordem de grandeza (10x). O SmartReal-MDP serve como base inicial comparativa para os outros algoritmos. O BCPP1 apresenta uma taxa de bloqueio até quase uma ordem de grandeza menor do que a do SmartReal-MDP no inicio, mantendo-se ainda cerca de 56% menor quando a carga se duplica, chegando a um pouco mais de 12% na carga máxima avaliada, 22 *Erlang*. Importante lembrar que o BCPP1 acrescenta as funcionalidade de roteamento *off-line* e a função custo. O BCPP2, que acrescenta o bloqueio preventivo, tem um rendimento inicial de até duas ordens de grandeza melhor que o BCPP1, sendo 30% melhor para a carga de 8 *Erlang*, mas com rendimento praticamente igual ao atingir a metade da carga máxima avaliada, 12 *Erlang*. O rendimento do ARPP, que traz as novas funcionalidades de tratamento diferenciado de tráfego, somente atinge uma taxa média de bloqueio de 5% na carga máxima avaliada. O ARPP tem um ganho com relação ao BCPP2 de uma ordem de grandeza. O ARPP suporta até seis vezes mais carga que o Smart-Real-MDP, cinco vezes mais que o BCPP1 e quatro vezes mais que o BCPP2.

**Tabela 2. Taxa de Bloqueio Médio - MBBR (%) - USANet**

| Carga (*Erlang*) | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|
| SmartReal-MDP | 6,69 | 15,36 | 23,65 | 30,08 | 34,76 | 38,37 | 42,62 | 45,12 | 47,35 | 49,14 |
| BCPP1 | 0,41 | 5,70 | 13,65 | 21,22 | 26,96 | 31,82 | 35,80 | 38,82 | 41,10 | 43,16 |
| BCPP2 | 0,11 | 1,55 | 9,84 | 18,85 | 25,93 | 31,75 | 35,56 | 38,96 | 42,31 | 44,42 |
| ARPP (sem classes) | 0,09 | 0,11 | 0,18 | 0,47 | 0,94 | 2,26 | 4,44 | 6,45 | 9,69 | 12,05 |
| ARPP | 0,12 | 0,16 | 0,22 | 0,29 | 0,29 | 0,37 | 0,48 | 0,63 | 1,16 | 2,17 |

**Tabela 3. Taxa de Bloqueio Médio - MBBR (%) - PanEURO**

| Carga (*Erlang*) | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
|---|---|---|---|---|---|---|---|---|
| SmartReal-MDP | 1,23 | 13,85 | 26,50 | 35,49 | 42,55 | 46,07 | 52,21 | 53,53 |
| BCPP1 | 0,19 | 7,60 | 20,32 | 29,61 | 37,31 | 43,40 | 47,79 | 50,92 |
| BCPP2 | 0,13 | 4,35 | 18,66 | 28,67 | 37,16 | 40,78 | 48,43 | 52,38 |
| ARPP (sem classes) | 0,11 | 0,17 | 0,24 | 0,40 | 0,58 | 1,07 | 2,04 | 4,18 |
| ARPP | 0,11 | 0,17 | 0,24 | 0,40 | 0,63 | 1,07 | 2,10 | 4,34 |

Na topologia PanEURO, Figura 6, o comportamento dos algoritmos é praticamente o mesmo, apenas diferenciando a escala da *Erlang* que é menor devido as características da topologia PanEURO que é mais limitada, pois possui muitos nós com grau de conectividade menor que a topologia USANet. Devido a maior limitação da rede PanEURO, menos conectividade, os resultados do ARPP (sem classes) e o ARPP são praticamente os mesmos. Quando o ARPP alcança o patamar de 5% de MBBR com uma carga de 16 *Erlang*, o ganho com relação a todos os outros algoritmos é de até uma ordem de grandeza. O BCPP2 tem um ganho de 58% com relação ao BCPP1. O BCPP1 também tem um ganho de até uma ordem de grandeza com relação ao SmartReal-MDP. Com relação a carga, o ARPP suporta até cinco vezes mais que o SmartReal-MDP, até cinco vezes mais que o BCPP1, e quatro vezes mais que o BCPP2.

Para avaliar a capacidade de prover justiça do ARPP, nas Figuras 7 e 8, os gráficos apresentam as taxas de MBBR por par. A distribuição da MBBR desta Figura indica que o algoritmo ARPP produziu valores para alguns pares bem melhores que o algoritmo BCPP2, onde a reserva de comprimentos de onda por classe de serviço permitiram o ARPP prevenir o bloqueio de forma mais eficiente do que o algoritmo BCPP2 o que, consequentemente, diminui o numero de pares comunicantes com altas taxas de requisições bloqueadas, estabelecendo, assim, um sistema mais justo. Na topologia USANet, Figura 7, o BCPP2 tem aproximadamente 51% dos pares com taxa de MBBR superior a sua

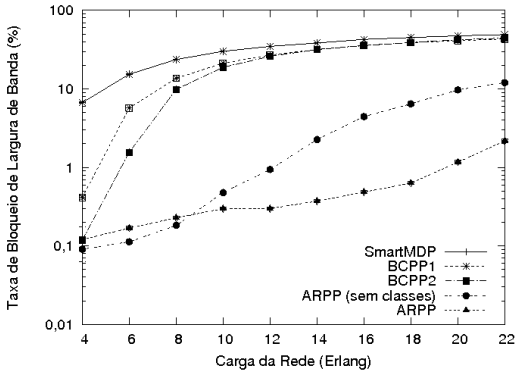



**Figura 5. MBBR para a rede USANet.**

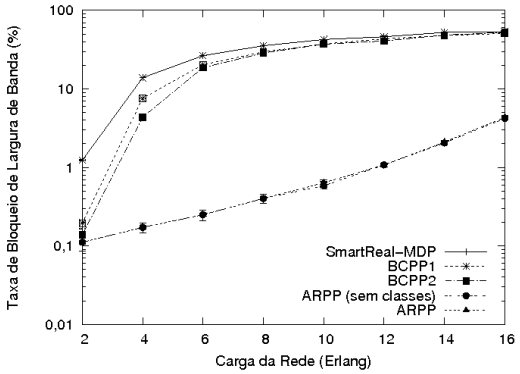



**Figura 6. MBBR para a rede PanEURO.**

média, enquanto que no ARPP apenas 24% dos pares estão acima de sua média, demonstrando que o ARPP é mais justo que o BCPP2. Na topologia PanEURO, Figura 8, o ARRP tem um redimento um pouco melhor, com apenas 20% de pares acima da média, enquanto o BCPP2 tem uma performance parecida que na topologia USANet com aproximadamente 55% de pares acima da média. Nos quatro gráficos, o valor das MBBR por par é maior ao lado direito, devido ao fato de que pares de comunicação representados nesta parte possuem grandes distâncias entre sua origem e destino.

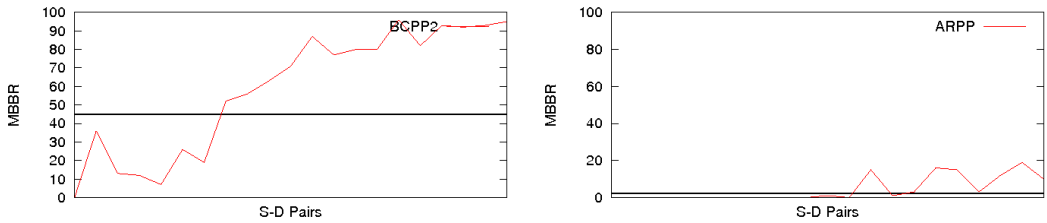



**Figura 7. MBBR de cada para origem-destino para uma carga de 22 *Erlang* para a rede USANet.**

## 5. Conclusão

Este artigo propõem uma solução que agrega várias técnicas para a provisão eficiente de recursos em redes WDM, como a agregação dinâmica de tráfego, o roteamento multicaminho, e o bloqueio preventivo, provisão de sobrevivência através da proteção parcial, além de um mecanismo de reserva de comprimentos de onda por classe de serviço, o que

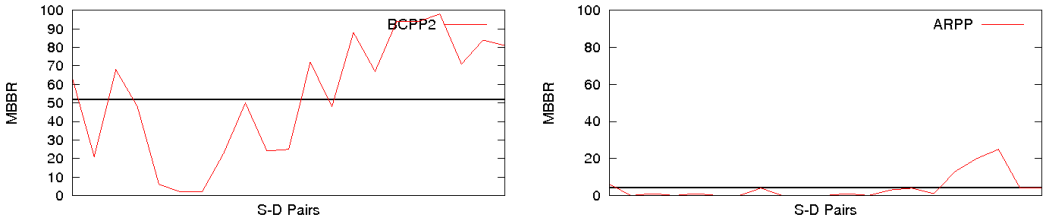



**Figura 8. MBBR de cada para origem-destino para uma carga de 16 *Erlang* para a rede PanEURO.**

leva a uma menor taxa de bloqueio na rede. O crescimento do tráfego heterogêneo com aplicações de alta capacidade e aplicações tradicionais *best-effort* tem trazido a necessidade da implementação de esquemas de roteamento com diferenciação em redes ópticas WDM.

O algoritmo proposto, *Adaptative Routing with Partial Protection* - ARPP, mostrou ser mais eficiente que os algoritmos desenvolvidos da literatura, atingindo um nível de eficiência superior a uma ordem de grandeza. O ARPP conseguiu melhorar os resultados com relação as métricas de taxa de bloqueio de banda, principalmente devido a reserva de comprimentos de onda por classe de serviço e a função custo adaptativa. Estes mecanismos trazem mais justiça ao atender aplicações heterogêneas como *Best-effort* e de Alta Capacidade, de forma a equilibrar as disponibilização dos recursos. Como estes dois tipos de classes de tráfego consomem o mesmo número de comprimentos de onda, a reserva de comprimento de onda deve ser proporcional a banda passante que cada classe consumirá entre elas. O resultado acaba sendo mais justo e mais eficiente para os dois tipos de classes.

Finalmente, o algoritmo ARPP também demonstrou grande robustez ao apresentar resultados similares frente a diferentes cenários de avaliação que empregaram topologias com níveis de conectividade distintos.

## Referências

Banerjee, D. and Mukherjee, B. (1996). A practical approach for routing and wavelength assignment in large wavelength-routed optical networks. *Selected Areas in Communications, IEEE Journal on*, 14(5):903 –908.

Cerutti, I., Fumagalli, A., and Sheth, S. (2004). Performance versus cost analysis of wdm networks with dynamic traffic grooming capabilities. In *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on*, pages 425 –430.

Chen, X., Jukan, A., Drummond, A., and da Fonseca, N. (2009). A multipath routing mechanism in optical networks with extremely high bandwidth requests. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1 –6.

Das, A., Martel, C., and Mukherjee, B. (2009). A partial-protection approach using multipath provisioning. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1 –5.

de Souza Júnior, P. J. and Drummond, A. C. (2013a). Proteção parcial para demandas de alta capacidade em redes wdm baseada em função de custo abrangente. *SBRC 2013, Workshop de Gerência e Operação de Redes e Serviços.*

de Souza Júnior, P. J. and Drummond, A. C. (2013b). Proteção parcial para demandas de alta capacidade em redes wdm utilizando mecanismo de bloqueio preventivo. *SBRC 2013- Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.*

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., Laity, A., Jacob, J. C., and Katz, D. S. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13(3):219–237.

Drummond, A. C. (2010). Wdmsim (http://www.lrc.ic.unicamp.br/wdmsim).

Drummond, A. C., de Souza Júnior, P. J., and Barreto, P. A. S. M. (2012). Protecao parcial para demandas de alta capacidade em redes wdm transparentes. *SBRC 2012, Workshop de Gerência e Operação de Redes e Serviços.*

Huang, S., Martel, C., and Mukherjee, B. (2011). Survivable multipath provisioning with differential delay constraint in telecom mesh networks. *Networking, IEEE/ACM Transactions on*, 19(3):657 –669.

Kim, H. J. and Choi, S. G. (2010). A study on a qos/qoe correlation model for qoe evaluation on iptv service. In *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, volume 2, pages 1377 –1382.

Padmanabhan, V., Wang, H., and Chou, P. (2003). Resilient peer-to-peer streaming. In *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on*, pages 16 – 27.

Rai, S., Deshpande, O., Ou, C., Martel, C., and Mukherjee, B. (2007). Reliable multipath provisioning for high-capacity backbone mesh networks. *Networking, IEEE/ACM Transactions on*, 15(4):803 –812.

Ramamurthy, S., Sahasrabuddhe, L., and Mukherjee, B. (2003). Survivable wdm mesh networks. *Lightwave Technology, Journal of*, 21(4):870 – 883.

Simeonidou, D., Hunter, D., Ghandour, M., and Nejabati, R. (2008). Optical network services for ultra high definition digital media distribution. In *Broadband Communications, Networks and Systems, 2008. BROADNETS 2008. 5th International Conference on*, pages 165 –168.

Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M. (2007). *Workflows for e-Science. Scientific Workflows for Grids*. Springer.

Tian, X., Yang, J., Ding, H., and Yang, Y. (2012). Multi-objective routing and wavelength assignment based priority algorithm in wdm networks. In *Information Science and Engineering (ISISE), 2012 International Symposium on*, pages 87–92.

Wang, H., Modiano, E., and Medard, M. (2002). Partial path protection for wdm networks: End-to-end recovery using local failure information. In *Seventh International Symposium on Computers and Communications*.

Xue, G., Zhang, W., Tang, J., and Thulasiraman, K. (2005). Establishment of survivable connections in wdm networks using partial path protection. Technical report, Department of Computer Science & Engineering, Arizona State University, Tempe, AZ.

Yao, W. (2005). Survivable traffic grooming with path protection at the connection level in wdm mesh networks. *Journal of Lightwave Technology*, 23(10).

# Proteção de Redes Ópticas Elásticas Contra até Duas Falhas Baseada em p-Cycle FIPP

**Helder M. N. da S. Oliveira**[1], **Nelson L. S. da Fonseca**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas 13089-971, SP, Brasil

helder@lrc.ic.unicamp.br, nfonseca@ic.unicamp.br

***Abstract.*** *Optical networks are subject to faults in links and nodes which can cause massive data loss. In the literature, many techniques have been proposed to deal with such failures in order to support continuous network operation. One of these techiniques defines p-Cycles which provide ring-type protection and the speed of restoration of meshes. Morever, Flexgrid networks emerged as a solution to deal with the diverse requirements of bandwidth. This paper presents two new algorithms to provide path protection using p-cycle path networks in Flexgrid networks. Our algorithms provide 100 % protection against single failure as well as against two failures.*

***Resumo.*** *Redes Ópticas estão sujeitas à falhas em enlaces e nós, o que pode causar perda maciça de dados. Muitas técnicas vem sendo desenvolvidas para lidar com essas falhas no intuito de dar suporte a operação continua da rede. p-Cycles fornecem proteção em anel bem como a velocidade de restauração da topologia em malha. Por outro lado, a tecnologia de redes Flexgrid emergiu como uma solução para lidar com a diversidade de largura de banda demandada pelas aplicações. O presente artigo apresenta dois novos algoritmos para fornecer proteção de caminho através do emprego de p-cycle de caminho em redes Flexgrid. Os algoritmos proporcionam 100% de proteção contra uma falha, bem como proteção contra duas falhas simultâneas.*

## 1. Introdução

Uma das principais características da arquitetura da Internet é não impor restrição sobre a camada de aplicação, permitindo, assim, o surgimento de novas aplicações. Essas aplicações têm demandas heterogêneas de largura de banda. Enquanto algumas aplicações, tais como e-mails requisitam pouca demanda de largura de banda, outras como IPTV e aplicações de *grid* requisitam largura de banda na ordem de Gbits por segundo [Xiaomin et al. 2009]. Tal diversidade exige uma rede de transporte flexível no que tange ao provisionamento de banda passante.

A técnica de multiplexação por divisão de comprimento de onda (WDM) empregada na camada de enlace da Internet possibilita a disponibilidade de grande capacidade e permite a multiplexação de vários comprimentos de onda em uma única fibra. WDM aloca faixas de frequência de largura fixa, o que leva a um conjunto de comprimentos de onda com a mesma capacidade seperados por uma frequência de banda de guarda. Tal alocação leva à subutilização do espectro uma vez que as demandas raramente coincidem

com a capacidade de um comprimento de onda. Em WDM , a capacidade de um comprimento de onda pode acomodar demandas de diferentes tamanhos através do emprego de agregação de trafego (*traffic grooming*), porém esta técnica implica em *overhead* e não elimina por completo o problema do desperdício de capacidade. Apesar da tecnologia de WDM com multiplas taxas apresentar alguma flexibilidade na alocação de recursos, sua granularidade de alocação é "grossa" e só melhora limitadamente o problema de desperdício de capacidade.

Tal rigidez de alocação de espectro motivou o surgimento de redes elásticas cujo espectro pode ser alocado com granularidade "fina" (spectrum-sliced elastic optical network), o que possibilita a alocação de faixas do espectro de tamanho variável. Redes ópticas elásticas empregam a multiplexação por divisão de frequências ortogonais (OFDM). OFDM é uma tecnologia de transmissão multi-portadora, que divide canais subportadoras ortogonais de menor capacidade.

Por ser um meio cabeado, as fibras ópticas são propensas a cortes. Dada a enorme capacidade de uma fibra óptica, qualquer interrupção implica em grandes perda de dados. Tal vulnerabilidade tem motivado o desenvolvimento de diferentes esquemas de restauração e proteção. *p-Cycle* é uma dessas técnicas de proteção, cujo emprego foi investigado intensamente em redes WDM. *p-Cycles* combinam as propriedades de proteção da topologia em anel de recuperação rápida e a eficiência de proteção das redes de malha restauráveis. *p-Cycles* protegem os enlaces que cobrem, bem como os enlaces que tem seus pontos finais no p-cycle ("enlaces tranzonais"). Um tipo de *p-cycle* de interesse especial é o *p-cycle* FIPP (*Failure-Independent Path Protecting*) que fornece caminhos de proteção pre-conectados.

No entanto, a grande dificuldade no projeto de redes que empregam *p-cycles* para proteção é a complexidade computacional para obter a solução do problema de provisão de proteção, que é exponencialmente dependente do número de nós e do número de enlaces. O problema de estabelecimento de caminhos protegidos por *p-cycles* é um problema NP-completo e heurísticas têm sido desenvolvidas para resolvê-lo. O presente artigo introduz dois algoritmos chamados FIPP-Flex e FIPP-Flex-twofailure para proteção de caminhos através do uso de p-cycle FIPP em redes ópticas elásticas (*Flexgrid*). O algoritmo FIPP-Flex provê proteção contra uma única falha e o algoritmo FIPP-Flex-twofailure provê proteção contra ocorrência de duas falhas simultâneas. Os algoritmos promovem compartilhamento de caminhos de proteção com o maior número de caminhos primários possível.

Este artigo está organizado da seguinte forma. A seção 2 revisa trabalhos relacionados. A seção 3 introduz os conceitos de *p-cycle* e FIPP . A seção 4 apresenta a notação usada no artigo. A seção 5 introduz os algoritmos RSA -FLEX e a seção 6 apresenta os algoritmos FIPP -FLEX e o FIPP-Flex-twofailure. A seção 7 avalia o desempenho do algoritmo proposto e a seção 8 conclui o artigo .

## 2. Trabalhos Relacionados

Apesar de terem sido propostos diversos algoritmos de roteamento e alocação de espectro(routing and spectrum alocation - RSA), pouca atenção tem sido dada a questão de proteção em redes elásticas.

Foi proposto em [Pages et al. 2012] um mecanismo que atua nos transponders

ociosos no nó de origem de uma requisição de enlace de alta taxa de dados para fragmentá-lo em vários de baixa taxa de dados, mais facilmente alocável na rede. Além disso, com o objetivo de apoiar a operação, um algoritmo RSA foi também proposto para alocar adequadamente os fragmentos gerados ao longo do espectro. Para fragmentar a demanda de banda passante da requisição em várias demandas com menor requisição de banda mais facilmente alocáveis na rede.

Em [Tanaka et al. 2013] os requisitos de nós ópticos em redes ópticas elásticas foram estudados e múltiplas arquiteturas de rede usando um esquema de alocação de recurso integrado foram comparadas.

Os autores de [Shao et al. 2012] propuseram políticas conservadoras de proteção em redes ópticas baseadas em OFDM com alocação de banda elástica. Eles introduziram uma política em que caminhos ópticos de *backup* protegem caminhos ópticos primários utilizando caminhos disjuntos, o que leva a maior robustez a proteção.

Os autores em [Patel et al. 2011] propuseram técnica de sobrevivência "transparente" em redes ópticas WDM (fwdm). Os autores utilizam a politica *First- Fit* para atribuir espectro para os "caminhos de trabalho" e a politica *Last-Fit* para atribuir espectro para os "caminhos de *backup*". O espectro de *backup* de diferentes caminhos podem se sobrepor. A abordagem de proteção proposta, chamada *elastic separate-protection-at-connection* (ESPAC), fornece proteção fim a fim.

Em Liu *et al.* [Liu et al. 2013], os autores propuseram uma nova técnica para a proteção compartilhada que proporciona aos caminhos ópticos primários compartilhar o espectro de caminhos de *backup* se os caminhos primários forem disjuntos fisicamente.

Nenhum dos trabalhos mencionados anteriormente emprega p-cycle para proteção de caminhos.

## 3. p-Cycle

O *p-cycle* é um esquema de proteção em que a capacidade reservada é pre-conectada e forma estruturas em anel para proteção de redes em malha [Asthana et al. 2010]. *p-Cycles* fornecem proteção semelhante a proteção fornecida por Anéis de Comutação de Linhas Bidirecionais (BLSR), que é considerado uma generalização do esquema de proteção 1:1 [Kiaei et al. 2009]. A diferença fundamental entre p-cycle e proteção em anel é a proteção de enlaces transzonais, que são enlaces que não estão no anel (ciclo) porém os dois nós finais estão. Esta propriedade melhora a eficiência de proteção dos p-cycles. Outra propriedade importante é a ausência da necessidade de caminhos comporem rotas em anel, a fim de serem protegidos. *p-Cycles* proveem rápida restauração pois são pre-conectados [Schupke 2006].

A Figura 1 ilustra o conceito de *p-cycle*. Na figura, uma linha simples representa um enlace, um ciclo em negrito representa o *p-cycle* e a seta representa o caminho de recuperação de uma falha denotada por um "X". Na figura 1(a), A-B-C-D-E-A é um *p-cycle* que usa a capacidade reservada nos enlaces de proteção. Quando o enlace A-B falha, a proteção é provisionada como ilustrado na figura 1(b). Quando o enlace transzonal B-D falha, cada p-cycle proporciona dois caminhos alternativos como mostram as figuras 1 (c) e 1(d).

Um caso especial de *p-cycle* para proteção de caminho é o chamado *p-cycle* de

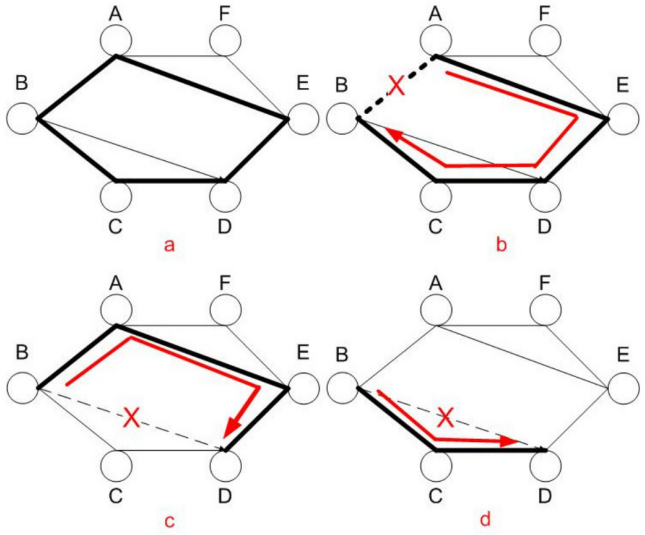



**Figura 1. Exemplo de p-cycle**

Proteção de Caminho com Independência de Falha (FIPP) [Kodian and Grover 2005]. *p-Cycles* FIPP fornecem proteção de caminhos para caminhos com nós finais sobre o *p-cycle*.

FIPP é uma extensão do conceito de *p-cycle* que permite que falhas não sejam necessariamente limitadas a um enlace ou segmento de caminho imediatamente adjacente aos nós terminais. FIPPs baseiam-se em diferentes "caminhos de trabalho" e de "*backup*", e proporciona a vantagem de que a detecção de falha é independente da localização da falha, ou seja, é "independente de falha". Tal propriedade é vantajosa quando a localização da falha é lenta ou difícil, como em redes transparentes e translúcidas [Kodian and Grover 2005].

A Proteção por compartilhamento de caminhos de reserva (SBPP) propostas para redes baseadas em sinalização IP também tem a propriedade de falhas independentes. No entanto, a principal diferença entre SBPP e FIPP é que em SBPP o caminho de *backup* precisa ser determinado em tempo real, em caso de falha, o que pode levar a adoção de um caminho de restauração sem a qualidade de transmissão adequada. Por outro lado em FIPP, caminhos de proteção pre-conectado são muito importantes para assegurar a QOS desejada. Além disso, SBPP exige uma extensa base de dados, devido à necessidade de todos os nós terem conhecimento da capacidade global, topologia e compartilhamento de caminho de *backup*.

A Figura 2 ilustra o conceito de *p-cycle* FIPP. Na Figura 2, o ciclo em destaque EFGHIQRONM representa o p-cycle e as setas mostram os vários caminhos que são protegidos. Nesses caminhos, estão tanto enlaces transzonais quanto enlaces do próprio p-cycle. Pode-se observar que um único p-cycle protege um conjunto de catorze caminhos mutuamente excludentes e que possuem seus nós de extremidade no p-cycle de caminho.

## 4. Notação

A fim de facilitar a compreensão dos algoritmos propostos, a presente seção introduz a notação usada no artigo. Seja:

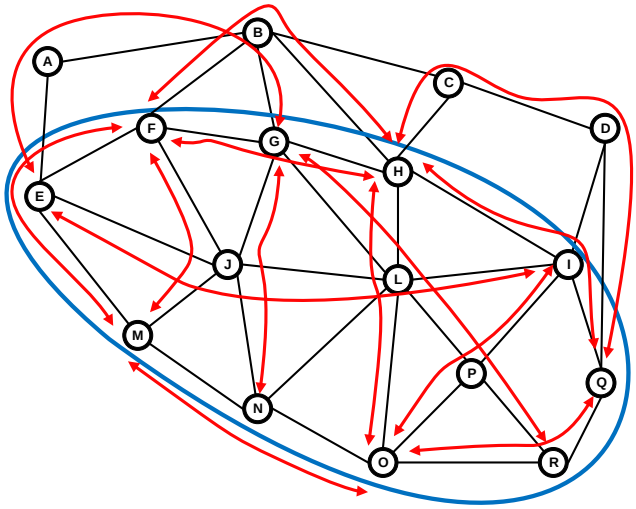



**Figura 2. Exemplo de p-cycle FIPP**

$s$: o nó fonte;

$d$: o nó destino;

$b$: a demanda de largura de banda nos *slots*, $b = 1 \ldots B$;

$r(s, d, b)$: requisição do nó $s$ para o nó $d$ com demanda de largura de banda $b$ em *slots*;

$N$: número de *slots* entre dois nós;

$G = (V, E, W)$: multigrafo marcado composto por um conjunto de nós $V$, um conjunto de arestas $E$ e um conjunto de pesos das arestas $W$, $|E| = N \cdot |V|$.

$E = \{e_{u,v,n}\}$: conjunto de $n$ arestas;

$e_{u,v,n}$: a n-ésima arestas conectando os nós $u$ e $v$;

$w(e_{u,v,n})$: o peso das arestas $e_{u,v,n}$;

$c(e_{u,v,n}) = 1$ se o n-ésimo *slot* no enlace conectando os nós $u$ e $v$ estão livres e $w(e_{u,v,n}) = \infty$ se o *slot* está alocado;

$W = \{c(e_{u,v,n})\}$:conjunto de pesos das arestas;

$\widetilde{G}_{n,b} = (\widetilde{V}, \widetilde{E}, \widetilde{C})$: O n-ésimo grafo marcado tal que $\widetilde{E}$ é o conjunto de arestas conectando $\{\widetilde{u}, \widetilde{v}\} \in \widetilde{V}$ e $\widetilde{C}$ é o conjunto de custos associados a $\widetilde{E}$. As arestas em $\widetilde{E}$ correspondem ao mapeamento de $b$ arestas de $G$, sendo a primeira aresta a n-ésima aresta conectando $u$ e $v$.

$\widetilde{V} = V$: conjunto de nós;

$\widetilde{e}_{u,v} \in \widetilde{E}$: arestas conectando $\widetilde{u}$ e $\widetilde{v}$; $\widetilde{e}_{\widetilde{u},\widetilde{v}} = \{e_{u,v,n}\} \in E$ é uma sequência tal que $e_{u,v,n}$ é a menor aresta ordenada, $e_{u,v,n+b}$ é a maior aresta ordenada e $|\widetilde{e}_{u,v}| = b$;

$\widetilde{w}_n(\widetilde{e}_{\widetilde{u},\widetilde{v}})$: peso da aresta $\widetilde{e}_{\widetilde{u},\widetilde{v}}$;

$\widetilde{W}_n = \{\widetilde{c}_n(\widetilde{e}_{\widetilde{u},\widetilde{v}})\}$: conjunto de pesos de arestas;

$P_n$: sequência de arestas $\widetilde{G}_n$ tal que o nó fonte $s$ é o menor nó ordenado e $d$ é o maior nó ordenado;

$W(\widetilde{P}_n)$: $\sum_{\widetilde{e}_{\widetilde{u},\widetilde{v}} \in \{\widetilde{P}_n\}} \widetilde{e}_{\widetilde{u},\widetilde{v}}$: o peso do caminho $\widetilde{P}_n$ é a soma dos pesos de todos as arestas na cadeia;

$W_{s,d}$ = peso do menor caminho entre $s$ e $d$;

$\widetilde{c}_{u,v,b}$: *p-cycle* que contém os vértices $u$ e $v$ e arestas correspondentes ao mapeamento de $b$ arestas do multigrafo $G$;

$\widetilde{C}_{u,v,b} = \widetilde{c}_{u,v,b}$: conjunto de todos os *p-cycles* que contém os vértices $u$ e $v$ e arestas correspondentes ao mapeamento de $b$ arestas do multigrafo $G$;

$\widetilde{C}$: conjunto de todos os *p-cycles* estabelecidos;

$P_1 \oplus P_2$: concatenação de dois caminhos $P_1$ e $P_2$

## 5. O Algoritmo RSA-Flex

Solução do problema de roteamento e atribuição de comprimentos de onda (RWA) em redes WDM são semelhantes as soluções do problema de roteamento e alocação de espectro (RSA) em redes ópticas *Flexgrid*. Além da restrição de continuidade de espectro que impõe a utilização do mesmo espectro em cada fibra ao longo da rota de um caminho óptico, nos algoritmos RSA *slots* devem ser alocados de forma contínua no espectro (restrição de contiguidade do espectro).

O problema de roteamento e alocação de espectro é um problema NP-difícil [Wang et al. 2011] e heurísticas são necessárias para resolver o problema. Os algoritmos propostos neste artigo modelam a disponibilidade de espectro na rede como um multigrafo rotulado. Um multigrafo é um grafo que pode ter várias arestas (também chamadas de "arestas paralelas"), conectando o mesmo par de nós. Neste grafo, os vértices representam OXCs e as arestas representam os *slots* dos enlace que conectam os OXCs. Nos multigrafos utilizados no presente artigo, todos os vértices são conectados por $N$ arestas, que é o número de *slots* no espectro de cada enlace da rede. O $W_{s,d}$ representa a disponibilidade de *slots*. O valor $\infty$ significa que o *slot* já está alocado enquanto o valor 1 significa que o *slot* está disponível para alocação. Estes valores foram definidos para facilitar o emprego de algoritmos tradicionais que encontram o caminho mais curto.

No procedimento proposto, o multigrafo é transformado em $N - b + 1$ grafos, sendo $b$ a demanda de largura de banda em *slots* requisitada. Estes grafos são gerados através da seleção de uma aresta no multigrafo e das $b$ arestas consecutivas a aresta selecionada. Este conjunto de $b$ arestas do multigrafo é mapeada em uma única aresta do grafo gerado. Seu peso é dado pela aplicação de uma função de peso específica, que considera o peso das $b$ arestas. A Figura 3 ilustra um multigrafo representando o espectro e um dos grafos gerado. Para cada grafo gerado, executa-se um algoritmo de caminho mais curto e o caminho escolhido é o que tem o menor peso entre todos os caminhos mais curtos encontrados.

Para uma demanda de $b$ *slots*, $N - b + 1$ grafos do tipo $\widetilde{G}_{n,b}$ são gerados. Cada aresta do grafo $\widetilde{G}_{n,b}$ corresponde ao mapeamento de $b$ arestas de $G$ iniciando na n-ésima aresta de $G$. Uma vez que as mesmas arestas ordenadas conectando dois nós em $G$ são

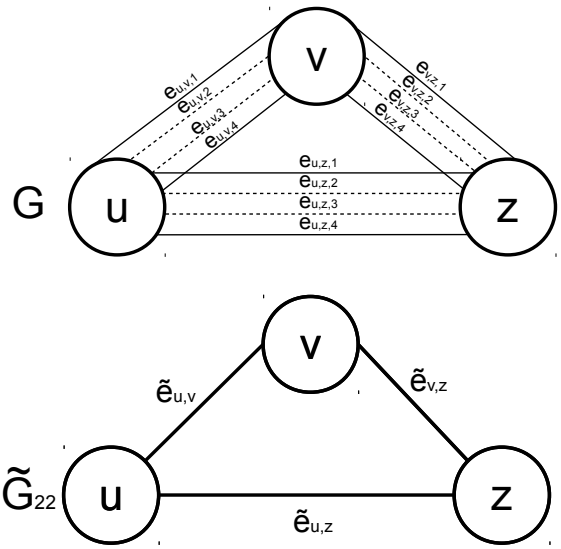



**Figura 3. Multigrafo em um grafo associado**

mapeadas em arestas de $\widetilde{G}_{n,b}$, a continuidade do espectro é garantida.

**Algorithm 1** RSA-Flex

1: $\forall n = 1...N-b$
2: $\quad (W(P_n), P_n) = ShortestPath(\widetilde{G}_{n,b}, r(s,d,b))$
3: $W_{s,d} = W(P_n) | \, \forall i \; W(P_n) \leq W(P_i)$
4: **if** $W_{s,d} = \infty$ **then**
5: $\quad block \; r(s,d,b)$
6: **else**
7: $\quad W(e_{u,v,i}) = \infty \quad \forall \{u,v\} \in \widetilde{P}_i \quad n = n...i{+}b-1$
8: **end if**

O algoritmo 1 detalha o algoritmo RSA-Flex. Neste algoritmo, a linha 1 estabelece o conjunto de arestas que será mapeada para $\widetilde{G}_{n,b}$ arestas. A linha 2 resolve um algoritmo de menor caminho para o grafo $\widetilde{G}_{n,b}$ e provê o menor caminho e seu respectivo peso. Se o peso do caminho mais curto for $\infty$, não é possível encontrar um caminho com a restrição de continuidade para a demanda $b$ com alocação começando com o n-ésimo *slot*. A linha 3 seleciona o caminho entre os $N - b + 1$ caminhos mais curtos, com o menor peso. No caso do peso de todos o caminhos mais curto serem $\infty$ (linha 4), não existe caminho na rede que satisfaça a requisição de $b$ *slots*, portanto, a requisição tem de ser bloqueada (Linha 5). Caso contrário, o caminho mais curto com o menor peso será escolhido (linha 7) e as arestas correspondentes no multigrafo $G$ terão seu peso alterado para $\infty$ (linha 8), o que significa que os *slots* são alocados para o caminho óptico recém-estabelecido.

Uma vez que o algoritmo RSA-Flex executa um algoritmo de caminho mais curto $N - b$ vezes e considerando o uso do algoritmo de caminho mais curto de Dijkstra, a complexidade computacional do algoritmo proposto é $N \cdot (|V| + |E|) \cdot \log(|V|)$.

## 6. Algoritmos FIPP-Flex e FIPP-Flex-twofailure

**Algorithm 2** FIPP-Flex

1: $(W(P_n), P_n) = RSA - Flex(G, s, d, b)$
2: **if** $W_{s,d} = \infty$ **then**
3: $block\ r(s, d, b)$
4: **else**
5: **if** $C_{u,v,i} \neq \emptyset \forall i \geq b$ **then**
6: establish $r(s, d, b)$ as $P_n$
7: **else**
8: $(W(P_1), P_1)$ = RSA-Flex $(G, r(s, d, b))$
9: $(W(P_2), P_2)$ = RSA-Flex $(G, r(s, d, b))$
10: **if** $W(P_1) = \infty$ or $W(P_2) = \infty$ **then**
11: block $r(s, d, b)$
12: **else**
13: establish $r(s, d, b)$ as $P_n$
14: establish $P_1$ and $P_2$
15: $\widetilde{c}_{u,v,b} = P_1 \oplus P_2$
16: **end if**
17: **end if**
18: **end if**

Os algoritmos apresentados nesta seção, chamados FIPP-Flex e FIPP-Flex-twofailure, resolvem o estabelecimento de caminhos ópticos em redes protegidas por *p-cycles* FIPP. Nesses algoritmos, caminhos ópticos são estabelecidos se e somente se puderem ser protegidos por um *p-cycle* FIPP, que protegem somente caminhos primários disjuntos. Requisições para estabelecimento de caminhos ópticos chegam dinamicamente e para cada requisição tenta-se associar um *p-cycle* existente para proteger o caminho óptico requisitado. Caso não haja *p-cycle* que possa proteger o caminho óptico, então, cria-se um novo *p-cycle* para a requisição. Se nenhum caminho for encontrado para proteger o caminho solicitado, então ele não é estabelecido. O algoritmo FIPP-Flex garante um caminho de proteção para cada caminho óptico estabelecido e a proteção é garantida para falhas individuais. O algoritmo FIPP-Flex-twofailure diferentemente do algoritmo FIPP-Flex garante um caminho de proteção totalmente disjunto do caminho primário, e garante assim proteção para duas falhas simultâneas.

No algoritmo FIPP-Flex, a linha 1 tenta encontrar um caminho para estabelecer a requisição $r(s, d, b)$. Se não houver caminho disponível (linha 2), então o pedido é bloqueado (linha 3). Caso exista, procura-se um *p-cycle* para proteger o caminho solicitado (linha 5). Se existir um *p-cycle*, o caminho óptico é estabelecido. Caso contrário, tenta-se descobrir um *p-cycle* em potencial para proteger o caminho óptico solicitado (linhas 8 e 9). Se não existir nenhum *p-cycle* que possa ser criado para proteger o caminho óptico, a requisição é bloqueada (Linha 11); caso contrário, o caminho óptico (linha 13), bem como o *p-cycle* (linhas 14 e 15) são estabelecidos para satisfazer a requisição.

## 7. Avaliação de Desempenho

Nos experimentos de simulação, foi utilizado o simulador de eventos discretos *Flex-GridSim* [Moura and Drummond ] desenvolvido em Java. A simulação de redes ópticas

elásticas pode ser realizada levando-se em conta uma demanda de conexões estática, expressa através de uma matriz de tráfego, ou levando-se em conta uma demanda dinâmica de conexões, que escolhe aleatoriamente os pares origem destino de uma conexão, o tempo de início da conexão e o período de duração da conexão. O simulador *FlexGridSim* considera um modelo dinâmico de requisição de conexão. *FlexGridSim* possui um escalonador de eventos de conexão e de encerramento de conexão. No simulador foi acrescentado um escalonador de falha de enlace e de recuperacão de falhas. Estes três tipos de eventos permite a simulação de falhas em rede óptica com requisição dinâmica de conexões. O simulador admite como parâmetros de entrada: a topologia da rede, o tipo de mecanismo de proteção a ser simulado (algoritmo de protecão) e a taxa de falha de enlace.

No algoritmo proposto nesse artigo, a cada requisicão de conexão, o algoritmo de roteamento e alocação de espectro (RSA) busca um caminho primário e outro secundário (backup), também chamado de proteção. Caso a rede consiga prover os dois canais, a conexão é estabelecida; caso contrário, a conexão não é estabelecida, ou seja ocorre uma situação de bloqueio de conexão. O algoritmo de RSA adotado é o RSA-Flex.

Para avaliar o desempenho dos algoritmos FIPP-Flex e FIPP-Flex-twofailure, foram empregados experimentos de simulação e os resultados comparados aos das redes sem qualquer esquema de proteção. O simulador *FlexGridSim* [Moura and Drummond ] foi utilizado nas simulações. Em cada simulação, 100.000 requisições foram geradas. Utilizou-se o método de replicação independente e adotou-se nível de confiança de 95 % para os intervalos de confiança. A topologia dfn-bwin–D-B-E-N-C-A-N-N (Figure 4) foi utilizada. Esta topologia contém tem 10 nós e 45 enlaces. Na rede elástica simulada, a análise do espectro foi dividido em 300 *slots* de 12,5 GHz cada.

A Figura 5 mostra a probabilidade de sobrevivência da rede com proteção contra falha dupla. A curva rotulada como "FIPP-Flex" denota a rede protegida por *p-cycle* criado pelo algoritmo FIPP-Flex, a curva rotulados como "FIPP-Flex-twofailure" denota a rede protegida por *p-cycle* criado com o algoritmo FIPP-Flex-twofailure, ou seja, a rede estará protegida contra duas falhas simultâneas. Embora o algoritmo FIPP-Flex não tenha sido projetado para proteção contra dupla falha, a probabilidade de sobrevivência foi plotada para o caso de dupla falha a fim de se avaliar a sua capacidade de prover proteção nestes casos. A probabilidade de sobrevivência contra duas falhas no algoritmo FIPP-Flex é 0,025. Embora este valor não seja aceitável para redes operacionais, ter uma garantia de probabilidade de sobrevivência de 0,025 sem ter que pagar o *overhead* associado para proteção de dupla falha é bastante vantajoso.

A Figura 6 apresenta o bloqueio de banda (*Bandwidth Blocking Ratio*) para redes sem proteção, com proteção contra uma falha e com proteção contra duas falhas. A curva rotulada com "No-FIPP-Flex"denota resultados para rede sem proteção. A BBR gerada para se ter proteção contra duas falhas está em torno de 0.5 devido a exigência de se ter três caminhos disjunto o que faz com que alguns nós da rede sejam sobrecarregados mais rapidamente. A BBR gerada para se proteger uma única falha está em torno de 0.3. Há, portanto, um crescimento da BBR de 0,7 ao se garantir proteção contra dupla falha em relação à sem proteção. Observa-se que a partir da carga de 50 *erlangs*, o impacto no BBR para se proteger contra uma única falha é de aproximadamente 0,2, ou seja, bloqueia-se apenas 20 % a mais da banda requisitada do que em uma rede que não provê qualquer

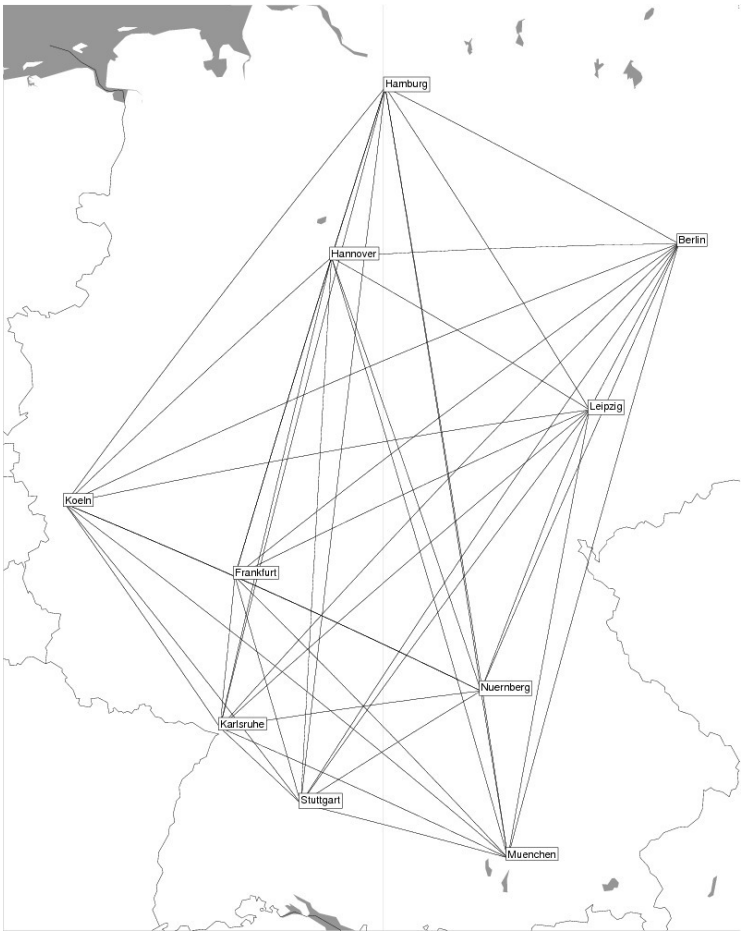



**Figura 4. Topologia dfn-bwin–D-B-E-N-C-A-N-N**

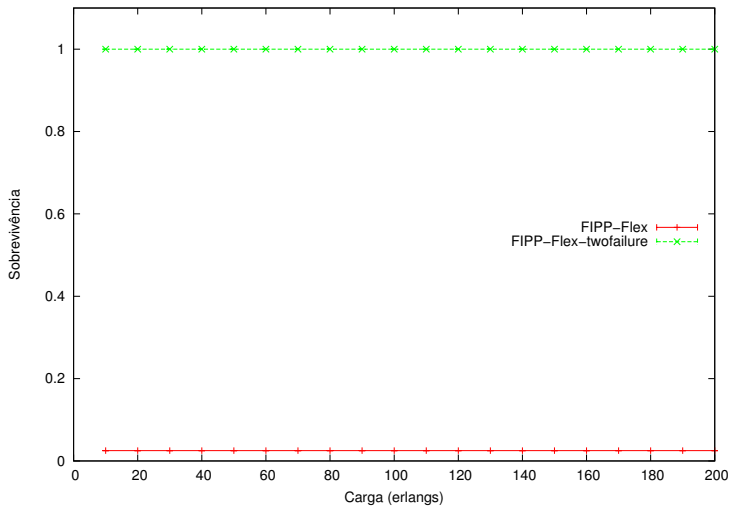



**Figura 5. Sobrevivência**

proteção de caminho.

A Figura 7 compara o número médio de saltos para o caminho primário. Observa-se que na rede com proteção contra falha dupla os caminhos são compostos por um maior

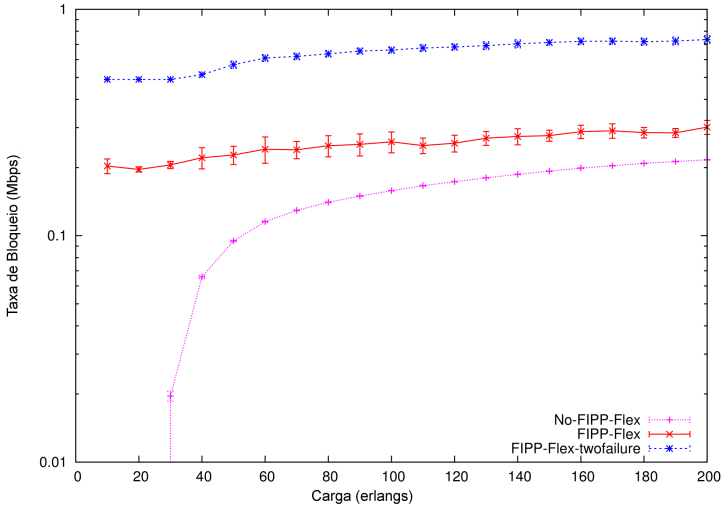



**Figura 6. Bloqueio de banda em função da carga da rede**

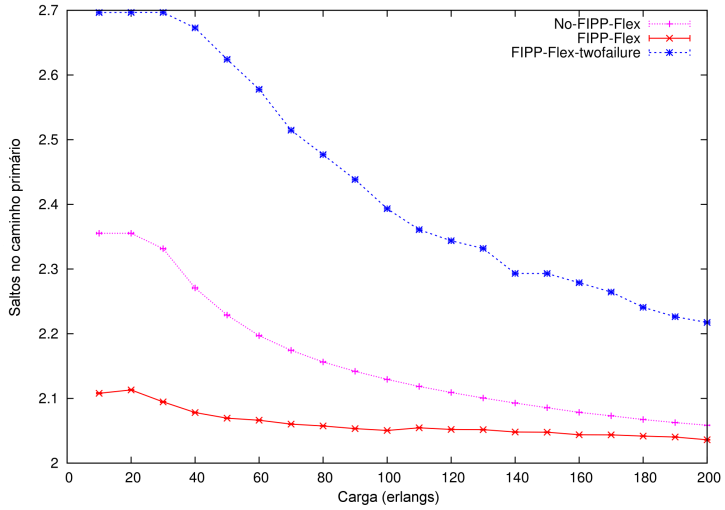



**Figura 7. Tamanho médio dos saltos no caminho primário**

número médio de saltos e que este decresce com o aumento da carga. Com o aumento da carga, diminui-se a disponibilidade de enlaces e aumenta-se o bloqueio da rede. Sob altas cargas, consegue-se estabelecer apenas caminhos menores. Vários caminhos que podem ser estabelecidos em redes sem proteção não podem ser estabelecidos em redes com proteção contra uma falha, consequentemente o tamanho médio dos caminhos em redes com proteção para uma falha são menores que os caminhos em redes com proteção.

Nas redes Flexgrid, o estabelecimento e rompimento de caminhos ópticos leva à fragmentação do espectro, que, é caracterizado pela existência de *slots* disponíveis que não podem ser aglutinados para aceitar requisições. Define-se razão de fragmentação como a razão entre a média do número de tipos de requisição (tamanho em slots) que não podem ser aceitas é o número total de tipos de requisição. A razão de fragmentação indica a probabilidade de diferentes tipos de requisições serem rejeitadas como consequência da fragmentação. Há uma grande diferença entre os índices de fragmentação para redes sem

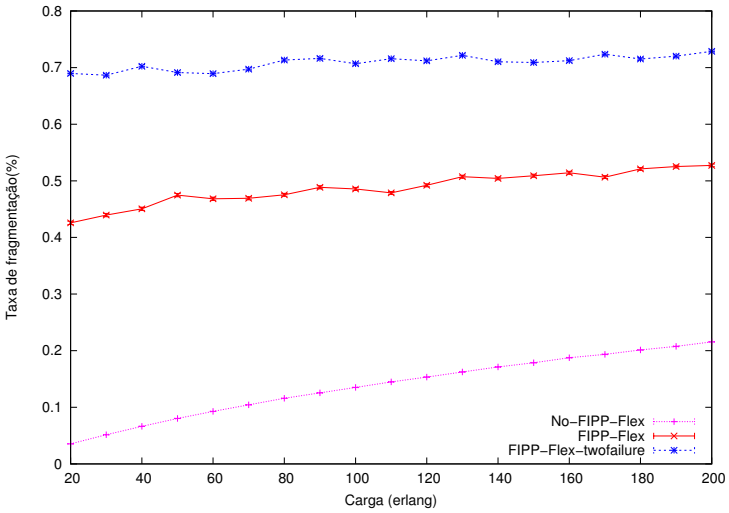



**Figura 8. Relação de Fragmentação**

proteção, com proteção contra uma falha e contra duas falhas. Quanto maior o grau de proteção, maior é a fragmentação da rede devido à alocação de maior número de caminhos de proteção de capacidade diversas.

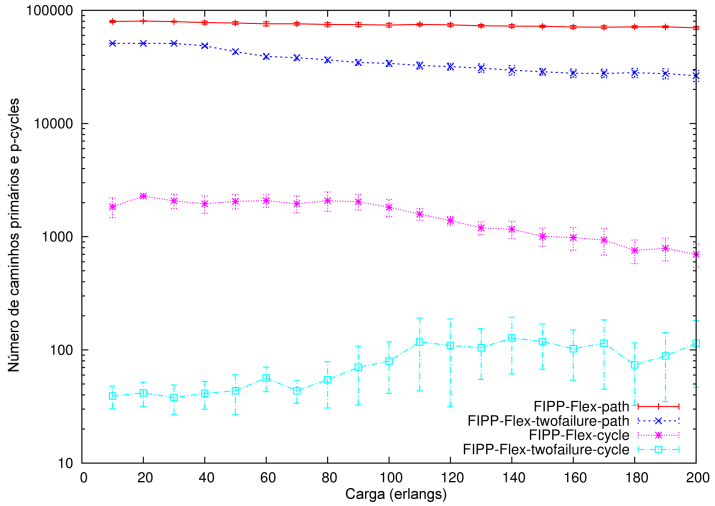



**Figura 9. Número de Caminhos Primários e de p-Cycles em função da carga**

A Figura 9 compara o número de *p-cycles* e caminhos primários estabelecidos. Observa-se que o número de caminhos primários estabelecidos ao se empregar o algoritmo FIPP-Flex difere em menos de uma ordem de grandeza do número de caminhos primários ao se utilizar o algoritmo FIPP-Flex-twofailure. No entanto, a diferença é de duas ordens de grandeza ao se comparar o número de p-cycles demandado. Dado que há mais bloqueio em redes que utilizam o algoritmo FIPP-Flex-twofailure, existe uma tendência para se estabelecer caminhos ópticos de proteção apenas para um número reduzido de *p-cycles*. Isto tende a aumentar o compartilhamento de *p-cycles*. Em redes que se utiliza o FIPP-Flex, um maior número de *p-cycles* com capacidade heterogênea é estabelecido e *p-cycles* tende a ser compartilhado por um número menor de caminhos

primários.

## 8. Conclusão

Este artigo apresentou dois algoritmos para apoiar a criação de caminhos ópticos em redes *Flexgrid* protegidos por *p-cycles* FIPP para uma falha bem como para falhas duplas. Os algoritmos foram avaliados em diferentes cenários. O algoritmo FIPP-Flex fornece 100 % de proteção para falhas individuais e uma baixa sobrevivência para dupla falha. O algoritmo FIPP-Flex-twofailure provê 100 % de proteção contra duas falhas. No cenário com proteção contra única falha, é possível observar um impacto de até 20% no bloqueio de largura de banda. No cenário com proteção contra duas falhas a necessidade de se criar três caminhos disjuntos afeta significativamente o bloqueio de largura de banda causando sobrecarga e aumentando em mais 20 % o bloqueio em relação a redes protegidas contra uma falha única. Em ambos os cenários, o número de caminhos primários é maior que o número de p-cycles, já que os caminhos primários compartilham os p-cycles. O número de caminhos primários para redes protegidas contra duas falhas é uma ordem de grandeza menor do que o número de caminhos primários protegidos contra uma única falha, além disso, o número de p-cycles é tipicamente uma ordem de grandeza maior. Como trabalho futuro diferentes esquemas de modulação e o impacto das condições de transmissão serão consideradas no algoritmo RSA.

## Referências

Asthana, R., Singh, Y., and Grover, W. (2010). p-cycles: An overview. *IEEE, Communications Surveys Tutorials*, 12(1):97 –111.

Kiaei, M., Assi, C., and Jaumard, B. (2009). A survey on the p-cycle protection method. *IEEE, Communications Surveys Tutorials*, 11(3):53 –70.

Kodian, A. and Grover, W. (Oct. 2005). Failure-independent path-protecting pcycles: efficient and simple fully preconnected optimal-path protection. *IEEE, J. Lightwave Technol.*, 23:3241–3259.

Liu, M., Tornatore, M., and Mukherjee, B. (2013). Survivable traffic grooming in elastic optical networks shared protection. *IEEE, Journal of Lightwave Technology*, 31(6):903–909.

Moura, P. M. and Drummond, A. C. FlexGridSim: Flexible Grid Optical Network Simulator. http://www.lrc.ic.unicamp.br/FlexGridSim/.

Pages, A., Perello, J., and Spadaro, S. (2012). Lightpath fragmentation for efficient spectrum utilization in dynamic elastic optical networks. In *Optical Network Design and Modeling (ONDM), 2012 16th International Conference on*, pages 1–6.

Patel, A., Ji, P., Jue, J., and Wang, T. (2011). Survivable transparent flexible optical WDM (FWDM) networks. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, pages 1–3.

Schupke (2006). Analysis of p-cycle capacity in WDM networks. *Photonic Network Communications*, pages p. 41–51.

Shao, X., Yeo, Y.-K., Xu, Z., Cheng, X., and Zhou, L. (2012). Shared-path protection in OFDM-based optical networks with elastic bandwidth allocation. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference*, pages 1–3.

Tanaka, T., Hirano, A., and Jinno, M. (2013). Impact of transponder architecture on the scalability of optical nodes in elastic optical networks. *Communications Letters, IEEE*, 17(9):1846–1848.

Wang, Y., Cao, X., and Pan, Y. (2011). A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *Proc of IEEE,INFOCOM 2011*, pages 1503–1511.

Xiaomin, C., Jukan, A., Drummond., A. C., and da Fonseca, N. L. S. (2009). A multipath routing mechanism in optical networks with extremely high bandwidth requests. In *Proc of IEEE, Global Telecommunications Conference, 2009.*, pages 1–6.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 18*

## *Redes de Sensores e de Rádios Cognitivos*

# Uma Solução Ciente do Consumo de Energia para os Problemas de Localização 3D e Sincronização em RSSFs

**Cristiano B. Cardoso[1], Daniel L. Guidoni[2], Guilherme Maia[3], Jo Ueyama[4], Antonio A. F. Loureiro[3], Leandro A. Villas[1]**

[1]Instituto de Computacao – UNICAMP
[2]Universidade Federal de São João del-Rei
[3]Instituto de Ciências Matemáticas e de Computação – USP
[4]Universidade Federal de Minas Gerais – UFMG

C120528@dac.unicamp.br, danielguidoni@gmail.com, loureiro@dcc.ufmg.br
guilhermemm@gmail.com, joueyama@icmc.usp.br e leandro@ic.unicamp.br

***Abstract.*** *Localization and synchronization are fundamental services for many applications in Wireless Sensor Networks (WSNs), since it is often required to know the sensor nodes' position and global time to relate a given event detection to a specific location and time. However, the localization and synchronization tasks are often performed after the sensor nodes' deployment in the sensor field. Since manual configuration of sensor nodes is usually an impractical activity, it is necessary to rely on specific algorithms to solve both the problems of localizing and synchronizing the clock of sensor nodes. With this in mind, in this work we propose a joint solution for the problem of 3D localization and time synchronization in WSNs using an unmanned aerial vehicle (UAV). A UAV equipped with a GPS flies over the sensor field area broadcasting its geographical position. Therefore, sensor nodes are able to estimate their geographical position and global time without the need to equip them with a GPS device. By means of simulations we show that our proposed join solution leads to smaller time synchronization and localization errors as well as a lower energy consumption when compared to solutions found in the literature.*

***Resumo.*** *Localização e sincronização são serviços fundamentais para muitas aplicações em redes de sensores sem fio (RSSFs). Uma vez que geralmente é necessário saber a posição dos nós sensores e a hora global para relacionar uma determinada detecção de evento a um local e hora específica. No entanto, as tarefas de localização e sincronização são frequentemente realizadas após a deposição dos nós sensores na área de interesse. Uma vez que a configuração manual de nós sensores é geralmente impraticável, é necessário contar com algoritmos específicos para resolver os problemas de localização e sincronização de relógio dos nós. Em função disso, este artigo apresenta uma solução conjunta para o problema de localização 3D e sincronização de relógio em RSSFs usando um veículo aéreo não tripulado (VANT). Um VANT equipado com um GPS sobrevoa a área do campo sensoriado transmitindo a sua posição geográfica e a hora global. Assim, os nós sensores são capazes de estimar sua posição geográfica e tempo global, sem a necessidade de possuírem um dispositivo GPS. Por meio de simulações mostramos que a solução proposta apresenta menores erros de sincronização de relógio e localização além de reduzir o consumo de energia quando comparado com outras soluções encontradas na literatura.*

## 1. Introdução

Uma rede de sensores sem fio (RSSF) pode ser definida como uma rede cooperativa composta por milhares de nós sensores com recursos limitados. Estes nós são equipados com uma interface de comunicação sem fio, processador, memória e sensores. Os nós sensores têm a capacidade de coletar dados sobre propriedades físicas próximas a sua localização física, tais como temperatura, umidade, pressão, movimento e outras propriedades. As soluções para este tipo de rede deve se concentrar em um baixo consumo de energia, a fim de maximizar o tempo de vida da rede [Villas et al. , Guidoni et al. , Villas et al. 2013a].

A principal tarefa de uma RSSF é o monitoramento de fenômenos físicos e a transmissão dos dados coletados para o nó *sink*, através de uma comunicação *multi-hop*. A importância de um sistema de localização 3D e sincronização em RSSFs surge a partir da necessidade de nomear os dados coletados [Oliveira et al. 2009] e eventos associados com o seu local e tempo de ocorrencia [Intanagonwiwat et al. 2000]. Alguns algoritmos de roteamento usam as informações de localização dos nós sensores na criação das rotas com o objetivo de melhorar o seu desempenho [Villas et al. 2013b], por exemplo, reduzir os prazos de encaminhamento, número de saltos, o consumo de energia e outros. No entanto, dependendo da precisão da informação de localização, tais rotas pode não conter os nós corretos, diminuindo, assim, o desempenho dos protocolos de roteamento. Além disso, os sistemas de sincronização pode ser utilizado também para aumentar o desempenho desses protocolos. Existem alguns algoritmos de roteamento que consideram o atraso de transmissão para aumentar o desempenho de roteamento. Finalmente, alguns algoritmos consideram em seus projetos uma solução conjunta de localização e sincronização. Por exemplo, os algoritmos de roteamento que criam a sobreposição de rotas para agregar dados correlacionados em espaço e tempo [Villas et al. 2013a].

Tipicamente, os sistemas de localização e sincronização propostos para RSSFs usam uma abordagem recursiva [Oliveira et al. 2009]. Neste exemplo, uma nó estima a sua localização e o tempo global do relógio, com base nas posições e tempos de relógio recebidos de outros nós que já conhecem sua posição e o tempo global do relógio. Quando um nó é localizado no espaço e no tempo, ele transmite suas informações para auxiliar outros nós na suas estimativas. No entanto, esta abordagem têm alguns inconvenientes. Por exemplo, devido a erros no processo de estimação, depois de um nó calcular sua própria posição e tempo global de relógio, o nó propaga o erro de estimação para outros nós. Além disso, em um cenário 3D, um nó deve receber pelo menos quatro posições a partir dos nós de referência para estimar a sua própria posição. Portanto, isso pode limitar o número de nós que são capazes de calcular a sua própria posição, uma vez que uma nó pode não receber a quantidade apropriada de informações para realizar a estimativa. Finalmente, para iniciar o processo de recursividade 4-10% da rede de nós deve estar equipado com um receptor de GPS (*beacon nodos*). Esta suposição aumenta o custo da rede, uma vez que o custo de um nó *beacon* é muito mais elevado quando comparado com uma nó sem uma receptor GPS.

O objetivo deste trabalho é eliminar algumas das desvantagens descritas acima (nós *beacon* e propagação de erro) que ocorrem nas abordagens de localização e sincronização existentes. Neste trabalho, propomos uma solução conjunta para o problema de localização 3D e sincronização em redes de sensores sem Fio, a solução proposta faz de um veículo aéreo não tripulado (VANT). O VANT é equipado com uma receptor GPS e

sobrevoa o campo de sensoriamento transmitindo sua posição e tempo do relógio, permitindo os nós sensores estimarem a sua posição e relógios. A solução proposta apresenta três principais contribuições para os sistemas de localização e sincronização: (i) todos os nós da rede são capazes de estimar sua localização e o tempo global do relógio com alta precisão, uma vez que os nós recebem informação diretamente do VANT, (ii) a nossa solução é eficiente para redes esparsas e densas, ao contrário da maioria das soluções na literatura que são apropriadas para redes densas e (iii) a nossa solução reduz drasticamente o custo da rede, uma vez que é necessário apenas um nó equipado com um GPS.

O restante deste artigo está estruturado da seguinte forma. A próxima seção, apresenta a definição do problema e na Seção 3 nós fornecemos uma visão geral das abordagens existentes para a localização e sincronização em RSSFs. A Seção 4 descreve o algoritmo proposto para o problema de localização 3D e sincronização em RSSFs, enquanto os resultados de avaliação de desempenho são apresentados na seção 5. Finalmente. a Seção 6 apresenta as conclusões e trabalhos futuros.

## 2. Descrição do Problema

Considere uma RSSF composta de $n$ nós sensores com recursos limitados, com um alcance de comunicação $r_c$, espalhados em um campo 3D. Tal rede pode ser representada por um Grafo Euclidiano $G = (V, E)$, com as seguintes propriedades:

- $V = \{v_1, v_2, \ldots, v_n\}$ é o conjunto de nós sensores;
- $\langle i, j\rangle \in E$ se e somente se $v_i$ alcança $v_j$, por exemplo, a distância entre $v_i$ e $v_j$ é menor do que $r_c$;
- $w(e) \leq r_c$ é o peso da aresta $e = \langle i, j\rangle$, por exemplo, a distância entre $v_i$ e $v_j$.

Além disso, os seguintes termos são usados para descrever o estado de um nó sensor durante os processos de localização e de sincronização:

- *Nós Desconhecidos – $D$:* ós que não sabem sua localização e seus relógios não estão sincronizados;
- *Nós Referência – $R$:* nós que foram capazes de estimar suas posições e sincronizar seus relógios. Portanto, o objetivo de qualquer algoritmo de localização e sincronização é transformar *Nós Desconhecidos* em *Nós Referência* consumindo a menor quantidade de recursos da rede neste processo;
- *Nós Beacon – $B$:* nós que conhecem a sua posição física real e seu relógio está sincronizado. Esta informação é obtida por meio de um receptor GPS ou configuração manual. Estes nós são a base da maioria dos sistemas de localização e de sincronização de RSSF. Além disso, esses nós geralmente não sofrem as mesmas restrições de recursos como nós sensores comuns que precisam ser localizados e sincronizados.

Dados os termos acima, uma definição para o problema de localização e sincronização pode ser declarado como segue:

**Definição 1 (Descrição do Problema)** *Assuma uma RSSF representado por $G = (V, E)$. Além disso, suponhamos que, para todos $b \in B$ existe um conjunto de nós Beacon $B$ com posições conhecidas $(x_b, y_b, z_b)$ e relógios sincronizados $(timestamp_b)$. Portanto a solução para ambos os problemas consiste em encontrar $(x_u, y_u, z_u, clock_u)$ para o maior número de $u \in D$, assim convertendo nós Desconhecidos em Nós Referência com*

*um custo de comunicação menor $w$. Assumindo-se que a tarefa de comunicação é a tarefa com o maior consumo de energia em redes de sensores, quanto menor for o custo de comunicação do algoritmo que faz a sincronização e localização dos nós, menor será a energia consumida pelos nós sensores; aumentando assim o tempo de vida da rede.*

Uma solução simples para o problema acima mencionado em uma RSSF é equipar todos os nós sensores com um receptor de GPS. Apesar de algumas vantagens claras, tais como erros de localização e de sincronização relativamente pequenos (2–15 m e 2–10 $\mu$s dependendo do receptor GPS) e de alta precisão, uma vez que os erros seriam semelhantes para todos os nós da rede de sensores, esta abordagem também possui vários inconvenientes, tais como o aumento dos nós de sensores, a falta de visibilidade dos satélites quando usado em áreas cobertas, e, finalmente, e mais gravemente, o aumento do consumo de energia e custos de nós sensores. Portanto, esta solução torna-se prontamente impraticável para redes com centenas ou milhares de nós sensores, o que nos leva a projetar e integrar um sistema de localização e sincronização que consome a menor quantidade de recursos dos nós sensores.

## 3. Trabalhos Relacionados

Nesta seção, é apresentado algumas propostas existentes na literatura. Primeiro, apresentamos os algoritmos para o problema de localização e em seguida algoritmos para o problema de sincronização para RSSFs.

### 3.1. Algoritmos de Localização

As soluções existentes na literatura, de uma forma ou de outra, tentam melhorar o Sistema de Posicionamento *Ad Hoc* (APS) [Niculescu and Nath 2003] ou a Estimação Recursiva de Posicionamento (RPE) [Albowicz et al. 2001]. No APS, um número reduzido de nós *beacon* (pelo menos 3) são implantados no campo sensor. Cada nó *beacon* inicia uma mensagem de difusão contendo a sua posição. Em seguida, cada nó desconhecido calcula a distância usando a comunicação em múltiplos saltos de cada nó *beacon*. Uma vez que as distâncias são calculadas, os nós desconhecidos podem estimar a sua posição, usando, por exemplo, triangulação, tornando-se, assim, nós de referência . O algoritmo de RPE utiliza uma abordagem diferente. Os nós desconhecidos estiman a suas posições com base em um conjunto de nós *beacon* (normalmente de 5% a 10% do número total de nós do sensores). O algoritmo é dividido em quatro fases. Na primeira, cada nó *beacon* envia a sua posição para os seus vizinhos. Na segunda fase, quando um nó desconhecido recebe as mensagens dos nós *beacons*, ele calcula a distância de cada nó *beacon*, utilizando a técnica de *RSSI*. Na terceira fase, os nós desconhecidos estimam as suas posições com base na informação recebida. Finalmente, na quarta fase, os nós desconhecidos se tornam nós de referência e enviam as suas posições para os seus vizinhos; o que aumenta o número de posições disponíveis, convertendo um nó desconhecido em um nó de referência. A desvantagem deste algoritmo é que o erro na estimativa de posição é propagado na rede, assim, o erro aumenta no processo de estimativa para outros nós. Existem outros algoritmos na literatura para o problema de localização em redes de sensores. A maioria deles evoluiu a partir do APS e RPE concentrando-se em características específicas em cenários específicos [Galstyan et al. 2009].

No entanto, todos os algoritmos mencionados anteriormente são de natureza interativa, ou seja, um nó desconhecido deve receber as posições de referência a partir de

um nó *beacon* ou de um nó de referência para calcular a sua posição, passando esse nó desconhecido a ser um nó de referência, que continua este processo para ajudar outros nós desconhecidos. Isso aumenta o custo da comunicação de tais algoritmos, levando assim a um aumento do consumo de energia dos nós de sensores, que poderiam comprometer o tempo de vida da rede. Com isto em mente, nós propusemos uma solução na qual os nós desconhecidos precisam apenas ouvir as mensagens disseminadas por um nó móvel (VANT), a fim de calcular as suas posições.

### 3.2. Algoritmos de Sincronização

O problema de sincronização de tempo pode ser dividido em três casos: (i) tempo de sincronização relativo, que é usado para ordenar as mensagens e os eventos; (ii) relógio independente, onde um nó mantém o controle de desvio e compensação, e (iii) sincronização de tempo global, em que há um tempo global para todos os nós da rede. Neste trabalho estamos interessados no último caso. Há um número de algoritmos de sincronização disponíveis para resolver a sincronização de tempo global em RSSFs [Maróti et al. 2004, Li and Rus 2006, Lenzen et al. 2009].

No Protocolo de Sincronização de Tempo por Inundação (*FTSP*) [Maróti et al. 2004], um nó sincroniza seu relógio com base em apenas uma única mensagem. O FTSP aproveita o tempo da camada MAC para enviar uma mensagem, chamada Sincronização de Um Salto (OHS). Um nó de raiz, o qual tem um relógio global sincronizado, cria uma mensagem com o seu relógio e transmite esta mensagem para os seus vizinhos. Quando um nó não sincronizado recebe essa mensagem, ele recebe a hora global dentro da mensagem e adiciona um valor OHS pré-definido a hora recebida e depois sincroniza o seu relógio. FTSP foi avaliada em uma rede de sensores sem fio real e o OHS apresentou uma precisão de 2–4 $\mu$s numa plataforma Berkeley Mica2 [Mica2 2007].

[Lenzen et al. 2009] propõe o algoritmo *PulseSync*. A ideia é que tem um nó de referência propaga o seu relógio, tão rapidamente quanto possível na rede para sincronizar os outros nós. [Li and Rus 2006] propôs três esquemas para conseguir a sincronização do relógio global: baseado em todos os nós, baseados em difusão localizada e baseado em agrupamento. Em todos os três sistemas, o número de mensagens trocadas entre os nós é alta devido a várias mensagens de referência trocadas entre um nó e seus vizinhos. Assim, os sistemas não são escaláveis quando o número de nós aumenta na rede.

## 4. Proposta

A Figura 1 ilustra a solução proposta para os problemas de localização 3D e sincronização usando um VANT. O VANT percorre a área do campo sensoriado, onde os nós estão implantados. Durante o vôo, o VANT é responsável por transmir periodicamente a sua posição geográfica e seu tempo de relógio. Esta informação é baseada em informações do receptor GPS. Depois de receber quatro ou mais mensagens contendo a posição e o tempo de relógio do VANT, o nó é capaz de calcular a sua posição e sincronizar o seu relógio. Observe que, enquanto em um cenário 2D leva apenas três pontos de referência para calcular a posição do nó, em um cenário 3D são necessários quatro pontos de referência. A partir de agora, apresentamos todos os componentes dos sistemas de localização e de sincronização para o funcionamento da solução proposta. Além disso, mostramos a solução integrada para uma RSSF usando VANT.

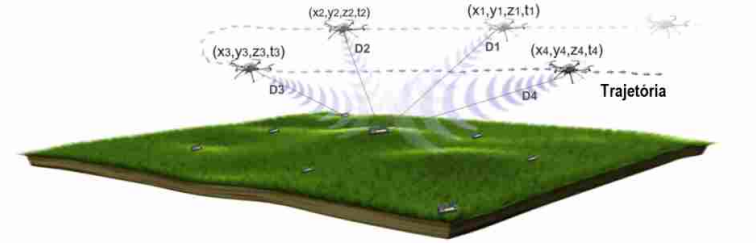



**Figura 1. Solução Proposta.**

### 4.1. Sistema de Localização

O sistema de localização pode ser dividido em duas fases: estimativa da distância e cálculo da posição. Em primeiro lugar, vamos mostrar como a nossa solução calcula a distância entre dois nós e, em seguida, vamos mostrar como ela calcula a posição dos nós.

Existem vários métodos para estimar a distância entre dois nós em uma rede de sensores [Oliveira et al. 2009]. O método mais utilizado é o RSSI, uma vez que não requer nenhum hardware adicional além de um rádio transmissor/receptor embutido no nó sensor. Ao contrário de outras técnicas de estimativa de distância [Oliveira et al. 2009], a técnica RSSI não requer nenhuma mensagem de controle para calcular a distância até o nó que transmitiu. Um nó pode calcular a distância entre ele próprio e um outro nó com base na intensidade do sinal recebido.

Como já foi dito, o nó sensor necessita de quatro pontos de referência para calcular a sua posição em um ambiente 3D. Estes pontos são fornecidas pelo VANT durante o seu vôo sobre o campo de sensoriamento. Além disso, o nó também necessita tem ciência da distância entre si e cada um dos pontos de referência. Isto pode ser alcançado através da medição da intensidade do sinal de recepção (RSSI) de cada transmissão do VANT. Finalmente, quando o nó sensor tem, pelo menos, quatro pontos de referência e a distância de cada ponto, o nó é capaz de calcular a sua posição. Multilateração é o método mais comum usado para estimar a posição quando um nó possui quatro ou mais pontos de referência.

No método multílateração, temos um sistema de pelo menos quatro equações e três variáveis em cada uma, coordenadas $(x, y, z)$. Cada equação do sistema é contruído a partir das posições de referência recebidos e as respectivas distâncias. O sistema de equações pode ser representada como segue:

$$
\begin{aligned}
(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 &= d_1^2 \\
(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 &= d_2^2 \\
&\vdots \\
(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2 &= d_n^2
\end{aligned}
$$

onde $(x_i, y_i, z_i)$ e $d_i$ são, respectivamente, uma posição de referência e a distância estimada para esta posição de referência calculado pela técnica de RSSI. Este sistema de equações é linearizado por subtração da última equação, que é $(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2 = d_n^2$ do outro. Uma vez linearizado, temos um sistema linear, que pode ser resolvido. Neste trabalho o método dos mínimos quadrados [Golub and Loan 1996] foi utilizado para resolver este sistema linear, uma vez que este é um simples e de baixo custo,

que são fatores importantes na concepção de soluções para redes de sensores. A solução do sistema linear representa a posição $(x, y, z)$ do nó que realizou os passos descritos.

### 4.2. Sistema de Sincronização

Alguns dos protocolos de sincronização são baseados em emissor-receptor e outros são baseados receptor-receptor [Sundararaman et al. 2005]. Neste trabalho, utilizamos um protocolo baseado emissor-receptor, que reduz o número de mensagens para calcular a sincronização do relógio. Em um projeto emissor-receptor, existem diferentes maneiras de executar a sincronização de tempo. Alguns protocolos usam uma comunicação em dois sentidos para descobrir a diferença entre os relógios e, assim, sincroniza-los. No entanto, como o objetivo deste trabalho é propor uma solução integrada para os problemas tanto de localização e de sincronização, comunicação de duas vias não se aplica ao nosso caso. Em vez disso, usamos uma forma de comunicação, em que o remetente envia apenas uma mensagem e o receptor é capaz de sincronizar seu relógio com base em uma informação de tempo.

Para este efeito, foi utilizado o conceito descrito no Protocolo de Sincronização de Tempo por Inundação (FTSP), onde os nós da rede sincronizam seus relógios usando comunicação unidirecional. Para sincronizar um relógio usando comunicação unidirecional, um nó deve calcular os seguintes valores: tempo de envio, o tempo de acesso MAC, o tempo de propagação e o tempo de recepção. O tempo de envio indica o tempo para criar uma mensagem para transmiti-la à rede. O tempo de recepção indica o tempo necessário para receber uma mensagem e transmiti-la para o host. Este requisito de tempo pode ser afrouxado se a hora é anexada à mensagem na camada MAC, antes da sua transmissão. O tempo de propagação pode ser facilmente calculado para um determinado modelo de propagação. Finalmente, o tempo de acesso MAC é o que é difícil de calcular, uma vez que epende do tráfego e outros parâmetros de rede. No entanto, se o algoritmo de sincronização é executada durante a iniciação da rede, podemos agendar a execução do processo de sincronização antes das outras funções concorrentes de rede, uma vez que outras tarefas, tais como protocolos de roteamento, são baseados no processo de sincronização. Neste caso, o tempo de acesso MAC está entre $2\,\mu s$ e $10\,\mu s$ [Maróti et al. 2004].

### 4.3. Solução Conjunta Proposta

Tal como ilustrado na Figura 2, as operações da solução proposta são divididas em duas fases. A primeira delas refere-se ao VANT que atravessa o campo de sensores. A segunda fase está relacionada com o cálculo da posição e a sincronização do relógio. Assim que um nó recebe uma mensagem do VANT, ele calcula a sua distância do VANT, utilizando a técnica de RSSI e armazena a posição e a hora do VANT. Quando um nó tem, pelo menos, quatro mensagens, já é capaz de calcular a sua posição em 3D e sincronizar o seu relógio. Como um nó recebe quatro valores de tempo, ele usa todos os valores recebidos para diminuir o erro de estimação, a fim de sincronizar seu relógio. A Figura 2 mostra as principais etapas da solução proposta. A primeira fase é descrita no retângulo pontilhado na esquerda (plano de vôo) e a segunda fase (estimativa da posição e sincronização do relógio) é descrita no retângulo pontilhado a direita.

Um plano de vôo contém a rota da aeronave, que é previamente projetada com base na rede. Após a decolagem, o algoritmo agenda a transmissão da posição e hora do VANT. Enquanto o fim do percurso não é atingido, o algoritmo de retorna o próximo

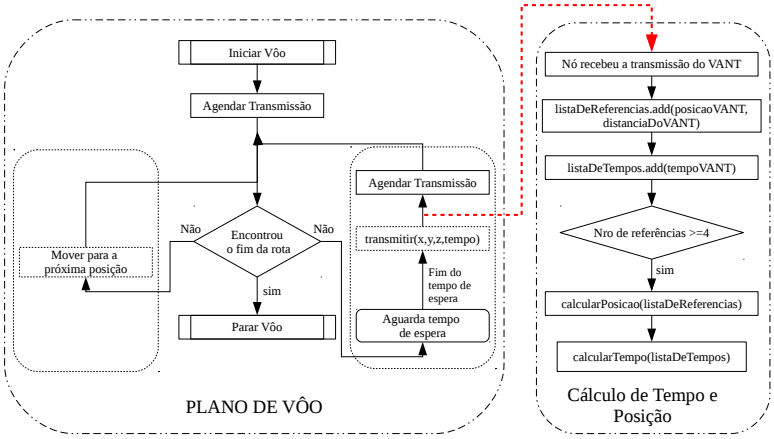



**Figura 2. Principais componentes da solução proposta.**

ponto onde o VANT deve mover-se, que se desloca até ponto específicado, a uma certa velocidade. É importante destacar que o VANT realiza uma transmissão periódica em paralelo com o seu deslocamento sobre o campo de sensor.

Quando um nó recebe uma mensagem do VANT, ele calcula a distância que está do VANT usando a técnica de RSSI. Depois disso, o nó armazena a distância e a posição relativa do VANT em *referenceSet*. O *timestamp* do VANT é armazenado em *stampSet*. Se o número de posições recebidas é maior ou igual a quatro, o nó é capaz de calcular a sua posição e sincronizar o seu relógio. Para calcular a sua posição, o nó usa o método dos mínimos quadrados descrito acima, para calcular o tempo local, o nó e faz uma média de todas as marcas de tempo recebidos. Além disso, para cada *timestamp* recebido, a função adiciona um erro de sincronização de um salto (OHS), pré-definido, que é o erro relacionado com o tempo de acesso MAC e tempo de propagação.

## 5. Avaliação de Desempenho

### 5.1. Metodologia

A solução proposta para o problema de localização 3D e sincronização é comparada com algumas soluções encontradas na literatura, no entanto, cada uma resolve cada problema individualmente. Para fazer uma comparação justa, nós combinamos uma solução de localização com uma solução de sincronização da literatura. Avaliamos a facilidade com que essas soluções podem ser colocadas juntas para resolver os problemas de localização 3D e sincronização. Identificamos que o algoritmo de Estimativa de Posição Recursiva (RPE) e o Protocolo de Sincronização por Inundação (FTSP) foram os mais apropriados. Conforme descrito na Seção 3, o FTSP usa comunicação unidirecional para sincronizar o relógio do nó. Desta forma, quando um nó *beacon* ou referência envia a sua posição no algoritmo de RPE, adicionamos a hora global do nó na mensagem de localização. Com base nessas informações, o nó pode estimar sua posição e sincronizar seu relógio. É importante salientar que um nó desconhecido torna-se um nó de referência, quando se calcula a sua posição e hora. Isso só pode ser feito depois de receber quatro mensagens, uma vez que estamos estudando o problema de localização 3D. Neste caso, o nó irá sincronizar o seu

**Tabela 1. Parâmetros de simulação.**

| Parametros | Valores |
|---|---|
| Numero de nós | 250 to 2000 |
| Densidade | 15 to 50 |
| Alcance de comunicação dos nós | 50 m |
| Alcance de comunicação do VANT | 50 m |
| Velocidade do VANT | 10 m/s |
| Erro de RSSI (%) | 0 to 20 |
| Erro de OHS ($\mu$s) | 0 to 30 |
| RPE-FTSP | 25 to 200 nós *beacon* |
| Área de Monitoramento ($x$ and $y$) | $x = y = \frac{n \times \pi \times r_c^2}{Density}$ |
| Terreno ($z$) | 0 to 10 m |
| Altura do VANT | 20 to 50 m |
| Broadcast msg interval (UAV) | 1/second |
| Concumo de energia na transmissão | 0.08 J |
| Consumo de energia na recepção | 0.02 J |
| Tamanho do pacote | 568 bits |

relógio quando tenha informações suficientes para calcular a sua posição. Esta estratégia combinada conduz à solução RPE-FTSP.

O principal objetivo da nossa avaliação de desempenho é avaliar o algoritmo integrado proposto, considerando as seguintes métricas importantes: (i) comsumo de energia, (ii) erro de estimativa de posição, (iii) erro de sincronização ($\mu$s) e (iv) número de nós que não foram capazes de estimar suas posições e sincronizar seus relógios. Para fazer isso, nós variamos dois parâmetros de rede importantes: (i) Número de nós da rede e (ii) a densidade da rede. Para a realização dessas avaliações, foi utilizado o plano de vôo[SBRC2012]. Os parâmetros de simulação são apresentados na Tabela 1. O alcance de comunicação dos nós sensores e do VANT é 50 m, esses valores foram usados para ter uma comparação justa com os algoritmos da literatura. O consumo de energia necessária para transmitir um pacote é 0.08 J (Joules) e o consumo de energia necessária para receber um pacote é 0.02 J. O comprimento do pacote é composto pelo cabeçalho do pacote e sua carga útil somando 568 bits. O cabeçalho do pacote tem 440 bits e tem a carga útil de 128 bits, o que é dividido em quatro valores de 32 bits (um ponto flutuante de quatro bytes) que estão na posição $(x, y, z)$ e tem um *timestamp* (baseados no nó MicaZ [MicaZ 2009]).

Introduzimos um erro na técnica de RSSI, que varia de 0 a 20%da distância a partir do remetente. O número de nós *beacons* na solução integrada RPE-FTSP varia de 25 a 200. Os nós *beacon* estão equipados com um receptor de GPS. É importante notar que, na nossa solução integrada, o VANT só está equipado com um receptor GPS. Para calcular a área de monitoramento $(x, y)$, utilizamos o número de nós $(n)$ e o alcance de comunicação $r_c$ dos nós sensores. A terceira dimensão $(z)$ para cada nó éum número aleatório entre 0 e 10 m. A altitude do VANT é um número aleatório entre 20 e 50 m, que muda a cada nova direção ao atravessar a área de monitoramento.

Usamos o simulador SinalGo v.0.75.3 [Sinalgo 2008] para avaliar os algoritmos. Cada cenário foi repetido 33 vezes, com diferentes sementes para a geração de números aleatórios. Em todos os resultados, as curvas representam os valores médios, enquanto que as barras de erro representam o intervalo de confiança de 95%.

### 5.2. Consumo Total de Energia

Nesta seção, nós avaliamos o consumo total de energia, tanto para RPE-FTSP como para a solução conjunta proposta. Para realizar esta avaliação, fixamos o número de nós em 500

e a densidade da rede em 30. O objetivo desta análise é verificar o consumo de energia dos nós que executam os algoritmos de localização e sincronização. Para esta avaliação, a solução integrada RPE-FTSP não consideramos o consumo de energia dos nós *beacon*, uma vez que esses nós gastarão mais energia devido ao receptor GPS. Neste caso, os nós *beacon* podem ser equipados com mais reservas de energia. Em nossa solução, nós também não consideram o consumo de energia do VANT.

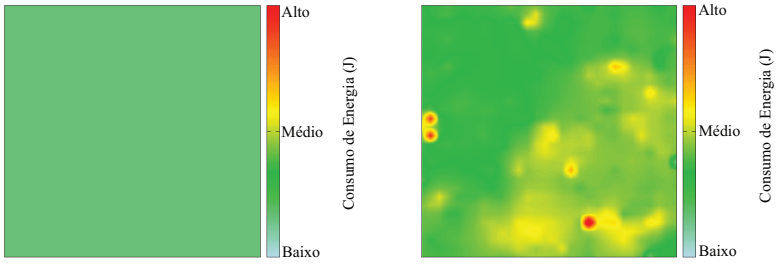
A Figura 3 mostra o mapa do consumo de energia para os dois algoritmos, considerando um número diferente de nós *beacon* no algoritmo RPE-FTSP. O mapa de energia é obtido depois de executar os algoritmos e considerando que os nós *beacon*são implantados aleatoriamente na rede. A Figura 3(a) mostra o mapa de energia para a nossa solução conjunta. Podemos verificar que o consumo de energia da solução proposta é homogêneo para todos os nós da rede. Este resultado pode ser explicado pelo fato de que o VANT transmite a sua posição para todos os nós, enquanto que atravessa a área de sensores. Alem disso, um nó sensor não comunica com os seus vizinhos para calcular a sua posição e para sincronizar o seu relógio. As Figuras 3(b), 3(c) e 3(d) mostram o mapa de energia considerando o RPE-FTSP para um número diferente de nós *beacon*. É importante salientar que o consumo de energia do RPE-FTPS não é homogênea entre os nós, o que pode fazer com que alguns nós fiquem sem energia mais cedo do que outros. Além disso, quando incrementamos o número de nós *beacon*, o consumo de energia também aumenta. Isto é devido ao fato de o algoritmo de RPE-FTSP depender de (i) a comunicação entre os nós de sensores, (ii) o número de nós *beacon*, e (iii) das posições dos nós *beacon* na área de monitorização. Quando o número de nós *beacon* é pequeno, menos as mensagens são transmitidas na rede, levando assim a um consumo de energia menor. Este fato não é observado quando aumentamos o número de nós *beacon*. Quando a rede tem mais nós *beacon*, um nó desconhecido receberá mais mensagens do que o necessário para estimar a sua posição e sincronizar seu relógio, assim, causando um grande impacto sobre o consumo de energia. É importante notar que, mesmo quando o número de nós *beacon* é pequeno, se eles estão geograficamente próximos um do outro, um nó desconhecido vai gastar mais energia, recebendo todas as mensagens de difusão a partir dos *beacon*.

Nas seções seguintes, analisamos, entre outras métricas, o consumo de energia. Levando em consideração o impacto do número de nós e densidade da rede.

### 5.3. Número de Nós

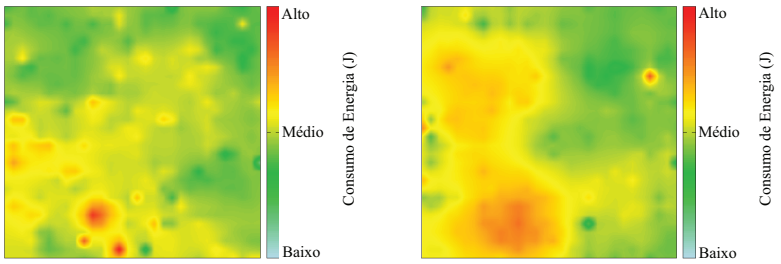
Nesta seção, nós avaliamos as soluções para um número diferente de nós na rede. Para esta análise, fixamos a densidade da rede a 30, o erro RSSI a 5 % eo OHS para 5 $\mu$s.

A Figura 4(a) mostra a estimativa de erro de posição. O sistema proposto tem um pequeno erro na estimativa de posição e o erro não é afetado pelo número de nós, o que não é observado no algoritmo de RPE-FTSP . O erro de estimativa de posição no RPE-FTSP é cerca de três vezes maior quando comparado com a nossa proposta e aumenta quando aumentamos o número de nós. Isso acontece quando fixamos o número de *beacons* e aumentamos o número de nós. Os nós desconhecidos estima a sua posição com base nos nós de referência, os quais têm uma posição estimada. Assim, o erro de estimativa espalha na rede. Podemos também observar que, quando aumenta o número de nós *beacon*, o erro de estimativa diminui, uma vez que os nós desconhecidos irão estimar a sua posição utilizando posições dos nós *beacons*. É importante salientar que,




(a) Solução proposta

(b) RPE-FTSP com 50 nós *beacon*




(c) RPE-FTSP com 100 nós *beacon*

(d) RPE-FTSP com 150 nós *beacon*

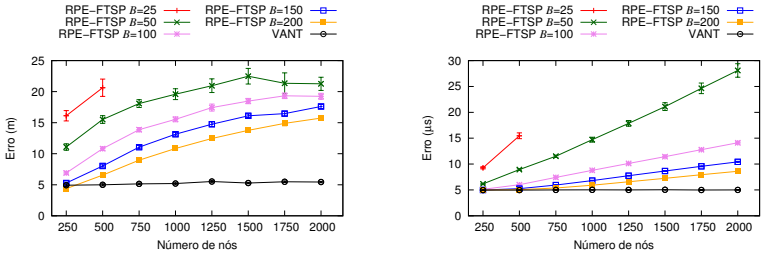
**Figura 3. Consumo total de energia.**

quando temos apenas 25 nós *beacon*, o algoritmo de RPE-FTSP não é capaz de estimar qualquer posição quando $n > 500$ . A principal desvantagem do uso de diversos nós *beacons* é o custo da rede, que aumenta substancialmente devido aos receptores GPS. Além disso, quando os problemas de localização e de sincronização são resolvidos, os nós *beacon* tornam-se inúteis, uma vez que este processo é executado apenas uma vez durante o tempo de vida da rede.

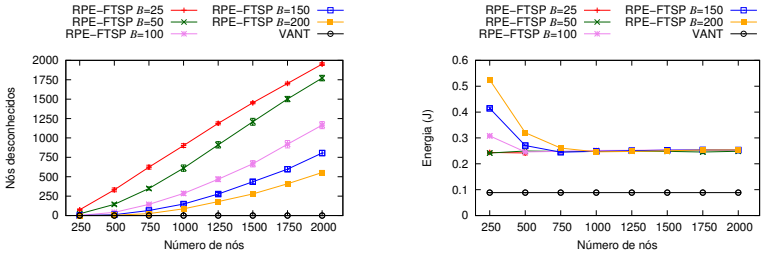
O erro de sincronização é mostrado na figura 4(b). Quando aumentamos o número de nós da rede, o erro de sincronização do algoritmo de RPE-FTSP também aumenta. Pelo mesmo motivo discutido acima. Quando a rede tem 2000 nós, o erro de sincronização do algoritmo de EPR-FTSP é 1,89 vezes maior do que a da solução proposta (quando $B = 200$). Nós também podemos ver que o sistema de sincronização proposto não é afetado pelo número de nós.

A Figura 4(c) mostra o número de nós desconhecidos. Um nó é rotulado como desconhecido quando não recebeu informações suficientes para estimar a sua posição/hora ou quando o erro de posição é maior do que o seu alcance de comunicação. Podemos observar que, quando a rede tem poucos nós *beacon*, o número de nós desconhecidos é aumenta quando o número de nós de rede aumenta. No entanto, quando a rede tem cerca de 750 nós, o número de nós desconhecidos é baixo. É importante salientar que, como o VANT atravessa toda a área de sensores, todos os nós da rede são capazes de calcular a sua posição e sincronizar seus relógios.

O consumo de energia é ilustrado na Figura 4(d). Podemos verificar que, quando a rede tem entre 250 e 500 nós, o número de nós *beacon* tem um grande impacto sobre o




(a) Erro de posição (b) Erro de sincronização




(c) Nós desconhecidos (d) Consumo de energia

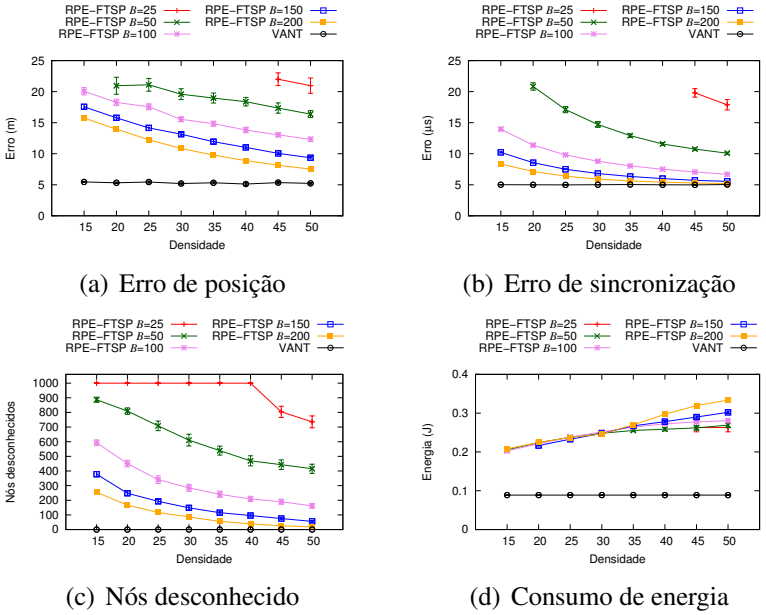
**Figura 4. Número de nós.**

consumo de energia. Como foi descrito acima, quando a rede tem um pequeno número de nós de sensores e um elevado número de nóa *beacon*, um nó desconhecido recebe mais mensagens do que a quantidade necessária para calcular a sua posição e relógio. Quando o número de nós é maior do que 750, o número de nós *beacon* não tem impacto sobre o consumo de energia, uma vez que a área de monitoramento é suficientemente grande para diminuir a densidade de *beacons* na área de monitoramento. Por exemplo, quando o número *beacon* é maior do que 750, o consumo de energia do EPR-FTSP é mais do que 2,5 vezes maior quando comparada com a solução proposta.

## 5.4. Densidade da Rede

Esta secção avalia os algoritmos para diferentes densidades de rede. O número de nós de rede é 1000, o erro de RSSI é 5% e o OHS é 5 $\mu$s. A Figura 5(a) mostra o erro no processo de estimativa de posição. Podemos observar que quanto maiores os valores para a densidade da rede, melhor será o desempenho RPE-FTSP. Isto é devido ao fato de que, quando aumenta a densidade da rede para um número fixo de nós, a área monitorada diminui. Neste caso, o erro de estimativa a posição não se espalha para muitos nós. Nossa solução, que usa um VANT, não é afetado pela densidade da rede, uma vez que o VANT atravessa toda a área de monitoramento.

O mesmo comportamento é observado no problema de sincronização, uma vez que ambos os algoritmos são executados em conjunto (Figura 5(b)). É importante notar que para altos valores de densidade de rede, não há diferença entre a nossa abordagem e o algoritmo de RPE-FTSP. A Figura 5(c) mostra os nós desconhecidao. Tal como esperado, quando se aumenta a densidade da rede, o número de nós desconhecidos diminui rapidamente, isto quando o número de nós *beacon* é maior do que 50. Para uma rede com elevada densidade e um grande número de nós *beacon*, os nós desconhecidos são menos de 10% dos nós. Na nossa proposta, o número de nós desconhecidos é zero, uma vez que o VANT atravessa toda a área monitorada, transmitindo a sua posição e tempo de relógio.




(a) Erro de posição (b) Erro de sincronização

(c) Nós desconhecido (d) Consumo de energia

**Figura 5. Densidade.**

A Figura 5(d) mostra o consumo de energia para diferentes densidades de rede. Quando a densidade da rede é até 30, o consumo de energia no RPE-FTSP não é afetado pelo número de nós *beacon*. Para densidades de rede superior a 30, quando aumentamos o número de nós *beacon*, o consumo de energia também aumenta. Quando a densidade da rede é de 50, o consumo de energia, considerando-se 200 nós *beacon*, é 1,27 vezes maior do que com 25 nós *beacon*. É importante notar que a solução proposta não é afectada pela densidade de rede. Considerando 200 nós *beacon*, o erro na estimativa posição do RPE-FTSP é apenas 1,5 vezes maior. Quando comparados com a solução proposta, o erro de sincronização é o mesmo, mas o consumo de energia é de 3,75 vezes maior.

## 6. Conclusões

A localização e sincronização em redes de sensores sem fio são serviços fundamentais para muitas aplicações. Há uma grande classe de aplicações que necessitam adicionar as informações de localização e de tempo nos dados coletados. Tipicamente, nas abordagens encontradas na literatura, o erro de estimativa depende da quantidade de nós *beacon* implantado no campo monitorado. Além disso, esses nós *beacon* aumentam significativamente o custo da rede. Neste trabalho , propomos uma solução conjunta para os problemas de localização 3D e sincronização em RSSFs usando um veículo aéreo não tripulado. Os resultados das simulações mostram que a solução proposta apresenta menores erros de sincronização e localização em relação às soluções existentes. Além disso, a eficiência de nosso solução é independente do número de nós na rede, o que é um aspecto importante no caso de escalabilidade. Finalmente, em nossa solução, todos os nós sensores são capazes de calcular o tempo global e sua posição. Como trabalho futuro pretende-se considerar diferentes planos de voo e realizar experimentos em um ambiente real.

## 7. Agradecimentos

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

## Referências


Albowicz, J., Chen, A., and Zhang, L. (2001). Recursive position estimation in sensor networks. In *The 9th International Conference on Network Protocols*, pages 35–41.

Galstyan, A., Krishnamachari, B., Lerman, K., and Pattem, S. (2009). Distributed online localization in sensor networks using a moving target. In *Third International Symposium on Information Processing in Sensor Networks*, pages 61–70.

Golub, G. H. and Loan, C. F. V. (1996). *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, third edition.

Guidoni, D. L., Boukerche, A., Villas, L. A., de Souza, F. S., Oliveira, H. A., and Loureiro, A. A. A small world approach for scalable and resilient position estimation algorithms for wireless sensor networks. In *Proceedings of the MobiWac '12*, pages 71–78, New York, NY, USA. ACM.

Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67.

Lenzen, C., Sommer, P., and Wattenhofer, R. (2009). Optimal clock synchronization in networks. In *SenSys '09*, pages 225–238.

Li, Q. and Rus, D. (2006). Global clock synchronization in sensor networks. *IEEE Trans. Comput.*, 55(2):214–226.

Maróti, M., Kusy, B., Simon, G., and Lédeczi, A. (2004). The flooding time synchronization protocol. In *SenSys '04*, pages 39–49.

Mica2 (2007). Mica2, crossbow technology.

MicaZ (2009). Micaz crossbow technology.

Niculescu, D. and Nath, B. (2003). Ad hoc positioning systems (aps) using oao. In *IEEE INFOCOM*, pages 1734–1743.

Oliveira, H. A. B. F., Boukerche, A., Nakamura, E. F., and Loureiro, A. A. F. (2009). Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm. *Perform. Eval.*, 66(3-5):209–222.

Sinalgo (2008). Simulator for network algorithms. Distributed Computing Group.

Sundararaman, B., Buy, U., and Kshemkalyani, A. D. (2005). Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281 – 323.

Villas, L., Boukerche, A., Guidoni, D. L., Oliveira, H. A., Araujo, R. B., and Loureiro, A. A. Time-space correlation for real-time, accurate, and energy-aware data reporting in wireless sensor networks. In *Proceedings of the MSWIM 2011*, pages 59–66, New York, NY, USA. ACM.

Villas, L., Boukerche, A., Ramos, H., de Oliveira, H., de Araujo, R., and Loureiro, A. (2013a). Drina: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. *Computers, IEEE Transactions on*, 62(4):676–689.

Villas, L., Guidoni, D., and Ueyama, J. (2013b). 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.

# Decisão de Espectro em Redes de Sensores Sem Fio Empregando Aprendizado de Máquina

**Vinicius F. e Silva**[1], **Daniel F. Macedo**[1], **Jesse L. Leoni**[1]

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

{viniciusfs,damacedo}@dcc.ufmg.br,jleoni@ufmg.br

***Abstract.*** *Wireless Sensor Networks (WSNs) employ ISM spectrum bands for communication, which are overloaded due to various technologies such as WLANs and other WSNs. Therefore, such networks must employ intelligent methods such as Cognitive Radio to coexist with other networks. This study evaluates the use of Supervised Machine Learning (ML) for channel selection in WSNs, considering the channel quality and communication metrics. The methods were evaluated experimentally and compared with energy-based methods. The results show that ML-based methods increase the communication performance by reducing the number of transmission attempts and therefore also reducing the delivery delay.*

***Resumo.*** *Redes de Sensores Sem Fio (RSSFs) utilizam faixas de espectro ISM, sobrecarregadas devido a diversas tecnologias como WLANs e outras RSSFs. Por isso, tais redes devem empregar métodos inteligentes, como por exemplo de Rádio Cognitivo, de forma a coexistir com outras redes. Este trabalho avalia o uso de métodos baseados em Aprendizado de Máquina (AM) Supervisionado para a seleção de canal em RSSFs, considerando métricas de qualidade do canal e de comunicação. Os métodos foram avaliados experimentalmente, e comparados com métodos baseados em energia. Os resultados mostram que métodos baseados em AM aumentam o desempenho da comunicação, reduzindo o número de tentativas de transmissão e com isso o atraso no envio de quadros.*

## 1. Introdução

Um estudo da FCC (*Federal Communications Commision*) [FCC 2002] mostrou que as regras atuais adotadas para alocação de canal causam perdas consideráveis na eficiência e aproveitamento do espectro, pois em inúmeros intervalos de tempo os usuários licenciados ao uso desses canais não estão em operação [Yucek and Arslan 2009, Zeng et al. 2010]. Além disso, o acúmulo de usuários não licenciados em frequências livres gera interferências, causando perdas de informação e um aumento no consumo de energia. Redes de Sensores Sem Fio (RSSFs) são suscetíveis a esse problema, o que leva à redução do seu tempo de vida.

Por isso, as RSSFs deveriam adotar técnicas que explorem o espectro com mais eficiência, de forma a coexistir com usuários de diferentes redes sem causar conflitos entre elas. Os Rádios Cognitivos (RCs) permitem uma melhor exploração de faixas de frequência pouco utilizadas [Zeng et al. 2010]. Neste trabalho seguimos a definição de [FCC 2005], onde o RC mensura o ambiente eletromagnético no qual ele opera, para

ajustar seus parâmetros de forma autônoma e dinâmica, de forma a aumentar o fluxo de dados, reduzir a interferência e facilitar a interoperabilidade.

No caso das RSSFs, que empregam a faixa de frequência não licenciada ISM (*Industrial, Scientific and Medical*), as funções de RC são utilizadas para aumentar a coexistência de múltiplas redes, ao mesmo tempo que garantem uma comunicação mais eficiente e confiável [Sherman et al. 2008]. Enquanto nos RCs "tradicionais" procura-se selecionar faixas de frequência livres de usuários licenciados (também chamados de *Usuários Primários*), nas RSSFs os RCs selecionam canais com melhores condições para a comunicação, evitando a interferência tanto de outras RSSFs instaladas na região quanto de outros tipos de redes operando na faixa ISM (Wi-Fi, Bluetooth, etc). Entretanto, o principal desafio na implementação de RCs em RSSFs é a limitação de recursos de processamento, memória e energia destas redes. Assim, as soluções de RC existentes devem ser adequadas às restrições das RSSFs.

Neste trabalho propomos o emprego de Aprendizado de Máquina (AM) para a seleção de canais em RSSFs. As soluções propostas são avaliadas experimentalmente, comparando o seu desempenho com soluções baseadas somente em energia, as quais são populares em ambientes reais. Foram coletados dados de desempenho acerca de uma sequência de transmissões e recepções realizadas, como forma de quantificar o nível de ocupação de cada canal. A partir dos dados coletados, foram construídas bases de treinamento que permitiram a criação de modelos de previsão de baixo consumo de memória e processamento, tornando possível sua implementação em uma RSSF real. Por meio de testes práticos, observamos que o uso de AM melhorou o desempenho da RSSF, reduzindo a quantidade de tentativas de transmissão, a quantidade de colisões e o atraso sobre os pacotes transmitidos, em cenários com interferências de outras redes.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta os conceitos envolvidos e os trabalhos relacionados. A modelagem do problema é apresentada na Seção 3, e a coleta dos dados para as bases de treinamento é apresentada na Seção 4. A Seção 5 apresenta uma análise da acurácia dos algoritmos de AM, seguida da avaliação dos melhores algoritmos em uma plataforma real na Seção 6. A Seção 7 apresenta as conclusões e trabalhos futuros.

## 2. Conceitos e Trabalhos Relacionados

As características de um RC podem ser divididas em quatro partes [Akyildiz et al. 2009]. O *Sensoreamento de Espectro* obtém informações sobre um conjunto de canais, detectando os intervalos de tempo não utilizados, bem como os canais menos utilizados e portanto mais propícios ao uso pelos RCs. A *Decisão de Espectro* analisa as informações obtidas na fase de Sensoreamento e determina os melhores canais a serem utilizados com base em um ou mais critérios de Qualidade de Serviço (*Quality of Service - QoS*). O *Compartilhamento de Espectro* aloca os canais selecionados e coordena as transmissões entre os diversos usuários, de forma a minimizar as colisões e a interferência causada sobre os usuários primários. Já a *Mobilidade de Espectro*, em canais não reservados como nas RSSFs, estima as condições futuras da rede, identificando se uma troca de canal é necessária. Neste trabalho o foco principal é na *Decisão de Espectro*.

Em RCs, as técnicas de AM são empregadas para otimizar métricas de *QoS*, ou parâmetros provenientes de uma ou mais camadas da pilha de protocolos. Uma das prin-

cipais vantagens no uso de algoritmos de AM é a sua capacidade de modelar o comportamento da rede em intervalos futuros, permitindo a aplicação de soluções em tempo hábil, como a troca de canal [Bantouna et al. 2012]. Em aplicações onde não é possível obter informações do meio em tempo real, são usadas técnicas alternativas como os modelos estatísticos de uso do espectro, propostos por [Zhao and Swami 2007].

Existem três abordagens em AM [Katidiotis et al. 2010]. O *Aprendizado Supervisionado* busca construir as regras de decisão com base em exemplos conhecidos e bem definidos. O *Aprendizado não supervisionado* não dispõe de uma base inicial de conhecimento, de forma que o sistema refina durante a sua operação o seu processo de decisão. O *Aprendizado semi supervisionado* realiza o aprendizado tanto com uma base de treino, antes da ativação do sistema, quanto a cada execução do sistema em operação.

Em aprendizado supervisionado, uma das técnicas mais utilizadas, devido à sua modelagem simples, são as Redes Neurais Artificiais (RNA). Em [Tsagkaris et al. 2008] são propostas estruturas de RNA para predição da vazão. Foi mostrado que a acurácia da predição pode mudar consideravelmente em função dos parâmetros considerados. Um exemplo é RNA desenvolvida por [Baldo et al. 2009], onde é considerada a variabilidade do tráfego ao longo do dia e da semana. Estes trabalhos, entretanto, não avaliaram a viabilidade da obtenção dos parâmetros de entrada e tomada de decisão em tempo hábil.

Existem poucos trabalhos em RCs que empregam aprendizado não supervisionado. Em [Tsagkaris et al. 2012], são apresentados os Mapas Auto-Organizáveis (*Self-Organizing Maps* - SOMs). SOMs coletam amostras multi-dimensionais e, a partir delas, constroem uma rede composta por neurônios cujas distâncias entre si determinam a sua similaridade. Neurônios próximos entre si, ou seja, cuja distância entre os valores dos atributos é reduzida, definem um *cluster*, o qual representa uma classe. A principal desvantagem foi a alta quantidade de memória necessária para sua execução, o que pode inviabilizar a aplicação de SOMs em RSSFs.

Em [Unnikrishnan and Veeravalli 2010] o sensoreamento e acesso ao espectro são tratados como um Processo de Decisão de Markov Parcialmente Observado, onde o uso dos canais não é conhecido de forma explícita pelos usuários não licenciados (também chamados de *Usuários Secundários*). Uma vez que o usuário acessa um canal livre, o mesmo recebe uma recompensa proporcional à largura de banda do canal acessado. De forma contrária, o canal sofre penalização caso esteja ocupado. [Felice et al. 2010] apresentam um conjunto de técnicas envolvendo aprendizado por reforço (*Reinforcement Learning - RL*) que podem ser aplicadas em RCs.

A principal desvantagem destas técnicas é a dependência de funções de recompensa e penalização que sejam altamente precisas. Muitas RSSFs, por possuírem *hardware* limitado, não conseguem realizar os cálculos exigidos por essas funções em tempo viável ou com a precisão necessária. Este trabalho procura adotar algoritmos de AM que, mesmo com a precisão de valores reduzida, oferecem desempenho adequado.

Fora do escopo de AM, é possível encontrar na literatura recente diferentes propostas de métodos de decisão de espectro para seleção de canal. No trabalho de [Masonta et al. 2013], são apresentados e descritos os estudos mais recentes de RCs para esse propósito. São também apresentados estudos relacionados à caracterização do canal, considerada pelos autores como uma etapa anterior à seleção de canal, e reconfiguração,

considerada uma etapa posterior, ambas inerentes à decisão de espectro. Os autores fazem uma divisão entre trabalhos com decisão centralizada ou não centralizada.

Em RSSF, [Correia et al. 2012] propõem dois métodos de decisão de espectro para seleção de canal. O primeiro deles toma como base medições de RSSI (*Received Signal Strength Indicator*) e resultados apresentados pelo algoritmo CCA (*Clear Channel Assessment*) em cada canal analisado. Já o segundo método toma como base informações provenientes da camada física e requisitos da aplicação em execução, os quais são ponderados por meio de entropia.

O uso de AM tem sido cada vez mais comum em algoritmos de sensoreamento e decisão de espectro. Entretanto, há uma deficiência na literatura de trabalhos que avaliem experimentalmente o uso de técnicas de AM em RSSFs. Trabalhos recentes, como o de [Thilina et al. 2013], seguem uma linha de pesquisa semelhante à aqui adotada, propondo o uso de diferentes algoritmos de AM supervisionado e não supervisionado em RC, porém, os resultados não são validados em uma rede real. O presente trabalho avalia diversos algoritmos de AM e implementa os mais eficazes em uma plataforma comercial de RSSFs, os quais serão utilizados como preditores no algoritmo de decisão de espectro proposto. Nossos preditores tomam como base o desempenho médio obtido em uma sequência de transmissões e recepções em cada um dos canais de operação, de forma a escolher o melhor canal.

## 3. Rádios Cognitivos Empregando AM

Esta seção apresenta a modelagem da escolha do melhor canal em RSSF como um problema de AM. Construímos um preditor baseado em regressão que procura aproximar o número de tentativas de transmissão necessário pra uma sequência finita de pacotes. O canal escolhido é aquele com o menor número de tentativas de transmissão esperado. Esse processo é executado periodicamente. Os seguintes atributos foram selecionados como entrada do preditor:

- **RSSI**: Medido em $dBm$, quantifica a potência com a qual o sinal do meio é recebido pelo rádio em um determinado instante. Para este atributo, é coletada uma quantidade fixa de amostras imediatamente antes de cada transmissão realizada. Aqui, o *RSSI* varia entre -91$dBm$ e -10$dBm$, com granularidade de 3$dBm$.
- **Número de Tentativas de Transmissão Total**: Quantidade de tentativas realizadas para se transmitir um pacote, com sucesso ou não, a um determinado destinatário da rede. Como determinado na implementação padrão do rádio RF230 no TinyOS, o número máximo de tentativas de transmissão é 10;
- **Contagens das Razões de cada Tentativa Realizada**: Em caso de uma ou mais falhas na transmissão de um determinado pacote, também serão contabilizadas as suas razões, definidas abaixo:
    1. **ERX**: O rádio estava ocupado com a recepção de outro pacote;
    2. **ECCA**: O algoritmo CCA (*Clear Channel Assessment*), implementado em *hardware*, indicou que o canal estava ocupado (detecção de portadora);
    3. **ENACK**: Falha ocorrida em razão do rádio ter transmitido um pacote e a sua confirmação (pacote *ACK - Acknowledgement*) não ter sido recebida. Esta métrica captura as colisões e erros de bit.

- **Dados do último pacote recebido**: São coletados a cada pacote recebido o seu RSSI e o LQI (*Link Quality Indication*). O LQI é uma métrica definida pelo padrão IEEE 802.15.4, que mensura a qualidade do enlace na recepção de dados, em termos do nível de distorção do sinal recebido. O RSSI aqui coletado possui maior granularidade que o RSSI coletado antes das transmissões (1 $dBm$), em razão do rádio utilizado.

Como as variáveis de entrada variam significativamente a cada transmissão e a cada recepção, processamos os dados para gerar um histórico de desempenho das últimas transmissões e recepções realizadas. Empregamos uma média móvel ponderada, desta forma designamos um menor peso às transmissões e recepções mais antigas e reduzimos a variação na entrada do preditor. A Equação 1 mostra o cálculo da média móvel, onde $E_t$ é o desempenho obtido no instante $t$, e $V_{t-1}$ é o valor da média móvel obtido no instante $t-1$.

$$V_t = V_{t-1} \times (1-\alpha) + E_t \times \alpha \tag{1}$$

O valor de $\alpha$ determina a rapidez com a qual a entrada reage a uma mudança no seu sinal (por exemplo um aumento no valor de entrada). Para evitar que o preditor tome uma decisão com base em informações que não condizem com a realidade, o meio deve ser sensoreado diversas vezes. Podemos estimar esta quantidade da seguinte forma. Em um cenário ideal, onde temos uma entrada constante $E$, o valor da média móvel no instante $t$ é dado por (a partir da Equação 1):

$$V_t = E \times \alpha \times \left( \sum_{i=0}^{t} (1-\alpha)^i \right) \tag{2}$$

O valor de $V_t$ tende a $E$ quando o número de termos tende a infinito. Na prática, devemos estimar um fator de aproximação $p$ empregando a fórmula abaixo.

$$(1-\alpha)^{n+1} = p \tag{3}$$

O valor $p$ da Equação 3 se encontra em um intervalo entre 0 (distância nula do valor real) e 1 (distância infinita). O termo $n$ define a quantidade de pacotes de sondagem enviados para que as condições reais do meio sejam aproximadas por um fator $p$. Esta sequência de pacotes de sondagem é aqui denominada de *janela de sensoreamento*.

O preditor proposto define um regressor para a variável *número de tentativas de transmissão dos próximos N pacotes*, aqui denominada de *janela de previsão*, onde $N$ é um parâmetro que será ajustado posteriormente. Esta variável foi escolhida pois o seu valor indica a qualidade do meio tendo em vista tanto interferências externas (ruídos do ambiente, interferências de outras redes no mesmo espectro) como internas (transmissões de dados no mesmo canal). Um canal que transmite a informação com menor quantidade de tentativas (devido a interferências internas e externas) deverá ser um canal com menor atraso e taxa de perda.

## 4. Coleta dos Dados para as Bases de Treinamento

Em seguida, capturamos a massa de dados para as bases de treinamento a serem aplicadas nos algoritmos de AM. Os cenários de coleta foram escolhidos de forma a representar

redes com variados graus de interferência. Para tanto, empregamos roteadores Wi-Fi para geração de interferência. Os dados foram coletados em três cenários distintos, descritos com mais detalhes abaixo, juntamente com uma descrição da geração de interferência pelos roteadores.

1. **Baixa Interferência**: Cenário em que apenas os nós sensores transmitiam dados entre si, com um nível de interferência externa quase nulo;
2. **Média Interferência**: Além da transmissão de dados entre os nós da RSSF, foram também colocados dois roteadores transmitindo dados entre si, a baixa potência e distantes um do outro, na mesma faixa de frequência da RSSF;
3. **Alta Interferência**: Semelhante ao cenário de Média Interferência, os mesmos roteadores foram colocados próximos um do outro e da RSSF, transmitindo na mesma faixa de frequência e a alta potência.

Foi escolhida a Plataforma *Iris* [Crossbow 2008] para coleta dos dados de desempenho a cada transmissão, empregando o rádio padrão da plataforma (RF230). A RSSF projetada para coleta dos dados foi composta por dez nós sensores, cada qual conectado a um computador para armazenar os dados de desempenho coletados.

Cada nó transmitia pacotes de 28 *bytes* a cada 500ms a um destinatário fixo. Antes de cada transmissão, eram coletadas dez amostras de RSSI, e após a transmissão eram coletados os atributos restantes descritos na Seção 3. Foram coletados dados durante quatro horas. A Figura 1 mostra a RSSF projetada, juntamente com os roteadores para geração de interferência.

**Figura 1. RSSF projetada para coleta dos dados de desempenho.**

## 5. Análise da Acurácia dos Algoritmos de AM

Foi utilizada a Ferramenta Weka [Hall et al. 2009] para a análise dos algoritmos de AM. Dentre os algoritmos disponibilizados na Ferramenta, foram avaliados os 16 algoritmos de regressão que permitiam a obtenção de resultados de desempenho em tempo hábil, inferior a uma semana de tempo de execução para o treinamento. Em vista disso, o algoritmo *SVM* (*Support Vector Machine*), considerado o estado-da-arte entre os algoritmos de AM, não foi selecionado. Os algoritmos avaliados foram:

- Algoritmos baseados em função:
    1. *Isotonic Regression*
    2. *Linear Regression*
    3. *MLP Regressor*
    4. *Multilayer Perceptron*
    5. *Pace Regression*
    6. *RBF Regressor*
- Algoritmos baseados em busca:
    1. *KNN - K-Nearest Neighbours*
- Algoritmos de meta-aprendizado:
    1. *Additive Regression* – com *Decision Stump* como algoritmo base
    2. *Bagging* – com *REP Tree* como algoritmo base
- Algoritmos baseados em regra de decisão:
    1. *Conjunctive Rule*
    2. *Decision Table*
    3. *M5 Rules*
- Algoritmos baseados em árvore de decisão:
    1. *Decision Stump*
    2. *Extra Tree*
    3. *M5P*
    4. *REP Tree*

Para a validação do desempenho dos algoritmos, foi utilizado o método *10-Fold Cross Validation*. Nele, o conjunto de instâncias é dividido em dez partes, obtendo-se dez combinações treinamento-teste (nove partes para treinamento, e uma para teste). Cada algoritmo foi executado dez vezes, obtendo-se assim o desempenho em cem combinações treinamento-teste. Cada execução realiza a divisão das partes de Teste de forma diferente, gerando assim diferentes combinações de dados.

Para a classificação dos algoritmos, foi utilizado o teste t-pareado, disponibilizado como teste padrão no Weka, o qual considera a dependência entre amostras, e portanto é o mais adequado para uso em conjunto com o método *10-Fold Cross Validation*. Foi considerada a métrica *Raiz do Erro Médio Quadrático*[1] [Hamilton 1994], a qual é mais sensível a *outliers*, o que é importante em RSSFs devido às oscilações bruscas do meio eletromagnético.

A análise de desempenho dos algoritmos se dividiu em duas etapas. A primeira analisou o impacto do valor de $\alpha$ no desempenho dos algoritmos de AM. Foram testados valores entre 0.01 e 0.09, com passo 0.02. Para cada valor de $\alpha$ testado, foi criada uma base de treinamento distinta, a qual reuniu dados de transmissão e recepção obtidos nos três cenários analisados. Para todas as bases, o tamanho da janela de previsão foi fixado em dez, ou seja, os *dez pacotes subsequentes*, dos quais a soma do número de tentativas de transmissão foi escalonado em um intervalo entre 0 e 100. A segunda etapa analisou a influência do tamanho da janela de previsão sobre o desempenho de cada algoritmo de AM. Assim como na etapa anterior, para cada tamanho de janela considerado, foi criada uma respectiva base de treinamento, com o mesmo escalonamento da janela de previsão

[1]Não empregamos a taxa de acertos ou erros, pois esta métrica somente pode ser empregada em algoritmos de AM baseados em *classificação*.

feito na primeira etapa. Nestes testes, o tamanho da janela de previsão foi variado entre 1 e 30, com $\alpha$ fixo em 0.01, valor que apresentou o melhor desempenho na primeira etapa, como será mostrado a seguir.

A Tabela 1 apresenta a Raiz do Erro Médio Quadrático média, obtida em dez execuções no Weka para cada um dos algoritmos de AM avaliados. Observamos que uma redução do valor de $\alpha$ causa um aumento no desempenho dos algoritmos de regressão. Isso se deve ao fato de que, quanto menor o valor de $\alpha$, menor será a variabilidade sobre os valores de entrada. Como aqui consideramos três cenários distintos, uma menor variabilidade nas entradas faz com que a base de treinamento se aproxime de três pontos distintos, cada um relacionado a um determinado cenário, simplificando a distribuição espacial entre as instâncias e com isso as soluções geradas por algoritmos de regressão, consequentemente aumentando o seu desempenho.

**Tabela 1. Raiz do Erro Médio Quadrático: valor de $\alpha$ variável.**

| **Algoritmo Aplicado** | **Peso ($\alpha$)** | | | | |
|---|---|---|---|---|---|
| | **0.01** | **0.03** | **0.05** | **0.07** | **0.09** |
| **Isotonic Regression** | 7.35 | 7.30 | 7.33 | 7.35 | 7.39 |
| **LinearRegression** | 7.51 | 7.55 | 7.79 | 8.06 | 8.33 |
| **MLP Regressor** | 7.06 | 7.07 | 7.10 | 7.13 | 7.16 |
| **Multilayer Perceptron** | 7.67 | 7.75 | 7.76 | 7.80 | 7.78 |
| **Pace Regression** | 7.51 | 7.55 | 7.78 | 8.04 | 8.30 |
| **RBF Regressor** | 7.25 | 7.15 | 7.17 | 7.22 | 7.27 |
| **KNN** | 3.22 | 3.31 | 3.59 | 4.40 | 5.57 |
| **Additive Regression** | 7.12 | 7.12 | 7.12 | 7.13 | 7.16 |
| **Bagging - REPTree** | 5.56 | 6.19 | 6.49 | 6.67 | 6.78 |
| **Conjunctive Rule** | 7.80 | 7.78 | 7.78 | 7.77 | 7.77 |
| **Decision Table** | 6.84 | 6.99 | 7.05 | 7.09 | 7.10 |
| **M5Rules** | 7.35 | 7.53 | 7.54 | 7.10 | 7.05 |
| **Decision Stump** | 7.87 | 7.81 | 7.79 | 7.78 | 7.79 |
| **Extra Tree** | 5.66 | 6.98 | 7.81 | 8.37 | 8.76 |
| **M5P** | 6.25 | 6.84 | 6.83 | 6.95 | 7.01 |
| **REPTree** | 5.99 | 6.70 | 6.95 | 7.05 | 7.10 |

A Figura 2 apresenta a Raiz do Erro Médio Quadrático média obtida em dez execuções para os cinco algoritmos de AM de melhor desempenho ao variarmos o tamanho da janela de previsão. Descartamos os resultados obtidos pelos outros algoritmos em razão de limitações de espaço e representação gráfica, além destes terem apresentado desempenho semelhante aos aqui mostrados. Observamos um comportamento logarítmico em todos os algoritmos apresentados, com o aumento do tamanho da janela de previsão. Esse fator indica a existência de um possível padrão de comportamento do meio ao longo de um intervalo específico, aqui representado pela janela de previsão.

## 6. Testes Práticos

Com base na Equação 3, escolhemos o valor de $\alpha = 0.7$ para a base de treinamento a ser utilizada na construção dos modelos de previsão a serem implementados. Consideramos um fator de aproximação das condições do meio igual ou menor que 1% (fator $p$ na

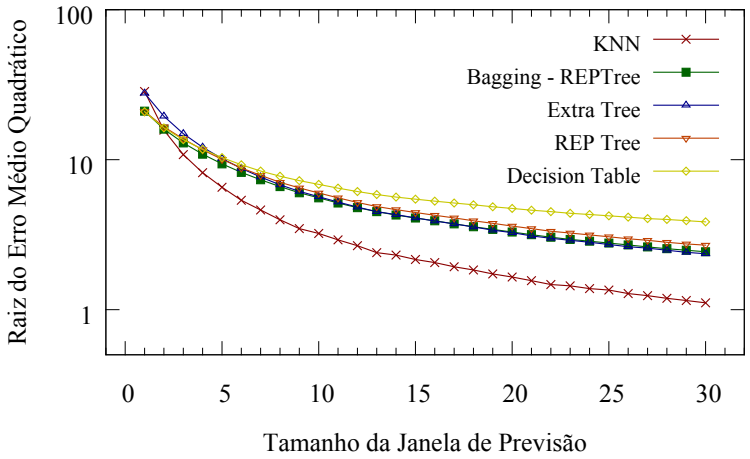



**Figura 2. Raiz do Erro Médio Quadrático - Tamanho de Janela Variável.**

Equação 3). Escolhemos estes valores de forma que o tempo necessário para coleta dos atributos de entrada, em cada canal, fosse de aproximadamente dois segundos, sendo um segundo para atributos relacionados ao envio de pacotes, e um segundo para atributos relacionados à recepção de pacotes. O tamanho da janela de previsão foi definido de forma empírica em aproximadamente seis minutos.

Foram realizados dois testes distintos, descritos nas seções a seguir. Para ambos, os nós da RSSF foram configurados para operar em cinco canais consecutivos. Foram empregados seis nós sensores, sendo que um deles, denominado *Nó Central*, executava uma das soluções de decisão de espectro, com um dos modelos de previsão gerados pelos algoritmos de AM implementado. Outros cinco nós sensores, denominados *Nós Auxiliares*, por sua vez, se mantinham fixos em um dos cinco canais. A função destes nós é se comunicar com o *Nó Central* quando necessário. A seguir é apresentado o funcionamento das soluções de decisão de espectro, além de uma representação gráfica desse funcionamento na Figura 3.

1. Para cada canal:
   (a) O *Nó Central* envia pacotes de sondagem durante um segundo ao *Nó Auxiliar*, de forma a coletar dados referentes à transmissão;
   (b) O *Nó Auxiliar* envia pacotes de sondagem ao *Nó Central* durante um segundo, para que o *Nó Central* colete dados referentes à recepção.
   (c) É realizada a previsão para o respectivo canal;
2. O *Nó Central* seleciona o canal com melhor previsão (menor valor de saída obtido);
3. O *Nó Central* transmite 30 pacotes de dados ao *Nó Auxiliar* localizado no canal selecionado, encaminhando o sumário do desempenho obtido ao computador, para mensurar o desempenho do algoritmo de troca de canal;
4. Reinicia-se o processo de decisão, retornando ao Passo 1.

Apesar de empregarmos um *Nó Auxiliar* por canal nos experimentos, o algoritmo pode ser modificado para que seja empregado somente um *Nó Auxiliar* para todos os canais sensoreados, de forma a reduzir o consumo de recursos.

Foram implementados os modelos de previsão obtidos pelos quatro melhores algoritmos de AM, os quais eram passíveis de implementação nos nós sensores (executável

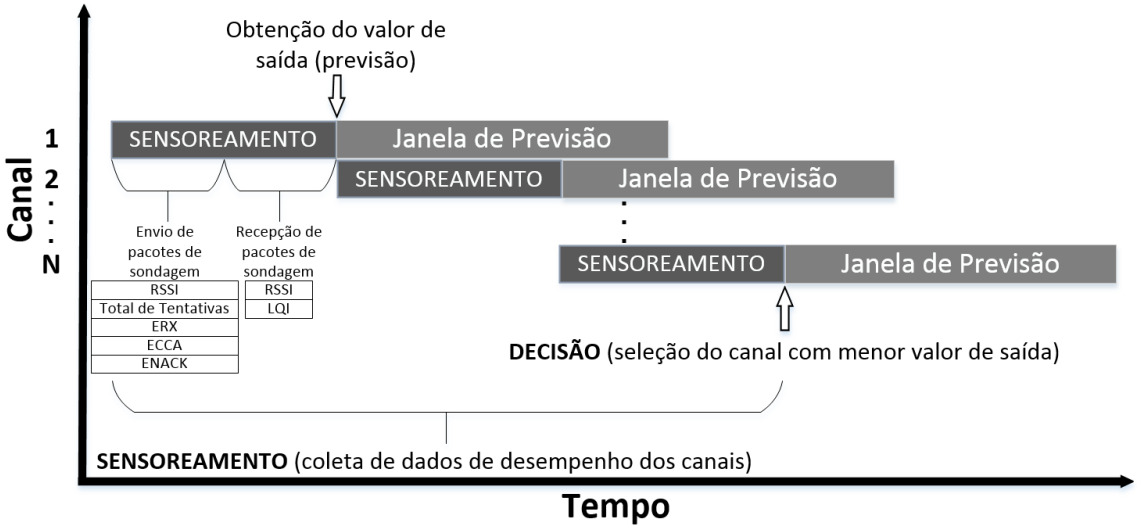



**Figura 3. Algoritmo de Decisão de Espectro.**

gerado inferior ao tamanho de ROM e RAM da plataforma *Iris*), segundo a métrica *Raiz do Erro Médio Quadrático*. Os algoritmos selecionados são listados abaixo:

1. *M5 Rules*: Algoritmo baseado em árvore de decisão com funções de regressão nas folhas, que gera um modelo de previsão sob a forma de uma lista de regras;
2. *REP Tree* (*Reduced Error Pruning Tree*): Algoritmo baseado em árvore de decisão, com valores fixos nas folhas;
3. *Additive Regression*: Algoritmo de Meta-Aprendizado que realiza incremento ou decremento no valor de saída de acordo com o resultado obtido pelo algoritmo-base. Nos testes realizados, foi utilizado como algoritmo-base o *Decision Stump*;
4. *Isotonic Regression*: Algoritmo que toma como base o atributo que proporciona o menor erro quadrático, gerando a previsão de acordo com o intervalo de valores no qual se encontra esse atributo.

Como *baseline*, foram implementadas duas soluções. A primeira, chamada *Fixo*, operava em um canal fixo. Já o segundo *baseline*, chamado *RSSI*, é baseado somente em energia, e foi inspirado no algoritmo proposto em [Stabellini and Javed 2010]. Ele coleta 100 amostras de RSSI em cada canal sensoreado, somando-se os valores obtidos. O canal escolhido é aquele que apresenta a menor soma. O funcionamento desta solução é semelhante ao das soluções de decisão, com diferença apenas no critério de escolha do melhor canal e no tempo de sensoreamento.

É importante observar o consumo de energia para execução das soluções não é considerado neste trabalho, uma vez que a medição desta variável de forma experimental, em especial na plataforma Iris, é um procedimento complexo. Por isso, o foco principal dos testes práticos foi quantificar o desempenho do enlace sem fio, de forma a verificar a eficácia das soluções propostas em nós sensores reais. A quantificação do consumo de energia é considerada aqui como trabalho futuro.

### 6.1. Primeiro Teste - Baixa Interferência Externa

Neste teste avaliamos o comportamento da RSSF em um ambiente com baixa interferência de outras redes. Aqui, os *Nós Auxiliares* foram posicionados a aproximadamente seis

metros do *Nó Central*, sendo utilizada como fonte de interferência principal as redes WLAN existentes nas proximidades do ambiente de testes. As soluções de decisão e os *baselines* foram executados em sequência durante 15 minutos cada, sendo realizadas cinco repetições independentes. Os resultados obtidos são apresentados com intervalo de confiança de 95% segundo a distribuição t de Student.

A Tabela 2 apresenta o desempenho obtido com as soluções implementadas, de acordo com taxa de entrega, o total de tentativas de transmissão, o total de colisões e a latência média a cada pacote enviado.

**Tabela 2. Resultados de desempenho - primeiro teste.**

| Solução Implementada | Taxa de Entrega (%) | Tentativas/ Pacote | Colisões/ Pacote | Latência/ Pacote (ms) |
|---|---|---|---|---|
| **REP Tree** | 96,44 (+/- 5,29) | 1,50 (+/- 0,32) | 0,37 (+/- 0,41) | 6,78 (+/- 1,64) |
| **M5 Rules** | 97,14 (+/- 2,33) | 1,49 (+/- 0,18) | 0,27 (+/- 0,18) | 6,62 (+/- 0,73) |
| **Additive Regression** | 98,88 (+/- 1,55) | 1,29 (+/- 0,18) | 0,14 (+/- 0,07) | 5,80 (+/- 0,42) |
| **Isotonic Regression** | 93,01 (+/- 16,48) | 1,54 (+/- 0,82) | 0,46 (+/- 0,87) | 7,13 (+/- 3,70) |
| **RSSI** | 97,50 (+/- 2,09) | 1,43 (+/- 0,25) | 0,25 (+/- 0,15) | 6,37 (+/- 0,85) |
| **Fixo** | 92,13 (+/- 6,15) | 2,27 (+/- 0,75) | 0,55 (+/- 0,34) | 8,78 (+/- 2,04) |

Com base nos resultados da Tabela 2, apenas uma das soluções de decisão (*Additive Regression*) se mostrou melhor em relação à solução baseada em canal fixo, para todas as métricas analisadas. Para as outras soluções de decisão, com exceção do *Isotonic Regression*, houve melhora segundo o total de tentativas de transmissão por pacote, ainda em relação à solução baseada em canal fixo. Além do total de tentativas de transmissão, o algoritmo *M5Rules* ainda foi superior ao mesmo *baseline* segundo a latência média por pacote. Nestes casos, foi possível verificar que as soluções propostas foram capazes de detectar variações de comportamento das redes WLAN locais, acarretando na escolha de canais distintos e melhorando o desempenho, ao contrário da solução baseada em canal fixo que esteve sujeita às variações do canal utilizado.

Por outro lado, a desvantagem das soluções de decisão, em relação à solução baseada em RSSI, se deve ao fato de que a fonte de interferência principal era o sinal das redes WLAN locais, que apresentaram poucas mudanças na sua configuração, como a potência de transmissão e o canal de operação, por exemplo. Além disso, essas redes atuaram em baixa magnitude sobre a RSSF. As soluções de decisão têm de forma intrínseca, em seus modelos de previsão, interferências externas de maior magnitude, que acarretam em impactos consideráveis no desempenho da transmissão. Por isso, a aplicação dessas soluções em cenários onde a fonte de interferência é de menor magnitude pode acarretar em erros na previsão em determinados momentos. Nesses casos, a simples leitura do RSSI permite uma conclusão adequada sobre a situação de cada canal.

### 6.2. Segundo Teste - Alta Interferência Externa

Neste teste avaliamos o comportamento das soluções de decisão em um cenário de coexistência. Os *Nós Auxiliares* foram colocados próximos do *Nó Central*, a uma distância de aproximadamente trinta centímetros. Como fonte de interferência principal, além das redes WLAN localizadas na região, foram colocados dois roteadores próximos da RSSF transmitindo dados entre si, em alta potência e a taxa máxima. Foi utilizado o programa *iperf*, e escolhemos canais próximos a um dos cinco empregados na RSSF. Os roteadores se comunicavam durante 15 segundos, e em seguida trocavam de canal de forma aleatória. Antes de voltar a transmitir, os roteadores aguardavam 10 segundos para que a troca de canal fosse realizada com sucesso. A Tabela 3 apresenta o desempenho médio obtido com as soluções implementadas, em cinco repetições independentes e intervalo de confiança de 95%, segundo a distribuição t de Student.

**Tabela 3. Resultados de desempenho - segundo teste.**

| Solução Implementada | Taxa de Entrega (%) | Tentativas/ Pacote | Colisões/ Pacote | Latência/ Pacote (ms) |
|---|---|---|---|---|
| **REP Tree** | 93,94 (+/- 4,07) | 1,96 (+/- 0,42) | 0,26 (+/- 0,19) | 7,33 (+/- 1,30) |
| **M5 Rules** | 92,88 (+/- 2,05) | 2,29 (+/- 0,52) | 0,26 (+/- 0,08) | 7,92 (+/- 0,99) |
| **Additive Regression** | 92,36 (+/- 1,88) | 2,23 (+/- 0,31) | 0,30 (+/- 0,09) | 7,93 (+/- 0,60) |
| **Isotonic Regression** | 94,78 (+/- 3,54) | 1,88 (+/- 0,59) | 0,18 (+/- 0,11) | 7,01 (+/- 1,33) |
| **RSSI** | 89,09 (+/- 3,12) | 2,86 (+/- 0,39) | 0,35 (+/- 0,04) | 9,23 (+/- 0,68) |
| **Fixo** | 88,55 (+/- 6,91) | 2,90 (+/- 0,76) | 0,36 (+/- 0,11) | 9,34 (+/- 1,60) |

Ao contrário do que foi observado no primeiro teste, observamos melhorias consideráveis no desempenho das transmissões, quando comparamos as soluções de decisão com os *baselines* implementados. Verificamos também a importância de se considerar uma gama maior de atributos durante o sensoreamento, como realizado nas soluções de decisão, quando variações de comportamento do meio de maior magnitude são observadas. Nesses casos, a simples leitura do RSSI acarreta em escolhas ruins em determinados momentos, uma vez que a relação sinal-ruído varia consideravelmente.

Outro fator apresentado por meio da Tabela 3 é a equivalência de desempenho entre as soluções de decisão, por meio do intervalo de confiança. Apesar das diferenças de desempenho obtidas no Weka, as imprecisões presentes nas medidas e a variação do meio podem explicar o desempenho similar na implementação em uma RSSF real. Isso também mostra que o fator predominante no desempenho prático de um algoritmo de decisão de espectro utilizando AM é a modelagem dos dados da base de treinamento, e não a escolha do melhor algoritmo de AM segundo o desempenho obtido na ferramenta de análise, ou o ajuste fino dos seus parâmetros. Portanto, vemos que para a decisão de espectro em RSSFs é mais interessante selecionar o algoritmo de AM que apresenta menor consumo de recursos computacionais.

## 7. Conclusões

Tendo em vista a poluição da faixa ISM e as restrições de energia em RSSFs, este trabalho propôs o uso de modelos de previsão gerados por algoritmos de AM, de baixo custo computacional, para a seleção do melhor canal de transmissão. Foram construídos modelos de regressão, com base em dados coletados a partir de plataformas reais. Estes modelos estimam o desempenho das transmissões a curto e longo prazo, permitindo a escolha do canal com melhor qualidade de comunicação para uso na RSSF.

Por meio da abordagem de regressão definiu-se, como valor de saída para os algoritmos de AM, a soma do número de tentativas de transmissão dos pacotes subsequentes, atributo que permitiu aproximar de forma simplificada o tempo necessário para a transmissão bem sucedida de uma sequência de pacotes em um determinado canal. Por meio da média móvel aplicada sobre os valores de entrada, incorporamos o comportamento histórico do canal em um único conjunto de atributos, simplificando as bases de treinamento a serem aplicadas nos algoritmos de AM. Com isso, foi possível criar modelos de previsão mais simples e passíveis de implementação em uma plataforma de RSSFs comercial.

Verificou-se, por meio dos testes práticos, que o uso de AM aumenta o desempenho da RSSF em relação a métodos de canal fixo e de decisão baseada em energia, principalmente em cenários de coexistência, onde a variação no comportamento do meio é de maior magnitude, ou seja, quando o desempenho médio das transmissões e recepções sofre impacto considerável. Isso reitera a aplicação final destinada a esses algoritmos, onde o meio se modifica constantemente, devido à existência de múltiplas redes no espectro.

Como trabalhos futuros, pretende-se refinar o ajuste dos parâmetros dos modelos, adicionando também mecanismos para que o número de canais sensoreados seja reduzido a longo prazo. Além disso, pretende-se investigar métodos de AM semi-supervisionados ou outras formas para a atualização periódica do modelo de previsão.

## Agradecimentos

Este trabalho foi realizado com financiamento das agências CAPES, CNPq e FAPEMIG.

## Referências

Akyildiz, I., Lee, W.-Y., and Chowdhury, K. (2009). Spectrum management in cognitive radio ad hoc networks. *Network, IEEE*, 23(4):6 –12.

Baldo, N., Reddy, T. B., Manoj, B. S., Rao, R., and Zorzi, M. (2009). A neural network based cognitive controller for dynamic channel selection. In *ICC*, pages 1–5. IEEE.

Bantouna, A., Stavroulaki, V., Kritikou, Y., Tsagkaris, K., Demestichas, P., and Moessner, K. (2012). An overview of learning mechanisms for cognitive systems. *EURASIP J. Wireless Comm. and Networking*, 2012:22.

Correia, L. H. A., Oliveira, E. E., Macedo, D. F., Moura, P., Loureiro, A. A. F., and Silva, J. S. (2012). A framework for cognitive radio wireless sensor networks. In *ISCC*, pages 611–616. IEEE.

Crossbow (2008). *Iris Datasheet*. Crossbow Technology. 2p.

FCC (2002). Spectrum policy task force report. Federal Communications Commission, Federal Communications Commission (FCC'02), Washington, DC, USA, November.

FCC (2005). Notice of proposed rule making and order: Facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies. ET Docket No. 03-108.

Felice, M. D., Chowdhury, K. R., Wu, C., Bononi, L., and Meleis, W. (2010). Learning-based spectrum selection in cognitive radio ad hoc networks. In Osipov, E., Kassler, A., Bohnert, T. M., and Masip-Bruin, X., editors, *WWIC*, volume 6074 of *Lecture Notes in Computer Science*, pages 133–145. Springer.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The Weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Hamilton, J. (1994). *Time series analysis*. Princeton Univ. Press, Princeton, NJ.

Katidiotis, A., Tsagkaris, K., and Demestichas, P. (2010). Performance evaluation of artificial neural network-based learning schemes for cognitive radio systems. *Computers & Electrical Engineering*, 36(3):518–535.

Masonta, M., Mzyece, M., and Ntlatlapa, N. (2013). Spectrum decision in cognitive radio networks: A survey. *Communications Surveys Tutorials, IEEE*, 15(3):1088–1107.

Sherman, M., Mody, A., Martinez, R., Rodriguez, C., and Reddy, R. (2008). IEEE standards supporting cognitive radio and networks, dynamic spectrum access, and coexistence. *Communications Magazine, IEEE*, 46(7):72–79.

Stabellini, L. and Javed, M. (2010). Experimental comparison of dynamic spectrum access techniques for wireless sensor networks. In *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*, pages 1 –5.

Thilina, K. M., Choi, K. W., Saquib, N., and Hossain, E. (2013). Machine learning techniques for cooperative spectrum sensing in cognitive radio networks. *IEEE Journal on Selected Areas in Communications*, 31(11):2209–2221.

Tsagkaris, K., Bantouna, A., and Demestichas, P. (2012). Self-organizing maps for advanced learning in cognitive radio systems. *Comput. Electr. Eng.*, 38(4):862–881.

Tsagkaris, K., Katidiotis, A., and Demestichas, P. (2008). Neural network-based learning schemes for cognitive radio systems. *Computer Communications*, 31(14):3394–3404.

Unnikrishnan, J. and Veeravalli, V. V. (2010). Algorithms for dynamic spectrum access with learning for cognitive radio. *IEEE Transactions on Signal Processing*, 58(2):750–760.

Yucek, T. and Arslan, H. (2009). A survey of spectrum sensing algorithms for cognitive radio applications. *Communications Surveys Tutorials, IEEE*, 11(1):116 –130.

Zeng, Y., Liang, Y.-C., Hoang, A. T., and Zhang, R. (2010). A review on spectrum sensing for cognitive radio: challenges and solutions. *EURASIP J. Adv. Signal Process*, 2010:2:2–2:2.

Zhao, Q. and Swami, A. (2007). A survey of dynamic spectrum access: Signal processing and networking perspectives. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1349 –IV–1352.

# Cooperação em Redes de Rádios Cognitivos

**Pedro Smith Coutinho**[1], **José Ferreira de Rezende**[1] **e Valmir Carneiro Barbosa**[1]

[1]COPPE – Universidade Federal do Rio de Janeiro (UFRJ)

{coutinho, rezende}@land.ufrj.br, valmir@cos.ufrj.br

***Resumo.*** *O funcionamento de redes de rádios cognitivos (RCs) envolve o acesso oportunista do espectro licenciado a usuários primários (UPs) por parte de usuários secundários (USs). Tradicionalmente, esse acesso é realizado de forma que os UPs sofram o mínimo de interferência em seu acesso ao espectro, tentando sempre tornar os USs o mais transparentes aos UPs quanto possível. No entanto, esse trabalho demonstra o fato de que os USs, ao influenciar positivamente a comunicação dos UPs, têm a capacidade de melhorar o desempenho de ambas as redes primária e secundária. Para isso, é proposto um esquema de cooperação através de retransmissões realizadas transparentemente pelos USs, que ao mesmo tempo melhoram o desempenho da rede primária e aumentam suas próprias oportunidades de utilização do espectro licenciado. O desempenho do esquema proposto é avaliado utilizando o simulador ns-3, apresentando melhoras tanto na vazão primária quanto na secundária quando comparadas a cenários sem cooperação.*

***Abstract.*** *The operation of cognitive radio (CR) networks involves opportunistic access to the spectrum licensed to primary users (PUs) by secondary users (SUs). Traditionally, this access is performed in such a way that the PUs suffer minimal interference to their access to the spectrum, trying to make the SUs as transparent as possible to PUs. However, this study demonstrates that the SUs, when positively influencing PU communication, can improve the performance of both primary and secondary networks. In order to achieve this, we propose a cooperation scheme based on retransmissions performed transparently by SUs, which simultaneously improves the performance of the primary network and increases their own opportunities to use the licensed spectrum. The performance of the proposed scheme is evaluated using the ns-3 simulator, showing improvements in primary and secondary throughputs when compared to a scenario without cooperation.*

## 1. Introdução

A pesquisa em Rádios Cognitivos (RCs) propõe a reutilização dinâmica de faixas do espectro eletromagnético alocadas a Usuários Primários (UPs), por Usuários Secundários (USs), de maneira oportunista [Akyildiz et al. 2006]. De acordo com este paradigma, USs apenas poderiam se comunicar usando uma determinada faixa quando não houvesse a presença de UPs, de tal forma a não causar interferência a estes. Assim, a proposta inicial era que os RCs, isto é, dispositivos de comunicação sem-fio dotados de capacidades cognitivas com o intuito de aproveitar oportunidades de espectro, ao serem usados pelos USs fossem totalmente transparentes aos UPs. No entanto, a interação e cooperação entre

UPs e USs pode ser fundamental em superar desafios e oferecer mais vantagens, tanto para UPs quanto para USs, do que o paradigma transparente.

Comunicação cooperativa é um princípio que diz respeito à cooperação de nós de uma rede com relação à comunicação de seus vizinhos. A comunicação tradicionalmente ocorre partindo de um conjunto (possivelmente unitário) de fontes em direção a outro conjunto (também possivelmente unitário) de destinos. A cooperação na comunicação introduz um terceiro conjunto (que, mais uma vez, pode ser unitário) chamado de retransmissores (do inglês *relays*). A utilização dos retransmissores traz vantagens à comunicação fonte-destino, como ganho de potência, economia de energia e maior diversidade espacial [Rashid et al. 2010], além do aumento da vazão de transmissão.

A proposta desse trabalho é que os nós cognitivos da rede secundária atuem como retransmissores dos nós primários, trazendo melhorias tanto no desempenho da rede primária quanto da rede secundária. Isto acontece pois, com auxílio dos nós secundários, os nós primários são capazes de escoar mais eficientemente o seu tráfego, liberando mais oportunidades para os nós secundários. Na proposta descrita neste artigo, fazemos uso dos mecanismos de priorização de tráfego, modulação e codificação adaptativa (ou controle automático de taxa) e de *spoofing*, os quais estão presentes em todas as tecnologias de rede sem-fio, para tornar transparente a cooperação dos USs em relação aos UPs. Assim, com o intuito de demonstrar os ganhos obtidos pela proposta, ela foi implementada utilizando-se os recursos oferecidos pelo padrão IEEE 802.11e [Mangold et al. 2002], os quais estão presentes no simulador ns-3 [ns-3 2013]. Pelo emprego correto dos mecanismos mencionados anteriormente e as suas respectivas parametrizações, obteve-se resultados que demonstram que a cooperação transparente dos USs faz com que os UPs obtenham uma melhor vazão, ao mesmo tempo que os próprios USs também o fazem em função de aumentarem as oportunidades de transmissão secundárias.

A Seção 2 descreve os principais trabalhos que tratam da cooperação em redes sem-fio. A Seção 3 apresenta o modelo do sistema sob qual reside a proposta deste artigo e a Seção 4 descreve o mecanismo proposto neste trabalho. Em seguida, a Seção 5 descreve o ambiente de simulação usado na avaliação de desempenho da solução proposta e traz os resultados das simulações realizadas, assim como uma discussão a respeito. E, por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

## 2. Trabalhos Relacionados

Existem propostas de utilização de comunicação cooperativa entre os próprios USs, com o objetivo de melhorar a qualidade da sua comunicação [Rashid et al. 2010]. Também há trabalhos [Rashid et al. 2010, Simeone et al. 2007, Krikidis et al. 2010, El-Sherif et al. 2011] que propõem o uso de retransmissores para a comunicação dos UPs. Utilizando-se da Teoria de Filas, é possível mostrar que os retransmissores na comunicação dos UPs fazem com que suas filas se esvaziem mais rapidamente, resultando em mais oportunidades de uso do espectro para os USs. Esses trabalhos utilizam a Teoria de Filas para analisar as taxas de chegada e de saída nas filas dos usuários.

O trabalho [Simeone et al. 2007] realiza a análise de um sistema com apenas quatro nós: um par fonte-destino de UPs e um par fonte-destino de USs. Esse trabalho propõe que o US fonte receba transmissões do UP fonte e mantenha uma fila de retransmissões, além de sua própria fila de envio de pacotes. Por meio de reconhecimentos

positivos/negativos (ACKs/NACKs), o US, quando houver uma oportunidade de espectro, retransmite um pacote do UP fonte ao UP destino com probabilidade $\epsilon$, ou transmite um pacote próprio ao US destino com probabilidade $1 - \epsilon$. Quando existe a possibilidade de erros de sensoriamento, a potência com a qual o US fonte pode realizar transmissões deve ser controlada, e é realizada uma otimização desse parâmetro. Em topologias em que o canal entre o UP fonte e o UP destino é ruim, mas os canais entre UP fonte e US fonte e entre US fonte e UP destino são bons, conclui-se que, quando o US realiza a comunicação cooperativa, sua vazão estável máxima aumenta em relação ao caso não-cooperativo.

O trabalho [Krikidis et al. 2010] utiliza a Teoria da Informação para encontrar taxas de transmissão dos UPs e USs de forma que não haja interferência nos UPs acima de um determinado limiar especificado, mesmo quando há transmissões simultâneas de ambas as partes, propondo um protocolo de acesso ao meio (MAC) com retransmissões. Ele realiza a comparação entre TDMA (*Time Division Multiple Access*) com e sem retransmissões e seu protocolo de transmissões simultâneas com e sem retransmissões, concluindo que sua proposta com comunicação cooperativa apresenta os melhores resultados.

Em [El-Sherif et al. 2011], é feita uma análise da comunicação cooperativa em redes maiores. Sua proposta possui nós primários operando exclusivamente como retransmissores, e a rede analisada, na maioria dos resultados, consiste em 20 UPs, 10 USs e 10 retransmissores primários. Em sua modelagem, os UPs acessam sua rede num esquema TDMA, com fatias de tempo (*timeslots*) ociosas se não tiverem algo a transmitir. Os USs e retransmissores acessam esses *slots* ociosos com um protocolo *Slotted* ALOHA. É interessante notar que de seus resultados, conclui-se que oferecer prioridade aos retransmissores primários, isto é, quando os USs só acessam o espectro nos *slots* ociosos depois que os retransmissores esvaziarem suas filas, oferece melhor desempenho não só aos UPs, mas também aos USs, em termos de vazão (taxa de saída das filas) obtida. A razão para isso é que há mais oportunidades de uso do espectro por parte dos USs quando as filas dos UPs e dos retransmissores são esvaziadas mais rapidamente.

Os trabalhos encontrados na literatura relacionados à comunicação cooperativa em redes de rádios cognitivos são, em geral, fortemente teóricos e ainda se encontram distantes de implementações práticas. A proposta que será apresentada nesse trabalho aborda a comunicação cooperativa de uma forma independente da tecnologia de rede sem-fio utilizada, mas se apresenta prática e próxima de tecnologias de redes existentes. Além disso, um ponto que está ausente na maioria dos trabalhos encontrados é a questão da decisão sobre quando realizar ou não as retransmissões cooperativas, já que os benefícios trazidos por elas dependem das condições dos canais encontrados entre os nós da rede. A proposta desse trabalho realiza retransmissões cognitivas, em que os *relays* atuam somente quando as condições encontradas são favoráveis, mas permanecem fora da comunicação dos outros nós, caso contrário.

## 3. Modelagem do Sistema

### 3.1. Modelo do Canal

O canal sem-fio entre um nó transmissor e um nó receptor pode ser modelado como um canal de Rayleigh [Rappaport 2002]. Segundo esse modelo, o sinal recebido no nó $j$ transmitido pelo nó $i$ pode ser representado como:

$$y_{ij}^t = \sqrt{G_i \rho_{ij}^{-\gamma}} h_{ij}^t x_i^t + n_{ij}^t, \quad (1)$$

onde $G_i$ é a potência de transmissão, $\rho_{ij}$ é a distância entre dois nós, $\gamma$ é o expoente de perda de propagação, $h_{ij}$ é o coeficiente de desvanecimento de canal entre os nós $i$ e $j$ no momento $t$ e é modelado como um processo gaussiano complexo independente e identicamente distribuído, circularmente simétrico de média zero e variância unitária. $x_i^t$ denota o pacote transmitido com potência média unitária e $n_{ij}^t$ denota ruído branco aditivo gaussiano com média zero e variância $N_0$.

O sucesso ou fracasso na recepção de um pacote são caracterizados por eventos de interrupção (*outage*), que são associados a uma probabilidade de interrupção (*outage probability*) [Benvenuto et al. 1988]. Uma interrupção é definida como o evento em que a relação sinal-ruído (SNR) cai abaixo de um limiar $\beta$ necessário para que o sinal seja decodificado com sucesso, ou seja, que o pacote de dados seja recebido.

Para o modelo de canal de Rayleigh da equação 1, temos:

$$\Pr(\text{SNR} < \beta) = 1 - \exp\left(-\frac{\beta N_0}{G_i \rho^\gamma}\right). \quad (2)$$

Para as simulações realizadas nesse trabalho, foi utilizado o modelo de propagação *log distance*, que é mais simples de ser calculado e utiliza valores obtidos empiricamente para aproximar a perda de propagação entre transmissor e receptor, levando em consideração em a potência média do sinal recebida diminui logaritmicamente de acordo com a distância entre transmissor e receptor [Rappaport 2002]. Segundo esse modelo, a perda de propagação é dada por:

$$L = L_0 + 10\gamma \log_{10}\left(\frac{d}{d_0}\right), \quad (3)$$

onde $L_0$ é a perda de propagação a uma distância de referência $d_0$, $\gamma$ é o expoente de perda de propagação e $d$ é a distância entre transmissor e receptor.

De acordo com esse modelo, a potência recebida $P_r$ é:

$$P_r = 10 \log_{10}(P_{r0}) - \gamma \log_{10}\left(\frac{d}{d_0}\right), \quad (4)$$

onde $P_{r0}$ é a potência recebida à distância de referência $d_0$.

A probabilidade de interrupção, então, pode ser vista de forma simplificada como:

$$\Pr(\text{SNR} < \beta) = \Pr(d > d^*). \quad (5)$$

Ou seja, a probabilidade de que a distância entre os nós seja maior do que uma distância $d^*$.

### 3.2. Modelo de Filas

Cada nó, primário ou secundário, possui um *buffer* para armazenar os pacotes que serão transmitidos. Essas filas possuem taxa de chegada de pacotes $\lambda$ e taxa de serviço (transmissão) $\mu$.

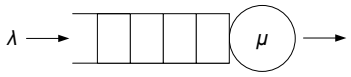



**Figura 1. Fila com taxa de chegada $\lambda$ e taxa de saída $\mu$.**

Além disso, USs realizam suas transmissões secundárias com prioridade menor do que as transmissões primárias dos UPs, ou seja, aguardam por oportunidades secundárias $op_s$ para transmitir. Essa priorização, apresentada na equação a seguir, garante o direito de utilização do espectro licenciado aos UPs, donos do mesmo, sobre os USs, que realizam o acesso oportunista:

$$P_{\text{tráf. prim.}} > P_{\text{tráf. sec.}} \tag{6}$$

O Teorema de Little relaciona o número de pacotes no sistema (aguardando na fila e sendo transmitidos), a taxa média de chegada de pacotes e o tempo médio que um pacote demora no sistema da seguinte forma [Cooper 1981]:

$$N = \lambda T, \tag{7}$$

onde $N$ é o número de pacotes na fila, $\lambda$ é sua taxa de chegada e $T$ é o tempo gasto por cada pacote.

Considerando a não saturação do canal, temos que $\mu \rightarrow \lambda$. Ou seja, em média, os pacotes tendem a ser transmitidos na mesma taxa em que chegam à fila. A partir da equação 7, pode-se ver que quanto maior a vazão $\lambda$, menor o tempo para transmitir o mesmo número de pacotes. Isto é, para um mesmo $N$, temos que $\uparrow \lambda \Rightarrow \downarrow T$.

### 3.3. Vazão e Capacidade do Canal

Com o uso de modulação e codificação adaptativa, as condições do canal (ruído, interferência, distância entre transmissor e receptor) fazem com que diferentes esquemas de codificação ou modulação sejam utilizados. Cada esquema de modulação/codificação está relacionados diretamente com a taxa de transmissão de dados no canal, que pode ser vista como a vazão obtida $\lambda$. Para cada esquema existe um limiar de interrupção $\beta$ diferente. Quanto maior a taxa de transmissão do esquema $k$ e, consequentemente, maior a vazão obtida $\lambda_k$, é necessário um limiar de SNR maior $\beta_k$. Isto é, $\uparrow \lambda_k \Rightarrow \uparrow \beta_k$.

## 4. Proposta

A proposta apresentada nesse trabalho pode ser implementada como rede secundária e ter diversas tecnologias de comunicação atuando como rede primária. Para isso, basta que três premissas sejam suportadas por essa tecnologia:

- Modulação e codificação adaptativa (*adaptive modulation and coding*) ou algum mecanismo de controle de taxa de transmissão que adeque os parâmetros de transmissão às condições do canal sem-fio sendo utilizado.
- Priorização de tráfego. Apenas os nós cognitivos (USs) precisam possuir essa capacidade. Qualquer tecnologia que já possua suporte a priorização de tráfego/qualidade de serviço (QoS), pode ter o mecanismo aproveitado para implementar a proposta nos nós secundários de forma compatível com os nós primários.

- *Spoofing*, isto é, permita que um nó possa se passar por outro. Para que a retransmissão seja transparente aos UPs, é necessário que o *relay* (US) possa se fazer passar pelos UPs cuja comunicação ele está auxiliando.

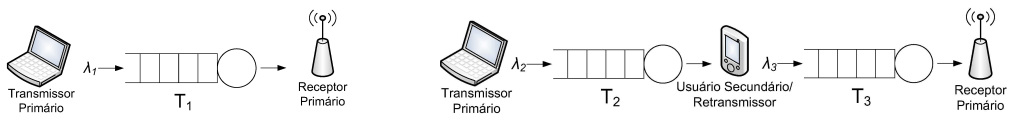
Praticamente todas as tecnologias de redes sem-fio como IEEE 802.11, IEEE 802.16 (WiMAX), Bluetooth, UMTS (3G), LTE (4G), que utilizam diversas formas de acesso ao meio, como CSMA/CA, TDMA, OFDMA, possuem essas características e podem ser beneficiadas atuando como a rede primária para a proposta desse trabalho [Vassis et al. 2005, WiMAX , Bluetooth , UMTS , LTE ].

Além disso, as retransmissões cognitivas podem funcionar com base na técnica *amplify-and-forward*, em que não é necessário nem mesmo decodificar a transmissão para retransmiti-la. Essa técnica aumenta a potência recebida e a diversidade espacial e permite um funcionamento de forma ainda mais independente da tecnologia de comunicação utilizada na rede primária [Huang and Zhang 2013].

Considerando pacotes de mesmo tamanho, de acordo com as equações 7 e 5, limiares de probabilidade de interrupção são maiores para capacidades maiores, e assim temos que o tempo para se transmitir um pacote é tanto menor quanto maior for a vazão do canal, ditada pelas suas condições. No caso de espaços abertos, onde não há sombreamento (*shadowing*), o principal fator dessas condições é a distância entre transmissor e receptor, conforme visto na equação 5. A equação abaixo ilustra esse fenômeno, para uma taxa de transmissão $k$:

$$\Pr(\text{SNR} \geq \beta_k) \propto \lambda_k \propto \frac{1}{T_k}. \tag{8}$$

A proposta apresentada nesse trabalho vai contra o princípio de redes de rádios cognitivos de que os usuários secundários não devem interferir na comunicação dos usuários primários. Ao contrário, como será visto mais à frente nos resultados do trabalho, quando um US atua como retransmissor (*relay*) dos pacotes de um par de UPs, o US causa uma interferência benéfica na comunicação primária, aumentando a vazão de transmissão dos UPs. Um US situado entre um par de UPs segmenta a comunicação deles quando pode trazer benefícios à vazão de tráfego da mesma, de acordo com a equação 8. Isso ocorre quando as taxas da comunicação segmentada são tais que a soma das durações das transmissões nesse caso é menor do que a duração da transmissão fim-a-fim, sem a atuação do *relay*. A Figura 2 e a equação 9, apresentadas a seguir, ilustram essa ideia.




(a) Transmissão direta. (b) Com retransmissor secundário.

**Figura 2. Comunicação de um par de UPs com ou sem atuação do retransmissor (*relay*) secundário.**

Na Figura 2, observa-se que, sem a atuação do retransmissor secundário na comunicação do par de usuários primários, a transmissão de um pacote dura um tempo $T_1$. Já com retransmissão, esse pacote demora $T_2 + T_3$ para chegar ao receptor primário. Assim,

é vantajoso realizar a retransmissão de um pacote quando se obtém um tempo total menor do que sem a atuação do *relay* secundário, como pode ser visto na equação a seguir:

$$\text{Se } T_1 \leq T_2 + T_3, \text{ não retransmite;} \tag{9}$$
$$\text{se } T_1 > T_2 + T_3, \text{ retransmite.} \tag{10}$$

Essa atuação do retransmissor secundário no tráfego primário ocorre de maneira transparente, de forma que tanto transmissor quanto receptor primários acreditem estar se comunicando diretamente, o que permite que uma rede sem-fio convencional seja considerada a rede primária sem que seus nós precisem sofrer qualquer modificação. Para isso, tanto os pacotes de dados quanto os reconhecimentos (ACKs) enviados pelo retransmissor ao par de nós primários devem ser criados como se fossem dos próprios nós primários, técnica conhecida como *spoofing*. Os nós secundários desta rede devem possuir capacidades cognitivas para realizar a decisão entre retransmitir ou não um pacote primário detectado de acordo com a equação 9, além de serem capazes de realizar essa retransmissão de forma transparente aos usuários primários licenciados.

Para que a retransmissão cognitiva possa funcionar corretamente e trazer benefícios à comunicação primária, é necessário que os pacotes retransmitidos pelos USs possuam prioridade diferente dos pacotes gerados pelos mesmos. De fato, para evitar disputas entre o tráfego primário transmitido por UPs e retransmitido pelos *relays* secundários, é melhor que as retransmissões possuam prioridade maior do que o tráfego primário direto. Desta forma, a equação 6 é modificada, dando origem à equação 11:

$$P_{\text{tráf. relay}} > P_{\text{tráf. prim.}} > P_{\text{tráf. sec.}} \tag{11}$$

Na equação 11, o tráfego primário é transmitido pelos UPs, e tanto o tráfego secundário quanto *relay* são transmitidos pelos USs.

Conforme as equações 7 e 8, temos que $\lambda_k \propto \frac{1}{T_k}$, quanto maior a vazão obtida pelo canal, menor o tempo necessário para transmitir um mesmo número de pacotes. Semelhantemente, de acordo com a mesma equação, $\lambda_k \propto N_k$, para um mesmo período $T$ fixo. Logo, quanto maior a vazão obtida, maior o número pacotes primários transmitidos. Consequentemente, com isso se obtém um maior o número de chances de ocorrer uma oportunidade de transmissão secundária, cuja probabilidade pode ser denominada $Pr(op_s)$. Desta forma, ao elevar o número de pacotes transmitidos pelos UPs ($N_p$), os USs também elevam suas chances de obterem oportunidades de transmissão secundária, elevando o número de pacotes transmitidos por eles ($N_s$), beneficiando a rede secundária ao ajudar a rede primária. Isto é:

$$N_s \propto \text{Pr}(op_s) \propto N_p. \tag{12}$$

Na Seção 5, a seguir, apresentaremos uma implementação no simulador ns-3, baseada no padrão IEEE 802.11.

## 5. Ambiente de Simulação e Resultados Numéricos

Para avaliar o desempenho da implementação da proposta desse trabalho foi utilizado o simulador de eventos discretos ns-3. Esse simulador conta com modelos bastante completos de redes IEEE 802.11 (Wi-Fi), incluindo qualidade de serviço (QoS), mecanismos de controle automático de taxa de transmissão e modulação de sinal na camada física que são essenciais para uma avaliação de desempenho mais fiel da proposta apresentada [ns-3 2013].

Para a avaliação da proposta desse trabalho, tanto os nós primários quanto secundários das simulações foram modelados como rádios Wi-Fi seguindo o padrão IEEE 802.11g [Vassis et al. 2005], com o suporte a qualidade de serviço (QoS) do padrão IEEE 802.11e. A camada MAC dos nós secundários foi modificada para realizar as retransmissões dos pacotes e reconhecimentos (ACKs) dos usuários primários a partir da implementação do padrão IEEE 802.11e, utilizando as diferentes classes de tráfego, devidamente parametrizadas, para implementar a priorização necessária para o funcionamento dos rádios cognitivos secundários. Os nós primários utilizaram a camada MAC do padrão IEEE 802.11g sem modificação em seu código, apenas com seus parâmetros ajustados para a priorização dos diferentes tipos de tráfego da rede. É importante frisar que a proposta desse trabalho independe da tecnologia de comunicação utilizada pelos USs, e trabalhos futuros visam comprovar essa generalidade modelando a proposta sobre outras tecnologias.

Para cada par de transmissor e receptor primários detectado pelo nó secundário, este mantém informações sobre as taxas de transmissão utilizadas tanto na comunicação direta como na segmentada ($\lambda_1$, $\lambda_2$ e $\lambda_3$) e, consequentemente, pode calcular as durações $T_1$, $T_2$ e $T_3$ da equação 9. De acordo com esses valores, o nó pode ter dois estados para cada par primário: retransmitindo ou não retransmitindo. Porém, se a soma das durações $T_2 + T_3$ for maior do que a duração $T_1$, a atuação de um *relay* é prejudicial à vazão da comunicação. Logo, periodicamente os nós secundários mudam de estado por um período curto para coletar informações sobre o estado complementar ao seu estado atual (seja esse retransmitindo ou não). A partir dessas informações, a decisão entre manter o estado atual ou mudar é tomada pelo nó secundário. A Figura 3 ilustra um exemplo desse mecanismo cognitivo em funcionamento.

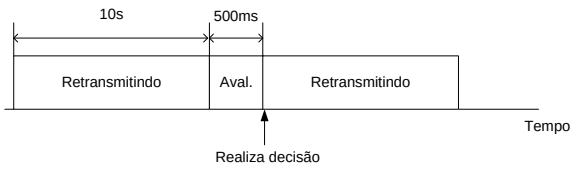
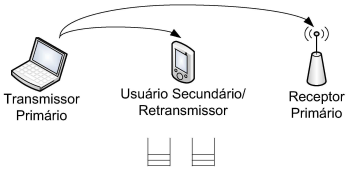


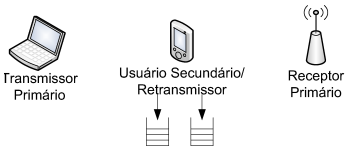

**Figura 3. Processo de decisão secundário entre realizar ou não a retransmissão de um par de nós primários.**

Quando um usuário secundário recebe um quadro endereçado a um usuário primário, ele armazena esse quadro em seu *buffer* de retransmissões e prepara um quadro ACK com *spoofing*, que será enviado ao transmissor primário. Caso o receptor primário receba o quadro com sucesso e envie um ACK, no momento que este é detectado pelo retransmissor, tanto o quadro de dados quanto de reconhecimento são descartados pelo US, como pode ser visto na Figura 4. Caso contrário, os quadros são transmitidos e a

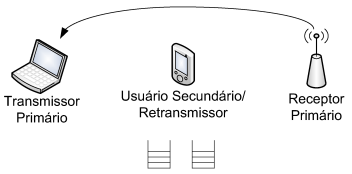
atuação do *relay* se inicia na comunicação desse par de UPs, como ilustrado na Figura 5. Quando um retransmissor secundário começa a atuar na comunicação de um par de nós primários, os mecanismos de controle automático de taxa de transmissão do padrão IEEE 802.11 [Kamerman and Monteban 1997] fazem com que as taxas de transmissão usadas sejam cada vez maiores, aumentando a vazão de dados primários e, consequentemente, as oportunidades de transmissão secundária $op_s$.




(a) Transmissor primário realiza a transmissão, que é recebida pelo *relay* e pelo receptor primário.



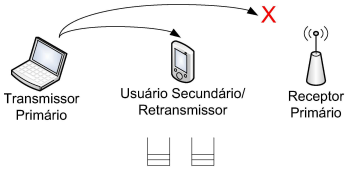

(b) *Relay* adiciona quadro de dados e ACK às suas respectivas filas.



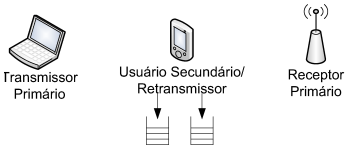

(c) Receptor primário transmite ACK para sinalizar que a transmissão do quadro de dados foi bem-sucedida.


(d) *Relay* descarta quadro de dados e ACK de suas filas.

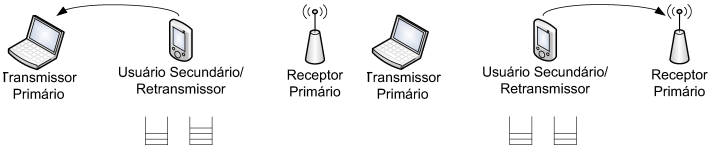
**Figura 4. Passo a passo do funcionamento da rede quando a transmissão primária é bem-sucedida e o *relay* não atua na comunicação.**




(a) Transmissor primário realiza a transmissão, que é recebida apenas pelo *relay*.




(b) *Relay* adiciona quadro de dados e ACK às suas respectivas filas.




(c) *Relay* transmite ACK com *spoofing* ao transmissor primário.

(d) *Relay* transmite quadro de dados com *spoofing* ao receptor primário.

**Figura 5. Passo a passo do funcionamento da rede quando há falha na transmissão primária e o *relay* atua na comunicação.**

De acordo com o padrão IEEE 802.11e, os períodos entre quadros (IFSs – *Inter Frame Spaces*) distintos usados para os *backoffs* das transmissões de cada classe de tráfego são denominados AIFS (*Arbitration Inter Frame Spaces*) [Mangold et al. 2002] e

são usados para priorizá-las. Desta forma, diferencia-se o tráfego de cada classe de usuários (UPs e USs) de maneira distribuída, com prioridades de acordo com a equação 6. Para se calcular o *backoff* que será utilizado para a transmissão de um quadro de uma determinada classe de tráfego, o valor do AIFS dessa classe é somado a um valor sorteado dentro da janela de contenção (CW – *Contention Window*), que aumenta a cada erro de transmissão e também é diferente para cada classe de tráfego.

Como os quadros de reconhecimento (ACKs) formados pelos retransmissores cognitivos devem entrar em uma fila para serem transmitidos ou descartados (caso o quadro enviado diretamente pelo transmissor primário tenha sido recebido corretamente), esses quadros foram implementados como uma nova classe de tráfego, com prioridade maior do que os dados retransmitidos. Cada classe de tráfego tem seus valores de AIFS e CW ajustados de forma que, à exceção de um erro de transmissão ou colisão que aumente o tamanho dos *backoffs*, as prioridades das classes são respeitadas de forma determinística. Desta forma, a equação 11, para essa implementação, recebe um novo termo, como pode ser visto a seguir:

$$P_{\text{relay ACK}} > P_{\text{relay dados}} > P_{\text{tráf. prim.}} > P_{\text{tráf. sec.}} \tag{13}$$

As classes "*relay* ACK", "*relay* dados"e "tráfego secundário"estão presentes em três filas diferentes na camada MAC dos nós cognitivos secundários, enquanto a classe "tráfego primário"se encontra nos nós primários da rede.

### 5.1. Simulações

Para avaliar o desempenho da implementação da rede de rádios cognitivos proposta nesse trabalho, foram realizadas simulações com dois nós primários (UPs) com e sem um retransmissor secundário situado entre eles, equidistante aos mesmos. O propósito disso foi validar o funcionamento e avaliar os benefícios trazidos pela retransmissão transparente dos *relays* cognitivos, comparando os resultados entre os seguintes cenários: sem *relays*, com *relays* que sempre realizam retransmissões (não cognitivos, obtem resultados piores do que sem retransmissões em alguns casos), e a proposta, *relays* cognitivos que realizam retransmissões apenas quando é benéfico aos UPs. Após essa fase, foram feitas simulações em que o nó secundário situado entre os nós primários também atuou como transmissor secundário para um segundo nó secundário, além de realizar as retransmissões cognitivas. Nessa fase, mostramos que, além de melhorar o desempenho dos nós da rede primária, a atuação dos nós secundários como *relays* cognitivos também melhora o desempenho da sua própria comunicação.

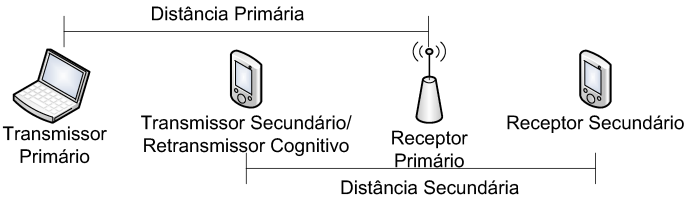



**Figura 6. Distâncias primária e secundária usadas nas simulações.**

A distância entre os nós primários foi variada de 5m a 175m, de 10m em 10m. Nas simulações com tráfego secundário, a distância entre os nós secundários variou de

30m a 120m, de 15m em 15m. A Figura 6 ilustra as distâncias primária e secundária das simulações. Os nós cognitivos seguiram o padrão IEEE 802.11g [Vassis et al. 2005], e o mecanismo de controle automático de taxas de transmissão utilizado foi o ARF (*Automatic Rate Fallback*) [Kamerman and Monteban 1997]. O modelo de propagação do meio utilizado foi o *log distance*. O tamanho dos pacotes gerados na camada de aplicação foi 2000 bytes, e o tráfego gerado na camada de aplicação foi do tipo CBR (*Constant Bit Rate*) assintótico, usando o protocolo UDP. As avaliações cognitivas para decidir se os nós secundários deveriam ou não realizar as retransmissões (*relay*) tinham duração de 500ms, e eram realizadas periodicamente a cada 10s. Foram realizadas 30 rodadas de 60s de duração e os resultados apresentados foram calculados com intervalos de confiança de 95%. A Tabela 1 consolida todos os valores de parâmetros usados nas simulações.

**Tabela 1. Parâmetros de simulação**

| Parâmetro | Valores |
|---|---|
| Distância entre os nós primários | 5m a 175m |
| Distância entre os nós secundários | 30m a 120m |
| Tamanho do pacote na camada de aplicação | 2000 bytes |
| Tipo de tráfego gerado pelos transmissores | CBR assintótico |
| Protocolo da camada de rede | UDP |
| Padrão de comunicação da rede sem-fio | IEEE 802.11g |
| Mecanismo de controle automático de taxas | ARF |
| Modelo de propagação de sinal | *log distance* |
| Intervalo entre avaliações cognitivas | 10s |
| Duração das avaliações cognitivas | 500ms |
| Duração de cada rodada | 60s |
| Número de rodadas para cada cenário | 30 |
| Intervalo de confiança | 95% |

## 5.2. Resultados

A Figura 7 mostra a vazão da comunicação primária de acordo com a distância entre os nós primários em três cenários diferentes: sem *relays*, com um *relay* não cognitivo que sempre realiza retransmissões (mesmo trazendo prejuízos à rede primária) e com um *relay* que avalia se deve ou não realizar as retransmissões dos pacotes primários, para somente beneficiá-los. Para cada distância primária, o nó secundário sempre está equidistante aos dois nós primários, na metade do caminho entre eles. Ao se analisar a figura pode-se notar que, até uma distância de 85m, o cenário sem retransmissões possui vazão maior, e em distâncias maiores do que esta, o cenário com retransmissões atinge maior vazão. Nota-se também, que a proposta desse trabalho, isto é, avaliar cognitivamente se deve-se ou não realizar retransmissões, atinge vazões muito próximas ao melhor cenário em cada uma das duas situações. A proposta apresenta ótimos resultados para distâncias primárias maiores do que 85m, oferecendo uma melhora na vazão de 200% (de aprox. 2 Mbps para aprox. 6 Mbps) com o protocolo UDP e de 50% (de aprox. 1 Mbps para aprox. 1,5 Mbps) com o protocolo TCP.

Nos experimentos com tráfego secundário, os resultados foram muito parecidos para distâncias secundárias de até 90m, com os ganhos de vazão primária sendo reduzidos para as distâncias secundárias maiores do que essa. Por esse motivo, a distância secundária de 60m foi escolhida para ter seus resultados analisados. A Figura 8 apresenta

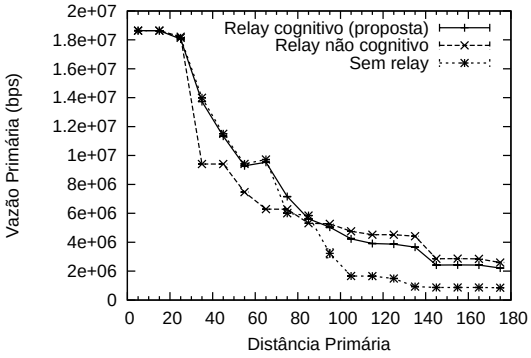
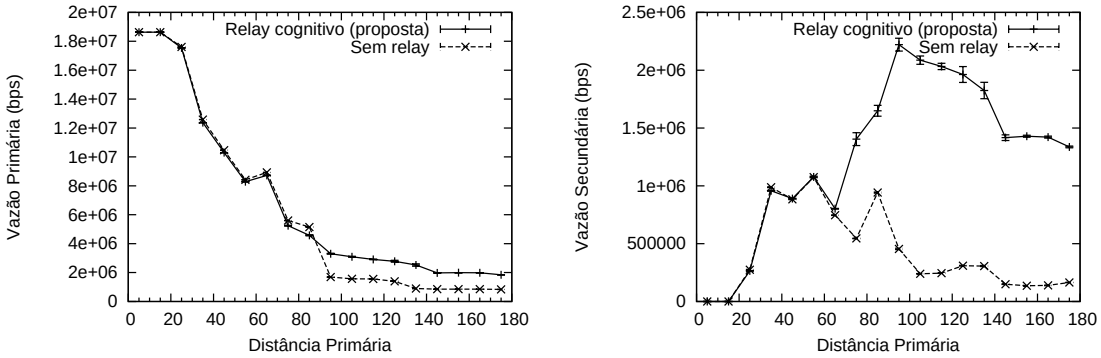



**Figura 7. Vazão da comunicação primária de acordo com a distância primária.**

as vazões das comunicações primária e secundária de acordo com a distância primária, para uma distância secundária de 60 m. Analisando essa figura, notamos que, o cenário com *relays* cognitivos apresenta melhoria no desempenho da comunicação primária assim como nos cenários da Figura 7, porém em proporções um pouco menores, com uma melhora de quase 100% (aprox.2 Mbps para quase 4 Mbps). Além disso, o cenário em que os nós secundários realizam as retransmissões cognitivamente também aumenta a vazão da comunicação secundária. Esse aumento é bastante significativo, aumentando essa vazão em 300% (de aprox. 500 kbps para mais de 2 Mbps). Veremos na Figura 9 as razões para as quais isso ocorre.

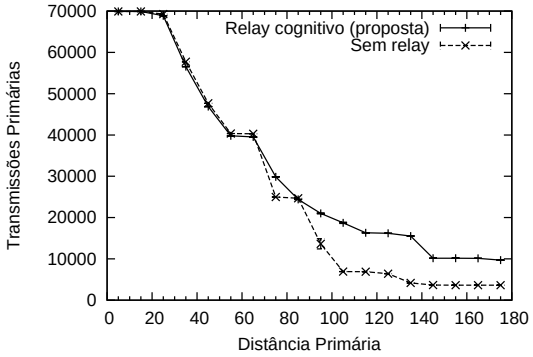



(a) Comunicação primária. (b) Comunicação secundária.

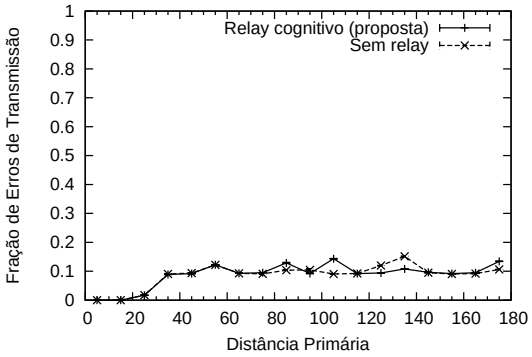
**Figura 8. Vazões das comunicações primária e secundária de acordo com a distância primária, com distância secundária de 60 m.**

A Figura 9(a) mostra o número de transmissões que ocorrem na comunicação primária em função da distância primária. Nota-se nessa figura que, na faixa de distâncias primárias a partir de 75m, na qual as retransmissões apresentam melhorias na vazão da comunicação secundária, há um número muito maior de transmissões primárias quando os *relays* atuam. Como o aumento das transmissões primárias também aumenta as oportunidades de transmissão secundárias $op_s$ e isso explica o aumento da vazão secundária. Como podemos observar na Figura 9(b), a fração de erros de transmissão pelo total de transmissões primárias é aproximadamente a mesma para ambos os cenários, o que demonstra que a taxa de erros de quadro permanece constante e não é influenciada pela

proposta desse trabalho.




(a) Número total de transmissões primárias.




(b) Fração de erros de transmissão pelo total de transmissões primárias.

**Figura 9. Total de transmissões e fração de erros pelo total de transmissões para a comunicação primária.**

## 6. Conclusões e Trabalhos Futuros

O funcionamento tradicional de redes de rádios cognitivos envolve o acesso oportunista do espectro licenciado a usuários primários por parte de usuários secundários. Nesse modelo de acesso, os usuários secundários devem interferir o mínimo possível na comunicação dos usuários primários. No entanto, esse paradigma de transparência pode ser quebrado quando os usuários secundários atuam como retransmissores (*relays*) dos usuários primários, auxiliando em sua comunicação. Essa melhora causada na comunicação dos usuários primários pelos secundários pode ser considerada uma "interferência benéfica".

A proposta de cooperação entre usuários primários e secundários desse trabalho apresenta as seguintes características: transparência, já que as retransmissões são feitas de forma que os usuários primários acreditam se comunicar diretamente e cognição/inteligência, pois os retransmissores secundários somente atuam como *relays* quando sua avaliação indica que haverá benefícios aos usuários primários. Além disso, a proposta independe da tecnologia de comunicação utilizada, desde que a mesma atenda aos requisitos apresentados na Seção 4: priorização de tráfego, modulação e codificação adaptativa ou controle automático de taxas e permita *spoofing*, mecanismos presentes em todas as tecnologias de redes sem-fio. A proposta foi avaliada para uma rede de acesso aleatório baseada na tecnologia IEEE 802.11 e apresenta resultados com melhora significativa na vazão primária, além de um outro efeito benéfico muito importante. Por aumentar o número de transmissões realizadas pelos usuários primários, ela aumenta as oportunidades de transmissão secundária, melhorando significativamente também a vazão secundária, sem aumentar a taxa de erros de transmissão primários.

Como trabalhos futuros, realizaremos simulações com redes de rádios cognitivos de múltiplos pares de nós, tanto primários como secundários, distribuídos aleatoriamente em uma determinada área, além da implementação utilizando outras tecnologias de comunicação na rede primária. O objetivo disso é avaliar os resultados da proposta em cenários de aplicação mais geral, uma vez que os resultados preliminares em cenários controlados, apresentados nesse trabalho, são bastante promissores.

## Referências

Akyildiz, I. F., Lee, W.-Y., Vuran, M. C., and Mohanty, S. (2006). Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 50:2127–2159.

Benvenuto, N., Pupolin, S., and Guidotti, G. (1988). Performance evaluation of multiple access spread spectrum systems in the presence of interference. *Vehicular Technology, IEEE Transactions on*, 37(2):73–77.

Bluetooth. Specification of the Bluetooth System, Core Version 3.0 + HS. Relatório Técnico. Bluetooth SIG.

Cooper, R. B. (1981). *Introduction to Queueing Theory (2a Edição)*. Elsevier North Holland.

El-Sherif, A., Sadek, A., and Liu, K. (2011). Opportunistic multiple access for cognitive radio networks. *IEEE Journal on Selected Areas in Communications*, 29(4):704 –715.

Huang, C. and Zhang, X.-P. (2013). Performances of amplify-and-forward cooperative relay networks with different topologies. *Wireless Personal Communications*, 69(2):561–577.

Kamerman, A. and Monteban, L. (1997). Wavelan®-ii: a high-performance wireless lan for the unlicensed band. *Bell Labs Technical Journal*, 2(3):118–133.

Krikidis, I., Devroye, N., and Thompson, J. (2010). Stability analysis for cognitive radio with multi-access primary transmission. *IEEE Trans. on Wireless Communications*, 9(1):72 –77.

LTE. 3GPP specification: 36.201; Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description. Relatório Técnico. 3rd Generation Partnership Project (3GPP).

Mangold, S., Choi, S., Klein, O., Hiertz, G., and Stibor, L. (2002). IEEE 802.11e Wireless LAN for Quality of Service. *Proceedings of the European Wireless Conference*, 1:32–39.

ns-3 (2013). ns-3 Network Simulator. Último acesso em 06-dezembro-2013.

Rappaport, T. S. (2002). *Wireless Communications: Principles and Practice (2a Edição)*. Prentice Hall.

Rashid, R., Aripin, N., Fisal, N., Ariffin, S., and Yusof, S. (2010). Integration of cooperative sensing and transmission. *IEEE Vehicular Technology Magazine*, 5(3):46 –53.

Simeone, O., Bar-Ness, Y., and Spagnolini, U. (2007). Stable throughput of cognitive radios with and without relaying capability. *IEEE Transactions on Communications*, 55(12):2351 –2360.

UMTS. 3GPP specification: 25.05U; UMTS Radio Aspects; Multiplexing and multiple access. Relatório Técnico. 3rd Generation Partnership Project (3GPP).

Vassis, D., Kormentzas, G., Rouskas, A., and Maglogiannis, I. (2005). The IEEE 802.11g standard for high data rate WLANs. 19:21–26.

WiMAX. IEEE standard for local and metropolitan area networks part 16: Air interface for fixed and mobile broadband wireless access systems amendment 2: Physical and medium access control layers for combined fixed and mobile operation in licensed bands.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 19*

## *Redes de Datacenters e Computação em Nuvem*

# Latência Versus Sobrevivência no Projeto de Centros de Dados Geograficamente Distribuídos

**Rodrigo S. Couto[1,2], Stefano Secci[2], Miguel Elias M. Campista[1], 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

***Resumo.*** *Uma tendência atual no projeto de centros de dados é realizar a distribuição geográfica de seus servidores, permitindo uma maior proximidade aos usuários finais e um aumento na capacidade de sobrevivência a falhas. Entretanto, essa solução aumenta a latência de comunicação entre os servidores, causada pela maior distância entre eles. Neste trabalho, aborda-se o compromisso entre latência e sobrevivência em centros de dados geodistribuídos, através da formulação de um problema de otimização. Os resultados obtidos a partir de cenários realistas mostram que, para redes de longa distância, o aumento na latência é significativo apenas no caso de exigências muito fortes de sobrevivência, enquanto que o mesmo é insignificante para exigências moderadas. Por exemplo, o pior caso de latência é menor que 4 ms ao garantir que 80% dos servidores estarão disponíveis após uma falha, em uma rede na qual o pior caso de latência poderia chegar a 33 ms.*

***Abstract.*** *A current trend in data center design is to geo-distribute its servers, allowing better proximity to the end users and higher survivability. As a shortcoming, this solution increases the latency between servers, caused by a greater distance between them. In this work we tackle the trade-off between survivability and latency in geo-distributed data centers, through the formulation of an optimization problem. The results obtained from realistic scenarios show that, for wide area networks, the latency increase is significant only in case of very strong survivability requirements, whereas it is negligible for moderate requirements. For instance, the worst-case latency is less than 4 ms when guaranteeing that 80% of servers are available after a failure, in a network where the maximum possible latency is 33 ms.*

## 1. Introdução

A computação em nuvem, através da virtualização, pode usufruir da consolidação de servidores de modo a facilitar o gerenciamento da infraestrutura. Por outro lado, a virtualização também possibilita que uma infraestrutura de computação em nuvem possua seus servidores espalhados por diversos sítios, o que pode tornar o centro de dados (DC - *Data Center*) mais próximo do usuário e com maior capacidade de sobrevivência

*Este trabalho foi realizado com recursos da FAPERJ, CNPq e CAPES, e do projeto NU@GE FSN.

a falhas. De fato, a tendência atual em redes de centro de dados é criar uma arquitetura geograficamente distribuída, utilizando alguns poucos sítios de uma rede de longa distância (WAN - *Wide Area Network*), de forma a reduzir a latência percebida pelos usuários finais [Lam et al., 2010]. Além disso, a distribuição geográfica tem a vantagem de permitir uma maior capacidade de sobrevivência a falhas nos sítios e enlaces da rede. As máquinas virtuais (VMs - *Virtual Machines*) alocadas para um determinado usuário podem, dessa forma, estar distribuídas em diversas localidades definidas pelo mecanismo de orquestração de recursos da nuvem [Endo et al., 2011]. Em suma, as principais motivações para a construção de um DC geograficamente distribuído são:

- o aumento da sobrevivência do DC, devido à redução de pontos únicos de falha;
- a redução do atraso para acesso à nuvem do ponto de vista dos usuários finais, devido a uma maior proximidade física do DC com alguns clientes;
- a possibilidade de ampliar o DC mesmo com limitações de capacidade (fornecimento de energia, espaço físico, etc) de um sítio.

Este trabalho foca no primeiro item, que diz respeito à sobrevivência. A sobrevivência de uma infraestrutura de nuvem pode ser aumentada pela distribuição do DC em diversos sítios dentro de uma determinada região geográfica. Quanto maior a região, menor o risco de o DC inteiro ser afetado por um pequeno conjunto de falhas inter-relacionadas como, por exemplo, rompimento de fibras ópticas, quedas de energia ou outros desastres de larga escala [Grover, 2004]. Um exemplo de desastre de larga escala que afetou a comunidade de redes recentemente foi o furacão Sandy em Novembro de 2012: os servidores da *IEEE Communications Society* (web, email, FTP, DNS, etc.) permaneceram completamente desconectados durante 5 dias.

Apesar de a distribuição geográfica possuir efeitos positivos, o projeto desse tipo de DC deve considerar o aumento da latência da comunicação entre seus servidores causado pela maior distância entre eles, além do custo necessário para interconectar sítios em WANs. O último aspecto geralmente depende de vários fatores externos, como a matriz de tráfego do DC e o montante de investimento necessário para sua construção física. O primeiro aspecto, entretanto, é de natureza operacional e possui crescente importância em redes para nuvem, visto que um acréscimo na latência de apenas alguns milissegundos pode causar considerável impacto nos serviços fornecidos [Develder et al., 2012b]. Assim, este trabalho foca no compromisso entre capacidade de sobrevivência e latência de interconexão no projeto de DCs geograficamente distribuídos. Este artigo objetiva responder às seguintes questões:

- O quanto a latência aumenta quando a sobrevivência do centro de dados é melhorada a partir da distribuição geográfica?
- Qual é a influência da capacidade (número de servidores suportados) dos sítios presentes na WAN no compromisso entre latência e sobrevivência?

Para responder essas questões, neste trabalho o projeto de DCs geograficamente distribuídos é modelado como um problema de otimização, possuindo ambos objetivos de sobrevivência e latência. Em um panorama geral, o projeto de DCs geograficamente distribuídos levando em consideração a sobrevivência começou a ser abordado recentemente na literatura [Develder et al., 2012a, Xiao et al., 2014, Habib et al., 2012]. O estado da arte foca no provisionamento de capacidade de fibra óptica entre os sítios. O presente trabalho se diferencia da literatura existente pois aborda a otimização da latência

e da sobrevivência. Assim, preenche uma lacuna no projeto de DCs geograficamente distribuídos, pois foca nesses dois importantes objetivos, ignorando outros fatores como a capacidade da rede. Os resultados de simulação mostram que um nível moderado de sobrevivência consegue ser alcançado em redes em malha WAN, sem comprometer significativamente a latência. Considerando todas as redes analisadas, o problema proposto encontra configurações de DC que, após a falha em um elemento, garantem a sobrevivência de 80% dos servidores, enquanto o aumento de latência em relação ao uso de apenas um sítio (sobrevivência zero) é de 3,6 ms. Por outro lado, o aumento da capacidade de sobrevivência, quando esta já possui um nível muito elevado, acarreta em um alto aumento da latência. Por exemplo, o aumento da sobrevivência de 94% para 95% pode levar a um aumento de 46% na latência.

Este artigo está organizado da seguinte forma. A Seção 2 apresenta o modelo de DC utilizado e os critérios escolhidos. A Seção 3 descreve o problema de otimização proposto, enquanto a Seção 4 apresenta os resultados. A Seção 5 descreve os trabalhos relacionados. Por fim, a Seção 6 conclui o trabalho e identifica direções futuras.

## 2. Modelo da Rede do Centro de Dados

O modelo da rede utilizado neste trabalho é baseado nas seguintes considerações:

- a menor unidade do DC é um bastidor (*rack*), que é um conjunto de servidores interligados. Em um sítio, os bastidores são interligados por uma topologia intra-sítio como, por exemplo, árvore, Fat-Tree, BCube, DCell, entre outras [Verdi et al., 2010]. Neste trabalho, a sobrevivência da topologia intra-sítio não é considerada, sendo abordada especificamente em [Couto et al., 2012];
- diversos sítios são dispostos em uma determinada região geográfica, nos quais é possível instalar um certo número de bastidores. Um sítio ativo é aquele que possui pelo menos um bastidor instalado e operacional;
- os usuários do DC acessam os serviços através de *gateways* espalhados pela rede;
- o DC é suscetível a falhas de enlace e nó. O nó é um roteador da rede WAN ou um sítio, enquanto o enlace é o meio físico que interconecta os nós, ou que provê acesso aos *gateways*. Cada enlace ou nó pertence a um ou mais grupos de risco compartilhado (SRGs - *Shared Risk Groups*). Um SRG é definido como um grupo de elementos suscetíveis a uma falha comum [Habib et al., 2013]. Por exemplo, um SRG pode ser composto de sítios ligados a uma mesma subestação elétrica.

A Figura 1 ilustra um exemplo do cenário descrito acima. Os sítios, que são interconectados pela WAN, podem hospedar diferentes números de bastidores. Dependendo da topologia da WAN e do número de *gateways*, a rede pode apresentar diferentes níveis de sobrevivência e latência de interconexão, como descrito nos próximos parágrafos.

O modelo descrito acima captura as características de um DC que fornece, principalmente, sua infraestrutura como serviço (IaaS - *Infrastructure as a Service*). Em serviços IaaS, cada cliente possui diversas VMs alocadas no DC. Além disso, o DC gerencia os recursos destinados às VMs, provendo alocação de recursos computacionais, enlaces virtuais, migração de VMs, etc. Um bastidor pode hospedar VMs de diversos clientes, e as VMs de cada cliente podem ser espalhadas por diferentes bastidores e sítios, de forma a aumentar a sobrevivência e a elasticidade.

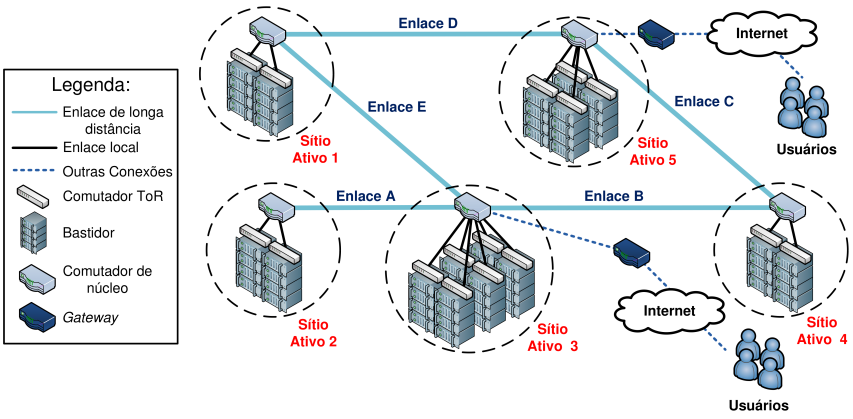



**Figura 1. Exemplo de um DC geodistribuído, composto por diferentes sítios.**

Como as VMs de um cliente podem estar geograficamente distribuídas, a sobrevivência de um serviço IaaS pode ser aumentada espalhando seus recursos entre diferentes sítios. Entretanto, como as VMs podem possuir padrões de comunicação entre elas (p.ex., o tráfego interno a um DC pode ser maior que o tráfego oriundo de usuários [Cisco, 2013]), a geodistribuição aumenta a distância entre bastidores e assim pode afetar o desempenho de suas aplicações. Note que este trabalho considera o projeto físico do DC, sendo a alocação de VMs executada por outro algoritmo que considera a infraestrutura do DC construída. Evidentemente, uma infraestrutura na qual o compromisso entre latência e sobrevivência é bem ajustado pode prover uma melhor alocação de VMs. Esse problema é uma preocupação constante em redes para nuvem, especialmente para redes de armazenamento de dados, nas quais um acréscimo de apenas 1 ms no RTT (*Round-Trip Time*) pode representar uma degradação de desempenho considerável [Telikepalli et al., 2004]. A seguir são detalhados os dois objetivos analisados.

## 2.1. Sobrevivência

Para quantificar a sobrevivência de um DC geodistribuído, utiliza-se neste trabalho o conceito de "sobrevivência de pior caso" definido em [Bodík et al., 2012]. Essa métrica é definida como a fração dos serviços do DC que permanecem operacionais após o pior caso de falha de um SRG. Neste trabalho, esse conceito é utilizado através da definição de "sobrevivência", como *a menor fração dos bastidores que permanecem disponíveis após a falha de um único SRG, considerando todos os possíveis SRGs*. Essa definição é apropriada para um DC geodistribuído que fornece serviços IaaS, uma vez que possuir menos de 100% dos bastidores disponíveis não significa que o DC perdeu sua capacidade de hospedar VMs. Formalmente, a métrica de sobrevivência pode ser calculada por:

$$s = \min_{f \in \mathcal{F}} \left( \frac{\sum_{k \in \mathcal{A}_f} r_k}{R} \right), \tag{1}$$

onde $\mathcal{F}$ é o conjunto formado por todos os SRGs, $R$ é o número total de bastidores, $\mathcal{A}_f$ é o conjunto das sub-redes acessíveis após a falha do SRG $f$, e $r_k$ é o número de bastidores na sub-rede acessível $k \in \mathcal{A}_f$. Uma sub-rede acessível é definida como uma parte da rede que está isolada das outra sub-redes, mas possui acesso a pelo menos um *gateway*. Note que, após uma falha, a rede pode ser particionada em diferentes sub-redes. Se uma

sub-rede não possui acesso a um *gateway*, seus bastidores não estão acessíveis ao mundo externo e então não podem fornecer os serviços IaaS. Por exemplo, se na Figura 1 os enlaces B e D falharem, a rede é dividida em duas sub-redes acessíveis: uma composta pelos sítios 1, 2 e 3, e outra composta pelos sítios 4 e 5. Considerando outro cenário, no qual apenas o enlace A falha, a rede se particiona em duas sub-redes. Uma delas é acessível e composta pelos sítios 1, 3, 4 e 5. A outra, composta pelo sítio 2, não é acessível pois não possui caminhos para um *gateway*. Assim, $\sum_{k \in \mathcal{A}_f} r_k$ na Equação 1 é o número total de bastidores acessíveis após a falha do SRG $f$.

De acordo com a definição fornecida acima, a métrica de sobrevivência assume valores no intervalo $[0, 1]$. Seu valor mínimo (zero) ocorre quando todos os bastidores são afetados por um mesmo SRG. O valor máximo, por sua vez, ocorre quando a rede tem um certo nível de redundância e o DC é distribuído de forma que nenhuma falha única de SRG pode desconectar um bastidor.

### 2.2. Latência de interconexão

Neste trabalho assume-se que a latência de interconexão entre os diferentes sítios do DC é afetada principalmente pelo atraso de propagação de seus caminhos. Assim, considera-se que a capacidade de rede é bem provisionada, tornando desprezível o atraso devido a congestionamento e retransmissões nos nós intermediários. Partindo desse pressuposto, é apropriado quantificar a latência de interconexão como *o maior atraso entre pares de sítios ativos, considerando todas as possíveis combinações*. A escolha do valor máximo como métrica de referência é importante para considerar o fato de que VMs alocadas para um determinado cliente podem ser espalhadas por diversos sítios. Assim, a latência máxima corresponde ao pior caso, no qual a consolidação de VMs é realizada independentemente da localização dos sítios ou quando não há capacidade suficiente para realizar uma melhor alocação. Formalmente, a latência de interconexão é definida como:

$$l = \max_{i,j \in \mathcal{D}} (\Delta_{ij} u_i u_j), \tag{2}$$

onde $\mathcal{D}$ é o conjunto de todos os sítios, ativos ou não, $\Delta_{ij}$ é o atraso de propagação entre os sítios $i$ e $j$ como definido acima, e $u_i$ é uma variável binária indicando se o sítio $i$ está ativo ou não (em outras palavras, se ele possui pelo menos um bastidor instalado). Considera-se que $\Delta_{ij} = \Delta_{ji}$, uma vez que os caminhos em WANs são geralmente configurados simetricamente. É importante salientar que, neste trabalho, a latência de interconexão é calculada sem considerar situações de falhas, de forma a melhor analisar o compromisso entre latência e sobrevivência. Entretanto, após uma falha, caminhos alternativos podem ser escolhidos. Se esse caminhos possuírem maiores comprimentos, a latência aumenta.

## 3. Formulação do Problema de Projeto de DCs Geodistribuídos

O problema de projeto de DCs geodistribuídos, formulado neste trabalho com programação linear inteira mista (MILP - *Mixed Integer Linear Programming*), possui o duplo objetivo de maximizar a sobrevivência enquanto minimiza a latência de interconexão. A otimização considera como parâmetros a latência entre os sítios, o número de bastidores suportados em cada sítio, informação sobre SRGs e o número total de bastidores a serem posicionados. A saída do problema fornece a quantidade de bastidores alocados para cada sítio. A Tabela 1 resume as notações utilizadas e seus tipos. Notações

**Tabela 1. Notações utilizadas no problema.**

| Notação | Descrição | Tipo |
|---|---|---|
| $\mathcal{D}$ | Sítios candidatos | Conjunto |
| $\mathcal{F}$ | SRGs | Conjunto |
| $M_{fi}$ | Valor binário indicando se o SRG $f$ desconecta da rede o sítio $i$ | Parâmetro |
| $\Delta_{ij}$ | Atraso de propagação (latência) entre os sítios $i$ e $j$ | Parâmetro |
| $L_{max}$ | Valor máximo de latência entre todos os sítios da rede, ativos ou não. | Parâmetro |
| $R$ | Número total de bastidores a serem posicionados | Parâmetro |
| $Z_i$ | Capacidade (máximo número de bastidores suportados) do sítio $i$ | Parâmetro |
| $\beta$ | Valor de ajuste da importância da latência em relação à sobrevivência | Parâmetro |
| $s$ | Sobrevivência do DC | Variável |
| $l$ | Latência de interconexão do DC | Variável |
| $x_i$ | Número de bastidores na localização $i$ | Variável |
| $u_i$ | Valor binário indicando se o sítio $i$ está ativo ($x_i > 0$) | Variável |

do tipo conjunto e parâmetro se referem aos dados do problema, enquanto as variáveis são ajustadas pelo algoritmo de otimização. A formulação MILP é apresentada a seguir:

$$\text{maximizar } (1-\beta)s - \beta\frac{l}{L_{max}} \tag{3}$$

$$\text{sujeito a } \sum_{i\in\mathcal{D}} M_{fi}x_i - sR \geq 0 \quad \forall f \in \mathcal{F}. \tag{4}$$

$$l - \Delta_{ij}u_i - \Delta_{ij}u_j \geq -\Delta ij \quad \forall i,j \in \mathcal{D},\ i<j. \tag{5}$$

$$Ru_i - x_i \geq 0 \quad \forall i \in \mathcal{D}. \tag{6}$$

$$u_i \leq x_i \quad \forall i \in \mathcal{D}. \tag{7}$$

$$\sum_{i\in\mathcal{D}} x_i = R. \tag{8}$$

$$x_i \leq Z_i \quad \forall i \in \mathcal{D}. \tag{9}$$

$$s \geq 0, \quad l \geq 0, \quad x_i \geq 0 \,\forall\, i \in \mathcal{D}. \tag{10}$$

$$s \in \mathbb{R}; \quad l \in \mathbb{R}; \quad u_i \in \{0,1\},\ \forall\, i \in \mathcal{D}; \quad x_i \in \mathbb{Z},\ \forall\, i \in \mathcal{D}. \tag{11}$$

O objetivo dado pela Equação (3) maximiza a sobrevivência $s$, como definida na Equação 1, enquanto minimiza a latência $l$, definida na Equação 2. O compromisso entre latência e sobrevivência é ajustado na Equação (3) pelo fator de escala $0 \leq \beta \leq 1$. Além disso, $L_{max} = \max_{i,j\in\mathcal{D}}(\Delta_{ij})$ normaliza $l$ no intervalo $[0,1]$. Assim, $l$ e $s$ assumem valores dentro do mesmo intervalo.

Como as Equações 1 e 2 não são lineares, a linearização de cada uma é dada respectivamente pelas Equações (4) e (5). Para a sobrevivência, aplicar a Equação (4) é equivalente a forçar $s$ a ser menor ou igual ao valor definido na Equação 1. Como a Equação (3) tenta aumentar a sobrevivência, $s$ assumirá o valor mais alto, que é dado de fato pela Equação 1. Utilizando o mesmo princípio, a Equação (5) força $l$ a assumir a máxima latência entre dois sítios ativos. Para considerar apenas sítios ativos no cálculo de $l$, utiliza-se as variáveis binárias $u_i, i \in \mathcal{D}$. Assim, se $u_i$ ou $u_j$ possuírem valor zero

para um determinado par de sítios, a restrição dada pela Equação (5) não será efetiva para esse par. Por exemplo, se $u_i = 0$ e $u_j = 1$, a restrição será $l \geq 0$. Os valores binários $u_i$ são definidos pelas Equações (6) e (7), fazendo $u_i = 0$ se $x_i = 0$ e $u_i = 1$ se $x_i > 0$. A Equação (8) restringe o número total de bastidores do DC ($R$), enquanto a Equação (9) limita o número de bastidores ($x_i$) permitido em cada sítio $i$, respeitando sua capacidade $Z_i$. Finalmente, as Equações (10) e (11) definem, respectivamente, os limitantes inferiores e o domínio de cada variável.

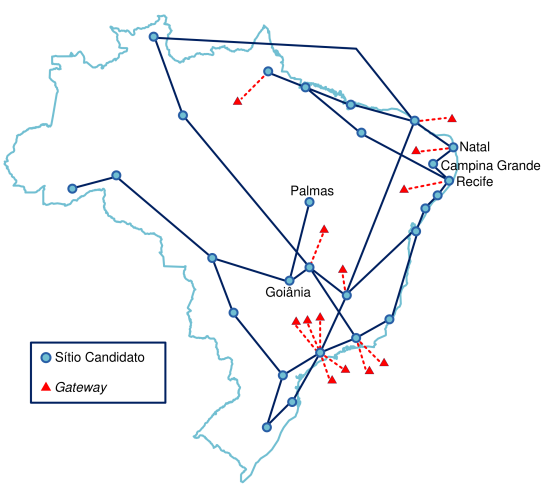
Para uma dada rede, os parâmetros de latência $\Delta_{ij}$ são calculados a partir dos caminhos mais curtos entre os sítios $i$ e $j$. Os parâmetros binários $M_{fi}$, para um SRG $f$, são obtidos pela remoção de todos os elementos (nós e enlaces) que pertencem a esse SRG. Assim, após a remoção, verifica-se para cada SRG quais os sítios possuem acesso aos *gateways*. Obviamente, se um sítio pertence a um determinado SRG, ele já é considerado como desconectado na análise desse SRG.

## 4. Avaliação do Compromisso entre Latência e Sobrevivência

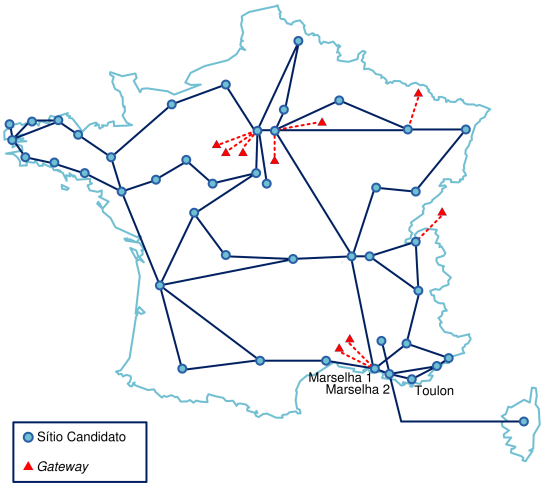
A análise deste trabalho utiliza topologias de WANs reais de educação e pesquisa, as quais são formadas por diversos pontos de presença (PoP - *Point of Presence*). Considera-se que cada PoP dessas redes é um sítio candidato a hospedar bastidores. Neste trabalho, são adotadas as topologias da RNP (Figura 2(a)) no Brasil, RENATER (Figura 2(b)) na França, e GEANT (Figura 2(c)) na Europa. Cada figura das redes utilizadas mostra seus sítios e *gateways*. Para maior clareza, são especificados nas figuras apenas os nomes de sítios mencionados ao longo do texto. Note que cada topologia cobre uma área de tamanho diferente. Em relação à RENATER, as redes RNP e GEANT cobrem uma área muito maior, com uma superfície mais de 10 vezes maior que a França metropolitana. Entretanto, a rede francesa possui mais nós que a RNP. Por fim, a GEANT possui um número de sítios próximo à RENATER.

O problema MILP é resolvido neste trabalho utilizando a ferramenta IBM ILOG CPLEX 12.5.1. Além disso, cada nó ou enlace da rede é um SRG. Dessa forma, considera-se o modelo de falha única (*single failure model*) [Habib et al., 2013]. A distância de um enlace é estimada como o comprimento de uma linha reta entre os centros das duas cidades que hospedam os sítios. Assim, o atraso de propagação é diretamente proporcional à distância, e utiliza-se uma velocidade de propagação de $2 \times 10^8$ m/s, comumente adotada para análise de redes ópticas [Ramaswami et al., 2009]. Para cálculo dos parâmetros da rede, utiliza-se a ferramenta de análise de grafos NetworkX.

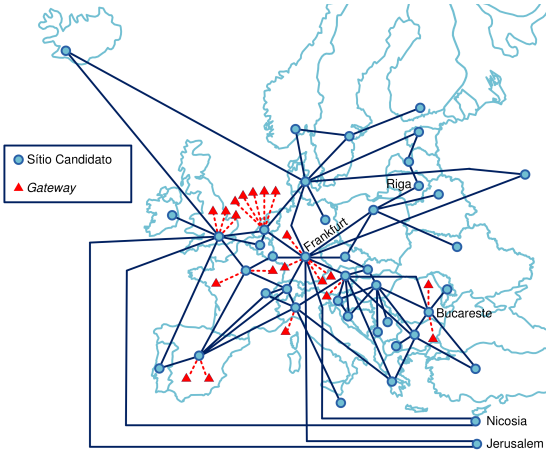
O número total de bastidores considerado nesta avaliação é $R = 1024$. Esse valor foi arbitrariamente escolhido, já que a alocação de bastidores depende da relação entre $R$ e a capacidade $Z_i$ de cada sítio $i$, e não de valores absolutos. Além disso, a análise é realizada para diferentes valores de $Z_i$ considerando, para simplificar, que todos os sítios possuem a mesma capacidade. Por fim, o compromisso entre sobrevivência e latência é analisado variando o parâmetro $\beta$ da Equação (3) de $0$ até $1$ em passos de $0,05$. Considerando todas as rodadas (isto é, todas as escolhas de $\beta$ para todas redes e valores de $Z_i$), em 89% delas o problema de otimização encontrou a solução em menos de 30 s em um PC com processador i7-3930K, e em 97% das rodadas a solução foi encontrada em menos de 5 min. Apenas 1 rodada (0,4% do número de rodadas) durou mais de 30 min, possuindo a duração de 117 min. Note que a complexidade dos métodos de




(a) RNP, 27 sítios conectados por 33 enlaces.




(b) RENATER , 45 sítios conectados por 54 enlaces.



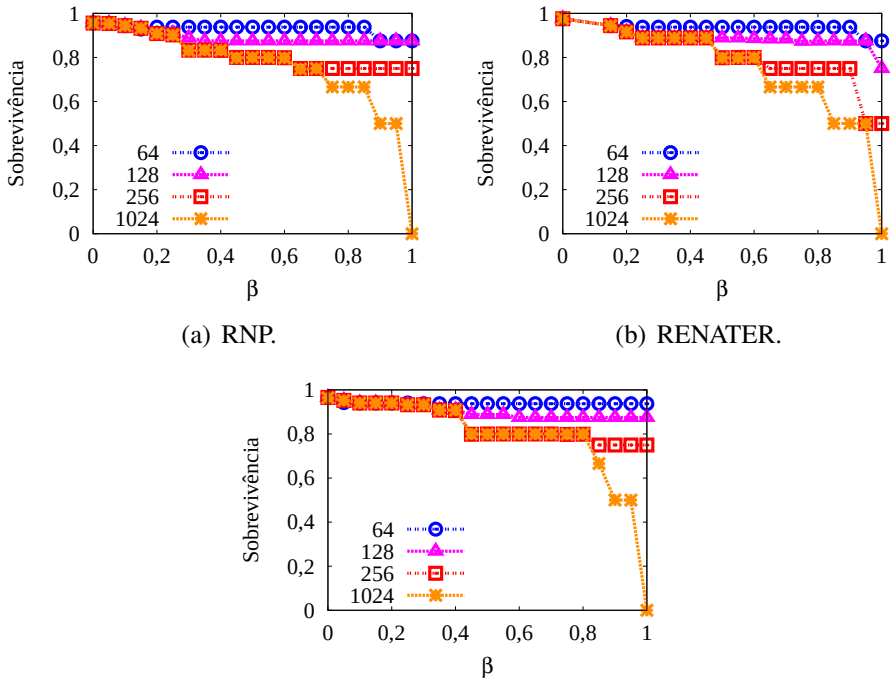

(c) GEANT, 42 sítios conectados por 68 enlaces.

**Figura 2. Topologias de redes de ensino e pesquisa consideradas na avaliação.**

otimização utilizados não apresenta barreiras para o cenário considerado, que geralmente possui poucas variáveis (p.ex. dezenas de sítios e de SRGs).

As Figuras 3 e 4 mostram, respectivamente, os valores de sobrevivência e latência obtidos para todas as redes analisadas. Cada curva representa uma capacidade $Z_i$ diferente (64, 128, 256, ou 1024), atribuída a todos os sítios. Como o número total de bastidores é constante, a redução da capacidade dos sítios força o problema a encontrar soluções com mais sítios ativos. Por exemplo, fixando a capacidade em 64, impõe-se a utilização de pelo menos $\frac{1024}{64} = 16$ sítios ativos. Por outro lado, fixando a capacidade em 1024 é o mesmo que desconsiderar a capacidade no problema, visto que um sítio poderá hospedar todos os bastidores. Como esperado, os resultados mostram que o aumento de $\beta$ compromete a sobrevivência enquanto melhora a latência (Figura 4). Além disso, ao diminuir a capacidade dos sítios o problema de otimização possui menos escolhas de alocação de bastidores, mantendo as métricas constantes para uma faixa de $\beta$ mais larga.

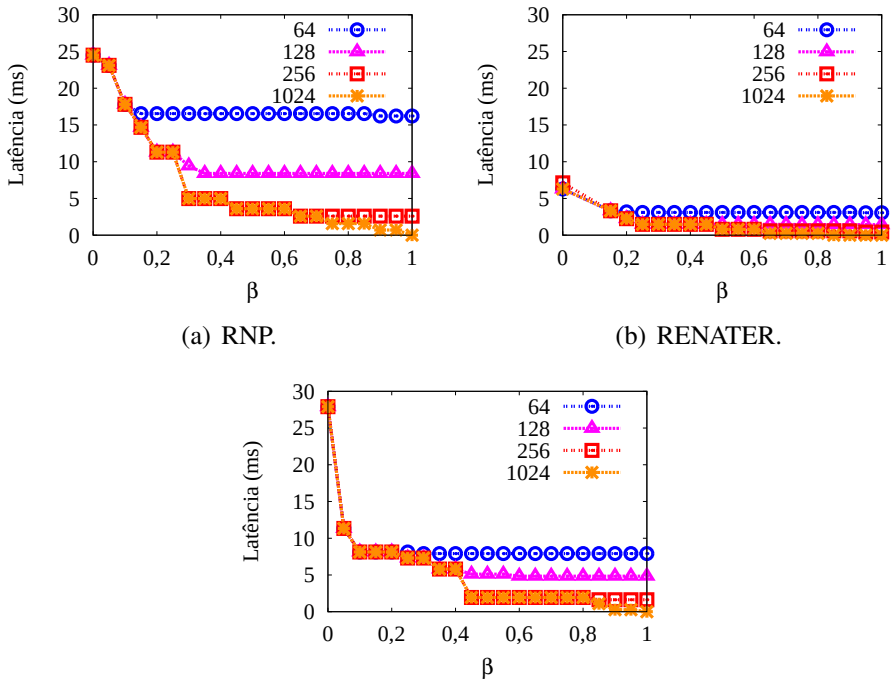
Como mostrado também nas Figuras 3 e 4, a capacidade dos sítios influencia os valores mínimos possíveis de sobrevivência e latência (valores alcançados para $\beta = 1$).




(a) RNP. (b) RENATER.

(c) GEANT.

**Figura 3. Sobrevivência em um DC com 1024 bastidores.**




(a) RNP. (b) RENATER.

(c) GEANT.

**Figura 4. Latência em um DC com 1024 bastidores.**

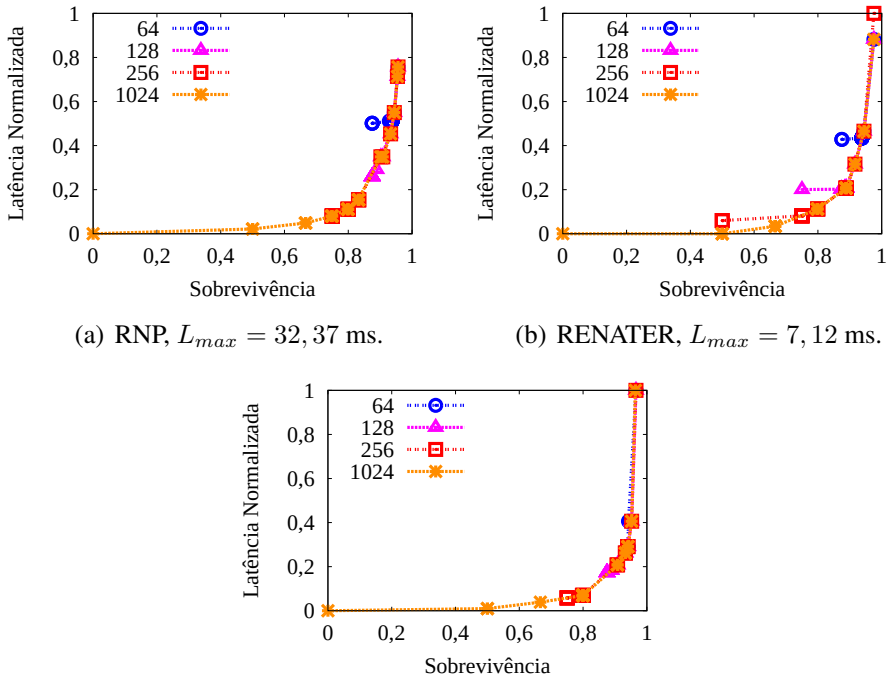
Essa influência é consequência do número mínimo de sítios ativos imposto pela capacidade, como visto mais adiante. Finalmente, pode-se observar que os valores mais baixos

de latência são alcançados pela rede RENATER, visto que ela cobre uma área menor que a RNP e a GEANT. Consequentemente, os caminhos na RENATER tendem a ser mais curtos se comparados com as outras duas redes.

Para melhor analisar o compromisso abordado neste trabalho, a Figura 5 mostra, para todas as redes, a latência normalizada em relação à sobrevivência. A latência normalizada é simplesmente $\frac{l}{L_{max}}$. $L_{max}$ é o maior valor possível de latência entre dois sítios, como definido anteriormente, e seu valor para cada rede está indicado na legendas dos gráficos. Cada ponto da Figura 5 é obtido plotando o valor de latência e o valor de sobrevivência alcançados para um mesmo $\beta$. Os resultados mostram um comportamento similar para todas as topologias: para valores altos de sobrevivência, um pequeno ganho na sobrevivência representa um alto acréscimo na latência. Isso ocorre pois, em todas as redes consideradas, sempre existem alguns nós com caminhos significativamente longos entre si. Assim, quando os requisitos de sobrevivência aumentam (i.e. $\beta$ diminui), esses nós são escolhidos devido à falta de opções. Consequentemente, uma pequena melhora na sobrevivência acarreta em um severo acréscimo na latência. Como exemplo, para uma capacidade de 1024 e $\beta = 0$ na GEANT, entre os sítios ativos encontram-se os de Nicósia (Chipre) e Jerusalém (Israel), cada um hospedando 34 bastidores. Apesar de esses sítios estarem próximos geograficamente, o caminho na rede entre eles é bastante longo. O tráfego entre eles deve passar por um nó em Frankfurt (Alemanha), percorrendo no total uma distância de 5.581 km, resultando em uma latência de 27,9 ms. Quando o valor de $\beta$ é incrementado para 0,05, o que representa uma diminuição pequena da importância da sobrevivência, o pior caso de latência passa a ser o caminho entre Riga (Letônia) e Bucareste (Romênia). Esse caminho possui um comprimento de 2.267 km, possuindo uma latência de 11,33 ms. Note então que a latência sofre uma alta redução (46%) para uma pequena variação de $\beta$, como visto na Figura 4(c). Entretanto, a consequente redução de sobrevivência é de apenas 2%, como visto na Figura 3(c).

Inversamente ao exposto acima, outro comportamento é observado para todas as redes: para valores mais baixos de sobrevivência, um aumento significativo da sobrevivência resulta em uma piora insignificante na latência. Como exemplo, a Figura 5(b) mostra que variando a sobrevivência de 0,5 para 0,75, com uma capacidade de 256 bastidores na RENATER, a latência normalizada possui um aumento desprezível de 0,06 ($l = 0,42ms$) para 0,08 ($l = 0,58ms$). Esse comportamento mostra que o DC pode atingir valores satisfatórios de sobrevivência com um acréscimo muito baixo na latência. Considerando todas as redes, o maior aumento na latência ao melhorar a sobrevivência de 0 para 0,5 é de 0,70 ms, que ocorre na rede RNP. Nessa mesma rede, uma sobrevivência de 0,8 é alcançada com um acréscimo de apenas 3,6 ms em relação ao caso de um único sítio ativo. Os baixos valores de latência alcançados, mesmo com valores de sobrevivência satisfatórios, são consequência de uma característica comum a todas as redes: todas as topologias consideradas possuem uma alta densidade de sítios em uma determinada região e eles não desconectam simultaneamente após a falha de qualquer SRG. Como exemplo, podem ser citados os sítios no Nordeste do Brasil (Natal, Campina Grande, Recife, etc.) e no Sul da França (Toulon, os dois sítios em Marselha, etc.). Assim, o DC pode ser espalhado nessas regiões sem um aumento significativo da latência.

Em suma, se a exigência de sobrevivência do DC não for muito alta, é fácil assegurar valores moderados de sobrevivência sem causar um aumento significativo da latência.




(a) RNP, $L_{max} = 32,37$ ms. (b) RENATER, $L_{max} = 7,12$ ms.

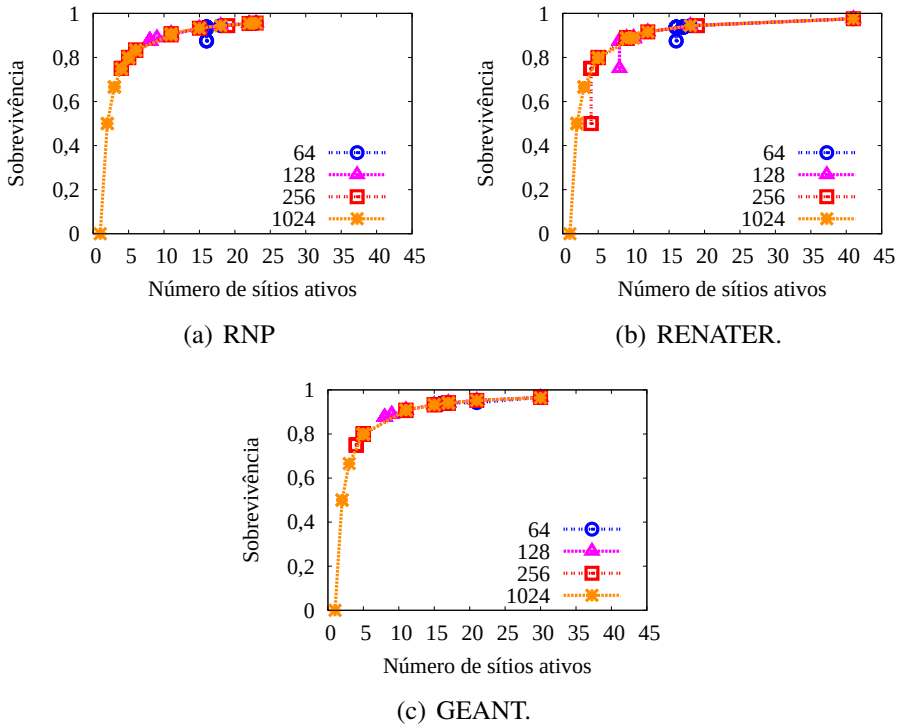
(c) GEANT, $L_{max} = 27,9$ ms.

**Figura 5. Latência versus sobrevivência.**

Entretanto, aumentar a sobrevivência para valores próximos de 1 produz um aumento grande da latência entre os bastidores, podendo impactar o desempenho de aplicações com exigências mais estritas de latência. É importante salientar que essas conclusões são válidas para todas as topologias consideradas, e os indícios apontam que são independentes da área geográfica atendida pela WAN e do número de nós e enlaces.

A Figura 5 também mostra que, ao diminuir a capacidade dos sítios, a latência mínima possível mantém-se a mesma, ou aumenta. No último caso, isso ocorre apenas para o mínimo valor de sobrevivência ($\beta = 1$) e, após isso, os pontos do experimento encontram-se na mesma curva na qual a capacidade é desconsiderada (capacidade de 1024). Esse comportamento ocorre, pois a limitação da capacidade reduz as possibilidades de alocação de bastidores, podendo inviabilizar soluções melhores. Para $\beta = 1$, o problema não considera a sobrevivência e tenta melhorar a latência o quanto possível. Entretanto, como a gama de soluções possíveis é limitada pelo número mínimo de sítios (p.ex. 16 para uma capacidade de 64), o problema tende a escolher sítios que estejam mais próximos geograficamente de forma a melhorar a latência. Esses sítios mais próximos tendem a estar no mesmo SRG. Por exemplo, considerando a RNP com uma capacidade de 64, a redução ocorre pois a escolha ótima é alocar 2 sítios afetados pelo mesmo SRG: Goiânia e Palmas. Para a RNP isso ocorre apenas para a capacidade de 64, enquanto para a RENATER esse comportamento começa a partir de uma capacidade de 256. Para a GEANT, todos os pontos encontram-se na mesma curva, independente da capacidade.

A Figura 6 mostra a métrica de sobrevivência em função do número de sítios ativos, para o mesmo experimento realizado anteriormente. Apesar de a otimização não controlar diretamente essa métrica, o acréscimo da sobrevivência é influenciado pela




(a) RNP (b) RENATER.

(c) GEANT.

**Figura 6. Sobrevivência em função do número de sítios ativos.**

distribuição do DC. Por sua vez, a distribuição do DC é influenciada pelo número de sítios ativos. Quanto mais sítios são usados, os bastidores do DC tendem a pertencer a mais SRGs e assim a sobrevivência tende a aumentar. Os resultados mostram que a sobrevivência cresce de forma logarítmica com o número de sítios ativos. Isso significa que o aumento do número de sítios ativos não melhora significativamente a sobrevivência quando o DC se torna amplamente distribuído em diferente regiões. Além disso, os resultados mostram que a rede pode possuir diferentes níveis de sobrevivência para um mesmo número de sítios ativos. Esse comportamento é observado apenas para o valor mínimo de sítios ativos impostos pela capacidade (p.ex., $\frac{1024}{256} = 4$ para uma capacidade de 256). A razão é a mesma daquela observada na Figura 5, na qual ao fazer $\beta = 1$ obtém-se uma alta redução de sobrevivência e uma pequena melhora na latência.

## 5. Trabalhos Relacionados

O projeto de DCs geodistribuídos possui diversos fatores em comum com o projeto de redes para computação em grade. Essas redes também podem exigir uma infraestrutura geograficamente espalhada para executar cálculos distribuídos como, por exemplo, em aplicações de e-ciência. [Develder et al., 2011] formula um problema de otimização para o projeto de redes para computação em grade considerando a sobrevivência, que pode também ser utilizado para DCs, como mostrado pelos autores em [Develder et al., 2012a]. O cenário de [Develder et al., 2011] consiste em diversas fontes que submetem tarefas para nós de destino. Os nós de destino possuem servidores, e todos os nós (fontes e destinos) são interconectados por uma rede óptica. Assim, o trabalho propõe um problema de otimização que determina quais nós da rede executarão cada tarefa, as rotas ópticas entre as fontes e destinos e a capacidade, em comprimen-

tos de onda, alocada para cada rota. O objetivo do problema é minimizar a capacidade necessária nos servidores e na rede. Para prover sobrevivência, o problema escolhe destinos e rotas alternativos para cada serviço. Problemas similares são propostos em [Xiao et al., 2014] e [Habib et al., 2012] para DCs. Uma característica comum dos trabalhos mencionados é o fato de a função objetivo do problema de otimização minimizar o custo para prover capacidade de rede e de servidores, deixando a sobrevivência como uma restrição. Além disso, esses trabalhos assumem que todos os serviços do DC e suas respectivas demandas são conhecidos no momento de sua construção. O atraso de propagação devido à distribuição geográfica é considerado apenas em [Xiao et al., 2014], porém esse trabalho não analisa o compromisso entre latência e sobrevivência.

O presente trabalho é diferente do atual estado da arte pois a função objetivo do problema de otimização consiste em métricas de latência e sobrevivência, de forma a explorar o compromisso entre esses fatores. Assim, este trabalho isola essas duas métricas, ignorando outros fatores como o custo (p.ex., a capacidade utilizada pelas rotas e o custo para ativar um sítio). Além disso, as conclusões apresentadas aqui são independentes dos serviços suportados pelo DC e da matriz de tráfego gerada por esses serviços. O custo e a matriz de tráfego são, sem dúvida, fatores importantes no projeto de um DC. Entretanto, esses fatores devem ser ignorados em uma primeira análise, para melhor observar o compromisso entre latência e sobrevivência.

## 6. Conclusões

Neste trabalho analisou-se o compromisso entre latência e sobrevivência no projeto de centros de dados (DC) geodistribuídos em redes de longa distância (WANs). As simulações realizadas em cenários diversos e realísticos mostram que quando a exigência de sobrevivência do DC é muito alta, uma pequena melhora na sobrevivência acarreta em um aumento significativo na latência. Esse comportamento é essencialmente devido à presença de caminhos muito longos para alguns poucos pares de sítios. Em níveis altos de sobrevivência, os resultados mostram que um aumento de 2% (de 0,94 para 0,96) na sobrevivência pode aumentar a latência em 46% (de 11,33 ms para 27,9 ms). Por outro lado, quando a exigência de sobrevivência permanece moderada, esta pode ser consideravelmente melhorada com um baixo acréscimo na latência. Considerando todas as redes analisadas, o máximo aumento na latência é de 0,7 ms quando a sobrevivência é melhorada de 0 para 0,5. Além disso, observa-se uma máxima latência de 3,6 ms quando a sobrevivência é 0,8. Por fim, os resultados mostram que a diminuição da capacidade dos sítios provoca um aumento nos menores valores possíveis de sobrevivência e latência (i.e. métricas alcançadas para $\beta = 1$). Apesar disso, o compromisso entre latência e sobrevivência não mudam significativamente para diferentes capacidades dos sítios.

Os cenários atuais de infraestrutura como serviço (IaaS) apresentam geralmente centros de dados monolíticos com um único ou um número baixo de sítios, provendo baixos níveis de sobrevivência a falhas, como frequentemente observado por usuários na Internet. Este estudo sugere que, considerando um cenário realístico de projeto de DCs em WANs, aumentar a sobrevivência a um nível moderado apresenta pouco ou nenhum impacto no desempenho de um IaaS.

Como trabalho futuro, pretende-se estender o estudo para incluir também falhas de larga escala modelando, por exemplo, grandes catástrofes naturais. Esse tipo de falha

pode afetar diversos sítios em uma grande área geográfica. Outra direção futura é adicionar outros objetivos ou restrições ao problema de otimização, como o custo da rede e a latência do ponto de vista dos usuários.

## Referências


Bodík, P., Menache, I., Chowdhury, M., Mani, P., Maltz, D. A. e Stoica, I. (2012). Surviving failures in bandwidth-constrained datacenters. Em *ACM SIGCOMM*, p. 431–442.

Cisco (2013). Cisco global cloud index: Forecast and methodology, 2012–2017.

Couto, R. S., Campista, M. E. M. e Costa, L. H. M. K. (2012). A reliability analysis of datacenter topologies. Em *IEEE GLOBECOM*, p. 1890—1895.

Develder, C., Buysse, J., De Leenheer, M., Jaumard, B. e Dhoedt, B. (2012a). Resilient network dimensioning for optical grid/clouds using relocation. Em *IEEE ICC*, p. 6262–6267.

Develder, C., Buysse, J., Shaikh, A., Jaumard, B., De Leenheer, M. e Dhoedt, B. (2011). Survivable optical grid dimensioning: anycast routing with server and network failure protection. Em *IEEE ICC*, p. 1–5.

Develder, C., De Leenheer, M., Dhoedt, B., Pickavet, M., Colle, D., De Turck, F. e Demeester, P. (2012b). Optical networks for grid and cloud computing applications. *Proceedings of the IEEE*, 100(5):1149–1167.

Endo, P. T., de Almeida Palhares, A. V., Pereira, N. N., Gonçalves, G. E., Sadok, D., Kelner, J., Melander, B. e Mangs, J. (2011). Resource allocation for distributed cloud: concepts and research challenges. *IEEE Network*, 25(4):42–46.

Grover, W. D. (2004). *Mesh-based survivable transport networks: options and strategies for optical, MPLS, SONET and ATM networking*. Prentice Hall Professional, primeira edição.

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.

Habib, M. F., Tornatore, M., Dikbiyik, F. e Mukherjee, B. (2013). Disaster survivability in optical communication networks. *Computer Communications*, 36(6):630 – 644.

Lam, C., Liu, H., Koley, B., Zhao, X., Kamalov, V. e Gill, V. (2010). Fiber optic communication technologies: What's needed for datacenter network operations. *IEEE Communications Magazine*, 48(7):32–39.

Ramaswami, R., Sivarajan, K. e Sasaki, G. (2009). *Optical networks: a practical perspective*. Morgan Kaufmann, terceira edição.

Telikepalli, R., Drwiega, T. e Yan, J. (2004). Storage area network extension solutions and their performance assessment. *IEEE Communications Magazine*, 42(4):56–63.

Verdi, F., Rothenberg, C., Pasquini, R. e Magalhães, M. (2010). Novas arquiteturas de data center para cloud computing. Em *Minicursos do XXVIII SBRC*, p. 103–152.

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.

# Um Esquema de Multicaminhos com Algoritmos Genéticos para Redes de Centro de Dados

**Lyno Henrique Gonçalvez Ferraz, Diogo Menezes Ferrazani Mattos e Otto Carlos Muniz Bandeira Duarte**

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{lyno,menezes,otto}@gta.ufrj.br

***Resumo.*** *Os centros de dados utilizados para computação em nuvem devem permitir a coexistência de serviços com padrões de tráfego distintos, garantir alta capacidade de transmissão de dados e tolerar falhas de enlaces. As topologias de interconexão dos centros de dados proveem redundância nas conexões físicas, que os mecanismos de encaminhamento utilizam para gerar múltiplos caminhos, e assim melhorar o desempenho de encaminhamento de pacotes. Este artigo propõe um esquema de geração de multicaminhos baseado em algoritmos genéticos que minimiza o tamanho dos caminhos e maximiza diversidade de enlaces usados na rede. Um simulador de eventos discretos foi desenvolvido para a avaliação das técnicas multicaminhos. O simulador modela o comportamento dos fluxos em diversos cenários de centros de dados. O artigo compara o esquema proposto com técnicas de uso de multicaminhos em redes de centro de dados propostos na literatura. Os resultados mostram que a proposta alcança maior taxa média de transmissão de fluxos, mesmo em cenários de alta utilização da rede.*

***Abstract.*** *Data centers used in cloud computing should allocate services with different traffic patterns, provide high data transfer capacity and link fault tolerance. The data center network topologies provide physical connection redundancy, which forwarding mechanisms avail to generate multiple paths. This paper proposes a multipathing scheme based on genetic algorithms to minimize path lengths and maximize link usage diversity. We develop a flow simulator to evaluate the multipathing techniques. The simulations model flow behaviors in different data center scenarios and compare the proposed scheme with multipathing techniques in literature. The results show the proposed scheme highest transmission rates, even in high network utilization scenarios.*

## 1. Introdução

A geração de dados cresce de forma exponencial e diversas tecnologias tais como Internet das coisas e redes elétricas inteligentes [Guimarães et al., 2013] devem contribuir ainda mais para esse cenário. O armazenamento e o tratamento dessas grandes massas de dados são uma área denominada *Big Data* [Costa et al., 2012] que impõe enormes desafios tecnológicos e, na qual, a computação em nuvem e os centros de dados vão desempenhar papéis fundamentais. O número de aplicações hospedadas nos centros de dados têm aumentado e uma das principais demandas de centros de dados em nuvem é a alta

Este trabalho foi realizado com recursos da CNPq, CAPES, FAPERJ, FINEP e FUNTTEL.

taxa de transmissão de fluxos entre os servidores. Os centros de dados oferecem uma grande capacidade de processamento e armazenamento para aplicações ao aglomerar servidores interconectados. As aplicações são distribuídas nesse aglomerado de servidores, então a rede de comunicação possui papel fundamental para que as aplicações executem de acordo com seus objetivos [Bari et al., 2013].

As topologias das redes de comunicação dos centros de dados são desenvolvidas para prover alta taxa de transmissão agregada, redundância de caminhos e confiabilidade. Para tal, as topologias são formadas com árvores com múltiplas raízes, de modo que ofereçam múltiplos caminhos entre pares de servidores. Por sua vez, mecanismos de encaminhamento normalmente utilizam os caminhos redundantes para tolerância a falhas [Couto et al., 2012]. No cenário de computação em nuvem, diversas aplicações de proprietários diferentes, ou inquilinos, compartilham tanto os servidores quanto a rede dos centros de dados de um provedor de infraestrutura. As cargas de trabalhos de cada inquilino são desconhecidas pelo provedor de infraestrutura, pois cada inquilino executa suas próprias aplicações e protocolos de comunicação. Portanto, o provedor de infraestrutura deve oferecer alta taxa de transmissão sem modificações em protocolos ou *software*. A infraestrutura deve empregar técnicas de uso dos multicaminhos disponíveis.

Este artigo propõe um esquema de criação de multicaminhos baseado em algoritmo genético, cuja função objetivo é minimizar os tamanhos dos caminhos e maximizar a diversidade de uso dos enlaces. Além disso, são modeladas heurísticas para seleção de caminhos já criados. O trabalho também apresenta as demandas de centros de dados em nuvem e analisa algumas técnicas de uso de multicaminhos. O artigo compara o esquema de multicaminhos proposto com as principais técnicas de encaminhamento de tráfego empregadas em centro de dados, como *Spanning Tree Protocol* [Touch e Perlman, 2009], *Equal Cost MultiPath* [Al-Fares et al., 2010], *Smart Path Assignment In Networks* (SPAIN) [Mudigonda et al., 2010].

Um simulador de fluxos de eventos discretos foi desenvolvido para a análise e comparação de desempenho das técnicas de multicaminhos. A simulação de fluxos permite uma maior escala e carga de simulação em relação a simuladores de pacotes. O simulador usa um modelo simplificado de fluxos que disputam a banda disponível nos enlaces dos centros de dados. Os resultados obtidos comprovam que o esquema proposto alcança maior taxa média de transmissão de fluxos, quando comparado com técnicas da literatura, tanto em cenários de comunicação todos-com-todos e todos-com-um.

O artigo está organizado da seguinte forma. A Seção 2 apresenta questões dos centros de dados em nuvem virtualizados. A Seção 3 apresenta o algoritmo genético para a criação de multicaminhos. O simulator desenvolvido é apresentado na Seção 4 e os resultados na Seção 5. Por fim, a Seção 6 conclui o artigo.

## 2. Sistema de Comunicação em Centros de Dados Virtualizados

Os centros de dados convencionais usam servidores dedicados para executar aplicações específicas, o que resulta na utilização ineficiente dos recursos devido à variação de demanda e da conseqüente ociosidade de recursos. Com o crescimento da computação em nuvem e as tecnologias de virtualização, diversos serviços e aplicações são agregados em um mesmo centro de dados para aumentar a utilização dos recursos físicos e, assim, reduzir os custos de operação e manutenção. Os centros de dados em nuvem virtualizados hospedam inquilinos diversos com suas próprias aplicações, o que aumenta a quantidade e variedade de aplicações que compartilham a infraestrutura do centro de dados [Bari et al., 2013, Mattos e Duarte, 2012].

A infraestrutura dos centros de dados é construída com o objetivo de prover alta capacidade de computação para a execução de aplicações, muitas vezes com topologias altamente redundantes, de modo que sempre haja múltiplos caminhos entre os servidores [Al-Fares et al., 2008, Costa et al., 2012]. Entretanto, uma aplicação é divida e distribuída entre os servidores, de maneira que o tráfego interno de um centro de dados é quatro vezes maior que o tráfego externo [Greenberg et al., 2011].

As aplicações que executam em um centro de dados são variadas, com padrões de tráfegos heterogêneos [Benson et al., 2010]. A maior parte do "tráfego" transmitido entre as aplicações é composta de fluxos com pequena quantidade de dados e muito breves, chamados de fluxos camundongos (*mice flows*). Por outro lado, a maior parte dos "dados transmitidos" está em fluxos com grande quantidade de dados e duradouros, que compõe uma minoria do total de fluxos transmitidos em um centro de dados. Esses fluxos volumosos e duradouros são chamados de fluxos elefantes (*elephant flows*). Os fluxos camundongos e fluxos elefantes possuem requisitos e comportamentos distintos, e a interação causa perdas de pacotes e atrasos no cumprimento de requisições o que impacta no desempenho das aplicações. Os fluxos camundongos são sensíveis à latência, pois normalmente provêm de aplicações do tipo partição/agregação, como *Map/Reduce* usadas em buscas *web*, composição de conteúdos de redes sociais e seleção de propaganda [Alizadeh et al., 2010]. Nesse tipo de aplicação, na fase de partição, uma requisição é dividida em diversas sub-requisições por um servidor de agregação e enviadas a diversos servidores de trabalho. Após a computação em paralelo das requisições, os servidores de trabalho enviam as repostas ao servidor de agregação que compõe a resposta à requisição e, assim, realiza a fase de agregação. Os fluxos desse tipo de aplicação possuem estritos limites de latência para que as repostas sejam apresentadas a tempo para os usuários. As respostas que ultrapassarem o limite de tempo são descartadas, degradando a qualidade da resposta para os usuários. A principal causa de aumento na latência é o congestionamento instantâneo dos enlaces causado pela fase de agregação, que corresponde a uma comunicação do tipo muitos-para-um. Assim, a prevenção desse problema é a priorização de fluxos com prazo curto e baixa utilização de enlaces [Zats et al., 2012].

Diversas propostas criam mecanismos para controlar a banda e o atraso dos fluxos, para garantir a qualidade e prazo das respostas para os usuários de aplicações em centros de dados. A proposta *Data Center TCP* (DCTCP) [Alizadeh et al., 2010] evita congestionamentos e perda de pacotes nos enlaces com um mecanismo fino de controle de congestionamento baseado em notificações explícitas de congestionamento. Assim, os comutadores do centro de dados marcam *bits* de congestionamento nos pacotes de retorno do fluxo quando detectam que as filas de transmissão estão ocupadas e os emissores reduzem a taxa de encaminhamento proporcionalmente à fração de enlaces congestionados.

A proposta *High-bandwidth Ultra-Low Latency* [Alizadeh et al., 2012] garante a latência mínima ao custo de forçar a taxa de transmissão para ser menor que a capacidade do enlace. Para fazer o controle de taxa, os autores utilizam o mecanismo DCTCP, mas marcam os *bits* de congestionamento com outra estratégia. Cada interface de saída possui uma fila fantasma que contabiliza a taxa de saída de pacotes e, quando a taxa de saída ultrapassa um limiar menor que o limite da fila de transmissão, os *bits* de congestionamento são marcados. A taxa de transmissão nunca é máxima, mas a proposta controla a banda dos fluxos antes de ocorrer congestionamento ou perda de pacotes.

O esquema de controle de protocolo *Deadline-Driven Delivery* [Wilson et al., 2011] prioriza os fluxos com menores prazos de entrega através

do controle de taxa. Periodicamente, as aplicações requisitam taxas aos roteadores de acordo com a quantidade de dados restantes nos fluxos e os prazos de entrega. Os roteadores distribuem taxas para as aplicações com um algoritmo guloso e, portanto, as aplicações transmitem os fluxos nas taxas máximas sem violar o prazo de entrega de nenhum fluxo.

Zats *et al.* propuseram *DeTail* [Zats et al., 2012] que é uma abordagem multicamadas para reduzir o tempo máximo de resposta a requisições. Na camada de enlace, os comutadores evitam perdas de pacotes devido à ocupação de filas com o uso de quadros de pausa para controlar a taxa de pacotes recebidos. Na camada de rede, os comutadores escolhem o próximo salto de pacotes com base na ocupação das filas para fazer o balanceamento de carga. O protocolo da camada de transporte é resistente a reordenamento para receber os pacotes vindos de vários caminhos e controla a taxa de transmissão com notificações de congestionamento de comutadores baseado na ocupação das filas. As aplicações especificam as prioridades para diferenciar fluxos sensíveis à latência.

As propostas até aqui mencionadas não só modificam a infraestrutura para melhorar o desempenho da rede, mas também modificam os servidores finais que devem interagir com a infraestrutura. Logo, essas propostas não são adequadas para centros de dados em nuvem virtualizados com multi-inquilinos, pois os inquilinos possuem protocolos próprios e diversificados. Além disso, mesmo padronizando os protocolos, há riscos de segurança na quebra de isolamento entre inquilinos, pois protocolos de aplicações comunicam-se diretamente com dispositivos da infraestrutura para reservar recursos.

Ao mesmo tempo em que se procura diminuir as perdas devido ao congestionamento instantâneo de enlaces, também se deve atender às características diferentes dos fluxos. Fluxos elefantes demandam alta capacidade de transmissão de dados e são resistentes à latência, pois transferem uma grande quantidade de dados e não têm os mesmos requisitos de entrega de fluxos camundongo. Assim, esses fluxos devem ser organizados de maneira a aproveitar o máximo possível da capacidade de transmissão dos centros de dados. Além disso, fluxos elefantes utilizam toda a capacidade de transmissão de um enlace e, assim, causam congestionamentos que afetam os fluxos camundongos. Diversas propostas aproveitam os múltiplos caminhos que as redes de centros de dados possuem para aproveitar o máximo da capacidade de transmissão dos enlaces. Uma forma de aproveitar os multicaminhos é através de *Equal Cost MultiPath* (ECMP). O protocolo de rede calcula múltiplos menores caminhos de mesmo custo e realiza um *hash* dos cabeçalhos dos pacotes para escolher o caminho no qual transmitir o fluxo. Assim, é esperado que os fluxos sejam distribuídos aleatoriamente nos múltiplos caminhos, ou seja, caminhos com custo igual. Diversos protocolos usam essa técnica de encaminhamento, como *Transparent Interconnection of Lots of Links* [Touch e Perlman, 2009] e 802.1aq *Shortest Path Bridging* [Allan et al., 2010]. O *Valiant Load Balancing* [Greenberg et al., 2011] funciona de maneira semelhante, mas a escolha de caminho é realizada através da escolha aleatória de um comutador intermediário.

A proposta *Smart Path Assignment In Networks* (SPAIN) [Mudigonda et al., 2010] explora a diversidade de caminhos das topologias de centros de dados para aumentar a vazão dos fluxos e a confiabilidade da rede. O SPAIN usa um algoritmo *offline* para configurar as árvores de VLANs (*Virtual Local Area Network*), de modo em que as árvores são construídas com base em menores caminhos e menor uso de enlaces. Outro algoritmo *online* executado nos servidores verifica as árvores e os servidores conectados e, então seleciona aleatoriamente uma árvore para ser usada em um fluxo.

O *MultiPath TCP* (MPTCP) [Raiciu et al., 2011] subdivide um fluxo TCP em diversos subfluxos, de maneira que cada um possua seu próprio controle de congestionamento. Cada um dos subfluxos é transmitido em um caminho diferente e, assim, cada subfluxo transmite na taxa máxima de cada caminho. Os multicaminhos utilizados pelo MPTCP podem ser definidos através de mecanismos como o SPAIN.

A proposta Hedera [Al-Fares et al., 2010] detecta os fluxos elefantes e os escalona nos caminhos dos centros de dados. O Hedera usa o controlador centralizado de redes NOX [Gude et al., 2008] comutadores programáveis OpenFlow [McKeown et al., 2008] para detectar os fluxos com alta taxa de transmissão de dados e tempo de vida. O controlador periodicamente migra os fluxos elefantes para outros caminhos baseado em um algoritmo de otimização de arrefecimento simulado (*simulated annealing*) que otimiza a taxa de transmissão dos fluxos. Essas propostas abordam o problema de organização dos fluxos nos enlaces da rede de centro de dados e todas, exceto MPTCP, são adequadas para centros de dados em nuvem virtualizados, pois não modificam os protocolos de inquilinos e podem ser utilizados por provedores de infraestrutura. Entretanto, a seleção de caminhos aleatória não leva em conta a utilização da rede e os tamanhos dos fluxos, o que causa colisões de caminhos que sobrecarregam enlaces e degradam o desempenho.

Este artigo foca em técnicas multicaminhos e propõe um esquema baseado na otimização da geração de múltiplos caminhos com algoritmos genéticos. A abordagem multicaminhos é mais adequada para o cenário de centros de dados em nuvem virtualizados do ponto de vista de um provedor de infraestrutura, pois não interferem nas aplicações dos inquilinos. Logo, as técnicas de multicaminhos consideradas não devem modificar protocolos de servidores finais, mas somente a infraestrutura de rede. A utilização dos multicaminhos é realizada em duas fases: a Configuração de Multicaminhos e a Seleção de Multicaminhos. Na fase Configuração de Multicaminhos, um algoritmo calcula os caminhos a serem configurados e, em seguida, configura os dispositivos de rede com diversos caminhos. Normalmente a configuração de caminhos é realizada de maneira *offline* ou quando ocorrem mudanças na topologia da rede. A fase de Seleção de Multicaminhos é *online* e ocorre constantemente enquanto a rede está em operação. Nessa fase, os dispositivos de rede usam algoritmos para selecionar qual dos multicaminhos configurados é utilizado para um fluxo.

## 3. O Esquema Proposto de Multicaminhos com Algoritmo Genético

Para aproveitar a redundância de caminhos na topologia dos centros de dados, este artigo propõe um esquema de Configuração de multicaminhos com algoritmos genéticos e de Seleção com heurísticas baseadas no uso dos caminhos. O esquema gera diversas árvores independentes para interconectar os comutadores de topo de *rack*, de maneira a minimizar tanto as distâncias entre comutadores de topo *rack*, quanto maximizar a diversidade de uso de enlaces na rede. As árvores são configuradas nos comutadores para o encaminhamento dos pacotes, com VLANs ou com controladores centralizados. Em cada árvore existe somente um caminho entre cada par de comutadores topo de *rack*. Somente uma árvore é selecionada para cada fluxo, portanto, o fluxo trafega por um único caminho o que não afeta a ordenação dos pacotes no receptor. A seleção das árvores a serem utilizadas por um fluxo pode ser determinada por diferentes heurísticas baseadas no uso de caminhos e de enlaces. A técnica SPAIN [Mudigonda et al., 2010] utiliza abordagem semelhante para explorar multicaminhos em centros de dados.

As árvores devem explorar ao máximo a diversidade de enlaces do centro de dados. Assim, o uso de diversas árvores aumenta a taxa de transmissão agregada, balanceia

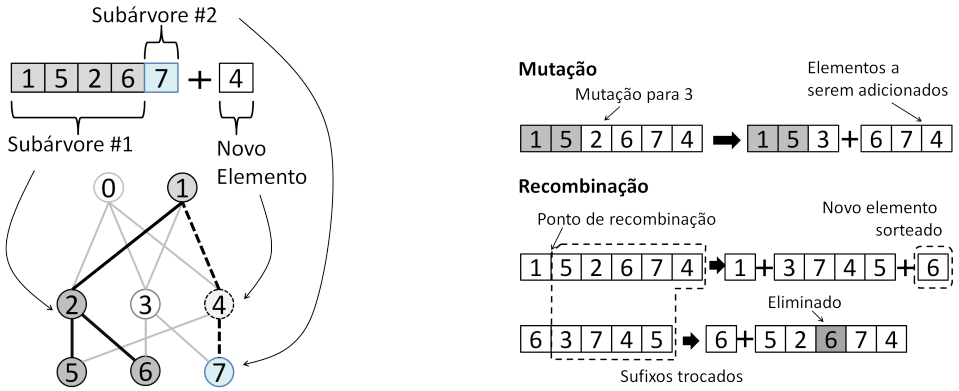
a carga entre os enlaces, diminui o impacto de falhas nos enlaces e previne perdas em rajadas devido a enlaces sobrecarregados. Para configurar as árvores, deve-se obter a topologia de rede do centro de dados para executar o algoritmo de geração de árvores. Após a geração das árvores, configuram-se as árvores nos comutadores. Esse processo é realizado *offline* antes do centro de dados entrar em operação e, portanto, não acarreta em atrasos de configuração e escalonamento de caminhos. Após a configuração das árvore, a seleção de caminhos é realizada por dispositivos inseridos entre servidores e comutadores topo de *rack* ou nos servidores.

Esta abordagem requer modificações mínimas na rede de comunicações do centro de dados. Além disso, essa abordagem não exige nenhuma modificação nas máquinas virtuais de inquilinos, logo é adequada para centros de dados em nuvem virtualizados. Neste artigo, é considerado o caso de uso de VLANs para definir uma árvore, pois exige apenas características de comutadores de prateleira. Para garantir a operação e alcançabilidade na ocorrência de falhas de enlaces das árvores de VLANs, todos comutadores da rede executam o *Spanning Tree Protocol* (STP) e configuram uma árvore de cobertura. Essa árvore de cobertura só é utilizada para o encaminhamento de pacotes que não pertençam a uma VLAN conhecida e ativa.

**Algoritmo Genético**

A representação de um indivíduo é uma árvore definida pelo arranjo sem repetição de identificadores de comutadores. Para gerar a árvore conexa e sem laços é realizado um processo que interconecta os comutadores do arranjo, que adiciona novos comutadores até conectar todos comutadores topo de *rack*. Uma nova árvore é gerada da seguinte maneira: sorteia-se um comutador e ele é adicionado a uma subárvore. Em seguida, sorteia-se outro comutador. Caso o comutador sorteado possua conexão direta com comutadores já sorteados, o comutador é adicionado na mesma subárvore, assim como todas as subárvores que o comutador possui conexão direta. Os enlaces considerados são aqueles que o comutador usa para conectar-se às árvores. Se o comutador não tem conexão direta com nenhum comutador de outra subárvore, o comutador é adicionado em uma nova subárvore. Esse processo é repetido até todos comutadores topo de *rack* estarem conectados na mesma subárvore. No final, os comutadores que não estão no caminho entre todos comutadores topo de *rack* são removidos da árvore. A Figura 1(a) apresenta um exemplo de genótipo de um indivíduo do algoritmo genético e a árvore formada.

As operações de mutação e recombinação são realizadas de maneira especial para manter a árvore conexa, sem laços. A mutação escolhe e muda aleatoriamente um dos comutadores do genótipo e, caso a mutação separe a árvore em subárvores, são adicionados novos comutadores. Na recombinação, sorteia-se uma posição para manter os comutadores no arranjo de dois genótipos e, os outros comutadores do arranjo são enviados para o outro genótipo. Os comutadores são adicionados sequencialmente e, caso a operação separe a árvore em subárvores, são adicionados novos comutadores até obter árvores conexas. A Figura 1(b) mostra as operações nos genótipos. O fenótipo, ou a avaliação de uma árvore, é representado por duas funções objetivo. A primeira calcula a distância média entre os comutadores topo de *rack* e a segunda calcula a soma do inverso de vezes que um enlace é utilizado em cada árvore. Assim, fenótipos melhores são os de árvores com diâmetros menores e que possuam enlaces menos utilizados. A comparação de fenótipos considera um melhor que outro caso uma função objetivo seja maior e a outra função seja maior ou igual.




(a) Representação de um indivíduo. (b) Operações sobre o genótipo.

**Figura 1. Algoritmo genético para a geração de árvores: a) representação de um indivíduo como um permutação de comutadores; b) operações de mutação e recombinação sobre o genótipo.**

O algoritmo é inicializado com certo número de indivíduos e em seguida o algoritmo entra em um laço com um número máximo de gerações. Em cada interação do laço, são sorteados pares de indivíduos proporcionalmente à qualidade do fenótipo. Os pares são recombinados para gerar novos pares de indivíduos até se obter o dobro da população. Em seguida os indivíduos são mutados e, para a próxima geração, sobrevivem apenas os indivíduos de melhor fenótipo dentre todos os indivíduos calculados.

### O Procedimento de Seleção de Multicaminhos

Para realizar a seleção de multicaminhos são definidas as seguintes heurísticas:

- seleção aleatória: o caminho é selecionado com probabilidade uniforme;
- seleção de caminhos menos utilizados: cada vez que o caminho é utilizado, diminui a probabilidade de ser selecionado;
- seleção de caminhos com enlaces menos utilizados: cada vez que um enlace de um caminho é utilizado por um fluxo, diminui a probabilidade dos caminhos que usam o enlace serem utilizados.

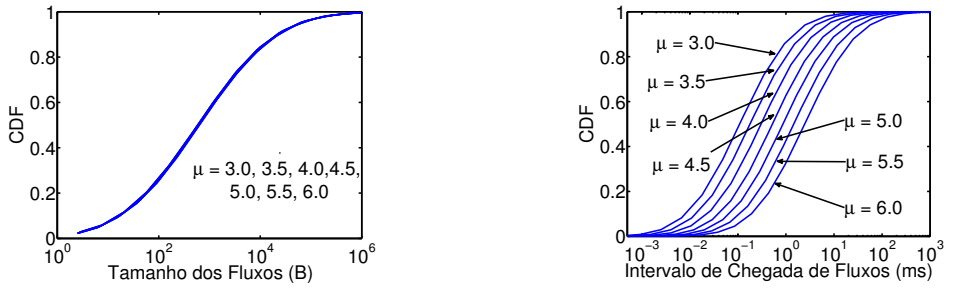
Deve ser ressaltado que os mecanismos que optam pela estratégia de selecionar os caminhos e enlaces menos utilizados precisam obter dados de um banco de informações do uso de caminhos e enlaces, o que causa atraso na seleção de caminhos de novos fluxos.

## 4. O Simulador de Fluxos de Eventos Discretos Desenvolvido

Para avaliar as técnicas de multicaminhos foi projetado e desenvolvido um simulador de eventos discretos que modela a transmissão de dados como um fluxo de dados. Como o objetivo deste trabalho é a avaliação de desempenho de estratégias de multicaminhos, um modelo de simulação de fluxos permite uma maior escalabilidade de simulação em comparação a modelos de simulação de pacotes como o provido pelo simulador NS3 [ns3, 2006], pois abstrai os procedimentos dos protocolos. Assim, o simulador de fluxos em centro de dados proposto cria uma fila de eventos ordenada pelo instante dos eventos. A cada passo, um novo evento é tratado e o tempo da simulação é atualizado. Eventos podem adicionar novos eventos na fila que são, na maioria, chegada e saída de fluxos. A simulação pára quando não existem mais eventos ou o tempo limite é atingido.

*O Modelo:* Um fluxo é definido pela tupla (origem, destino, tamanho em *bytes*, taxa de transmissão atual, quantidade *bytes* já transferidos). O modelo do simulador assume

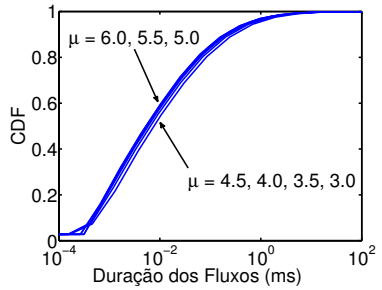
que a taxa de transmissão dos fluxos é máxima taxa obtida em regime permanente e só é modificada com a chegada ou saída de outro fluxo. Assim, a taxa de transmissão é calculada como a divisão igual de banda do enlace entre os fluxos. Caso um fluxo já tenha a taxa definida e ocupe menos que a banda máxima de um enlace, o restante da banda do enlace é dividido igualitariamente entre os fluxos restantes. Além disso, os fluxos são considerados em uma direção e os fluxos correspondentes aos pacotes de retorno dos fluxos são desprezados. Dessa maneira, o modelo é otimista e não considera o início lento do TCP, as perdas de pacote e a banda utilizada pelo fluxo de retorno. O simulador considera os cabeçalhos dos protocolos TCP/IP e da camada de enlace Ethernet e que todos os pacotes são transmitidos com o tamanho máximo permitido, exceto o último pacote de cada fluxo que contém somente os *bytes* restantes do fluxo.



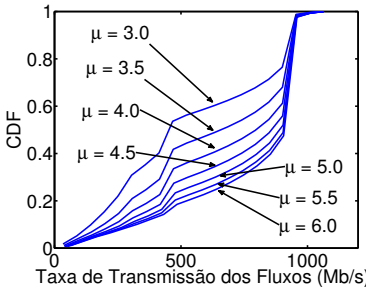

(a) CDF dos tamanhos dos fluxos. (b) CDF dos intervalos de chegada de fluxos.

**Figura 2. Tráfego utilizado nas simulações para modelar o intervalo de chegada de fluxos medidos em [Benson et al., 2010]: a) tamanho dos fluxos com distribuição lognormal ($\mu = 7, \sigma = 2.8$); b) intervalos de chegada de fluxos com distribuição lognormal ($\mu = 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, \sigma = 2$).**

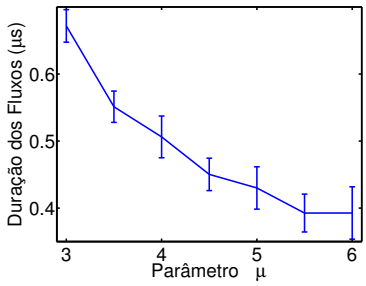
*Os Parâmetros de Simulação:* O simulador pode ser configurado com diversos parâmetros que permitem avaliar diferentes aspectos do desempenho dos centros de dados. A topologia do centro de dados indica a diversidade de caminhos e, portanto, avalia a eficácia e desempenho das técnicas de multicaminhos. O número de nós da topologia altera a escala das simulações. A configuração de comportamentos dos fluxos pode ser alterada para variar a carga de trabalho da simulação. O tamanho dos fluxos indica quanto tempo o fluxo ocupa os enlaces do caminho. Quanto maior o tamanho do fluxo em *bytes*, maior o tempo para transmiti-lo por completo, maior a probabilidade de fluxos novos usarem os mesmos enlaces, diminuir a taxa de transmissão e aumentar ainda mais o tempo de transmissão do fluxo. Nas simulações deste artigo, considerou-se uma distribuição lognormal para gerar os tamanhos dos fluxos. Os fluxos são gerados aleatoriamente com distribuição lognormal de média $\mu = 7$, e desvio padrão $\sigma = 2.8$, de modo que a função de distribuição cumulativa tenha os seguintes valores $F(x) = \{\approx 0.5 | x = 1000, \approx 0.95 | x = 100000\}$ de acordo com as medidas empíricas apresentadas por Benson *et al.* [Benson et al., 2010]. De maneira semelhante, os intervalos de chegada de novos fluxos também alteram a carga de trabalho dos centros de dados. Quanto menor o intervalo de chegada, mais fluxos disputam a banda dos enlaces, logo as taxas dos fluxos diminuem e o tempo de transmissão aumenta. As simulações desse artigo usam uma distribuição lognormal para os intervalos de chegada de fluxos com desvio padrão $\sigma = 2$ e média, $\mu$, variando para aumentar a carga de $3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0$. Os valores do parâmetro $\mu$ foram escolhidos para modelar o intervalo de chegada de fluxos medidos em [Benson et al., 2010]. Ambos os modelos utilizados nas simulações são apresentados na Figura 2.



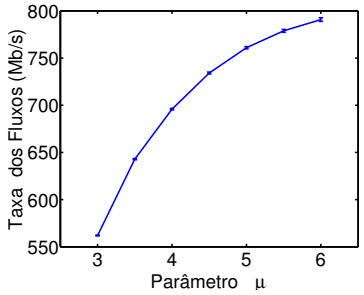

(a) Duração dos fluxos.




(b) Taxas de transmissão dos fluxos.




(c) Duração média dos fluxos.




(d) Taxas média de transmissão dos fluxos.

**Figura 3. Teste de sanidade do simulador. Duração e taxa de transmissão para diferentes médias $\mu$ da distribuição de intervalo de cegada de fluxos. Com a diminuição de $\mu$, mais fluxos disputam a banda do enlace, diminuindo a taxa de transmissão e aumentando a duração dos fluxos.**

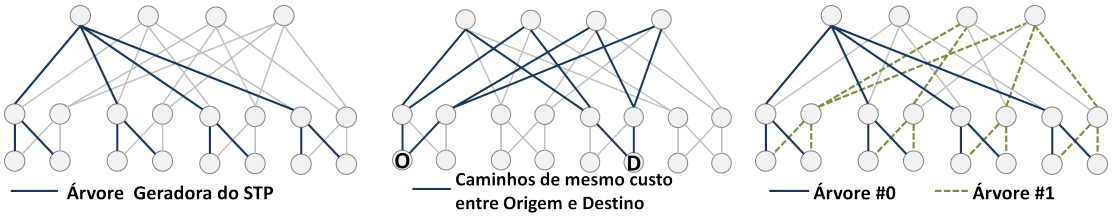
Outra característica importante dos comportamentos dos fluxos é o conjunto de destinos dos fluxos. Se os destinos são escolhidos uniformemente entre todos os nós do centro de dados, é esperado que toda a carga de trabalho seja distribuída uniformemente entre os destinos, apesar das cargas geradas aleatoriamente. Esse cenário utiliza intensamente todos caminhos disponíveis. Por sua vez, caso o tráfego seja concentrado em um ou poucos nós de destino, os caminhos até os destinos são utilizados intensamente enquanto caminhos para outros nós são menos utilizados. Logo, caminhos alternativos que usem enlaces de caminhos para outros nós se tornam uma boa escolha para os fluxos.

Foi realizado um teste de sanidade com uma topologia com dois nós interconectados. Esse teste mostra o impacto do aumento de carga por diminuição do intervalo de chegada de fluxos. Os resultados do teste de sanidade são apresentados na Figura 3. O parâmetro variado foi a média de $\mu$ da distribuição lognormal dos intervalo de chegada de fluxos. A Figura 2(a) mostra a função distribuição acumulada (*Cumulative Distribution Function* - CDF) dos tamanhos dos fluxos. A Figura 2(b) mostra a CDF dos intervalos de chegada de fluxos e que a diminuição de $\mu$ reduz o valor de intervalo de chegada dos fluxos em média. As Figuras 3(a) e 3(c) mostram a CDF e média das durações dos fluxos e as Figuras 3(b) e 3(d) mostram a CDF e as médias das taxas de transmissão dos fluxos para os diferentes valores de $\mu$s. É possível perceber que com a diminuição do $\mu$, a duração dos fluxos tende a valores maiores como apresentado nas Figuras 3(a) e 3(c). A diminuição do $\mu$ aumenta o número de fluxos que compartilham o enlace, portanto as taxas de transmissão obtidas são menores como mostram as Figuras 3(b) e 3(d). O restante dos gráficos são apresentados na forma de média das durações dos fluxos e média das taxas de transmissão dos fluxos.

## 5. Simulações e Resultados

Este artigo avalia as técnicas de multicaminhos considerando as fases de Configuração e Seleção de Multicaminhos. Assume-se que não há falhas em enlaces, nem a reconfiguração de multicaminhos. A topologia utilizada foi a fat-tree [Al-Fares et al., 2008] com comutadores de quatro portas, que oferece quatro caminhos distintos de mesmo custo entre pares de servidores. Essa topologia beneficia a técnica *Equal Cost MultiPath* (ECMP) pela existência de múltiplos caminhos distintos de menor custo, dessa maneira esse é um cenário justo para de comparação. As cargas de trabalho são definidas aleatoriamente por distribuições lognormal, tamanho dos fluxos com distribuição lognormal ($\mu = \mathbf{7}, \sigma = \mathbf{2.8}$), intervalos de chegada de fluxos nos comutadores topo de *rack* com distribuição lognormal ($\mu = \mathbf{3.0}, \mathbf{3.5}, \mathbf{4.0}, \mathbf{4.5}, \mathbf{5.0}, \mathbf{5.5}, \mathbf{6.0}, \sigma = \mathbf{2}$). O simulador executa a simulação até 1000 segundos. Os resultados são apresentados com intervalo de confiança de 95%. Os gráficos são apresentados na forma de média dos valores obtidos na simulação.

A seguir são apresentadas algumas técnicas que foram modeladas, implementadas e comparadas com o esquema proposto de multicaminhos com algoritmo genético.




(a) STP. (b) ECMP. (c) SPAIN.

**Figura 4. Multicaminhos na topologia *fattree* criados pelas técnicas a) STP: uma árvore com um caminho único entre pares; b) ECMP: diversos caminhos de mesmo custo entre pares; c) SPAIN: diversas árvores disjuntas.**

### O Modelo do Spanning Tree Protocol

O *Spanning Tree Protocol* (STP) calcula uma árvore de cobertura entre comutadores, De maneira que os comutadores só aprendem caminhos sobre a árvore. Desse modo, todos os caminhos entre pares de servidores compartilham os enlaces da árvore de cobertura. A Configuração de Multicaminhos dessa técnica considera um único caminho sobre a árvore de cobertura e, portanto, a Seleção de Multicaminhos usa o único caminho disponível. As simulações desse artigo consideram uma árvore de cobertura mínima.

### O Modelo do Equal Cost MultiPath

A técnica *Equal Cost MultiPath* (ECMP) calcula todos os caminhos para um destino que têm custo mínimo na fase de Configuração. Neste artigo, o cálculo é realizado com o algoritmo Dijkstra modificado. A Figura 4(b) apresenta os múltiplos caminhos de mesmo custo para a topologia *fattree*. Na fase de Seleção, o elemento de rede aplica uma função *hash* em campos do cabeçalhos dos pacotes para indicar qual próximo salto usar e, assim, um fluxo segue por um único caminho. As simulações utilizam a função *hash* `md5` sobre os identificadores da origem e destino para selecionar o caminhos de um fluxo.

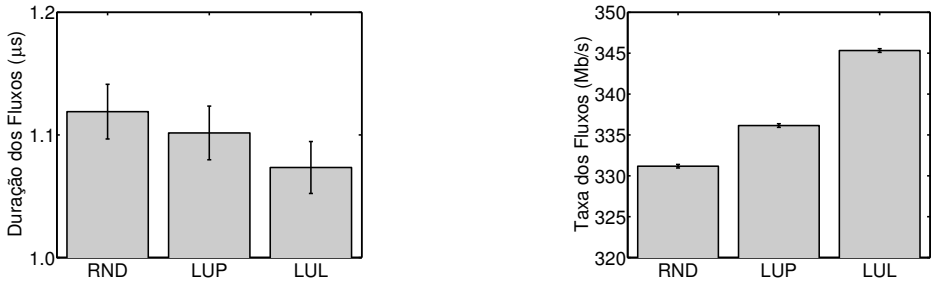
### O Modelo do Smart Path Assignment In Networks

Os mecanismos e algoritmos criados pelo *Smart Path Assignment In Networks* (SPAIN) configuram VLANs nos comutadores para cada árvore e, durante a operação da

rede, servidores selecionam uma árvore para um fluxo e marcam a etiqueta de VLAN da árvore. A fase de Configuração calcula de maneira *offline* as múltiplas árvores, que são criadas com dois algoritmos, um forma conjuntos de caminhos distintos entre pares, e outro usa algoritmo guloso para agregar os caminhos entre pares para formar as árvores. Assim, obtém-se um conjunto de árvores com menores caminhos entre pares com enlaces disjuntos. A Figura 4(c) apresenta exemplos de árvores formadas pelo SPAIN. A seleção de caminhos é realizada com um mecanismo que executa nos servidores. O mecanismo consulta uma base de dados de árvores e verifica a disponibilidade dos caminhos. Ao enviar um fluxo, o mecanismo escolhe uniformemente um dos caminhos ativos e marca a etiqueta de VLAN em todos os pacotes desse fluxo.

### 5.1. Resultados de Simulação com Tráfego Todos-para-Todos

Avalia-se o comportamento dos centros de dados com tráfego de todos os comutadores topo de *rack* para todos outros comutadores topo de *rack*. Como os destinos são escolhidos uniformemente, é esperado que a carga de trabalho seja distribuída uniformemente entre os caminhos com alta ocupação dos enlaces.



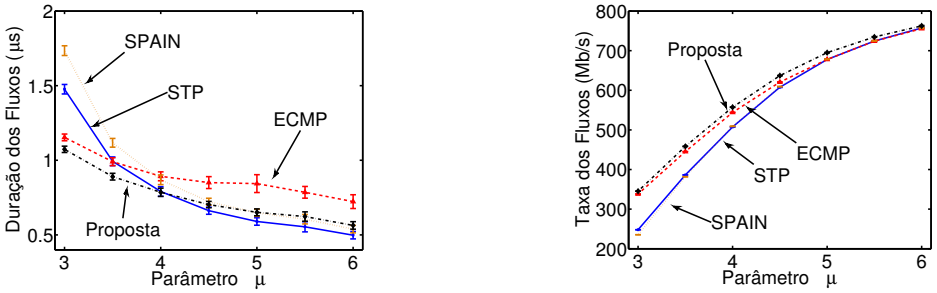

(a) Duração dos fluxos. (b) Taxa de transmissão dos fluxos.

**Figura 5. Avaliação das Heurísticas de Seleção dos fluxos para o esquema de multicaminhos com algoritmos genéticos. As heurísticas comparadas são seleção aleatória (*RND*), seleção dos caminhos menos utilizados (*LUP*) e seleção dos caminhos com enlaces menos utilizados (*LUL*).**

#### Avaliação das Heurísticas de Seleção de Multicaminhos

Para escolher a heurística para seleção de multicaminhos do algoritmo genético, comparam-se os seguintes tipos de seleção: seleção aleatória (*Random - RND*), seleção dos caminhos menos utilizados (*Least Used Path - LUP*) e seleção dos caminhos com enlaces menos utilizados (*Least Used Link - LUL*). As heurísticas de seleção de multicaminhos foram comparadas considerando a configuração de multicaminhos com algoritmos genéticos. Nessa simulação, somente uma alta carga de trabalho foi utilizada com intervalos de chegada de fluxos com distribuição lognormal($\mu = \mathbf{3.0}, \sigma = \mathbf{2}$). A Figura 5 apresenta os resultados da simulação para as diferentes heurísticas. A Figura 5(a) mostra que as durações dos fluxos tem valores de aproximadamente $1.1\mu$s, com pouca variação para cada tipo de seleção de caminhos. A Figura 5(b) mostra que apesar da duração dos fluxos serem próximas, as taxas diferem com 331Mb/s para a seleção aleatória (*RND*), 336Mb/s para a seleção dos caminhos menos utilizados (*LUP*) e 345Mb/s para a seleção dos caminhos com enlaces menos utilizados (*LUL*). A seleção dos caminhos com enlaces menos utilizados (*LUL*) diminui a probabilidade de uso de um caminho cujos enlaces já foram escolhidos por muitos fluxos, logo a taxa de transmissão média é maior. O algoritmo genético considerado no restante do artigo é o LUL.

**Avaliação e Comparação de Configuração de Multicaminhos**



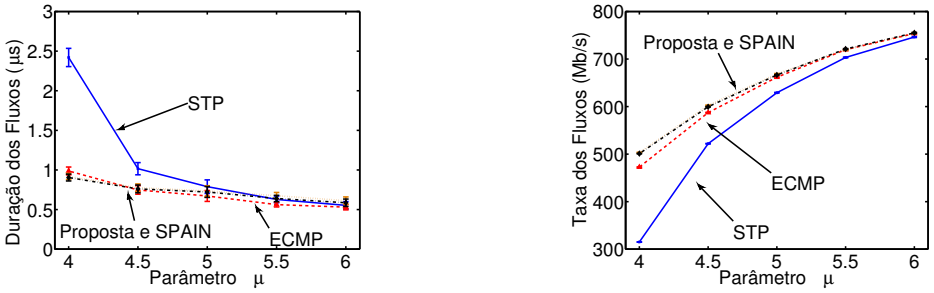

(a) Duração dos fluxos. (b) Taxa de transmissão dos fluxos.

**Figura 6. Avaliação de desempenho da Fase de Configuração dos fluxos com tráfego todos-para-todos do esquema proposto de árvores com algoritmos genético e comparação com as técnicas *Spanning Tree Protocol* (STP), *Equal Cost MultiPath* (ECMP), *Smart Path Assignment In Networks* (SPAIN).**

O desempenho do esquema proposto de multicaminhos com algoritmos genéticos é avaliado e comparado com as técnicas STP, ECMP e SPAIN, da fase de configuração de multicaminhos. A distribuição dos tamanhos dos fluxos é a mesma da simulação da avaliação das heurísticas de seleção de multicaminhos e o intervalo de chegada de fluxos possui distribuição lognormal com desvio padrão $\sigma = 2$ e com a média $\mu$ variando de 3.0 a 6.0. A Figura 6 apresenta os bons resultados da proposta deste artigo para a configuração de multicaminhos. Observa-se que para pequenos valores de $\mu$, a média da duração dos fluxos é maior e a taxa de transmissão é menor, de modo que as técnicas STP e SPAIN apresentam os piores valores. Como no STP existe somente uma mesma árvore em que ocorre a comunicação, todos os fluxos compartilham os mesmos enlaces, o que limita as taxas de transmissão obtidas. De maneira semelhante, a técnica SPAIN cria árvores através de um algoritmo guloso sem considerar caminhos cujos enlaces são menos utilizados nas árvores, então as árvores compartilham muitos enlaces. Além disso, a seleção de caminhos aleatória do SPAIN não prioriza enlaces pouco utilizados, assim o tráfego entre todos os comutadores topo de *rack* ocupa por muito tempo poucos enlaces, o que limita as taxas de transmissão obtidas. Como a topologia *fattree* disponibiliza quatro caminhos diferentes de mesmo custo, a técnica ECMP tem sucesso ao distribuir o tráfego entre os enlaces e atinge altas taxa de transmissão. Entretanto, a seleção de caminhos do ECMP por funções *hash* causa colisões na seleção de caminhos para cada fluxo, aumentando a duração média dos fluxos. O esquema proposto possui menores durações médias de fluxos e maiores taxas médias de transmissão, com ganhos em cima do ECMP que variam de 7,0 a 17,6 Mb/s. O esquema cria árvores com algoritmo genético considerando a utilização de enlaces nas outras árvores, portanto os fluxos usam caminhos distintos. Além disso, a heurística de seleção de caminhos prioriza caminhos com enlaces menos utilizados, o que balanceia o uso de enlaces na rede.

## 5.2. Avaliação e Comparação de Configuração de Multicaminhos com Tráfego Todos-para-Um

No cenário todos-para-um, o tráfego é concentrado em um comutador de destino que corresponde a fase de agregação de aplicações partição/agregação como *Map/Reduce*. Os comutadores topo de *rack* transmitem diversos fluxos de diferentes tamanhos para um único comutador topo de *rack*. A Figura 7 mostra os resultados da simulação nos quais o esquema proposto sempre apresenta menores durações médias e maiores taxas médias de




(a) Duração dos fluxos. (b) Taxa de transmissão dos fluxos.

**Figura 7. Avaliação de desempenho com tráfego todos-para-um do esquema proposto de configuração de árvores com algoritmos genético e comparação com as técnicas *Spanning Tree Protocol* (STP), *Equal Cost MultiPath* (ECMP), *Smart Path Assignment In Networks* (SPAIN).**

transmissão. No STP, todos os fluxos compartilham o enlace diretamente conectado ao comutador de destino e, portanto, cria-se um gargalo. Por outro lado, as técnicas SPAIN, ECMP e também o esquema proposto de algoritmo genético na *fattree* utilizam caminhos diferentes para cada fluxo. ECMP perde em desempenho para o esquema proposto devido às colisões na seleção de enlaces causada pelo *hash* e com $\mu = 4$ o esquema proposto tem vazão 30 Mb/s maior. O esquema proposto com algoritmos genéticos e a técnica SPAIN consideram caminhos alternativos pouco utilizados que podem ser maiores que os caminhos mínimos da rede e, assim, atingem as menores durações médias e as maiores taxas médias de transmissão.

## 6. Conclusão

As redes de centros de dados em nuvem virtualizados demandam novas técnicas que suportem os fluxos gerados pela grande quantidade e diversidade de aplicações dos múltiplos inquilinos. Os provedores de infraestrutura dos centros de dados em nuvem devem utilizar mecanismos que não interfiram na autonomia e prejudiquem o isolamento dos inquilinos. Esse artigo propõe um esquema de multicaminhos com algoritmos genéticos para que provedores de infraestrutura realizem o encaminhamento eficiente de fluxos. Um simulador de fluxos de eventos discretos foi desenvolvido para a avaliação e comparação da proposta com trabalhos da literatura. A proposta distribui o tráfego em enlaces da rede alcançando maiores taxas de transmissão de fluxos, mesmo em cenários de tráfego todos-para-todos e todos-para-um. O principal ganho da proposta deve-se ao uso de heurísticas de seleção de caminhos que consideram enlaces menos usados ao definir um novo fluxo. A etapa de seleção proposta diferencia-se das demais abordagens de multicaminhos, pois considera a quantidade de fluxos já existente nos enlaces ao definir o novo fluxo. A etapa de configuração de multicaminhos garante também o melhor desempenho da proposta à medida que calcula árvores otimizadas e disjuntas de cobertura da rede. Outro ponto importante da proposta é que se baseia somente na definição de regras de encaminhamento na camada de enlace e, portanto, não depende de mudanças em sistemas operacionais ou protocolos. A principal vantagem é as estações que acessam a rede não precisam ser alteradas para adoção da proposta, mas somente há a configuração de comutadores no núcleo da rede. Assim, a proposta é adequada para o cenário de múltiplos inquilinos compartilhando a rede com multicaminhos de um centro de dados para nuvem.

Como trabalhos futuros serão avaliadas diferentes topologias e estender o simulador para considerar os atrasos dos enlaces e o tamanho de *buffers* dos comutadores para estudar os impactos de uso de multicaminhos em fluxos camundongos.

## Referências


[Al-Fares et al., 2008] Al-Fares, M., Loukissas, A. e Vahdat, A. (2008). A scalable, commodity data center network architecture. Em *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, p. 63–74. ACM.

[Al-Fares et al., 2010] Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N. e Vahdat, A. (2010). Hedera: Dynamic flow scheduling for data center networks. Em *Proceedings of the 7th USENIX NSDI conference*, p. 19–19. USENIX Association.

[Alizadeh et al., 2010] Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S. e Sridharan, M. (2010). Data center TCP (DCTCP). Em *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10, p. 63–74, New York, NY, USA. ACM.

[Alizadeh et al., 2012] Alizadeh, M., Kabbani, A., Edsall, T., Prabhakar, B., Vahdat, A. e Yasuda, M. (2012). Less is more: Trading a little bandwidth for ultra-low latency in the data center. Em *Proceedings of USENIX NSDI conference*.

[Allan et al., 2010] Allan, D., Ashwood-Smith, P., Bragg, N., Farkas, J., Fedyk, D., Ouellete, M., Seaman, M. e Unbehagen, P. (2010). Shortest path bridging: Efficient control of larger Ethernet networks. *Communications Magazine, IEEE*, 48(10):128–135.

[Bari et al., 2013] Bari, M., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q. e Zhani, M. (2013). Data center network virtualization: A survey. *Communications Surveys Tutorials, IEEE*, 15(2):909–928.

[Benson et al., 2010] Benson, T., Akella, A. e Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. Em *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, p. 267–280. ACM.

[Costa et al., 2012] Costa, L. H., de Amorim, M. D., Campista, M. E. M., Rubinstein, M., Florissi, P. e Duarte, O. C. M. B. (2012). Grandes massas de dados na nuvem: Desafios e técnicas para inovação. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*.

[Couto et al., 2012] Couto, R. S., Campista, M. E. M. e Costa, L. H. M. K. (2012). A reliability analysis of datacenter topologies. Em *Global Telecommunications Conference (GLOBECOM 2012), IEEE*, p. 1 –6.

[Greenberg et al., 2011] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P. e Sengupta, S. (2011). VL2: A scalable and flexible data center network. *Commun. ACM*, 54(3):95–104.

[Gude et al., 2008] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N. e Shenker, S. (2008). NOX: Towards an operating system for networks. Em *SIGCOMM Comput. Commun. Rev., 2008*, p. 105–110. ACM.

[Guimarães et al., 2013] Guimarães, P. H. V., Murillo P., A. F., Andreoni L., M. E., 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. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*, p. 101–164, Brasília, DF, Brazil.

[Mattos e Duarte, 2012] Mattos, D. M. F. e Duarte, O. C. M. B. (2012). QFlow: Um sistema com garantia de isolamento e oferta de qualidade de serviço para redes virtualizadas. Em *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2012*.

[McKeown et al., 2008] 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., 2008*.

[Mudigonda et al., 2010] Mudigonda, J., Yalagandula, P., Al-Fares, M. e Mogul, J. C. (2010). SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. Em *Proceedings of the 7th USENIX NSDI conference*, NSDI'10. USENIX Association.

[ns3, 2006] ns3 (2006). The ns3 network simulator. `http://www.nsnam.org/`.

[Raiciu et al., 2011] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D. e Handley, M. (2011). Improving datacenter performance and robustness with multipath TCP. Em *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, p. 266–277. ACM.

[Touch e Perlman, 2009] Touch, J. e Perlman, R. (2009). Transparent interconnection of lots of links (TRILL): Problem and applicability statement. RFC 5556 (Informational).

[Wilson et al., 2011] Wilson, C., Ballani, H., Karagiannis, T. e Rowtron, A. (2011). Better never than late: Meeting deadlines in datacenter networks. Em *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, p. 50–61. ACM.

[Zats et al., 2012] Zats, D., Das, T., Mohan, P., Borthakur, D. e Katz, R. (2012). Detail: Reducing the flow completion time tail in datacenter networks. *SIGCOMM Comput. Commun. Rev.*, 42(4):139–150.

# Redes Virtuais de Data Center Mapeadas como Serviço em Redes Definidas por Software

**Raphael Vicente Rosa, Christian Esteve Rothenberg, Edmundo R. M. Madeira**

[1] Universidade Estadual de Campinas (UNICAMP) – Campinas, SP – Brazil

raphaelvrosa@lrc.ic.unicamp.br, chesteve@dca.fee.unicamp.br, edmundo@ic.unicamp.br

***Abstract.*** *Based on Software Defined Network paradigm we apply the Network-as-a-Service model to build a data center network architecture well-suited to the problem of virtual networks embedding. This task is performed by the algorithm proposed in this paper, which is based on aggregated information from a virtual routing plane using the BGP protocol and a physical network topology based on OpenFlow 1.3 devices. The experimental evaluation shows that the proposed algorithm performs efficient load balancing on the data center network and altogether yields better utilization of the physical resources.*

***Resumo.*** *Com base nos conceitos de Redes Definidas por Software, aplicamos o modelo de Rede como Serviço na construção de uma arquitetura para dar suporte a alocação de redes virtuais de data center. Esta tarefa é realizada pelo algoritmo proposto neste trabalho, o qual tem como base a agregação de informações de roteamento em um plano virtual definido pelo protocolo BGP e de uma topologia física de rede com suporte a OpenFlow 1.3. Com base em experimentos realizados, mostramos que o algoritmo proposto realiza com eficácia o balanceamento de carga na rede permitindo melhor aproveitamento de recursos da topologia física de rede de data centers.*

## 1. Introdução

Provedores de infraestrutura (Infrastructure Providers - InPs) têm virtualizado os data centers que possuem criando uma diversidade de recursos (ex: máquinas virtuais, switches/roteadores virtuais) que são utilizados pelos próprios InPs e por diversos provedores de serviços (Service Providers - SP). Um data center virtualizado pode fornecer recursos compartilhados para diferentes SPs e seus respectivos data centers virtuais (Virtual Data Centers - VDCs) [Bari et al. 2013]. Tal modelo de negócio requer a virtualização de servidores e de redes de data center (Data Center Networks - DCNs). A repartição e uso eficiente de recursos de rede, como largura de banda, ainda é um desafio que diversas propostas de arquiteturas de DCNs vem tentando solucionar [Verdi et al. 2010].

Um SP compartilhando uma infraestrutura de DCN pode possuir em suas redes virtuais requisitos como isolamento de recursos, gerenciamento flexível de alocações de tráfego, tolerância a falhas, balanceamento de carga e implantação de novas aplicações. Estes, muitas vezes necessitam ser alterados dinamicamente conforme planos de serviço oferecidos por SPs. Nesse sentido, soluções tradicionais, atreladas às pilhas de protocolos TCP/IP (ex: VLANs), possuem diversas limitações tais como: não isolamento de desempenho; riscos de segurança; difícil implementação de novas aplicações; flexibilidade limitada de gerenciamento; e ausência de suporte a inovações de rede [Bari et al. 2013].

Seguindo o paradigma de Redes Definidas por Software (Software Defined Networking - SDN), neste trabalho propomos utilizar o conceito de Rede como Serviço (Network-as-a-Service - NaaS), como em [Keller and Rexford 2010], aplicado à construção de redes de data center virtualizadas. Propomos uma arquitetura que possibilita o tratamento de alocações dinâmicas de redes virtualizadas de data center seguindo critérios de largura de banda e resiliência. Consideramos um cenário onde um InP possua a necessidade de compartilhar sua infraestrutura física de data center para diversos SPs os quais já possuam suas demandas de alocações de máquinas virtuais (Virtual Machines - VMs) bem definidas.

Utilizando a plataforma RouteFlow [Nascimento et al. 2011], definimos um plano virtual utilizando o protocolo de roteamento BGP com suporte a múltiplos caminhos, considerando as premissas de [Lapukhov et al. 2013] para operar em DCNs com topologia de rede folded-Clos. No plano de dados, utilizamos uma infraestrutura física com suporte a versão 1.3 do padrão OpenFlow [ONF 2012]. E no plano de controle da plataforma, construímos serviços que agregam informações dos planos virtual e de dados, e logo, possibilitam que estas sejam utilizadas em conjunto para tratar o mapeamento de redes virtuais. O algoritmo proposto neste trabalho tem como tarefa principal alocar largura de banda em enlaces da topologia física do InP por meio do mapeamento de rotas da topologia virtual do InP para a criação de topologias virtualizadas para cada SP. Políticas, representando demandas de SPs, definem requisitos de endereçamento, largura de banda e resiliência, os quais são utilizadas pelo algoritmo de mapeamento para construir redes virtuais por meio da configuração de rotas na topologia física de rede do data center. Tais rotas são definidas para prover isolamento de recursos e suporte às funcionalidades que o padrão OpenFlow 1.3 oferece, como *group tables* e *meter tables* [ONF 2012].

A abordagem de alocação de redes virtuais proposta neste artigo avalia o algoritmo proposto para prover balanceamento de carga a DCNs e o compara a propostas existentes na literatura. Neste sentido, este trabalho se distingue dos demais pelos seguintes aspectos: (*i*) propõe a configuração de uma topologia virtual a qual utiliza o protocolo BGP de forma eficiente para o roteamento em data centers com topologia de rede folded-Clos; (*ii*) estende a plataforma RouteFlow para utilizar o padrão OpenFlow 1.3 e prover suporte a diferentes aplicações de políticas de rede, tais como reserva de largura de banda e rotas por múltiplos caminhos; (*iii*) e define um algoritmo eficaz de mapeamento de redes virtuais que propicia garantias de resiliência e largura de banda.

A estrutura deste artigo segue da seguinte forma, na seção 2 são apresentados os conceitos básicos relacionados a este trabalho. A seção 3 mostra a definição da arquitetura montada assim como a descrição de seus principais elementos. Na seção 4 apresentamos o algoritmo proposto neste trabalho. A seção 5 apresenta os experimentos realizados, os dados obtidos e uma discussão sobre a arquitetura e o algoritmo proposto. E a seção 6 conclui o artigo abordando trabalhos futuros a serem feitos.

## 2. Conceitos Básicos

Nesta seção descrevemos os principais componentes em uma DCN virtualizada, as propostas existentes na literatura sobre este tópico, assim como os elementos principais utilizados na construção da arquitetura proposta neste trabalho.

### 2.1. Virtualização de Redes de Data Center

Um data center é formado por servidores, equipamento de rede (ex: switches, roteadores, cabos) e sistemas de distribuição de energia e de resfriamento. A rede de um data center é definida pela topologia (ex: BCube, Fat-tree, Clos) em que seus switches/roteadores encontram-se conectados e pelos protocolos de rede utilizados na comunicação entre seus componentes (ex: IP, Ethernet). Uma DCN geralmente possui a seguinte configuração topológica. Servidores são agregados em racks e se conectam a um switch Top-of-Rack (ToR). Este se conecta a switches End-of-Row (EoR), que não se interconectam, e são utilizados como intermediários em conexões de ToRs com switches Core [Verdi et al. 2010]. Um VDC se define como um conjunto de recursos virtualizados (ex: VMs, switches/roteadores virtuais) conectados por enlaces virtuais. Neste cenário, uma rede virtual é definida como um conjunto de recursos de rede virtualizados conectados por enlaces virtuais. Ou seja, a virtualização de redes, similar a virtualização de servidores, permite a criação de múltiplas redes virtuais sobre um mesmo substrato de rede físico as quais são implementadas e gerenciadas de forma independente [Bari et al. 2013].

Em trabalhos recentes sobre virtualização de redes de data center, os seguintes desafios são abordados: tecnologias de virtualização; topologias de rede; isolamento de desempenho; escalabilidade; tolerância a falhas; e técnicas de encaminhamento de pacotes. Nessas pesquisas, recursos de DCNs são utilizados de duas formas, por competição ou compartilhamento. No primeiro caso, destacam-se as propostas Seawall [Shieh et al. 2010], Netshare [Lam et al. 2012] e FairCloud [Popa et al. 2011], as quais propõem disputa justa de recursos de rede por multiplexação estatística e requisitos mínimos de largura de banda a VMs, mas não fornecem garantias determinísticas de recursos de rede (ex: latência, largura de banda). Já no segundo caso, as principais propostas são Gatekeeper [Rodrigues et al. 2011], SecondNet [Guo et al. 2010], Oktopus [Ballani et al. 2011] e Proteus [Xie et al. 2012]. Estas realizam a alocação de garantias mínimas de largura de banda a conjuntos de VMs, os quais são definidos de diferentes formas, tais como heurísticas e análises temporais de padrões de comunicação entre VMs.

### 2.2. RouteFlow

RouteFlow [Nascimento et al. 2011] é uma arquitetura de roteamento que segue o paradigma de SDNs ao centralizar logicamente o controle da rede, unificar o estado de informação e desacoplar as lógicas de encaminhamento e configuração de equipamentos de rede. A plataforma RouteFlow permite oferecer serviços de roteamento IP definidos em um plano virtual de controle e mapeados aos recursos de uma infraestrutura física de rede com suporte ao padrão OpenFlow permitindo a oferta da plataforma como serviço à rede pelo mapeamento flexível de recursos de uma topologia virtual sobre uma infraestrutura física de rede a qual pode ser distribuída e compartilhada. Constituída por três planos, virtual, físico e de controle, RouteFlow possui respectivamente em cada um destes planos seus seguintes componentes principais: rfclient, rfproxy e rfserver.

O plano virtual executa roteadores virtuais, interconectados por um switch com suporte a OpenFlow, definidos em nível de virtualização do sistema operacional Linux (Contêineres Linux - LXCs). Neles, a aplicação rfclient captura as rotas computadas por uma suíte de roteamento (ex: Quagga, XORP, BIRD) e as envia ao plano de controle. O plano físico é formado por switches com suporte a OpenFlow interligado ao(s) controlador(es) de rede OpenFlow, que executa(m) a aplicação rfproxy, responsável por realizar

a interface entre o plano de controle e o controlador de rede. No plano de controle, a aplicação rfserver armazena todo o estado de configuração e define o mapeamento entre topologias física e virtual. Ela também gerencia o estado do mapeamento físico/virtual contido em um banco de dados (ex: MongoDB) e realiza a troca e formatação de mensagens entre as aplicações rfclient e rfproxy, tais como rotas computadas em LXCs no plano virtual. Desde sua concepção, há mais de três anos [Nascimento et al. 2011], RouteFlow tem evoluído significativamente até o ponto de atualmente estar operando em pontos de troca de tráfego [Stringer et al. 2013].

## 3. Arquitetura Proposta

Com base na necessidade de uma arquitetura que desse suporte à criação e instanciação de redes virtuais em DCNs, vimos na plataforma RouteFlow as características que permitem o controle direto da rede, a unificação do estado das informações dos planos virtual e de dados, e o desacoplamento das lógicas de encaminhamento e configuração de equipamentos de rede. Também encontramos na proposta [Lapukhov et al. 2013] as características de configuração do protocolo BGP para o roteamento intra-AS em DCNs que foram ao encontro das funcionalidades da plataforma RouteFlow. Essa proposta define o uso do protocolo BGP com suporte a múltiplos caminhos sobre uma topologia de rede folded-Clos. Dentre suas vantagens podemos citar: design prático de roteamento para grandes data centers; protocolo de simples implementação com baixa complexidade de código e fácil suporte operacional; minimização do domínio de falha de equipamentos e protocolo de roteamento; diminuição de custos operacional e de capital.

Logo, utilizamos a plataforma RouteFlow para propor uma arquitetura para DCNs (vide Figura 2) com topologia virtual definida por roteamento com protocolo BGP, plano de dados com suporte a OpenFlow 1.3, e plano de controle com total gerenciamento sobre as topologia física e virtual do data center para dar suporte ao algoritmo de mapeamento de redes virtuais proposto neste trabalho. A arquitetura permite que *group tables* e *meter tables* sejam definidas pelo plano de controle para serem programadas pela aplicação rfproxy, de modo que fique transparente para ambos os planos quais os detalhes de implementação das funcionalidades de cada um. Nesta seção, descrevemos as modificações na plataforma RouteFlow feitas para sua adequação aos requisitos de mapeamento de elementos da infraestrutura física de um InP em uma topologia virtual para operação de uma DCN. Estas descrições estão separadas de acordo com os planos da arquitetura nas seguintes subseções: plano físico, plano virtual e plano de controle.

### 3.1. Plano Físico

Neste plano utilizamos uma topologia de rede folded-Clos contendo switches com suporte a OpenFlow 1.3. Em um controlador de rede com suporte a esta mesma versão de OpenFlow construímos a aplicação rfproxy. Esta, auxiliada pela aplicação de descoberta de topologia, captura eventos de adição e remoção de switches e enlaces na topologia física e os encaminha ao plano de controle para a definição da topologia física disponível aos mapeamentos. Funcionalidades do padrão OpenFlow 1.3 são utilizadas para dar suporte aos mapeamentos de redes virtuais, tais como a utilização de *group tables* para encaminhamento de dados semelhante ao mecanismo de roteamento ECMP (Equal-Cost Multipath), e reservas de largura de banda definidas por *meter tables*.

A programação de rotas de topologias virtuais de SPs na topologia física ocorre pela identificação única de uma rede de um SP por tags MPLS. Em switches Core e EoR, o tráfego é encaminhado somente por *matches* destas tags, definidas em rotas programadas na tabela 0. Já em ToRs, quatro tabelas são programadas. A tabela 0 trata *matches* em tags MPLS com tráfego destinado ao interior do rack e realiza a ação de retirada desta camada MPLS e o encaminhamento para a tabela 1. Nesta, o *match* ocorre por endereços de rede, e tem como ações a reescrita de endereços MAC nos pacotes e o encaminhamento para a próxima tabela. Na tabela 2, o *match* com os enredeços MAC de servidores gera a ação de encaminhamento de pacotes para as devidas portas em que eles estão conectados. Por fim, a tabela 3 trata o tráfego a ser encaminhado para fora do rack. Logo, esta tem como ação a marcação da tag MPLS relativa à rota de uma rede virtual de um SP, definida no plano de controle. Todas as rotas de uma política são programadas em switches com um tempo de *hard timeout* o qual define o tempo de permanência de uma topologia virtual no plano de dados. Este tempo é configurado pelo plano de controle, e nos switches do plano de dados, quando expirado, define a exclusão das rotas por ele definidas. Limitadores de largura de banda são programados em ToRs delimitando em racks o tráfego de entrada e saída por *meter tables* associadas respectivamente às regras das tabelas 2 e 3.

### 3.2. Plano Virtual

No plano virtual foi adicionada à aplicação rfclient a característica de leitura de rotas *multipath*. O roteamento em LXCs foi definido pela suíte de roteamento Quagga executando o protocolo BGP, configurado com as vantagens de uso intra-AS para DCNs em topologia folded-Clos [Lapukhov et al. 2013]. Como visto na Figura 1, realizamos o mapeamento das topologias base, física e virtual, as quais representam os planos de dados e virtual de um InP da seguinte forma: agregamos elementos que contém o mesmo ASN em uma camada da topologia física folded-Clos. Isto foi possível, pois elementos de uma mesma camada de switches da topologia folded-Clos não estão interconectados entre si, possuem rotas equidistantes a quaisquer outros elementos da topologia, e computam rotas com mesmo AS-PATH para todos os destinos da rede. Neste plano, também definimos que inicialmente todas as mensagens de controle são transferidas entre os planos virtual e de dados através do controlador de rede. Após mapeadas as topologias base, física e virtual, rotas são definidas no switch do plano virtual para que as mensagens de controle permaneçam na topologia virtual, não sobrecarregando o controlador de rede.

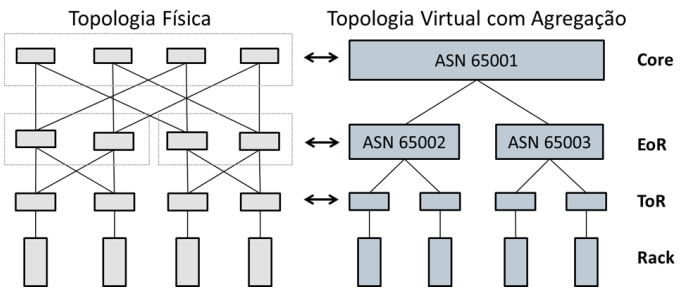



**Figura 1. Mapeamento de topologias física e virtual com agregação**

### 3.3. Plano de Controle

Neste plano é executada a aplicação rfserver, o banco de dados que mantém o estado do mapeamento entre os planos físico e virtual, e os componentes desenvolvidos neste trabalho (vide Figura 3): **Recurso**; **Politica**; **Configuração**; **Alocador**; e **Escalonador**.

**Recurso** realiza o armazenamento e gerenciamento de informações dos planos virtual e de dados. Para isso faz a criação de classes as quais representam os recursos de cada plano. As classes *TopologiaFísica* e *TopologiaVirtual* tem como atributos componentes que representam respectivamente os planos físico (ex: switches, enlaces) e virtual (ex: LXCs, interfaces, rotas). Estas duas classes são herdeiras da classe *Topologia* a qual constrói grafos utilizados para representá-las. Neste componente também define-se a classe *Topologias* a qual possui funções de instanciação de topologias físicas e virtuais, gerenciamento do mapeamento entre elas, preenchimento de suas configurações tais como rotas, enlaces e switches ou LXCs, além de funções de análise das configurações e estados das topologias física e virtuais. Além disso, a classe *Servidores* trata toda a representação de VMs em racks de servidores.

**Política** é responsável pelo gerenciamento de demandas de alocação de redes virtuais com requisitos de largura de banda, resiliência e esquemas de endereçamento entre VMs e servidores que irão constituir a topologia virtual do VDC estabelecido a um SP. Ou seja, a partir da determinação do mapeamento de VMs em servidores interconectados a ToRs, previamente estipulada por SP e InP, uma política representa este mapeamento. Ela contém os endereços das VMs, servidores e ToRs desta demanda, assim como a matriz de tráfego entre eles. A topologia de rede virtual de uma política, criada pelo mapeamento desta no plano de dados pelo componente Configuração e representada pela instância de uma *TopologiaVirtual* do componente Recurso, é armazenada e associada a um identificador MPLS único desta política, utilizado na programação de rotas no plano de dados. Dentro deste componente a classe denominada Políticas realiza o cadastramento, gerenciamento e armazenamento das políticas criadas bem como de suas respectivas topologias virtuais construídas e mapeadas.

**Configuração** contém algoritmos de mapeamento que realizam tanto a associação entre topologias quanto o mapeamento de rotas da topologia base virtual para a criação de topologias virtuais. Além disso, dentro deste componente existem algoritmos que auxiliam nas funções principais de mapeamento, os quais lidam com modificações de recursos das topologias base para melhor atender as demandas de mapeamento.

**Alocador** realiza toda a interface de configuração de topologias virtuais no plano físico. Por meio dele são configuradas topologias virtuais, mapeamentos são feitos utilizando o componente Configuração e topologias virtuais são mapeadas na topologia física do InP com o envio de mensagens à aplicação rfproxy. Além disso, ele também realiza a configuração de rotas no switch virtual para o gerenciamento do tráfego de controle do plano virtual do InP.

**Escalonador** é o componente responsável por gerir todo o funcionamento dos componentes anteriores. Por meio dele, são criadas políticas, constituídas e configuradas topologias virtuais e físicas e onde demandas de mapeamento de topologias são definidas em políticas para serem alocadas e desalocadas dinamicamente. Ou seja, este componente realiza toda a interface de comunicação com a aplicação rfserver.

### 3.4. Execução da Arquitetura

O funcionamento da arquitetura proposta ocorre da seguinte forma. Os elementos da plataforma são inicializados nesta ordem: LXCs e aplicações rfclient; switch do plano virtual; banco de dados; aplicação rfserver; controlador de rede e aplicação rfproxy; switches da topologia física. A aplicação rfserver realiza a inicialização de todos os componentes descritos na subseção anterior, tratando-os como serviços da plataforma RouteFlow.

O componente Escalonador (vide Figura 3), após aguardar a convergência de rotas do plano virtual e a descoberta de topologia do plano físico, cria uma política padrão a qual é definida pelo mapeamento das topologias base, física e virtual. Neste primeiro momento, os objetos que representam estas topologias são instanciados para serem utilizados nos mapeamentos posteriores. Além disso, o objeto que representa servidores é criado para definir a instanciação de políticas, ou seja, o mapeamento de VMs em servidores e, consequentemente, em ToRs. Em seguida, três *threads* são iniciadas para realizarem as tarefas de: (*i*) criação de demandas de mapeamento (ex: geradas pelo OpenStack) e definição de políticas; (*ii*) alocação de políticas para criação de topologias virtuais; e (*iii*) desalocação de políticas conforme tempo de mapeamento atingido.

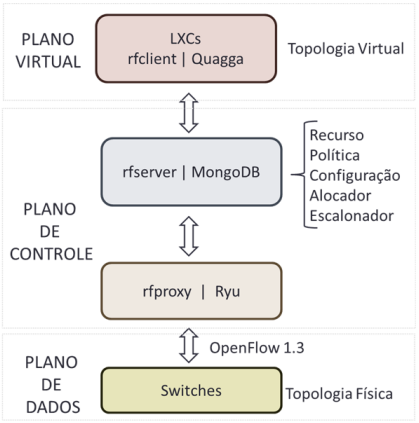



**Figura 2. Arquitetura Proposta**


**Figura 3. Componentes do Plano de Controle**

A primeira *thread*, com base em uma matriz de tráfego entre VMs, aloca estas em racks de servidores utilizando o objeto *Servidores*, criando uma matriz de tráfego entre ToRs. Estas informações são agregadas juntamente com requisitos de largura de banda, resiliência e endereçamento, definindo uma política que é colocada em uma fila para ser alocada. A *thread* de alocação verifica esta fila e realiza a alocação de políticas na topologia física base por meio do algoritmo de mapeamento proposto neste trabalho, e logo, cria a topologia virtual associada a esta política. Em função do tempo de alocação definido para cada política, a *thread* de desalocação retira o mapeamento de topologias virtuais da topologia física base.

## 4. Algoritmos do Plano de Controle

Existem dois algoritmos principais no plano de controle: (*i*) o *algoritmo de suporte* auxilia o gerenciamento de recursos da topologia física para que (*ii*) o *algoritmo de mapeamento* proposto neste trabalho realize a alocação de topologias virtuais de forma consistente.

### 4.1. Algoritmo de Suporte

Em cada um dos switches da topologia física base do plano de controle, dois atributos se destacam. O primeiro define uma tabela (*bw_port_table*) de porcentagem de largura de banda em cada porta de um switch. Cada vez que um enlace adjacente a um switch é adicionado a topologia física base ou uma rota é mapeada a este enlace, o percentual de largura de banda disponível neste enlace é calculado e armazenado na *bw_port_table* dos switches adjacentes, na forma {*porta adjacente a enlace : porcentagem de largura de banda disponível*}. O outro atributo diz respeito à tabela (*bw_table*) de porcentagem de largura de banda disponível para endereços de destino das rotas de um switch. Cada vez que uma rota é mapeada a um enlace, os switches adjacentes atualizam suas tabelas de largura de banda conforme o endereço de destino da rota mapeada, constituindo entradas na forma {*endereço de destino : [porta de destino de rota a endereço : porcentagem de largura de banda disponível para a rota]* }.

**Algoritmo 1** : Executado para atualizar *bw_tables* de switches da topologia física base

**Entrada:** topologia física base (topo_phy_base)
**Saída:** *bw_tables* atualizadas de todos switches de topologia física base (topo_phy_base)

```
1:  para todo switch ToR em topo_phy_base faça
2:      filaSwitches.adicionaItem(ToR, PrefixoDeRedeDeToR, ToR.bw_table, porta=1)
3:  fim para
4:  enquanto filaSwitches não vazia faça
5:      (switch, PrefixoDeRede, bw_tableDeSwitch, porta) = filaSwitches.retiraItem()
6:      se (switch, PrefixoDeRede, porta) presente em SwitchesEmFila então
7:          bw_tableDeSwitch ← SwitchesEmFila(switch, PrefixoDeRede, porta)
8:          SwitchesEmFila.removeItem(switch, PrefixoDeRede, porta)
9:      fim se
10:     SwitchesVisitados.adicionaItem(switch, PrefixoDeRede, porta)
11:     bw_usage_list ← lista de todos os valores de largura de banda em switch.bw_table com destino a PrefixoDeRede
12:     bw_usage_mean ← Maior valor a ser alocado igualmente em todas entradas de bw_usage_list para PrefixoDeRede
13:     se porta contida em enlaces a outros switches então
14:         switch_bw_table[PrefixoDeRede][porta] ← min(bw_usage_mean,switch._bw_port_table[porta])
15:     fim se
16:     para todo enlace adjacente a switch não contendo porta faça
17:         se (enlace.switchDestino,PrefixoDeRede,enlace.portaDestino) não presente em SwitchesEmFila e SwitchesVisitados
            então
18:             filaSwitches.adicionaItem(enlace.switchDestino,PrefixoDeRede,switch_bw_table,enlace.portaDestino)
19:         fim se
20:         se (enlace.switchDestino,PrefixoDeRede,enlace.portaDestino) presente em SwitchesEmFila então
21:             SwitchesEmFila(switch,PrefixoDeRede,porta) ← switch_bw_table
22:         fim se
23:     fim para
24: fim enquanto
```

É importante ressaltar que uma *bw_table* compreende a largura de banda fim-a-fim disponível para uma rota, enquanto que a *bw_port_table* representa a largura de banda disponível local, ou seja, somente em enlaces adjacentes a switches. Além disso, switches ToR tem em suas *bw_table* os valores de largura disponíveis para cada um dos servidores que possuem, definindo assim a largura de banda disponível em um rack.

Tendo como base estes atributos, o algoritmo 1 atualiza as estruturas *bw_table* de todos os switches da topologia física base tornando-as consistentes com o estado das políticas mapeadas, e logo, propicias à utilização no algoritmo 1. Ou seja, este algoritmo, realizando uma busca em largura no grafo da topologia física base, faz com que todos os switches da topologia física base tenham informações atualizadas sobre a largura de banda disponível em cada rack da rede. Esta atualização ocorre somente na topologia física base, toda vez que uma política é criada, alocada e desalocada.

### 4.2. Algoritmo de Mapeamento Proposto

O algoritmo 2 abaixo, define uma busca em largura no grafo da topologia física base. Os nós de entrada são os switches ToR que uma política possui definidos pela matriz de tráfego de VMs mapeadas em racks de servidores. A saída do algoritmo é uma topologia virtual construída com base em informações de rotas da topologia virtual base e disponibilidade de largura de banda da topologia física base. O mapeamento ocorre por anotações de largura de banda, definidas pelo identificador único de uma política, feitas em enlaces da topologia física base. Estas marcações são realizadas de acordo com rotas selecionadas da topologia virtual pelo algoritmo 3 perante requisitos de largura de banda e resiliência definidos nas políticas de mapeamento.

**Algoritmo 2** : Executado para mapear topologias virtuais na topologia física base

**Entrada:** topologia física base (topo_phy_base), topologia virtual base (topo_virt_base) e política
**Saída:** topologia virtual

1: **para todo** switch ToR em matriz de tráfego de política **faça**
2: **para todo** PrefixoDeRede em matriz de tráfego de política ≠ faixa de endereços de rede de switch ToR **faça**
3: lxc ← lxc de topo_virt_base mapeado a ToR
4: filaSwitches.adicionaItem(lxc, ToR, PrefixoDeRede)
5: PropriedadesSwitches(lxc, ToR, PrefixoDeRede) ← Largura de banda de ToR a PrefixoDeRede em matriz de tráfego de política
6: **fim para**
7: **fim para**
8: **enquanto** filaSwitches não vazia **faça**
9: (lxc, switch, PrefixoDeRede) ← filaSwitches.retiraItem()
10: **se** PrefixoDeRede não presente em entradas de SwitchesEmFila **então**
11: SwitchesVisitados.adicionaItem(lxc, switch, PrefixoDeRede)
12: **fim se**
13: larguraDeBandaRequisitada = PropriedadesSwitches(lxc, switch, PrefixoDeRede)
14: switch_rotas_selecionadas ← SelecionaRotas(switch,lxc,PrefixoDeRede,larguraDeBandaRequisitada) # Algoritmo 3
15: larguraDeBandaRotas←larguraDeBandaRequisitada dividida pela quantidade de switch_rotas_selecionadas
16: **para todo** rota em switch_rotas_selecionadas **faça**
17: **se** larguraDeBandaRotas[rota] alocada em enlace de topo_phy_base definido por rota **então**
18: Adicione switch, rota em switch e enlace em TopologiaVirtual
19: Defina como lxcDestino e switchDestino os respectivos lxc e switch de destino do enlace definido por rota
20: filaSwitches.adicionaItem(lxcDestino, switchDestino, PrefixoDeRede)
21: PropriedadesSwitches(lxcDestino, switchDestino, PrefixoDeRede) ← larguraDeBandaRotas[rota]
22: **else**
23: Mapeamento ← Falso
24: Pare os laços de execução
25: **fim se**
26: **fim para**
27: **fim enquanto**
28: **se** Mapeamento ≠ Verdadeiro **então**
29: Desfaça mapeamentos até então feitos de politica em topo_phy_base
30: **fim se**

**Algoritmo 3** SelecionaRotas: Executado para selecionar rotas para mapeamento

**Entrada:** (switch, lxc, PrefixoDeRede, larguraDeBandaRequisitada)
**Saída:** rotas selecionadas para mapeamento

1: switch_rotas ← lxc.get_rotas(PrefixoDeRede)
2: Seleciona switch_rotas com maior capacidade em switch.bw_port_table que satisfaçam critério de resiliência de política
3: **se** Opção SelecRotas Agreg Proposto **então**
4: Calcula média, desvio padrão, maior e menor valor de entradas de switch.bw_port_table com portas definidas por switch_rotas
5: Seleciona combinações de rotas de switch_rotas que satisfazem larguraDeBandaRequisitada dividida entre elas **e** que definam valores de switch._bw_port_table maiores ou iguais a média subtraída do dobro do desvio padrão de switch.bw_port_table
6: Para as combinações de rotas selecionadas na etapa anterior selecione aquela que possui a menor diferença entre os valores máximo e mínimo caso seus valores de largura de banda sejam selecionados e aplicados a switch.bw_port_table
7: Retorne conjunto de rotas selecionadas na etapa anterior
8: **fim se**
9: **se** Opção SelecRotas Tradicional **então**
10: Retorna rota de switch_rotas que satisfaz larguraDeBandaRequisitada em entrada PrefixoDeRede de switch.bw_table
11: **fim se**

Na opção "SelecRotas Agreg Proposto" do algoritmo 3, os critérios de resiliência de políticas são definidos por porcentagens, as quais expressam a quantidade de rotas que serão selecionadas para a avaliação de disponibilidade de caminhos. Além disso, nesta opção, a utilização do parâmetro de seleção de rotas, média subtraída do dobro do desvio padrão, permite que as rotas selecionadas estejam dentro de dois percentis da média dos valores de *bw_port_tables*. Isto garante que haja balanceamento de carga nas portas de um switch, ao contrário da opção "SelecRotas Tradicional", onde somente uma rota com menor disponibilidade de largura de banda é selecionada e nenhum critério de balanceamento de carga é realizado.

## 5. Avaliação Experimental

O testbed utilizado para a avaliação da arquitetura proposta neste trabalho foi montado utilizando a plataforma RouteFlow com as respectivas modificações mencionadas anteriormente. No plano de dados construímos a aplicação rfproxy sobre o controlador de rede Ryu[1], com as devidas modificações para suporte a OpenFlow 1.3. Utilizamos o switch ofsoftswitches13[2] tanto no plano de dados quanto no plano virtual. Realizamos a construção de duas topologias folded-Clos utilizando o emulador Mininet[3], com 12 e 48 switches. No plano virtual realizamos a construção do mapeamento com agregação para as topologias físicas criadas. Consideramos na topologia física e no plano de controle que cada enlace entre swiches possui 10.000 unidades de largura de banda, e entre switches e servidores 1.000 unidades de largura de banda. Além disso, consideramos que nas topologia físicas com 12 e 48 switches existam respectivamente 20 e 40 servidores em cada rack, podendo cada servidor possuir no máximo 20 VMs alocadas.

### 5.1. Experimentos e Resultados

Inicialmente fizemos um estudo analítico sobre a abordagem de construção da arquitetura com a utilização do protocolo BGP em roteadores virtuais definidos pelo mapeamento com agregação de switches da topologia física. Nesse aspecto, buscamos observar uma comparação entre as topologias folded-Clos comum e com agregação, referenciadas respectivamente por topoClos e topoClosAgreg. Na Tabela 1 vemos que a topologia virtual com agregação determina um número menor de conexões entre roteadores virtuais, menor quantidade de conexões TCP para as sessões BGP e, consequentemente, menor número de mensagens trocadas em cada rodada de atualizações do protocolo BGP.

**Tabela 1. Comparação de topologias**

| Topologia | Switches Plano de Dados | Roteadores Virtuais | Conexões entre Roteadores | Mensagens por rodada de atualização |
|---|---|---|---|---|
| topoClos | 12 | 12 | 16 | 32 |
| topoClosAgreg | 12 | 7 | 6 | 12 |
| topoClos | 48 | 48 | 64 | 128 |
| topoClosAgreg | 48 | 9 | 8 | 16 |
| topoClos | 96 | 96 | 256 | 512 |
| topoClosAgreg | 96 | 11 | 34 | 68 |
| topoClos | 128 | 128 | 1024 | 2048 |
| topoClosAgreg | 128 | 35 | 68 | 136 |

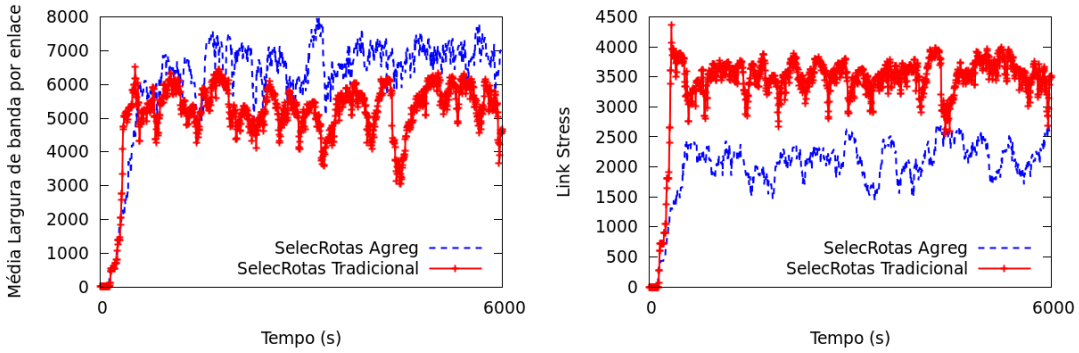
[1]https://github.com/osrg/ryu

[2]https://github.com/CPqD/ofsoftswitch13

[3]https://github.com/mininet/mininet

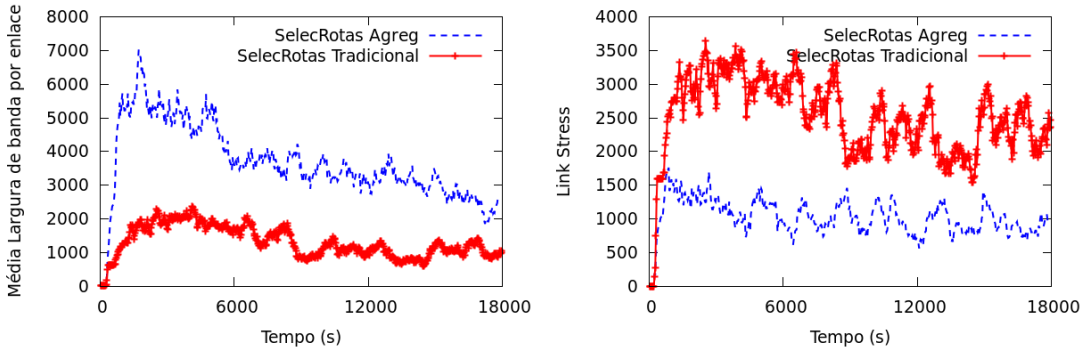
Adiante tratamos de avaliar o algoritmo "SelecRotas Agreg" proposto neste trabalho e compará-lo ao algoritmo "SelecRotas Tradicional". Para isso, realizamos a criação de demandas de mapeamento em um processo de poisson com VMs sendo alocadas de forma ordenada em servidores de ToRs conforme a disponibilidade de recursos em servidores. Consideramos os seguintes parâmetros para a realização deste experimento: chegada de demandas por processo de Poisson com média de 30 requisições por minuto. Cada requisição possui, respectivamente para as topologias com 12 e 48 switches: quantidade de máquinas virtuais uniformemente distribuída entre 10 e 30 e entre 30 e 70; demanda de tráfego entre VMs uniformemente distribuída entre 1 e 10 unidades de largura de banda; redes virtuais mapeadas com tempo de permanência na rede uniformemente distribuído entre 180 e 300 e entre 540 e 660 segundos; e tempo total de experimento definido em 6000 e 18000 segundos.

Neste sentido, avaliamos a largura de banda e sua variação (*link stress*) nos enlaces da rede, para entendermos o balanceamento de carga na topologia física base. Vemos em todos os experimentos que para as topologias com 12 (Figura 4) e 48 (Figura 5) switches, que o algoritmo "SelecRotas Agreg" obteve maior utilização de largura de banda na rede e menor link stress do que o algoritmo "SelecRotas Tradicional", contribuindo portanto, para o uso eficiente dos recursos da rede. É importante ressaltar, que em todos os experimentos obtivemos 100% de taxa de aceitação das requisições de mapeamento.




(a) Média de largura de banda utilizada por enlace (b) Link stress

**Figura 4. Dados de topologia física folded-Clos com 12 switches**




(a) Média de largura de banda utilizada por enlace (b) Link stress

**Figura 5. Dados de topologia física folded-Clos com 48 switches**

Buscando obter critérios de desempenho da arquitetura criada e do algoritmo proposto, realizamos experimentos para avaliar o tempo de mapeamento de demandas conforme o tamanho de requisições de redes virtuais. Discriminamos os tempos de operações de: pré configuração e checagem de topologias; mapeamento da rede virtual; e configuração das rotas na topologia física. Realizamos experimentos variando o tamanho das políticas de mapeamento, com matrizes de tráfego entre 1 e 16 racks, na topologia com 48 switches. Observamos na Figura 6 que os tempos de pré-configuração tem definição praticamente constante por não dependerem do tamanho das políticas de mapeamento. Já os tempos de mapeamento e configuração crescem conforme o tamanho das políticas devido a quantidade de rotas a serem analisadas, selecionadas, mapeadas e configuradas nos racks. No caso da topologia analisada, notamos que existem pequenos saltos de tempo conforme uma política utiliza racks não interconectados aos mesmos switches EoR, o que se deve à necessidade de mapeamento de rotas em switches Core.

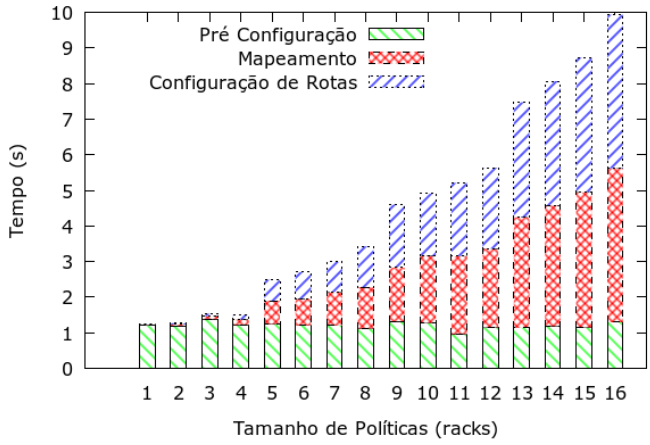



**Figura 6. Tempos de Mapeamento**

## 5.2. Discussão

Notamos nos resultados obtidos, com experimentos nas topologias folded-Clos com 12 e 48 switches (Figuras 4 e 5), que o algoritmo proposto no nosso trabalho realiza o balanceamento de carga na rede de forma eficiente quando comparado ao algoritmo "SelecRotas Tradicional", comumente visto na literatura (ex: [Ballani et al. 2011] e [Xie et al. 2012]). Isto pois, determina maior utilização e melhor espalhamento de alocações de largura de banda nos enlaces da rede. Além disso, vimos na Tabela 1 que o protocolo BGP, além de possuir as vantagens citadas no início da seção 3, quando implementado na topologia virtual folded-Clos com agregação, propicia as vantagens de menores utilização de roteadores virtuais e overhead do protocolo BGP. Por fim, vemos que a complexidade dos algoritmos apresentados condiz com os resultados apresentados na Figura 6, onde os tempos de mapeamento e de configuração de rotas crescem linearmente com o tamanho da política de requisição.

Diferente das propostas existentes na literatura, até onde temos conhecimento, a arquitetura construída neste trabalho é a única que faz utilização do padrão OpenFlow 1.3 e aborda explicitamente requisitos de balanceamento de carga na rede. Além disso, não só a utilização de múltiplos caminhos, como também de critérios de resiliência de políticas,

é uma abordagem até então desconhecida na literatura de DCNs. Agregar informações em grafos de topologias física e virtual em um plano de controle logicamente centralizado, permite uma diversidade de oportunidades de controle da rede e mapeamento de redes virtuais. Logo, tanto questões relacionadas a formas de endereçamento, flexibilidade da rede, desenvolvimento e criação de novas aplicações de rede, entre outras, podem ser feitas com total liberdade sobre a arquitetura proposta. Tal fato permite a abordagem de diversos problemas de DCNs pela extensão deste trabalho, considerando para isso diferentes requisitos de operação destas redes.

A arquitetura construída também permite um alto grau de liberdade na criação de redes virtuais. Toda a agregação de estado dos planos físico e virtual da arquitetura propicia ao plano de controle uma visão não conhecida nos trabalhos relacionados a esta proposta. Primeiro, pois objetivos de alto nível podem ser facilmente implementados tanto em políticas criadas conforme demandas de alocação de redes virtuais quanto na rede interna do próprio InP, seja pela programação de regras na seleção de rotas ou pela definição de configurações na própria execução do protocolo BGP no plano virtual. Do mesmo modo, a permissividade de características do padrão OpenFlow 1.3 dá ao mapeamento de redes virtuais todos os parâmetros necessários para programá-las de diferentes formas, como por exemplo, definindo a seletividade de tráfego de uma rede virtual somente por portas TCP ou diferentes protocolos de rede (ex: VXLAN, NVGRE).

## 6. Conclusões e Trabalhos Futuros

Neste artigo buscamos avaliar a alocação de redes virtuais em data centers utilizando o modelo de Network-as-a-Service com a aplicação de conceitos de Redes Definidas por Software. Construímos uma arquitetura utilizando a plataforma RouteFlow a fim de moldar todo o ambiente de rede, assim como os requisitos de operação de uma arquitetura de data center. Utilizamos para isso o mapeamento da infraestrutura física da rede de data center em uma topologia virtual onde rotas provenientes do protocolo BGP, configurado precisamente para este ambiente, foram obtidas para a alocação de topologias de data centers virtuais. Por meio de algoritmos implementados sobre o plano de controle da plataforma RouteFlow, conseguimos mostrar eficácia na alocação de redes virtuais considerando os fatores de utilização de largura de banda e balanceamento de carga da rede, e o overhead do protocolo de roteamento.

Vemos neste trabalho que muitos desafios surgem da aplicação de conceitos de SDN em DCNs, principalmente pelo uso do conceito de Network-as-a-Service. A avaliação de critérios de desempenho e escalabilidade em DCNs, utilizando novas formas de endereçamento, mapeamento de VMs em servidores, outras funcionalidades do OpenFlow 1.3, são tópicos de pesquisa que já estão em andamento. Futuramente iremos estender o funcionamento da plataforma para suportar tolerância a falhas em todos os seus planos. Buscaremos também elaborar novos algoritmos que levem em consideração requisitos de gerenciamento de energia no que diz respeito à utilização de recursos de rede e como estes interesses podem ir de encontro às abordagens resilientes de mapeamento de redes virtuais em data centers. Por exemplo, a seleção de rotas poderia ser estabelecida de diversas formas, considerando pesos de enlaces, políticas de roteamento interno, ou mesmo critérios de utilização de energia em enlaces e resiliência de caminhos. E por fim, vemos que os nós de entrada do algoritmo proposto podem ser estendidos a servidores, tornando esta solução flexível à programação de switches virtuais.

## 7. Referências Bibliográficas

Ballani, H., Costa, P., Karagiannis, T., and Rowstron, A. (2011). Towards predictable datacenter networks. In *Proceedings of the ACM SIGCOMM 2011*, pages 242–253, New York, NY, USA. ACM.

Bari, M., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q., and Zhani, M. (2013). Data center network virtualization: A survey. *Communications Surveys Tutorials, IEEE*, 15(2):909–928.

Guo, C., Lu, G., Wang, H. J., Yang, S., Kong, C., Sun, P., Wu, W., and Zhang, Y. (2010). Secondnet: A data center network virtualization architecture with bandwidth guarantees. In *Proceedings of Co-NEXT '10*, pages 15:1–15:12, New York, NY, USA. ACM.

Keller, E. and Rexford, J. (2010). The "platform as a service" model for networking. In *Proceedings of INM/WREN'10*, pages 4–4, Berkeley, CA, USA. USENIX Association.

Lam, V. T., Radhakrishnan, S., Pan, R., Vahdat, A., and Varghese, G. (2012). Netshare and stochastic netshare: Predictable bandwidth allocation for data centers. *SIGCOMM Comput. Commun. Rev.*, 42(3):5–11.

Lapukhov, P., Premji, A., and J. Mitchell, E. (2013). Use of bgp for routing in large-scale data centers. Internet-Draft draft-lapukhov-bgp-routing-large-dc-06.txt, IETF Secretariat.

Nascimento, M. R., Rothenberg, C. E., Denicol, R. R., Salvador, M. R., and Magalhaes, M. F. (2011). Routeflow: Roteamento commodity sobre redes programáveis. *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos-SBRC*.

ONF (2012). Openflow switch specification version 1.3.0 (wire protocol 0x04). Technical report, ONF.

Popa, L., Krishnamurthy, A., Ratnasamy, S., and Stoica, I. (2011). Faircloud: Sharing the network in cloud computing. In *Proceedings of the 10th ACM HotNets*, pages 22:1–22:6, New York, NY, USA. ACM.

Rodrigues, H., Santos, J. R., Turner, Y., Soares, P., and Guedes, D. (2011). Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *Proceedings of WIOV'11*, pages 6–6, Berkeley, CA, USA. USENIX Association.

Shieh, A., Kandula, S., Greenberg, A., and Kim, C. (2010). Seawall: Performance isolation for cloud datacenter networks. In *Proceedings of the 2Nd USENIX HotCloud*, pages 1–1, Berkeley, CA, USA. USENIX Association.

Stringer, J. P., Fu, Q., Lorier, C., Nelson, R., and Rothenberg, C. E. (2013). Cardigan: Deploying a distributed routing fabric. In *Proceedings of the Second ACM SIGCOMM HotSDN '13*, pages 169–170, New York, NY, USA. ACM.

Verdi, F. L., Rothenberg, C. E., Pasquini, R., and Magalhaes, M. (2010). Novas arquiteturas de data center para cloud computing. In *28th Brazilian SBRC 2010 - Minicursos*.

Xie, D., Ding, N., Hu, Y. C., and Kompella, R. (2012). The only constant is change: Incorporating time-varying network reservations in data centers. In *Proceedings of SIGCOMM '12*, pages 199–210, New York, NY, USA. ACM.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 20*

## *Redes Ópticas II*

# Novel Differentiated Service Methodology based on Constrained Allocation of Resources for Transparent WDM Backbone Networks

**Joana Sócrates Dantas[1,2], Regina Melo Silveira[1], Davide Careglio[2], José Roberto Amazonas[3], Josep Solé-Pareta[2], Wilson Vicente Ruggiero[1]**

[1]Laboratory of Computer Networks and Architecture (LARC)
Escola Politécnica da Universidade de São Paulo (EPUSP)
05508-900 – São Paulo – SP – Brazil

[2]Department of Computer Architecture (DAC)
Technical University of Catalonia (UPC)
08034 – Barcelona – Catalonia – Spain

[3]Laboratory of Communications and Signal (LCS)
Escola Politécnica da Universidade de São Paulo (EPUSP)
05508-900 – São Paulo – SP – Brazil

{joanadantas,regina,wilson}@larc.usp.br, jra@lcs.poli.usp.br,

{joana,careglio,pareta}@ac.upc.edu

***Abstract.*** *Current Internet traffic comprises diverse service requirements. Service providers differentiate connections characteristics according to demand's requirements and pricing policy. In this paper we propose a differentiated service methodology that implements constrained resources allocation according to demand's priority level for WDM networks. With our proposed mechanism higher priority connections are guaranteed a larger amount of resources, still, lower priority demands are guaranteed a minimum level of service.*

***Resumo.*** *O tráfego da Internet atual possui diversas variedades de requerimento por serviços. Provedores de serviço diferenciam as características das conexões de acordo com as requisições das demandas e a política de cobrança pelo serviço. Neste artigo propomos uma metodologia de diferenciação de serviço que implementa a alocação de recursos constringida de acordo com o nível de prioridade de serviço da demanda em redes WDM. Com o nosso mecanismo proposto conexões com prioridade mais alta são garantidas uma quantidade maior de recursos, mas, demandas de prioridade mais baixa são garantidas um nível mínimo de serviço.*

## 1. Introduction

Backbone networks offer long-haul data transport to their clients, Internet Service Providers (ISPs). This is a crucial task as a massive amount of data communication is currently provided and Internet communication rates demands are constantly increasing.

Wavelength Division Multiplexing (WDM) is the technology most commonly implemented by Backbone networks. In tandem with a control plane based on the Generalized Multi-Protocol Label Switching (GMPLS) architecture [Mannie 2004], WDM

enables the deployment of transparent networks. A transparent network can currently provide channels with capacity of up to 100 Gbps [Pedrola et al. 2012].

In a WDM network a channel corresponds to a wavelength and an end-to-end path and is known as a lightpath. When a grooming technique is implemented on a WDM network, two or more connections share a wavelength's resources. Grooming techniques are performed electronically and therefore, in a transparent network, traffic grooming is solely executed by the end nodes of a route and connections between a source and destination pair share resources from the same lightpath. A grooming technique may be implemented without following any specific policy or it can follow a given strategy that may regard Quality of Service, differentiated service and/or protection strategies.

When network's resources are not enough to serve all current clients' requests some demands will be blocked. When different level of services are established in the network, network managers may follow a policy to determine to which demands prioritize service. Minimum service level guarantees are specified by Service Level Agreements (SLA) signed between the data transport provider and its various clients. By implementing a differentiated service provisioning strategy data transport providers may obtain a cost effective service provisioning by avoiding SLAs penalties and optimizing network resources use [Song et al. 2007].

In order to offer different service levels, backbone networks implement differentiated service mechanisms. In WDM networks the most common service mechanisms assign a path based on its availability profile or providing different protection and/or restoration schemes according to a demand's priority of service.

Traffic pattern is usually unknown in advance, for this reason preemption procedures are necessary to avail resources to higher priority connection demands that may arrive during network operation. In a virtual circuit packet based network, as Multi Protocol Label Switched (MPLS), a traditional and commonly differentiated service approach is the preemption of lower priority connections to accommodate arriving higher priority traffic as described in [Le Faucheur et al. 2003] and [de Oliveira et al. 2007]. In a network where preemption is enabled, a higher priority traffic may use resources originally assigned to a lower priority traffic by disruption of its Label Switched Path (LSP).

Following SLA guidelines it is the network manager's responsibility to classify a network's traffic into priority levels. In MPLS networks the priority levels determine the priority of a demand to request preemption of an LSP. A backbone network connection transports a large volume of data and therefore should, ideally, not be disrupted. It is for this main reason that preemption strategies are not strictly adequate to backbone networks. Furthermore, preemption policies tend to cause instability, excessive number of re-routing decisions [de Oliveira et al. 2004] and a large number of Resource Reservation Protocol (RSVP) messages exchange each time an LSP is preempted [Salvadori and Battisti 2004]. All the aforementioned characteristics have a negative impact on large and complex networks as backbone networks.

Different preemption methodologies have been proposed with the objective of minimizing instability and message exchange overhead, proposals may be observed in Section 2. A differentiated distribution of resources approach is described for MPLS architecture in [Le Faucheur et al. 2003]. In DiffServ-TE MPLS LSPs are established with

a given bandwidth guarantee. Connections are organised into classes of traffic defined by a class-type (CT). The amount of bandwidth reserved varies according to a connection's CT value.

We propose a mechanism to implement differentiated resources distribution in WDM networks for long-haul data transport. With our proposed mechanism backbone network managers may design different profiles of service for the network's traffic, and distribute the network's resources aiming at financial optimization. We tested the proposed mechanism in an ad-hoc simulator and results indicate that by implementing our proposed mechanism higher priority connections have their service benefited, therefore optimizing network's revenue/cost ratio. In addition, lower priority connections are not disrupted and are guaranteed a minimum amount of allocated resources.

The rest of this paper is organized as follows. Section 2 presents related work on the topic. Section 3 describes the proposed mechanism. Section 4 explains the performance study and presents the results and their analysis. Finally, Section 5 concludes and introduces future works.

## 2. Related Work

Both GMPLS and MPLS networks are circuit switched networks where a connection's resources requirement is reserved before data transport begins. Preemption or differentiated distribution of resources policies may be implemented in MPLS as well as in GMPLS networks. In our work we extend a mechanism originally designed for MPLS networks to GMPLS networks. Therefore we review available mechanisms for MLPS networks as they may be adapted for GMPLS networks.

Authors in [Meyer and Vasseur 2010] propose a preemption scheme in which a connection selected to be preempted is re-routed and therefore not disrupted. In their proposal the ingress Label Edge Router (LER) under alert may trigger the necessary process to enable a temporary under-provisioning. Similarly, in [de Oliveira et al. 2004], an adaptive scheme is proposed for MPLS to select lower priority LSPs that can afford having their rate reduced in order to satisfy a higher priority connection requirement. The mechanism we propose is also based on under provisioning of lower priority connections, adapted for GMPLS WDM networks.

In [Sone et al. 2011] authors propose temporary under provisioning as a recovery scheme in Spectrum Switched Optical Networks (SSON). In case of a failure a low priority connection is switched to a backup path and its bandwidth provision is reduced. The proposal does not specify a model for resource reduction. In our proposal we study an specific resource constraint model.

In a WDM network most common differentiated service mechanisms are based on resilience strategy and/or link availability profile. A link's availability profile is determined by the amount of time the link works without failure. Both in [Song et al. 2007] and [Tornatore et al. 2012] authors propose a grooming approach in which connection requests are groomed into a lightpath according to its availability profile. In [El-Gorashi and Elmirghani 2009] authors propose a model with three resilience classes: shared, dedicated protection or restoration. Following this model traffic grooming is performed regarding demands' resilience requirements.

An alternative differentiated service approach was studied in [Dantas et al. 2012]. In this work authors focus on the Path Computation Element (PCE) architecture [Farrel et al. 2006] and analyse the impact of the priority in queue of a path computation request message (PCReq), described in [Vasseur and Roux 2009], on the resulting lower blocking probability.

To the best knowledge of the authors, there is no published work proposing differentiated distribution of resources nor there is any work concerning partial resources preemption for WDM networks. Our proposed mechanism offers an innovative differentiated service mechanism providing tools not currently available for WDM network managers.

## 3. Proposed Mechanism

In this paper, we propose a new mechanism for WDM networks based on differentiated distribution of resources developed MPLS networks in [Le Faucheur 2005b]. In the original MPLS mechanism a CT may be understood as a set of traffic trunks crossing a link and on which a specific set of bandwidth constraints is enforced [Tornatore et al. 2012].

When a differentiated distribution of resources mechanism is implemented in a network the resources assignment process must follow the constraints determined by a bandwidth constraint model. Three models were proposed for MPLS networks. In Maximum Allocation Model [Le Faucheur 2005a] (MAM) each CT has a maximum amount of resources that can be assigned to it as an upper limit. Maximum Allocation with Reservation (MAR) model is similar to MAM but the upper limit of bandwidth can be disregarded if the link being considered is not above a certain level of usage. In the Russian Dolls Model (RDM) [Le Faucheur 2005c] each class shares its maximum amount of bandwidth assignment with higher CTs. The lowest CT value's maximum bandwidth assignment equals the link's capacity and resources that are not being used by a higher CT may be used by lower CT connections.

Each resources constraint model has its benefits and disadvantages depending on networks management needs and network's characteristics. Still, there are performance analysis in the literature that point to advantages on implementing RDM. The RDM mechanism presented better performance regarding low priority connections, [Goldberg et al. 2006] and higher levels of connections' QoS [Molnar and Vlcek 2009]. RDM was proven to best adapt to the Diff-Serv bandwidth allocation policy by achieving high link utilization and guaranteeing quality of service to the various Diff-Serv classes [Lai 2005]. For the reasons aforementioned we have selected the RDM as the model used by our proposed mechanism.

In our proposed mechanism the resources constrained distribution idea is extended to a WDM network. In the WDM network scenario, we consider in our study, we assume path computation and resource assignment would be performed centrally by a PCE as illustrated by Figure 1. In the figure it is possible to observe a PCE exchanging messages with GMPLS nodes that work as Path Computation Clients (PCC). PCCs require path and resource computation to the PCE for a given demand (or demands) at hand. After a PCE's successful reply message it is up to the GMPLS architecture protocols to reserve and assign lightpaths to connections. In order to be able to implement the RDM algorithm a domain's PCE would have to be enhanced.

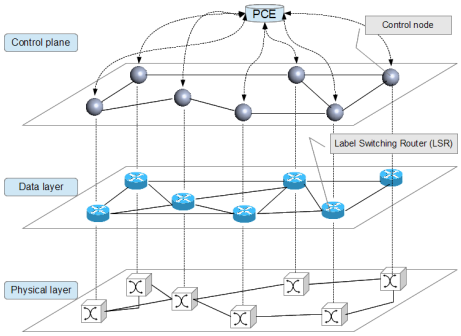



**Figure 1. A WDM network with path computation centralized by PCE with communication, via PCE protocol, with control nodes**

In the Russian Dolls Model, a link's maximum capacity use is achieved by accumulating successive CTs and bandwidth constraints (BC). For example CT7 may use resources up to BC7 limit, while CT6 shares its BC6 constraint with CT7, CT5 shares its BC5 constraint with CT6 and CT7 and so on as illustrated by Figure 2. Therefore, a lower priority traffic may use a higher priority class's bandwidth reservation, in case it is not in use, up to its own bandwidth constraint value, while a higher priority connection can preempt a lower priority connection up to its own bandwidth constraint value [Le Faucheur 2005c].

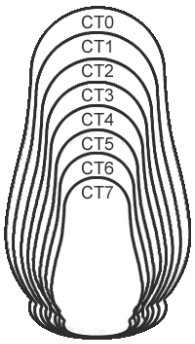



**Figure 2. RDM Russian Dolls Model**

In our proposed mechanism resources preemption is performed on a first fit basis and, to prevent excessive connection disruption and resource re-assignment, when a higher priority connection is torn down its resources are not re-assigned to lower priority connections.

The bandwidth constraints for each class must be determined by network management taking into account SLA details. The BC values must be carefully determined as they define the level of privilege over resources that higher priority connections will have. For instance, in a situation where one channel is assigned connections with all possible CT values and the difference between each BC value is equal for most CT values, different CT connections could end up having the same amount of resources assigned. However, if the difference between BC values increases as the CT value increases higher priority connections would be guaranteed more resources than lower priority ones.

The RDM is a resource distribution model. Throughout this paper, however, we will refer as RDM mechanism as the mechanism that implements this resource distribution model when performing the RWA algorithm.

### 3.1. Algorithm Description

In our proposal, the RDM mechanism is triggered when contention for resources occur, which will be detected in the network when a connection request is blocked by the grooming mechanism. When triggered, the RDM algorithm implements a resource distribution constraint model so when a demand arrives the lightpath computation process will limit the bandwidth requirement to its respective BC value. We also tested a variation of the mechanism where the assignment of resources follows no constraint. In this mechanism version the BC values are used to determined the constrained amount of resources that may be preempted from the various CT value connections. We will call this version of the mechanism RDM Aggressive herein.

In the algorithm, path computation is performed offline as alternate fixed routes. Having established the amount of resources to be assigned, the algorithm selects the first route from 3 alternate fixed possible routes. Then, in first fit order, it searches for a lightpath containing a connection using the same route as the one selected, so that grooming may be performed. If such a lightpath is found, the algorithm checks if the lightpath's available resources are enough to accommodate the demand at hand. In case it does, the demand is assigned as a new connection to the lightpath.

If there are not enough resources available, the lightpath's id is recorded in case the RDM mechanism needs to be triggered later. If, however, a lighpath containing a connection using the selected route is not found, the algorithm searches, in first fit order, an empty lightpath in the selected route. If the process is not successful the tasks described above are consecutively performed for 2nd best and then 3rd best routes.

If a connection cannot be established after the previous steps, but a lightpath containing a connection using the same route was found, then the RDM mechanism is triggered. If the version adopted is not RDM Aggressive, at this moment, the bandwidth constraint is enforced on demand's BW requirement. By accessing the information outputted from the grooming process the RDM algorithm already knows the lightpath supports data to be groomed into it. In the selected lightpath, it searches for connections with the lowest possible priority value and calculates the amount of resources that can be preempted from the selected connection. In case the resulting re-usable resource is not enough the procedure is repeated for all connections with the same lower CT value. This process is repeated for the consecutive CT values as long as their CT value is not equal or higher than the demands and until the demand's BW requirement is met by the sum of all preemptable resources.

If after checking all the lightpath's connections the resulting total amount of resources that can be preempted is not enough to satisfy the demand's request then the demand is blocked. Conversely, if enough resources are found the lightpath is selected by the PCE as the resulting computed path. For the demand to be assigned as a connection to the lightpath, the suggested lightpath and possible reduced suggested resources must be accepted, in which case, the channel's current connections are updated.

**Table 1. BW requested values distribution**

| BW requested value | 10 | 40 | 100 |
|---|---|---|---|
| Probability | 0.80 | 0.15 | 0.05 |

**Table 2. CT values distribution**

| CT value | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Probability | 0.60 | 0.25 | 0.10 | 0.05 |

## 4. Performance Study and Results Analysis

We developed a Java based discrete event driven simulator in order to evaluate the performance of our differentiated resources distribution algorithm. We performed a simulation using the German Nobel reference network topology with 17 nodes and 26 bidirectional links [Orlowski et al. 2010]. We consider that each link in the network corresponds to one fibre comprising wavelengths with 100 Gbps bandwidth capacity each [Pedrola et al. 2012].

Connection requests are generated dynamically following a uniform distribution among node pairs for source and destination. All runs are averaged over $5x10^5$ requests following a Poisson process with mean holding time ($\lambda$) set to 500 sec (exponential). Request inter-arrival time ($\mu$) are also exponential and vary with loading. Offered traffic load is calculated in Erlang using the relation E=$\lambda\mu$ [Ghimire and Mohan 2012]. Bandwidth requests are non-uniformly distributed among 10, 40 and 100 Gbps, with distribution approximate to values from [Pedrola et al. 2012] that can be observed in Table 1.

Previously, we tested the algorithm using different numbers of CT values. We observed that 4 CT values represent a good trade-off between complexity and overall performance. Therefore we decided to determine 4 possible CT values for the demands in the simulation, where 0 is the lowest priority value and 3 is the highest priority value. The 4 CT values follow a non-uniform distribution that can be observed in Table 2.

We evaluated the algorithm for different BC models but, due to space limitations, we selected to present results for the model that presented a higher level of benefit. The selected model's values are expressed in Table 3. BC values refer to the maximum amount of resources allowed to be allocated to a demand, minimum BW guaranteed is the amount of resources that will not be preempted from an active connection with a given CT value and Max. robbed is the maximum amount of resources a demand with a given CT value

**Table 3. BC Model values**

| CT value | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| BC values (in Gbps) | 100 | 97 | 79 | 47 |
| Minimum BW guaranteed (in Gbps) | 3 | 18 | 32 | 47 |
| Max. robbed (in Gbps) | 0 | 3 | 21 | 53 |

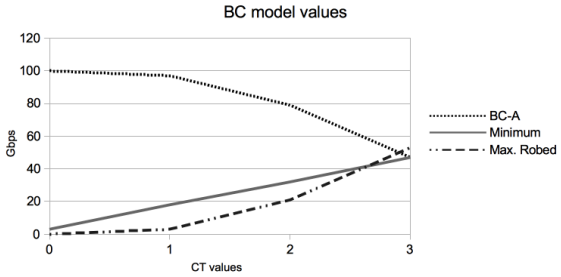



**Figure 3. Constraint values for BC model A**

may use from resources preempted from connections with lower CT values. Figure 3 illustrates the relation between the BC values and each of the CT values.

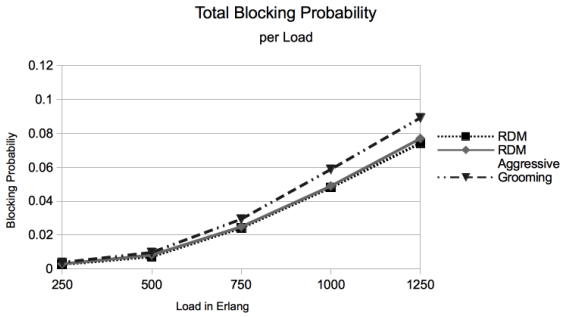



**Figure 4. Blocking probability for RDMs and grooming scenarios for various network offered loads**

Performance metrics analysed in the simulations were blocking probability and amount of resources allocated to connections. Simulation results reached a 98% confidence interval level. We compared the performance of the proposed mechanism's two versions, RDM and RDM Aggressive, with the resulting performance of a network scenario where a grooming mechanism is implemented without connection assignment policy.

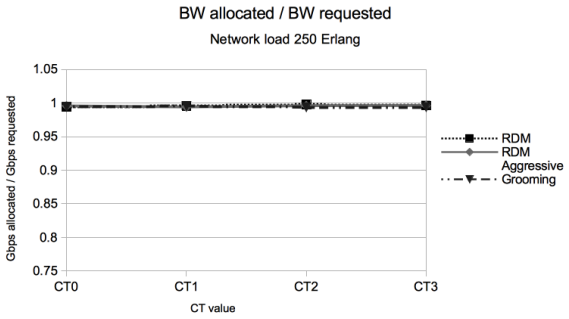



**Figure 5. Blocking probability by CT values for RDMs and grooming scenarios for a network offered load of 250 Erlang**

Simulation results indicate that as network offered load increases the RDM mechanism is triggered more often and the blocking probability tends to be increasingly smaller than the blocking probability of the grooming scenario as can be observed in Figure 4. In

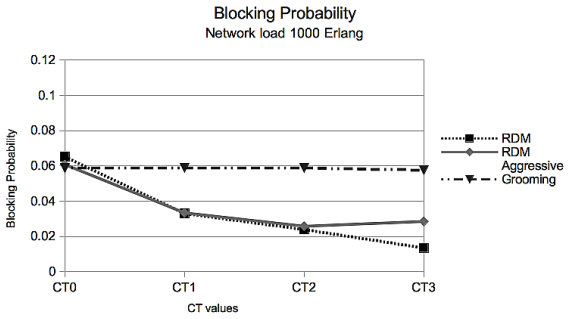



**Figure 6. Blocking probability by CT values for RDMs and grooming scenarios for a network offered load of 1000 Erlang**

our experiment, when the network offered load is 250 Erlang the blocking probability for all the three scenarios analysed may be considered to be the same, that is justified by the fact that under light offered load contention for resources do not occur often and the RDM mechanism is thus seldom triggered. In a network scenario with a heavy load the difference in blocking probability can be of 1% for a network offered load of 1000 Erlang and up to 2% for a network load of 1250 Erlang as can be observed in Figure 4. It is also possible to observe that, when the total number of demands is being considered, both RDM mechanism versions implemented provide approximately the same decrease in blocking probability when compared to the grooming scenario.

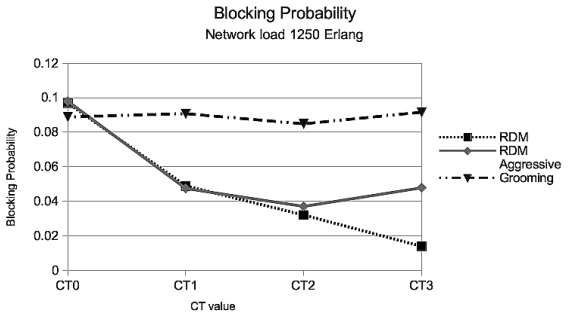



**Figure 7. Blocking probability by CT values for RDMs and grooming scenarios for a network offered load of 1250 Erlang**

When the blocking probability is analysed according to the demands' different CT values, in the scenario where the network offered load was 250 Erlang (Figure 5) the blocking probability is very low for all demands with different CT values, and therefore difference in the blocking probability values is unnoticeable. However, when the network offered load is 1000 Erlang or 1250 Erlang the blocking probability decreases as the connections' CT values increase. As can be observed in Figure 6 and in Figure 7 the largest beneficial impact on blocking probability performance occurred for connections with the highest priority, the ones belonging to CT value 3. In the RMD scenario the blocking probability for demands belonging to CT value 3 decreased almost 5% if compared to the grooming scenario for the scenario with an offered load of 1000 Erlang (Figure 6) and a decrease of almost 8% for an offered load of 1250 Erlang (Figure 7). In the RDM Aggressive scenario, however, the level of blocking probability decrease was less pronounced, being around 3% decrease for 1000 Erlang (Figure 6) and less than 5% decrease for the

1250 Erlang (Figure 7) offered load scenarios.

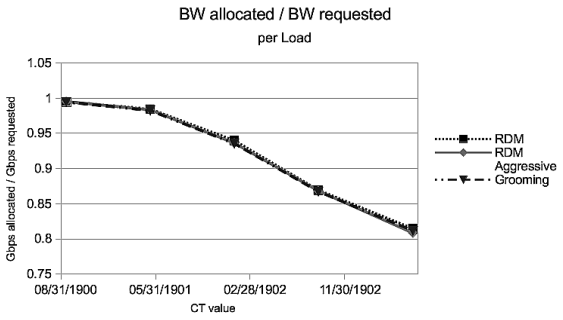



**Figure 8. Proportional resources allocation for RDMs and grooming scenarios according to different network offered load**

This result shows that the RDM Aggressive mechanism version does not offer best results for highest CT value connections. This phenomenon occurs because, in the RDM Aggressive version, each CT3 connection is allocated a larger amount of resources, and therefore, is responsible for a larger use of channels resources than in the RDM scenario. CT3 demands cannot preempt resources from other CT3 connections, therefore tend to compete between each other. In a network where a larger amount of resources are being used by CT3 connections, CT3 demands will have a higher blocking probability as they will have less resources to preempt for their own use. CT2 and lower CT values connections were also granted their original BW requirement, however due to the fact their allocated resources may be preempted by a higher CT value demand the resulting amount of resources allocated to these connections are limited. The fact that lower CT value connections have a limit on their resources allocation represents less competition for resources between connections of the same lower CT value.

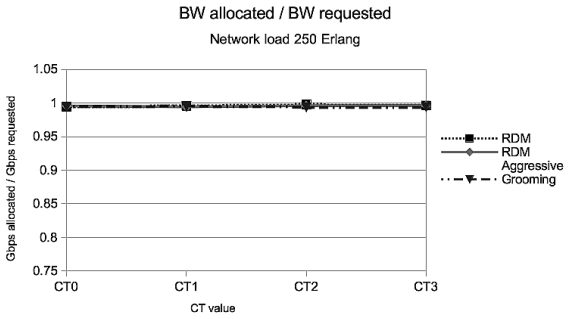



**Figure 9. Proportional resources allocation for RDMs and grooming scenarios for different CT value connections with a network offered load of 250 Erlang.**

We have also analysed the proportion of resources allocated to connections in relation to the amount of resources requested by demands. When all the connections were considered together it was possible to observe that implementing any of the two versions of the RDM mechanism the amount of proportional resources allocation did not increase in any of the network offered load scenarios as illustrated by Figure 8. In order to avoid repetition of terms, in the rest of this paper, we refer to proportional resources allocation simply as resources allocation.

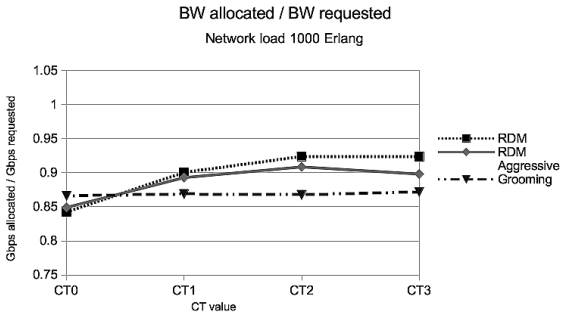



**Figure 10. Proportional resources allocation for RDMs and grooming scenarios for different CT value connections with a network offered load of 1000 Erlang.**

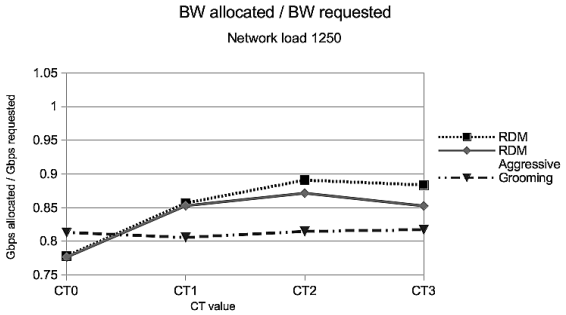



**Figure 11. Proportional resources allocation for RDMs and grooming scenarios for different CT value connections with a network offered load of 1250 Erlang.**

When the allocation of resources is analysed according to connections' CT values, the difference in the BW allocation between the different scenarios is only noticeable for network offered load of 500 Erlang or higher. Indeed with a network offered load of 250 Erlang it is possible to affirm there is no difference in resources allocation and distribution between the different scenarios as can be observed in Figure 9.

For higher network offered loads, however, the RDM mechanism tends to allocate more resources to higher CT value connections than the RDM Aggressive mechanism. For a scenario where the network offered load is 1000 Erlang, as illustrated by Figure 10, the resources allocated to connections with CT values 3 and 2 have an increase of approximately 5% while for the same network offered load the RDM Aggressive mechanism resulted in approximately 3% increase for connections with CT values 3 and 2. For a scenario where the network offered load is of 1250 Erlang, as illustrated by Figure 11, by implementing the RDM mechanism the proportional resources allocated to connections with CT3 and CT2 increase in approximately 7% while for the same network offered load the RDM Aggressive mechanism resulted in approximately 4% increase for connections with CT values 3 and 2.

This phenomenon can be explained by the fact that in RDM Aggressive mechanism version less connections of CT3 are being established, but at the same time they require a larger amount of resources than in RDM mechanism scenario. These larger amount of resources required from higher CT value demands will be preempted from the lower CT value connections, therefore CT2 connections also present a decrease in the resources allocated when compared to the RDM mechanism scenario.

As for the lowest CT value connections both versions of the RDM mechanism decrease the resources allocated for those connections when compared to the grooming scenario. This fact explains why the resulting resources allocated to all connections do not differ between different values of offered load: for all scenarios analysed the total amount of resources allocated is virtually the same, but the allocation is distributed differently to the different CT value connections.

Blocking probability and resources allocation results have a direct impact on a network's revenue. The service fee of higher priority connections is usually higher than the fee of lower priority connections. A pricing policy may be based on the number of connections established or the amount of resources allocated to a connection. In both cases it is possible to infer that by increasing service level to higher priority connections increases the amount of collected service fee. As an example, [Xia et al. 2009] presents service fee values for four different classes of service. We calculate the proportional service fee values according to a service fee value of reference, in this case the value of service fee for CT1, which results in the following proportional fee according to CT value : CT0 0.65; CT1 1.00; CT2 1.50; CT3 2.00. From these values it is possible to observe that offering service to a connection with CT3 generates twice as much collected fee as offering service to a connection with CT1.

In this section we demonstrated that the proposed RDM mechanisms perform differently from the grooming mechanism in terms of blocking probability and resources distribution according to connections' CT values. As the network's offered load increases the RDM mechanism is triggered more often. As a higher number of connections are established via the RDM mechanism higher priority connections benefit from a smaller blocking probability and larger amount of allocated resources. Since higher priority connections tend to have stricter SLA requirements, and are usually charged more by SPs, increasing the service for higher priority connections will eventually increase a SP's revenue.

## 5. Conclusion and Future Work

We proposed a novel mechanism for differentiation of service for a WDM network based on the constrained distribution of resources. The mechanism implements a resources distribution constrained model called the Russian Dolls Model (RDM). The RDM determines resources distribution and allocations constraints according to different levels and priorities of service. We tested two versions of the mechanism in an event driven simulation and analysed their resulting performance in comparison to a scenario where a grooming technique is implemented without a resource distribution policy.

From simulation results we observed that our proposed mechanism increases the distribution of resources to higher priority connections. Furthermore, the proposed mechanism decreases overall blocking probability for heavy load traffic scenarios. We believe that, the implementation of the mechanism, in a network, promotes better service to higher priority connections resulting on a net revenue increase. This economic benefit is justified by the fact that a SP's both service fees and penalty fees are higher for higher priority connections.

We are aware that in order for the mechanism to be properly implemented on a network control plane enhancements are required. Specifically, extensions on PCE pro-

tocol and GMPLS' RSVP-TE are necessary to guarantee the mechanism's deployment. However this subject is out of the scope of this article and will be addressed by future works.

As future works we will extend the study of differentiated service mechanism proposals for Elastic Optical Networks. And we test the proposed mechanisms in different topologies.

## 6. References

## References

Dantas, J., Careglio, D., Silveira, R., Ruggiero, W., and Sole-Pareta, J. (2012). PCE QoS tools and related scalability in WDM networks. In *Transparent Optical Networks (ICTON), 2012 14th International Conference on*, pages 1–7. IEEE.

de Oliveira, J., Vasseur, J., Chen, L., and Scoglio, C. (2007). Label switched path (LSP) preemption policies for MPLS traffic engineering. Technical report, IETF RFC 4655, August.

de Oliveira, J. C., Scoglio, C., Akyildiz, I. F., and Uhl, G. (2004). New preemption policies for DiffServ-aware traffic engineering to minimize rerouting in MPLS networks. *IEEE/ACM Transactions on Networking (TON)*, 12(4):733–745.

El-Gorashi, T. E. and Elmirghani, J. M. (2009). Differentiated resilience with dynamic traffic grooming for WDM mesh networks. In *Transparent Optical Networks, 2009. ICTON'09. 11th International Conference on*, pages 1–6. IEEE.

Farrel, A., Vasseur, J.-P., and Ash, J. (2006). A path computation element (PCE)-based architecture. Technical report, IETF RFC 4655, August.

Ghimire, R. and Mohan, S. (2012). A token-based routing mechanism for GMPLS-controlled WDM networks. *Optical Switching and Networking*, 9(2):170–178.

Goldberg, J. B., Dasgupta, S., and de Oliveira, J. C. (2006). Bandwidth Constraint Models: A Performance Study with Preemption on Link Failures. In *GLOBECOM*.

Lai, W. S. (2005). Bandwidth constraints models for differentiated services (DiffServ)-aware MPLS traffic engineering: performance evaluation. Technical report, IETF RFC 4128, June.

Le Faucheur, F. (2005a). Maximum allocation bandwidth constraints model for DiffServ-aware MPLS traffic engineering. Technical report, IETF RFC 4125, June.

Le Faucheur, F. (2005b). Protocol extensions for support of Diffserv-aware MPLS traffic engineering. Technical report, IETF RFC 4124, June.

Le Faucheur, F. (2005c). Russian dolls bandwidth constraints model for DiffServ-aware MPLS traffic engineering. Technical report, IETF RFC 4127, June.

Le Faucheur, F., Lai, W., et al. (2003). Requirements for support of differentiated services-aware MPLS traffic engineering. Technical report, IETF RFC 3564, July.

Mannie, E. (2004). Generalized multi-protocol label switching (GMPLS) architecture. Technical report, IETF RFC 3945, October.

Meyer, M. and Vasseur, J. (2010). MPLS traffic engineering soft preemption. Technical report, IETF RFC 5712, January.

Molnar, K. and Vlcek, M. (2009). Evaluation of bandwidth constraint models for MPLS networks. *J Electronic*, pages 1313–1842.

Orlowski, S., Wessäly, R., Pióro, M., and Tomaszewski, A. (2010). SNDlib 1.0 Survivable network design library. *Networks*, 55(3):276–286.

Pedrola, O., Castro, A., Velasco, L., Ruiz, M., Fernández-Palacios, J., and Careglio, D. (2012). CAPEX Study for a multilayer IP/MPLS-over-flexgrid optical network. *Journal of Optical Communications and Networking*, 4(8):639–650.

Salvadori, E. and Battisti, R. (2004). Quality of service in IP over WDM: considering both service differentiation and transmission quality. In *Communications, 2004 IEEE International Conference on*, volume 3, pages 1836–1840. IEEE.

Sone, Y., Watanabe, A., Imajuku, W., Tsukishima, Y., Kozicki, B., Takara, H., and Jinno, M. (2011). Bandwidth squeezed restoration in spectrum-sliced elastic optical path networks (SLICE). *Journal of Optical Communications and Networking*, 3(3):223–233.

Song, L., Zhang, J., and Mukherjee, B. (2007). Dynamic provisioning with availability guarantee for differentiated services in survivable mesh networks. *Selected Areas in Communications, IEEE Journal on*, 25(3):35–43.

Tornatore, M., Lucerna, D., Mukherjee, B., and Pattavina, A. (2012). Multilayer protection with availability guarantees in optical WDM networks. *Journal of Network and Systems Management*, 20(1):34–55.

Vasseur, J. and Roux, J. (2009). Path computation element (PCE) communication protocol (PCEP). Technical report, IETF RFC 5440, March.

Xia, M., Tornatore, M., Martel, C. U., and Mukherjee, B. (2009). Service-centric provisioning in WDM backbone networks for the future internet. *Journal of Lightwave Technology*, 27(12):1856–1865.

# Um Novo Algoritmo para Alocação de Rota e Comprimento de Onda com Restrições de Energia e da Camada Física em Redes Ópticas

**Pedro H. T. M. Nogueira[1,2], Victor A. P. Oliveira[2], Iguatemi E. Fonseca[1]**

[1]Programa de Pós-Graduação em Informática – Centro de Informática – Universidade Federal da Paraíba (UFPB). CEP: 58.051-900 – João Pessoa – PB – Brasil

[2] Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB) – Campi Patos e Sousas. PB – Brasil

pedro.nogueira@hotmail.com.br, victorcactusti@gmail.com, iguatemi@ci.ufpb.br

***Abstract.*** *This paper presents a novel Impairment and Energy-Aware Routing and Wavelength Assignment algorithm (IEA-RWA) that operates in a dynamic transparent optical network. The IEA-RWA algorithm is compared with other ones, such as: i) an Impairment-Aware Routing and Wavelength Assignment algorithm (IA-RWA); ii) an Energy-Aware Routing and Wavelength Assignment algorithm (EA-RWA); iii) a traditional RWA algorithm that do not take into account physical impairments and energy consumption. Numerical simulations suggest that the proposed IEA-RWA presents better performance, using different types of metrics, when compared with other algorithms.*

***Resumo.*** *Este artigo propõe um novo algoritmo para Alocação de Rota e Comprimento de Onda com Restrições de Energia e da Camada Física (IEA-RWA – Impairment and Energy-Aware Routing and Wavelength Assignment) em redes ópticas transparentes dinâmicas. O IEA-RWA é comparado com um algoritmo RWA que leva em conta restrições da camada física (IA-RWA), com um algoritmo RWA que leva em conta consumo de energia (EA-RWA) e com um algoritmo RWA tradicional/clássico que não leva em conta no momento de admissão de conexões restrições da camada física e nem consumo de energia. As simulações numéricas mostram que o IEA-RWA apresenta melhor desempenho em vários aspectos quando comparado com seus competidores.*

## 1. Introdução

A literatura tem mostrado que o consumo de energia elétrica nas redes de comunicação tem crescido devido ao rápido incremento do tráfego nessas redes [Tucker 2011a], [Tucker 2011b]. Em 2009, as redes de comunicação e a sua infraestrutura de telecomunicações associada foram responsáveis por cerca de 8% do total da energia elétrica consumida no mundo [Heddeghem 2009]. Considerando o aumento de tráfego esperado para os próximos anos e caso novas tecnologias atualmente praticadas não sofram melhorias e aperfeiçoamentos, em 2020 a fatia no consumo global de energia elétrica pelas redes de comunicação poderá atingir os 20% [Pickavet 2008]. Nesse sentido, o projeto tanto de dispositivos com baixo consumo de energia quanto de técnicas e algoritmos que diminuem o consumo em redes de comunicação tem ganhado atenção nos últimos anos [Tucker 2011a], [Tucker 2011b]. Esse tipo de rede tem sido

chamada pela comunidade acadêmica de Redes Energeticamente Eficientes (*Green Networks*). Recentemente, o desenvolvimento de redes ópticas transparentes (TON – *Transparent Optical Network*) deu importante contribuição para a redução do consumo de energia elétrica nas redes núcleo, pois a característica principal das TON's é que o sinal permanece no domínio óptico durante toda a transmissão e, portanto, conversões óptico-eletro-óptico (OEO) são evitadas [Ramaswami 2010].

Apesar desse avanço, um estudo recente mostrou que o consumo de energia nas redes ópticas ainda pode ser melhorado. Analisou-se o caso dos três domínios das redes ópticas: (i) Acesso; (ii) Metropolitanas; (iii) TON's (*backbone*). Em (i), redes de acesso, atualmente menos que 15% da energia consumida é efetivamente utilizada; já no caso (ii), redes metropolitanas, menos que 30% da energia consumida é efetivamente utilizada. No caso (iii), TON's, menos que 50% da energia consumida é efetivamente utilizada [Mukherjee 2011], [Farias 2012]. Percebe-se, portanto, que embora as redes TON's sejam mais eficientes na transmissão da informação do ponto de vista de consumo energético, ainda há uma grande margem para a diminuição no consumo de energia. Além disso, a ausência de conversão OEO possui desvantagens, pois a qualidade do sinal óptico (QoT – *Quality of Transmission*) durante a transmissão é prejudicada pelos efeitos da camada física (*physical layer impairments*) [Rahbar 2012]. Nessa linha, recentemente, alguns trabalhos têm proposto técnicas e algoritmos para minimização do consumo de energia em TON's através do uso de algoritmos de Alocação de Rota e de Comprimento de Onda com restrições de energia (EA-RWA – *Energy-Aware Routing and Wavelength Assignment*) [Bianco 2013], [Coiro 2011], [Drakulic 2013], [Leiva 2011], [Manousakis 2013], [Ricciardi 2012]. Em geral, essas técnicas tentam empacotar o máximo possível os caminhos ópticos em uma mesma rota ou em uma mesma região da rede de forma que partes da rede possam ser coladas em estados com baixo consumo de energia, como por exemplo, desligadas ou em estado de dormência (*stand-by*). Como consequência, os EA-RWA's projetados dessa forma podem priorizar: (a) o estabelecimento de rotas longas na rede; (b) alta utilização de enlaces/rotas mais ativos, ou seja, o número médio de comprimentos de onda usados nessas rotas é maior quando a rede usa um algoritmo EA-RWA do que quando usa um RWA tradicional. Por um lado, os algoritmos EA-RWA's apresentam ganhos quanto ao consumo de energia na rede. Porém, por outro lado, os fatores (a) e (b) levam ao baixo desempenho desses algoritmos EA-RWA do ponto de vista da qualidade do sinal óptico (QoT – *Quality of Transmission*). Rotas longas na rede torna o sinal mais exposto aos efeitos da atenuação da fibra, ao acúmulo do ruído de emissão espontânea (ASE – *Amplified Spontaneous Emission*) dos amplificadores ópticos a fibra dopada com érbium (EDFA – *Erbium-doped Fiber Amplifiers*) e aos efeitos de dispersão na fibra óptica. A concentração de vários caminhos ópticos em um mesmo enlace/rota torna esses sinais mais propícios às degradações causadas por efeitos não-lineares da fibra óptica [Fonseca 2005].

Uma evolução natural seria então desenvolver/projetar algoritmos que pudessem oferecer uma redução no consumo de energia da rede e também garantir níveis aceitáveis da QoT na rede. Nessa linha, a literatura tem proposto os algoritmos para Alocação de Rota e Comprimento de Onda com Restrições de Energia e da Camada Física (IEA-RWA – *Impairment and Energy-Aware Routing and Wavelength Assignment*) [Cavdar 2012]. Ainda há pouquíssimos trabalhos com propostas de algoritmos IEA-RWA's, a maioria dos artigos da literatura são relativos a algoritmos

EA-RWA's e, em grande parte dos casos, para cenários de rede com tráfego estático. Em [Cavdar 2012], é proposto um algoritmo IEA-RWA para um cenário de rede no qual a demanda de tráfego é conhecida previamente, ou seja, tráfego estático ou *off-line*. O IEA-RWA é modelado como um problema de programação linear inteira mista (MILP - *Mixed Integer Linear Programming*), no qual tanto o consumo de energia e quanto a QoT são otimizados em conjunto no projeto da rede. Os autores mostram que a estratégia proposta apresenta níveis de consumo de energia próximo a um algoritmo EA-RWA ao mesmo tempo em que garante a QoT em valores próximos de um algoritmo IA-RWA.

Diferentemente dos trabalhos encontrados na literatura, que propõe estratégias para redes ópticas estáticas, esse artigo propõe um algoritmo IEA-RWA em redes ópticas transparentes dinâmicas. O IEA-RWA foi desenvolvido de forma que leva em consideração restrições da camada física, como atenuação da fibra, acúmulo de ruído ASE e o efeito de saturação dos amplificadores ópticos EDFA's. Para o cálculo do consumo de energia na rede foi utilizado o modelo apresentado em [Tucker 2011a], o qual considera o consumo de energia por bit transmitido na rede levando em consideração todos os dispositivos ativos na rede, tais como, amplificadores ópticos, transmissores e receptores. O IEA-RWA proposto é comparado com um algoritmo IA-RWA, um algoritmo EA-RWA e com um algoritmo RWA tradicional/clássico que não leva em conta no momento de admissão de conexões restrições da camada física e nem consumo de energia. As simulações numéricas mostram que o IEA-RWA apresenta melhor desempenho em vários aspectos quando comparado com os outros algoritmos tanto do ponto de vista de QoT quanto da economia de energia na rede, além de apresentar probabilidade de bloqueio similar aos outros algoritmos. Para o melhor de nosso conhecimento, essa é a primeira vez na literatura que um algoritmo IEA-RWA é analisado/proposto em uma rede óptica transparente dinâmica.

O restante desse artigo é organizado da seguinte forma. A Seção 2 apresenta o modelo de consumo de energia utilizado nos algoritmos IEA-RWA e EA-RWA. A Seção 3 mostra a definição dos algoritmos RWA tradicional, EA-RWA, IA-RWA e IEA-RWA com suas respectivas métricas de desempenho. Na Seção 4 são debatidos os resultados das simulações numéricas nos diversos cenários de rede investigados. E finalmente, a Seção 5 traz a conclusão do artigo e as considerações sobre trabalhos futuros.

## 2. Modelo de consumo de energia em um sistema óptico WDM

O modelo de consumo de energia utilizado nesse trabalho é baseado no modelo apresentado em [Tucker 2011a]. Para quantificar o consumo energético, utiliza-se a métrica energia por bit de dado transmitido em um caminho óptico ou simplesmente energia por bit, a qual tem sido considerada como uma importante métrica para medida de consumo de energia em sistemas de transmissão digital [Tucker 2009]. Como será visto, o diferencial dessa métrica é que o seu cálculo depende de grandezas qu**e estão diretamente relacionadas** com parâmetros importantes para os algoritmos IA-RWA, EA-RWA e, portanto, IEA-RWA, como por exemplo, comprimento da rota na rede óptica, potência dos canais ópticos, relação sinal-ruído óptica (OSNR – *Optical Signal Noise Rate*), número de amplificadores ópticos na rota sob análise, dentre outros.

A Figura 1 mostra a visão geral do sistema de transmissão óptica com multiplexagem por divisão em comprimento de onda (WDM – *Wavelength Division Multiplex*) considerado para o cálculo da energia por bit [Tucker 2011a]. Esse sistema consiste de uma sucessão de *m spans* (ou seja, a distância entre um amplificador e outro) com comprimento $L_{span}$ e atenuação da fibra óptica $\alpha$ dB/km, de forma que o comprimento total é $L=m*L_{span}$. Nessa figura, $G$ é o ganho dado por cada amplificador, $P_A$ é potência elétrica consumida por cada amplificador, $P_{TX}$ é a potência elétrica consumida pelo transmissor óptico e $P_{RX}$ é a potência elétrica consumida pelo receptor óptico. Considera-se também que cada transmissor, amplificador e receptor óptico suportam até $k$ comprimentos de onda ou caminhos ópticos (*lightpaths*), que um filtro óptico com largura de banda $B_0$ é colocado após cada amplificador e que a potência média na saída do transmissor óptico Tx é $P$.

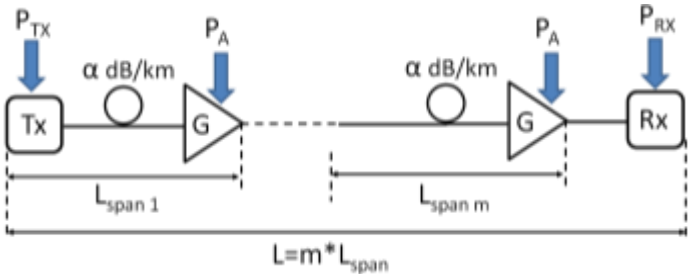



**Figura 1: Sistema WDM ponto a ponto transparente com comprimento L composto por m estágios de comprimento *Lspan*.**

É possível determinar que, para cada comprimento de onda no enlace da Figura 1, a OSNR na saída o $m$-ésimo estágio de amplificação é dado por [Desurvire 2002]

$$OSNR = \frac{P}{2n_{sp}m(e^{\alpha L_{span}} - 1)h\nu B_0}, \quad (1)$$

em que, $n_{sp}$ é o fator de emissão espontânea de cada amplificador, α é a atenuação da rede em $km^{-1}$, $h$ é a constante de Plank e ν é a frequência do sinal óptico ou caminho óptico em análise. Em termos de SNR por bit, i.e. $SNR_{bit}$, tem-se

$$SNR_{bit} = 2\tau_{bit}B_0 OSNR, \quad (2)$$

em que, $\tau_{bit}$ é o tempo ou período do bit, ou inverso da taxa de transmissão do caminho óptico em bit/s. Portanto, a energia por bit do sinal óptico na saída do $m$-ésimo amplificador é dada por [Tucker 2011a]

$$E_l = P\tau_{bit} = SNR_{bit}n_{sp}m(e^{\alpha L_{span}} - 1)h\nu. \quad (3)$$

Considerando que a potência elétrica total consumida pelo sistema da Figura 1 é $P_{tot} = mP_A + P_{TX} + P_{RX}$, obtem-se que a energia total consumida por bit por comprimento de onda é [Tucker 2011a]

$$E_{bit} = E_{AMP} + E_{TX} + E_{RX}, \quad (4)$$

em que,

$$E_{AMP} = \frac{mP_A\tau_{bit}}{k} = \frac{SNR_{bit}n_{sp}m^2(e^{\alpha L_{span}} - 1)(1 - e^{-\alpha L_{span}})h\nu}{\eta_{EP}} \quad (5)$$

é a energia total por bit por comprimento de onda no m-ésimo EDFA e $E_{TX}$ e $E_{RX}$ são as energias consumidas no transmissor e receptor ópticos, respectivamente, e $\eta_{EP}$ é a eficiência de conversão de potência elétrica do amplificador óptico [Desurvire 2002].

Nos algoritmos RWA utilizados e propostos nesse artigo, a equação (5) é usada para calcular a energia consumida por bit transmitido em um caminho óptico. Como foi mencionado anteriormente, perceba que essa equação depende de termos que são importantes para o cálculo das rotas pelos algoritmos RWA, como por exemplo, o OSNR do caminho óptico, o número de amplificadores e o comprimento da rota, a taxa de transmissão utilizada na rede óptica, além da potência elétrica consumida por dispositivos como EDFA's. Em geral, a maioria dos trabalhos da literatura utilizam apenas aspectos relacionados à potência elétrica consumida pelos dispositivos [Bianco 2013], [Coiro 2011], [Drakulic 2013], [Leiva 2011], [Manousakis 2013], [Ricciardi 2012]. Como comentário final, note que não é considerada a existência de conversão OEO rede óptica, já que esse trabalho tem foco em uma TON dinâmica, por isso os termos $E_{TX}$ e $E_{RX}$ não são considerados, pois são constantes para todas as requisições de conexão que chegam em uma TON com tráfego dinâmico.

## 3. Algoritmos RWA

### 3.1. RWA Tradicional

Para estabelecer um caminho óptico é necessário alocar e rotear um comprimento de onda para cada enlace da rota definida. Este problema é conhecido como o problema de Alocação de Rota e de Comprimento de Onda (RWA – *Routing and Wavelength Assignment*) [Ramaswami 2010]. Na Figura 2a é apresentado o fluxograma simplificado com a estrutura do *RWA Tradicional*, que recebe esse nome em virtude de não considerar efeitos da camada física ou de restrições de energia no momento da alocação da rota e do comprimento de onda. Nesse trabalho são utilizados o *Dijkstra* para encontrar a rota com menor distância e a heurística *First Fit* ao buscar por um comprimento de onda.

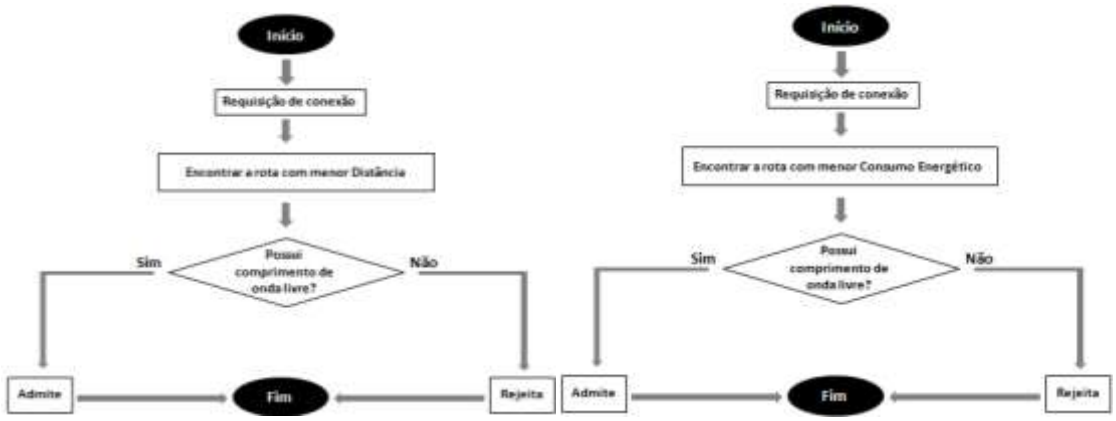
### 3.2. Algoritmo EA-RWA

Na Figura 2b é apresentado o fluxograma simplificado com a estrutura do *EA-RWA*. Percebe-se que o algoritmo mantém estrutura similar ao *RWA Tradicional*, porém, em vez de considerar a distância como custo do enlace é utilizado a energia por *bit* do sinal óptico. Ou seja, enquanto o *RWA Tradicional* faz a escolha da rota de acordo com a menor distância em quilômetros, o *EA-RWA* faz a escolha da rota baseada no menor consumo energético em J/bit. Para isso, a equação (5) é usada para o cálculo do consumo de energia.

### 3.3. Algoritmo IA-RWA

A Figura 3 apresenta o fluxograma do *IA-RWA*. O algoritmo *IA-RWA* consiste em resolver o problema *RWA*, tendo em conta as restrições de qualidade de serviço, como por exemplo, a QoT [Fonseca 2005]. É considerado mais sofisticado que o *RWA*, pois leva em conta as deficiências da camada física, bem como a disponibilidade de comprimento de onda [Fonseca 2005].

Como pode ser observado, primeiramente é gerada uma requisição de conexão. Depois é encontrada a rota com menor distância utilizando o algoritmo de *Dijkstra* e,




(a) (b)

**Figura 2: Fluxograma do algoritmo: (a) RWA Tradicional. (b) EA-RWA.**

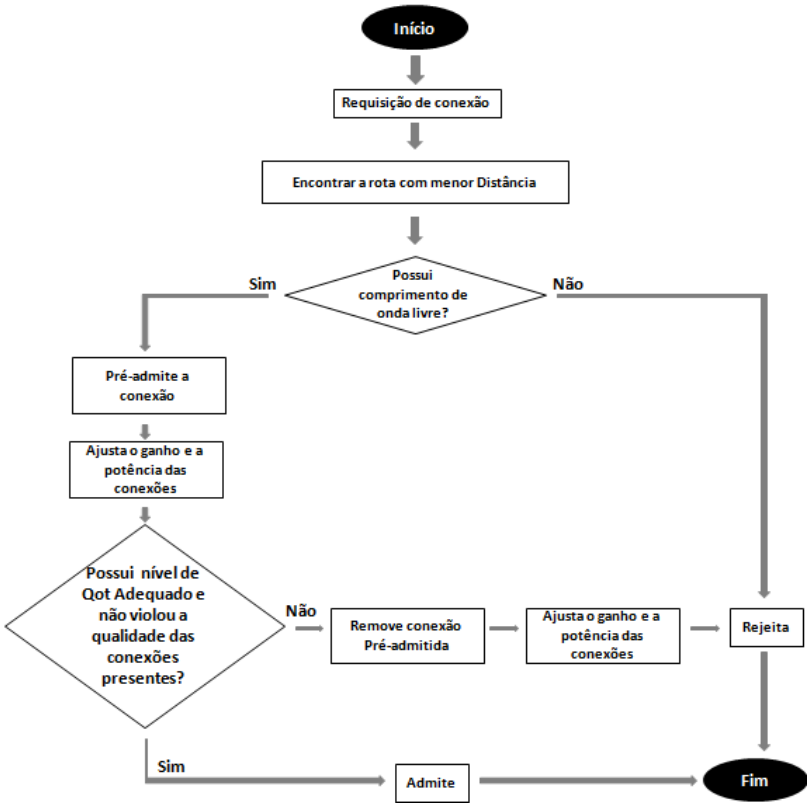



**Figura 3: Fluxograma do algoritmo IA-RWA.**

em seguida, o algoritmo busca por um comprimento de onda disponível através da heurística *First Fit*. Nesse ponto, se não existir comprimento de onda livre, a conexão é imediatamente rejeitada. Em caso contrário, a conexão é pré-admitida apenas para ser computada a QoT e em seguida é verificado se tal conexão não interferiu de forma a degradar as conexões já estabelecidas na rede. Em caso positivo, a conexão deve ser retirada e rejeitada. Senão, a conexão é finalmente admitida. Observe também que, como mostrado no fluxograma, são ajustados o ganho e a potência das conexões logo que a conexão é pré-admitida (e caso ela seja rejeitada). O modelo para o cálculo da OSNR e dos ajustes de ganho e potência podem ser encontrados em [Oliveira 2012a, 2012b].

## 3.4. Algoritmo IEA-RWA

Na Figura 4 é apresentado o fluxograma do *IEA-RWA*. Note que o algoritmo busca unir funcionalidades do *EA-RWA* e *IA-RWA*, ou seja, o algoritmo busca rotas energeticamente mais econômicas e leva em conta as limitações da camada física.

Como será visto na seção de resultados, o uso da rota com menor distância pelos algoritmos *Tradicional* e *IA-RWA* nem sempre é a rota mais econômica energeticamente. O *IEA-RWA* até permite que rotas longas sejam selecionadas, mas o algoritmo procura penalizar tais rotas e busca um compromisso entre a energia por bit transmitido, a ocupação dos enlaces e o tamanho da rota. Outra característica do algoritmo é que, enquanto o *EA-RWA* faz o roteamento pela menor energia por bit do sinal óptico, o *IEA-RWA* dar preferência por rotas com maior consumo de energia por bit do sinal óptico. Isso faz com que o algoritmo economize energia utilizando enlaces que já estão sendo utilizados por outras conexões da rede.

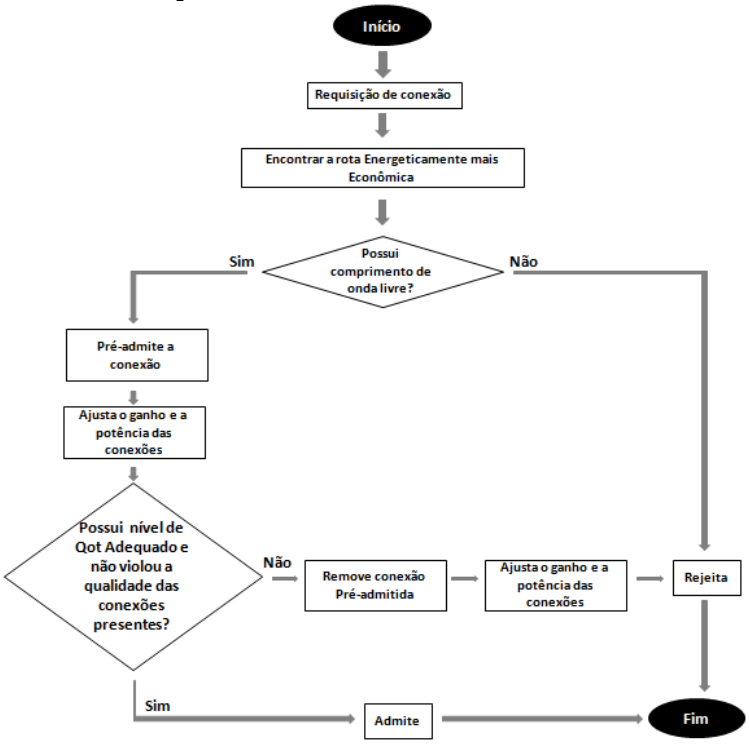



**Figura 4: Fluxograma do algoritmo *IEA-RWA*.**

A equação proposta nesse trabalho que relaciona o custo do enlace (LC) considera parâmetros como a distância, porcentagem de canais alocados, estágios de amplificação, energia por bit do sinal óptico e é expressa da seguinte forma

$$LC = m\ x\ floor\left(\frac{l}{E_{AMP}} + \frac{w}{W} + \beta_i^{\beta_i}\right). \quad (6)$$

Em que: $l$ – representa a distância do enlace; $E_{AMP}$ –calculado de acordo com a equação (5); $w$ – representa a quantidade de comprimentos de onda alocados no enlace; $W$ – representa a quantidade total de comprimentos de onda da rede; $\beta_i$ – representa um parâmetro em função de $m$ e do número de saltos ($i$) do caminho que está sendo

escolhido. A função *floor* retorna o número inteiro imediatamente superior ao número entre parênteses.

Note que o termo $\frac{1}{E_{AMP}}$ tem relação com a capacidade de transmissão por potência consumida. Fazendo uma análise da dimensão desse termo, tem-se que

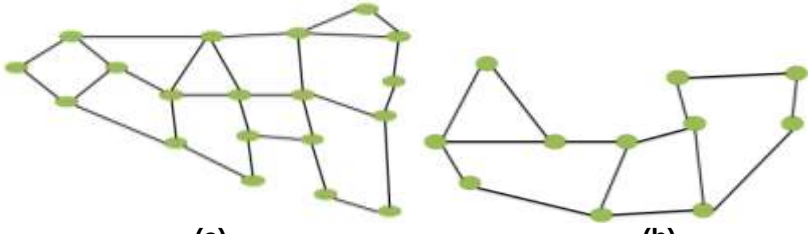
$$\left[\frac{l}{E_{AMP}}\right] = \frac{km}{J/bit} = \frac{km}{w} \cdot \frac{bit}{s} = \left[\frac{capacidade\ de\ transmissão}{potência\ consumida}\right]$$

Ou seja, é a relação entre capacidade de transmissão e potência consumida por bit transmitido. Esse termo busca priorizar rotas que tenham baixa capacidade de transmissão por potência consumida. A ideia é usar a estratégia conhecida na literatura como *just enough* [Zulkifli 2008], que aloca a rota/comprimento de onda que seja capaz de atender de maneira suficiente (*just enough*) as necessidades da requisição, deixando recurso disponíveis para demandas futuras; e não alocar a melhor rota/comprimento de onda. O segundo termo da função custo busca fazer um balanceamento do tráfego na rede, enquanto o último termo penaliza as rotas mais longas de forma que a equidade (*fairness*) da rede seja preservada.

## 4. Resultados

### 4.1. Cenário de simulação

Através de um ambiente de simulação implementado em C/C++ pelos próprios autores, foi simulado um cenário de uma TON dinâmica, na qual foram geradas 100000 requisições de conexões. Tais conexões possuem um padrão de tráfego uniforme entre os nós da rede e seguem uma distribuição poissoniana tendo duração com distribuição exponencial (média = 1s). Foram simuladas duas topologias de rede, uma com 19 nós adaptada da topologia da rede *NSFNet-2* e outra com 11 nós adaptada da rede *Abilene* (ambas bastante usadas na literatura) e estão mostradas nas Figuras 5a e 5b. Todos os enlaces são bidirecionais e têm comprimentos variando entre 80 e 320 km de uma fibra óptica monomodo com atenuação de 0,2 dB/km. O comprimento de um *span*, ou seja, a distância entre dois amplificadores ópticos é $L_{span}$ = 80 km. É utilizado um conjunto de W=16 e W=24 comprimentos de onda em uma TON sem conversão de comprimento de onda operando a uma taxa de transmissão de 10 Gbit/s. Os algoritmos *IA-RWA* e *IEA-RWA* utilizam $BER_{TH}$=$10^{-12}$ como critério de QoT para admitir ou rejeitar requisições de conexões na rede. A partir dos parâmetros apresentados na Tabela 1, a relação entre a quantidade de conexões rejeitadas e o número total de pedidos de conexões que chegam na rede óptica é a Probabilidade de Bloqueio da rede.



**(a)** **(b)**

**Figura 5: Redes simuladas: (a) NSFNet-2 com 19 nós, (b) Abilene com 11 nós.**

**Tabela 1: Parâmetros da simulação.**

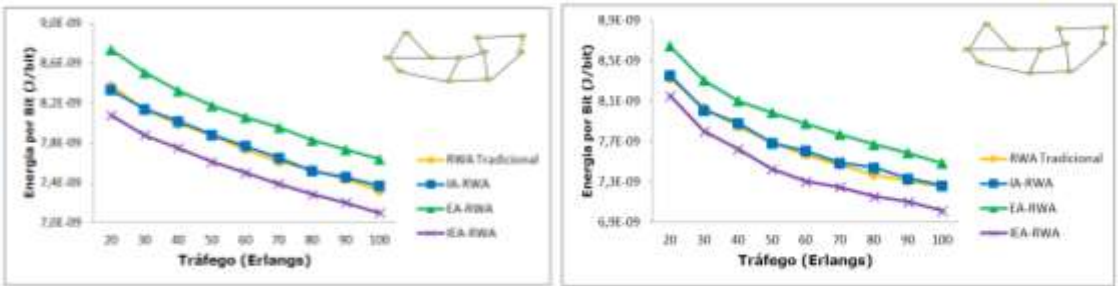
| Parâmetro | Valor |
|---|---|
| Potência dos canais ópticos (P) | 0 dBm |
| Ganho máximo do amplificador ($G_{max}$) | 16 dB |
| Potência de saturação ($P_{sat}$) | 10 dBm |
| Fator de emissão espontânea do amplificador ($n_{sp}$) | 4 |
| Eficiência de conversão de potência elétrica do amplificador óptico [Desurvire 2002] ($\eta_{EP}$) | 1% |
| Largura de banda do filtro ótico | 50 GHz |
| Largura de banda do filtro elétrico | Taxa de Bits x 0,8 |

### 4.2. Resultados das simulações numéricas

Os resultados são apresentados para as redes *Abilene* e *NSFNet-2*, em diversos cenários de simulação e avaliando o desempenho dos algoritmos *IA-RWA*, *EA-RWA* e *IEA-RWA*. O algoritmo *RWA Tradicional* é usado na comparação também, pois bloqueia conexões somente por continuidade, não bloqueando por ausência de QoT.

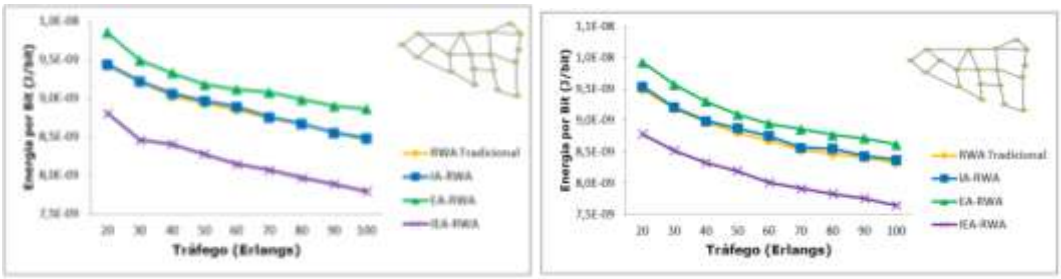
#### 4.2.1 Impacto no consumo de energia da rede

A Figura 6a mostra um comparativo do consumo energético médio utilizando 16 comprimentos de onda dos algoritmos *RWA Tradicional*, *IA-RWA*, *EA-RWA* e o *IEA-RWA* quando a rede *Abilene* opera a uma taxa de transmissão de 10 *Gbit/s*. O eixo y corresponde à energia consumida por bit e o eixo x a intensidade do tráfego. Para um tráfego entre 20 e 100 *Erlangs*, o *RWA Tradicional* e o *IA-RWA* apresentaram praticamente o mesmo consumo de energia, o algoritmo mais econômico foi o *IEA-RWA* enquanto o *EA-RWA* foi o que gastou mais energia. Os resultados para 24 comprimentos de onda são similares como pode ser visualizado na Figura 6b. Os resultados são também similares para a rede *NSFNet-2*, como mostrado na Figura 7.



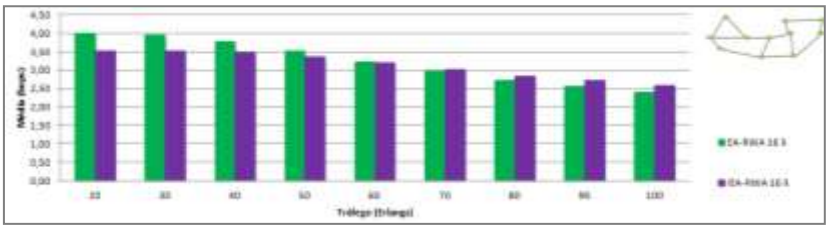

**(a)** **(b)**

**Figura 6: Consumo energético médio das conexões com: (a) 16, (b) 24 comprimentos de onda. Rede *Abilene*.**

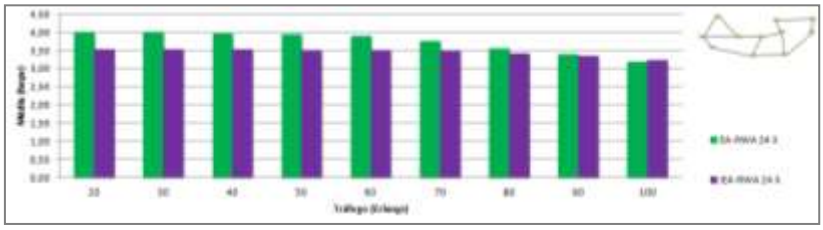



**(a)** **(b)**

**Figura 7: Consumo energético das conexões: (a) 16, (b) 24 comprimentos de onda.**



**(a)**



**(b)**

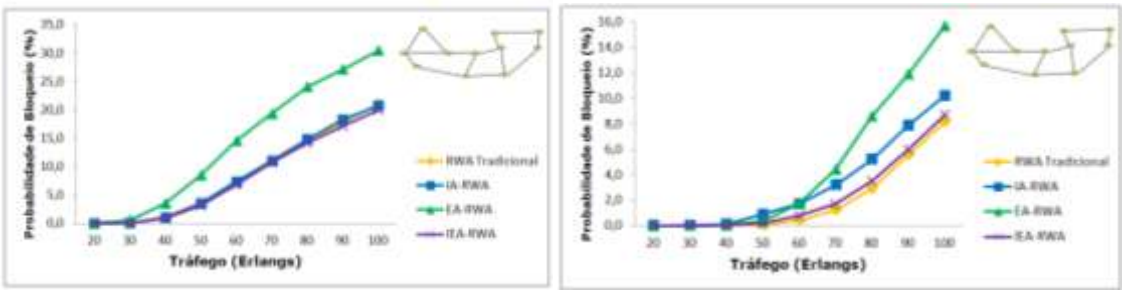
**Figura 8: Tamanho médio das conexões com (a) 16, (b) 24 comprimentos de onda.**

Um dos motivos que levou o *EA-RWA* a apresentar esse desempenho é que o algoritmo tende a fazer o roteamento usando rotas longas, ou seja, a quantidade de recursos que serão alocados para cada conexão será maior. Já o *IEA-RWA* tende a fazer o roteamento usando rotas mais curtas, ou seja, a quantidade de recursos que serão alocados para cada conexão será menor. Como o *EA-RWA* usa a energia por bit como custo do enlace, o que por sua vez depende da $OSNR_{bit}$ conforme equação (5), a tendência é a escolher rotas que estão livres de tráfego pois apresentam melhores valores de $OSNR_{bit}$. A medida que o tráfego da rede cresce esse comportamento é minimizado, já que a maioria dos enlaces estarão transmitindo tráfego. A função custo utilizada pelo algoritmo *IEA-RWA* tenta corrigir essa patologia, pois como discutido na Seção 3.4, possui termos que buscam o equilíbrio entre o número de recursos, o comprimento e a ocupação da rota, além de penalizar rotas mais longas. Isso pode ser verificado na Figura 8, que apresenta o tamanho médio das conexões admitidas do *EA-RWA* e *IEA-RWA* com 16 e 24 lambdas para a rede *Abilene*. Para tráfego baixo, a média das conexões admitidas pelo *EA-RWA* na *Abilene* é em torno de 4,00 (para 16 lambdas) e 3,98 (para 24 lambdas), enquanto que para tráfego alto essa média cai para 2,39 (para 16 lambdas) e 3,18 (para 24 lambdas). No caso do *IEA-RWA*, para baixos tráfegos, a média fica em torno de 3,52 (para 16 lambdas) e 3,53 (para 24 lambdas), enquanto que para altos tráfegos a média fica próxima de 2,59 (para 16 lambdas) e 3,23 (para 24 lambdas). Os resultados são similares para a rede *NSFNet-2* e foram omitidos por economia de espaço no artigo. Assim, com média de rotas menores, sobram mais

recursos e, em consequência, podem-se admitir mais conexões, o que tem impacto positivo também na probabilidade de bloqueio como será visto na Subseção 4.2.2.

### 4.2.2 Impacto na probabilidade de bloqueio da rede

A Figura 9 mostra uma comparação da probabilidade de bloqueio da rede utilizando 16 e 24 comprimentos de onda dos algoritmos *RWA Tradicional*, *IA-RWA*, *EA-RWA* e o *IEA-RWA* na rede *Abilene*. Com 16 comprimentos de onda observa-se que o desempenho do *EA-RWA* é inferior em relação aos outros algoritmos, porém, para 24 comprimentos de onda apresentou um bom desempenho para tráfegos até 60 *Erlangs*. Já o bloqueio do *IEA-RWA* utilizando 16 comprimentos de onda (em qualquer tráfego da rede) foi sempre aproximadamente menor ou igual ao bloqueio do *RWA Tradicional* e do *IA-RWA*. Para 24 lambdas o *IEA-RWA* apresentou um desempenho melhor em relação ao *EA-RWA* e ao *IA-RWA*. Também pode ser visto que, como esperado, quando se aumentou a quantidade de comprimentos de onda de 16 para 24 ouve uma redução na probabilidade de bloqueio das conexões. Para a rede *NSFNet-2* os resultados são similares e omitidos por questões de espaço do artigo.




**(a)** **(b)**

**Figura 9: Comparação da probabilidade de bloqueio total com: (a) 16, (b) 24 comprimentos de onda. Rede *Abilene*.**

Para tentar entender a diferença no desempenho apresentada pelos algoritmos, foi investigada também a probabilidade de bloqueio por continuidade e por QoT. Uma requisição será bloqueada por continuidade se não existir um comprimento de onda disponível na rota por onde tal requisição seria roteada. Analisando a Figura 10, que mostra a probabilidade de bloqueio somente por continuidade de comprimento de onda do IA-RWA e do IEA-RWA, percebe-se que o bloqueio causado pelo IEA-RWA é ligeiramente menor ou igual para 16 comprimentos de onda e praticamente o mesmo para 24 comprimentos de onda. Também pode ser observado que há uma redução na probabilidade de bloqueio das conexões quando se aumenta a quantidade de comprimentos de onda, a um tráfego de 100 *Erlangs*, por exemplo, a probabilidade de bloqueio cai de 20% para 6%. A Figura 11 mostra um comparativo do bloqueio da QoT apresentado pelos algoritmos do IEA-RWA e IA-RWA utilizando 16 e 24 lambdas na rede *Abilene*.

Observa-se que o algoritmo IEA-RWA também apresenta melhor desempenho quanto ao bloqueio por QoT quando comparado ao IA-RWA. Os resultados para a rede *NSFNet-2* são mostrados nas Figuras 12 e 13. Note que o algoritmo IEA-RWA apresenta melhor desempenho em praticamente todos os cenários, exceto para a rede *NSFNet-2* com 24 comprimentos de onda. O algoritmo IA-RWA tende sempre a alocar a melhor rota com melhor OSNR. Todavia, alguns trabalhos da literatura têm alertado que talvez essa estratégia não seja a mais adequada [Zulkifli 2008], em vez disso, é

melhor alocar uma rota que seja capaz de atender de maneira suficiente (*just enough*) aos requisitos de QoT e bloqueio por continuidade. O algoritmo IEA-RWA utiliza para o cálculo da rota uma função custo que depende de vários parâmetros, alguns com relação direta com os efeitos de atenuação, acúmulo de ASE e saturação dos amplificadores, como por exemplo, o número de amplificadores, comprimento e tamanho da rota. Portanto, busca encontrar uma rota que seja um compromisso entre o consumo de energia e QoT. Com isso, apresenta, na maioria dos cenários de rede investigados, melhor desempenho tanto quanto ao bloqueio por continuidade quando a preservação da qualidade de transmissão das conexões na TON e economia de energia. A exceção foi o cenário de rede mostrado na Figura 13, i.e. rede *NSFNet-2*, o bloqueio por QoT foi menor para o algoritmo IA-RWA. Nesse caso, a maior conectividade da rede *NSFNet-2* pode ter favorecido e dado mais opções de rotas com melhor QoT para o IA-RWA.

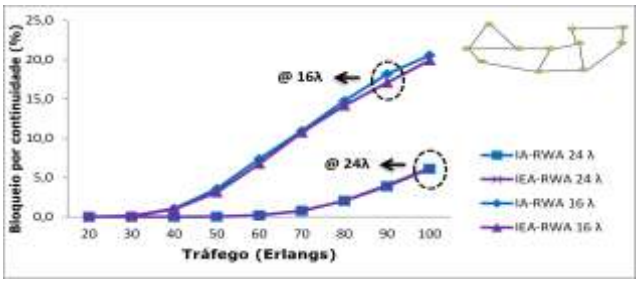



**Figura 10: Bloqueio por continuidade. Rede *Abilene* com W=16 e 24.**

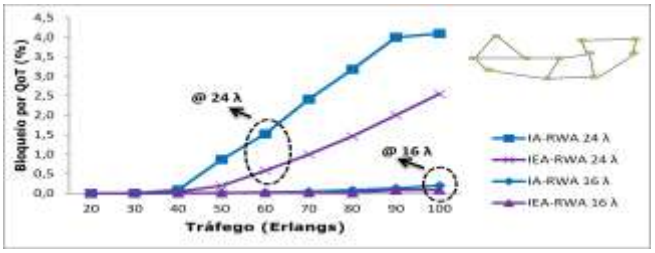



**Figura 11: Bloqueio por QoT. Rede *Abilene* com W=16 e 24.**

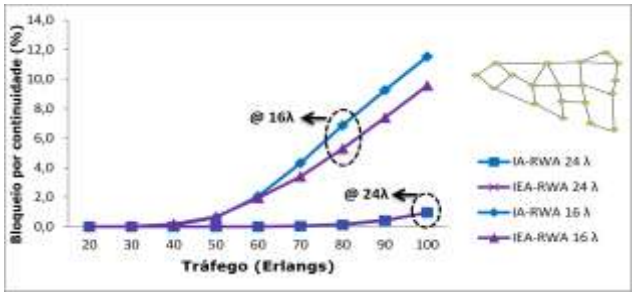



**Figura 12: Bloqueio por continuidade. Rede *NSFNet* com W=16 e 24.**

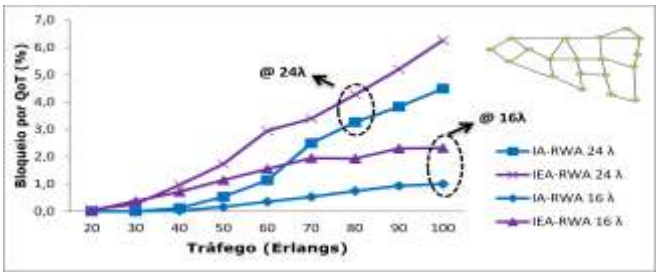



**Figura 13: Bloqueio por QoT. Rede *NSFNet* W=16 e 24.**

## 5. Conclusão

Este trabalho apresentou um estudo sobre algoritmos RWA com restrições de energia e da camada física em TON dinâmicas. Foi proposto um algoritmo IEA-RWA que apresentou bom desempenho tanto em aspectos relacionados ao consumo de energia quanto na preservação da qualidade do sinal das conexões da rede óptica. Os resultados das simulações numéricas sugeriram que utilizar um modelo de energia que, diferentemente da maioria dos modelos usados nos algoritmos RWA encontrados na literatura, é baseado em parâmetros relacionados com o algoritmo RWA, pode trazer maiores benefícios quanto ao consumo de energia elétrica pela rede. Ao mesmo tempo, o uso de uma função custo que relacione aspectos do consumo energético, com parâmetros ligados a efeitos da camada física e grandezas relacionadas com o tamanho da rota na rede, pode resultar em um algoritmo IEA-RWA com desempenho superior quando comparado com outros algoritmos. O uso de outras estratégias de roteamento, bem como a incorporação de outros efeitos da camada física ao IEA-RWA, estão atualmente em análise.

### Agradecimentos

Os autores agradecem à CAPES e ao CNPq por financiar parcialmente esse trabalho.

### Referências

Bianco, A. ; Bonetto, E. ; Ahmad, A. (2013) “Energy awareness in the design of optical core networks” In: IEEE/OSA Optical Fiber Communication Conference (OFC 2013), p. 1 – 3.

Cavdar, C.; Ruiz, M.; Monti, P.; Velasco, L.; and Wosinska, L. (2012) “Design of Green Optical Networks With Signal Quality Guarantee” In: IEEE International Conference on Communications (ICC 2012), p. 3025 – 3030.

Coiro, A.; Listanti, M.; and Valenti, A. (2011) “Dynamic Power-Aware Routing and Wavelength Assignment for Green WDM Optical Networks” In: IEEE International Conference on Communications (ICC 2011), p. 1 – 6.

Desurvire, E.; Bayart, D.; Desthieux, B.; and Bigo, S. (2002) “Erbuim-Doped Fiber Amplifiers: Device and System Developments”, ISBN-10: 0471419036, Wiley.

Drakulic, S., and Skorin-Kapov, N. (2013) “Green routing and wavelength assignment in optical networks” In: 15th IEEE International Conference on Transparent Optical Networks (ICTON 2013), p. 1 – 4.

Farias, J. E. (2012) “Crescimento do tráfego IP e consumo de energia em redes ópticas de transporte” In: 15º Simpósio Brasileiro de Micro-ondas e Optoeletrônica e o 10º CBMag – Congresso Brasileiro de Eletromagnetismo (MOMAG 2012), p. 1 – 6.

Fonseca, I. E. (2005) “Uma abordagem para aprovisionamento e diferenciação de QoS óptico na presença de FWM em redes ópticas transparentes”, Tese de Doutorado em Engenharia Elétrica, Universidade Estadual de Campinas (UNICAMP).

Heddeghem, W. V. (2009) “Energy in ICT - Trends and Research Directions”, In: IEEE 3rd International Symposium on Advanced Networks and Telecommunication Systems (IEEE ANTS 2009), p. 1-3.

Leiva, A.; Finochietto, J.M.; Huiszoon, B.; López, V.; Tarifeño, M., Aracil, J.; and Beghelli, A. (2011) "Comparison in Power Consumption of Static and Dynamic WDM Networks" Elsevier Optical Switching and Networking, Vol. 8, No. 3, p. 149 – 161.

Manousakis, K.; Angeletou, A.; and Varvarigos, E., (2013) "Energy efficient RWA strategies for WDM optical networks", IEEE/OSA Journal of Optical Communications and Networking, Vol. 5, No. 4, p. 338 – 348.

Mukherjee, B. (2011) "Energy Savings in Telecom Networks", Tutorial SBRC 2011, Campo Grande, MS, 01/06/2011.

Oliveira, V. A., Fonseca, I. E. (2012a) "EDFA Gain Variation Problem in Transparent Optical Networks", XXX Simpósio Brasileiro de Telecomunicações (SBrT 2012).

Oliveira, V. A. (2012b) "Algoritmos e um Sistema Distribuído para Simulação de Redes Óticas com Variação do Ganho dos Amplificadores", Dissertação de Mestrado UFERSA/UERN.

Pickavet, M.; Vereecken, W.; Demeyer, S.; Audenaert, P.; Colle, D.; Demeester, P.; Dhoedt B. (2008) "Energy footprint of ICT: future outlook and challenges", In: 35th International ICT Symposium, p. slides presentation.

Rahbar, A. G. (2012) "Review of Dynamic Impairment-Aware Routing and Wavelength Assignment Techniques in All-Optical Wavelength-Routed Networks" IEEE Communications Surveys & Tutorials, Vol. 14, No. 4, p. 1065 – 1089.

Ramaswami, R.; Sivarajan, K. N.; Sasaki, G. H. (2010) "Optical Networks A Practical Perspective", ISBN-10: 0123740924, Morgan Kaufmann, 3nd Edition.

Ricciardi, S.; Palmieri, F.; Fiore, U.; Careglio, D.; Santos-Boada, G.; and Solé-Pareta, J. (2012) "An energy-aware dynamic RWA framework for next-generation wavelength-routed networks" Elsevier Computer Networks, Vol. 56, No. 10, p. 2420 – 2442.

Tucker, R. S.; Parthiban, R.; Baliga, J.; Hinton, K.; Ayre, R.W.A.; Sorin, W.V. (2009) "Evolution of WDM Optical IP Networks: A Cost and Energy Perspective", IEEE/OSA Journal of Lightwave Technology, Vol. 27, No. 3 p. 243 – 252.

Tucker, R. S. (2011a) "Green Optical Communications—Part I: Energy Limitations in Transport", IEEE Journal of Selected Topics in Quantum Electronics, Vol. 17, No. 2, p. 245 – 260.

Tucker, R. S. (2011b) "Green Optical Communications – Part II: Energy Limitations in Networks". IEEE Journal of Selected Topics in Quantum Electronics, Vol. 17, No. 2, p. 261 – 274.

Zulkifli, N.; Almeida, R.C., Jr.; Guild, K.M. (2008) ""Just-enough" resource allocation for heterogeneous 10/40-gb/s services in dispersion-limited transparent optical networks", In: 5th IEEE International Conference on Broadband Communications, Networks and Systems (BROADNETS 2008), p. 640 – 646.

# Uma Nova Abordagem de Roteamento Multirrestritivo para Redes Ópticas Translúcidas

**Gilvan Durães[1], José Suruagy Monteiro[2], William Giozza[3]**

[1]Departamento de Ciência da Computação (DCC) - Universidade Federal da Bahia (UFBA) – Salvador – BA – Brasil

[2]Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE) – Recife – PE – Brasil

[3]Departamento de Engenharia Elétrica (ENE) – Universidade de Brasília (UnB) – Brasília – DF – Brasil

gilvanmd@dcc.ufba.br, suruagy@cin.ufpe.br, giozza@unb.br

***Abstract.*** *This paper presents a new approach for multi-restrictive routing in translucent optical networks which is simultaneously aware of optical link, wavelength, and optical node constraints. Based on this novel approach, two new multi-restrictive routing algorithms were proposed, named Offline Multi-Restrictive Routing (Off-MRR) and Multi-Restrictive Routing (MRR). These two algorithms were compared in terms of execution times, network utilizations, and blocking probabilities. In all evaluated scenarios, Off-MRR achieved a superior performance in terms of execution times, while MRR achieved a superior performance in terms of network utilizations and blocking probabilities.*

***Resumo.*** *Este trabalho apresenta uma nova abordagem de roteamento multirrestritivo para as redes ópticas translúcidas, ciente tanto das restrições relacionadas ao enlace óptico, como das restrições relacionadas ao comprimento de onda e ao nó óptico ao mesmo tempo. Considerando esta abordagem são propostos dois novos algoritmos de roteamento multirrestritivo, chamados Offline Multi-Restrictive Routing (Off-MRR)* e *Multi-Restrictive Routing (MRR). Os algoritmos propostos são comparados entre si, em termos de probabilidade de bloqueio, utilização da rede e tempo de execução. O Off-MRR apresentou um melhor desempenho em termos de tempo de execução, enquanto que o MRR alcançou um melhor desempenho em termos de probabilidade de bloqueio, em todos cenários avaliados.*

## 1. Introdução

As redes ópticas tiveram um grande avanço com o surgimento e amadurecimento da tecnologia de multiplexação por divisão de comprimento de onda (WDM – *Wavelength Division Multiplexing*), a qual permite dividir a banda passante de uma fibra óptica em diversos canais ópticos de comunicação formados por diferentes comprimentos de onda [1]. Desta forma, as redes ópticas WDM se tornaram o principal veículo dos *backbones* dos atuais sistemas de comunicações.

Uma nova arquitetura de rede óptica WDM que utiliza conversor *Óptico-Eletro-Óptico* (OEO) em alguns nós intermediários da rede e comutador puramente óptico em

outros nós, se torna cada vez mais uma realidade. Essa arquitetura de rede óptica que tem como objetivo agregar a agilidade de uma rede óptica transparente à qualidade do sinal óptico garantido com a conversão OEO é conhecida como rede óptica translúcida [2-3].

Para o estabelecimento de um circuito óptico (*lightpath*) entre um nó origem e um nó destino de uma rede óptica, é necessário definir uma rota e alocar um comprimento de onda em cada enlace dessa rota. O problema de roteamento e alocação de comprimento de onda, visando atender um maior número de solicitações dinâmicas de *lightpath* é conhecido como *Routing and Wavelength Assignment* (RWA) ou RWA dinâmico [1-7].

Em [7] é apresentado um dos primeiros estudos de análise de desempenho dos algoritmos RWA em redes ópticas modeladas com degradações na camada física. Além de avaliar a disponibilidade de rotas e comprimentos de onda, neste trabalho [7] é evidenciada a necessidade dos algoritmos de RWA considerarem as degradações de camada física para a escolha de recursos apropriados para o estabelecimento de circuitos ópticos.

Para incluir esta limitação não tratada nos algoritmos RWA pioneiros foram concebidos os algoritmos da classe *Physical Layer Impairments constrained and aware RWA* (PLI-RWA) [8]. Os algoritmos PLI-RWA são algoritmos de roteamento e alocação de comprimento de onda cientes das restrições associadas à degradação da camada física óptica. O objetivo de um algoritmo PLI-RWA consiste em encontrar uma rota e um comprimento de onda para o estabelecimento de um circuito óptico, levando em consideração as degradações da camada física óptica (*e.g.,* atenuação da fibra, perdas de inserção, ruído da emissão espontânea de amplificador óptico, dispersão por modo de polarização e *crosstalk*) [8]. O problema de roteamento dinâmico ciente das degradações da camada física em redes ópticas translúcidas é considerado mais difícil que o problema correspondente para redes ópticas transparentes [9].

O roteamento baseado em restrições refere-se à classe de algoritmos de roteamento que buscam caminhos que atendam a uma lista de requisitos ou restrições. Desta forma, o roteamento pode ser mono ou multirrestritivo, quando há apenas uma restrição ou quando há múltiplas restrições, respectivamente [10].

Este trabalho introduz uma nova abordagem de roteamento multirrestritivo ciente das degradações da camada física em redes ópticas translúcidas, considerando tanto as restrições relacionadas ao enlace óptico como restrições relacionadas ao comprimento de onda e ao nó óptico, ao mesmo tempo. A partir desta nova abordagem são propostos dois algoritmos de roteamento multirrestritivo, chamados *Offline Multi-Restrictive Routing* (*Off-MRR*) e *Multi-Restrictive Routing* (*MRR*). O *Off-MRR* calcula uma rota viável levando em conta apenas as restrições *offline*, ou seja, aquelas que não se alteram dinamicamente, tais como, número de saltos e distância. Já o *MRR* é dividido em duas fases: fase de planejamento (ou fase *offline*) e fase de operação (ou fase *online*). Na primeira fase do *MRR* (*offline*) é realizado o cálculo de todas as rotas viáveis, considerando as restrições *offline*. Na segunda fase de operação do *MRR* (*online*) é realizada uma "poda" das rotas calculadas na primeira etapa, ficando apenas aquelas rotas que satisfazem às restrições dinâmicas, tais como, dispersão por modo de polarização e restrições de *crosstalk*.

As demais seções deste artigo estão organizadas da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 apresenta a nova abordagem de roteamento multirrestritivo proposta para as redes ópticas translúcidas. A Seção 4 apresenta as heurísticas de roteamento multirrestritivo propostas cujos desempenhos são avaliados na Seção 5. Por fim, as considerações finais são feitas na Seção 6.

## 2. Trabalhos Relacionados

O problema de roteamento e alocação de comprimento de onda ciente da degradação do sinal óptico vem sendo muito estudado nos últimos anos [2, 8, 9, 11, 12, 13, 14].

Rai, Su e Mukherjee, em [11], propõem uma heurística de roteamento para redes ópticas translúcidas, baseada em um algoritmo de busca com informação, visando escolher rotas viáveis com o menor número de saltos.

Azodolmolky e outros, em [8], realizam um estudo com classificação de diversos algoritmos RWA cientes da degradação da camada física óptica. As estratégias de roteamento são classificadas em único caminho (*single path*) ou múltiplos caminhos (*multi-path*). As estratégias de múltiplos caminhos, ou múltiplas rotas, buscam em $n$ rotas alguma rota que seja *viável* para o atendimento de uma requisição de *lightpath*. Todas elas utilizam o algoritmo de menor caminho para cálculo das rotas.

Em [12], Kuipers e outros propõem uma abordagem de roteamento multirrestritivo para as redes ópticas translúcidas. Nessa abordagem, a rede óptica é modelada por um grafo $G(N,L)$, onde N representa o conjunto de nós e $L$ representa o conjunto de enlaces. Associado a cada enlace $(u,v) \in L$ existem $m$ valores reais representando degradações da camada física óptica $r_i(u,v), i=1,...,m$. $N_R \subseteq N$ representa o conjunto de $R$ nós opacos que possuem capacidade de regenerar o sinal óptico. Uma requisição é representada pela t-upla $(s, t, \Delta)$, onde $s, t \in N$ são os nós origem e destino da requisição de circuito óptico e $\Delta = \{\Delta_1, \ldots, \Delta_m\}$ representa os $m$ valores de limites de degradações para cada uma das $m$ degradações de camada física óptica. O problema de roteamento ciente das degradações de camada física óptica consiste em encontrar uma rota com origem em $s$ e destino em $t$ que não ultrapasse nenhum dos limites $\Delta_i, i=1,...,m$ em nenhum dos segmentos transparentes da rota, ou seja, segmentos entre dois nós regeneradores.

Em [12] também é proposta a heurística *Loop Avoidance Heuristic* (LAH) para redes ópticas translúcidas com distribuição esparsa de nós regeneradores. A heurística LAH considera múltiplas restrições no enlace óptico e evita laços. Além disso, Kuipers e outros [12] mostram que o problema modelado de roteamento ciente das degradações de camada física óptica em redes ópticas translúcidas é *NP*-Completo para $m$ restrições, $m>1$.

As estratégias de roteamento utilizadas nesses trabalhos relacionados fazem o uso de métricas apenas no enlace óptico para a busca de uma solução de roteamento monorrestritivo. Em [12], Kuipers e outros apresentam uma modelagem multirrestritiva de roteamento, mas que também considera apenas métricas associadas ao enlace óptico.

Diferentemente da modelagem de múltiplas restrições utilizada em [12], a modelagem multirrestritiva proposta neste trabalho contempla não apenas as restrições relacionadas ao enlace óptico, como também restrições relacionadas ao comprimento de onda e ao nó óptico, ao mesmo tempo. Além disso, este trabalho propõe duas novas

estratégias de roteamento multirrestritivo, cientes dos diferentes tipos de degradações da camada física óptica que afetam os nós, enlaces e comprimentos de onda em redes ópticas translúcidas.

## 3. Uma Nova Abordagem de Roteamento Multirrestritivo para Redes Ópticas Translúcidas

Esta seção apresenta e exemplifica as modelagens de restrições por enlace, comprimento de onda e nó óptico. Além disso, esta seção introduz uma nova taxonomia para algoritmos de roteamento baseados em restrição e propõe uma nova abordagem de roteamento multirrestritivo para redes ópticas translúcidas.

### 3.1 Roteamento com restrição por enlace

O roteamento com restrição por enlace em redes ópticas translúcidas é modelado tipicamente como visto na Seção 2, atribuindo-se um valor numérico por enlace óptico para representar uma métrica de degradação do sinal óptico ao longo de um caminho na rede óptica translúcida, como por exemplo, a distância em quilômetros.

Os algoritmos *Polynomial-time Impairment-Aware Routing Algorithm* (PIARA) [9], *Loop Avoidance Heuristic* (LAH) [12] e *Best Shortest Translucent Lightpath (BSTL)* [15] são exemplos de algoritmos baseados em uma modelagem de rede óptica translúcida com restrição por enlace.

### 3.2 Roteamento com restrição por comprimento de onda

Para exemplificar e justificar a abordagem de roteamento com restrição por comprimento de onda, a seguir é definida uma modelagem específica de *crosstalk* para redes ópticas translúcidas.

O sinal óptico sofre degradações ao longo de sua rota devido a diversos fatores relacionados à camada física óptica. Um desses fatores diz respeito ao *crosstalk* [16]. Nesta modelagem adota-se parcialmente a abordagem de *crosstalk* apresentada em [17], considerando-se apenas a interferência entre canais ópticos vizinhos.

Na modelagem de *crosstalk* utilizada como restrição para comprimento de onda neste trabalho, a rota que passa no comprimento de onda $\lambda_i$ sofre interferência da rota que passa no comprimento de onda adjacente $\lambda_j$. Isto ocorre devido à imperfeição do multiplexador (MUX) ao "misturar" parte do sinal do comprimento de onda $\lambda_i$ com o do comprimento de onda $\lambda_j$ na inserção do sinal óptico no enlace físico. Desta forma, o sinal óptico associado a cada um dos comprimentos de onda percorre o enlace com uma pequena perda da qualidade. Esta perda fica maior no final do percurso ao passar pelo demultiplexador (DEMUX) e aumentar o índice de *crosstalk* de maneira semelhante ao que ocorreu no multiplexador. Esta modelagem de *crosstalk* é chamada neste trabalho de *Link Crosstalk* (LC).

Considera-se então um número $c_w$ que representa o índice de *crosstalk* do comprimento de onda $w$ (número de comprimentos de onda vizinhos ocupados) e $c_w(P_{s,t})$ representa o acúmulo de *crosstalk* ($c_w$) na rota com origem no nó $s$ e destino no nó $t$. Desta forma, um limite de degradação está associado ao limite máximo de acúmulo de *crosstalk* que uma rota pode ter, também chamado de *LC Threshold*. A Figura 1 exemplifica o estado dinâmico de uma rota e os valores de $c_w(P_{s,t})$ de cada comprimento

de onda *w* disponível naquela rota (*w=1 e w=4)*. Vale ressaltar que ao passar por um nó regenerador os valores de degradação são reiniciados.

| *w* | $c_w$ |
|---|---|
| *1* | *1* |
| *2* | *1* |
| *3* | *1* |
| *4* | *2* |
| *5* | *0* |

| *w* | $c_w$ | $c_w(P_{s,t})$ | *LC Threshold = 2* |
|---|---|---|---|
| **1** | *1* | ***2*** | ***viável*** |
| **2** | *0* | - | *sem disponibilidade* |
| **3** | *1* | - | *sem continuidade* |
| **4** | *1* | ***3*** | ***inviável*** |
| **5** | *0* | - | *sem disponibilidade* |

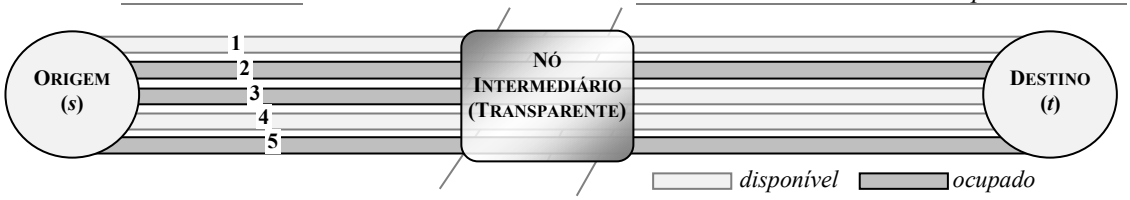



**Figura 1. Estado dinâmico de um enlace óptico e a tabela de índices de *Link Crosstalk*.**

## 3.3 Roteamento com restrição por nó

Esta última classe de roteamento restritivo em redes ópticas translúcidas diz respeito às restrições associadas ao nó *Optical-Crossconect* (OXC) como, por exemplo, restrição de número máximo de OXCs em uma rota.

Para melhor exemplificar e justificar esta classe, é proposta uma segunda modelagem de *crosstalk*, diferente da primeira modelagem apresentada na Seção 3.2 (*Link Crosstalk*). Esta segunda modelagem não invalida a primeira, apenas considera outro tipo de *crosstalk* que ocorre exclusivamente no nó óptico, chamada de *OXC Crosstalk*. Tal abordagem de *crosstalk* é inspirada no trabalho apresentado em [17].

Nós *Optical-Crossconect* (OXCs) são componentes ópticos intermediários em uma rota responsáveis pela comutação do sinal óptico na granularidade porta/comprimento de onda [18]. Além disso, um nó OXC também opera como nó origem e destino, realizando, respectivamente, a inserção e a terminação de *lightpaths*.

Cada OXC possui portas de entrada, portas de saída e comutadores de comprimento de onda. As portas de entrada possuem a função de demultiplexar o sinal óptico nos seus vários comprimentos de onda. As portas de saída realizam a função inversa, ou seja, de multiplexar o sinal de cada comprimento de onda em um único feixe de luz. Por fim, os comutadores ópticos possuem um módulo de comutação para cada comprimento de onda com a função de realizar a comutação de um comprimento de onda oriundo de uma porta de entrada para uma porta de saída e, além disso, adicionar e retirar *lightpaths* na rede óptica.

A Figura 2 ilustra uma ocorrência de *OXC crosstalk*. Nesta modelagem, é considerado o tipo de *crosstalk* que degrada o sinal de diferentes *lightpaths* que operam em um mesmo comprimento de onda. A degradação ocorre quando os *lightpaths* passam, simultaneamente, no mesmo módulo de comutação. Por exemplo, na Figura 2 dois *lightpaths* passam no OXC: o $LP_1$ da porta de entrada 1 para a porta de saída 1 e o $LP_2$ da porta de entrada 2 para a porta de saída 2. Como ambos os *LPs* entram no módulo de comutação do comprimento de onda $\lambda_i$, o *OXC crosstalk* ocorre neste módulo de comutação. Quando os dois *LPs* saem do módulo de comutação, o $LP_1$ transporta uma pequena parte do sinal oriundo do $LP_2$ e vice versa.

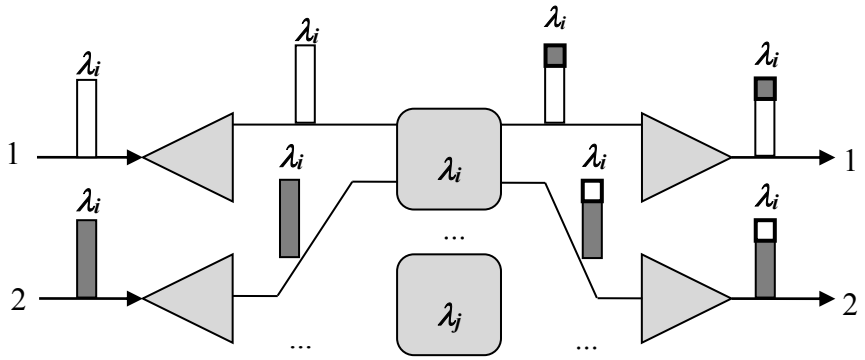



**Figura 2. Restrição de *OXC Crosstalk* no nó óptico.**

Para computar o *OXC crosstalk* é considerado um número *x* que representa o índice de *OXC crosstalk* de um determinado comprimento de onda em um OXC específico e $x(P_{s,t})$ representa o acúmulo de *OXC crosstalk* na rota com origem no nó *s* e destino no nó *t*. Vale destacar que os valores destas restrições são calculados dinamicamente, em função da atualização do estado da rede.

Semelhantemente às abordagens de restrição por enlace e comprimento de onda, o limite de degradação pode estar associado ao limite máximo de acúmulo de degradação do tipo nó (*e.g., OXC Crosstalk)* que uma rota pode ter. Vale ressaltar mais uma vez que quando um *lightpath* passa por um nó regenerador os seus valores de degradação são reiniciados.

## 3.4 Taxonomia das estratégias de roteamento baseadas em restrições

Apesar da escassez de algoritmos multirrestritivos em redes ópticas translúcidas, os trabalhos relacionados e as possíveis abordagens de roteamento apresentadas anteriormente permitem uma análise abrangente das estratégias de roteamento em redes ópticas translúcidas. Desta forma, esta subseção propõe uma taxonomia (classificação hierárquica) para algoritmos de roteamento que consideram restrições em redes ópticas translúcidas.

De maneira geral, as estratégias de roteamento podem ser classificadas em monorrestritivas ou multirrestritivas de acordo com o número de restrições que são consideradas por elas [10]. Conforme mostrado nas subseções 3.1 a 3.3, uma restrição pode estar associada ao enlace óptico (*e.g.* distância), ao comprimento de onda (*e.g. link crosstalk*) ou ao nó óptico (*e.g. OXC crosstalk* e número de OXCs intermediários). Além disso, cada medida de restrição pode, ou não, variar em função da operação da rede, sendo assim, uma restrição pode também ser classificada em *offline* ou *online*.

Independente de uma estratégia ser monorrestritiva ou multirrestritiva, a(s) restrição(ções) que a compõe pode(m) ser dita(s) *offline* ou *online*. Seguem alguns exemplos: *offline* - salto e distância; *online* - *link crosstalk* e OXC *crosstalk*. Na próxima subseção é definida a nova abordagem de roteamento proposta neste trabalho com o objetivo de considerar qualquer tipo de restrição, seja ela uma restrição de nó, de comprimento de onda ou de enlace, *offline* ou *online*.

## 3.5 Definição da nova abordagem de roteamento multirrestritivo

A rede óptica é representada por um grafo *G(N,L),*onde *N* representa o conjunto dos nós da rede e *L* o conjunto de enlaces da rede. *W* representa o conjunto de comprimentos de onda da rede óptica. O conjunto de nós opacos regeneradores que possuem capacidade de regenerar o sinal óptico e reiniciar todos os índices de degradação de um *lightpath* é representado por $N_R$, sendo $N_R \subseteq N$. É considerado de forma genérica um número real *r*, através da qual se representa o índice de degradação de uma determinada métrica associada à camada física óptica.

As restrições são representadas da seguinte forma (Fig. 3). Para cada enlace $(u,v) \in L$, estão associados *y* valores reais, representando as restrições por enlace $r_i(u,v), i=1,...,y$. Associados a cada comprimento de onda $w \in W$ existem *z* valores reais, representando as restrições por comprimento de onda $r_i(w), i=1,...,z$. Por último, associado a cada nó *n* existem *x* valores reais representando as restrições por nó óptico $r_i(n), i=1,...,x$. Desta forma, o total de *m* restrições é dado pelo somatório da quantidade de cada tipo de restrição, ou seja, $m=y+z+x$.

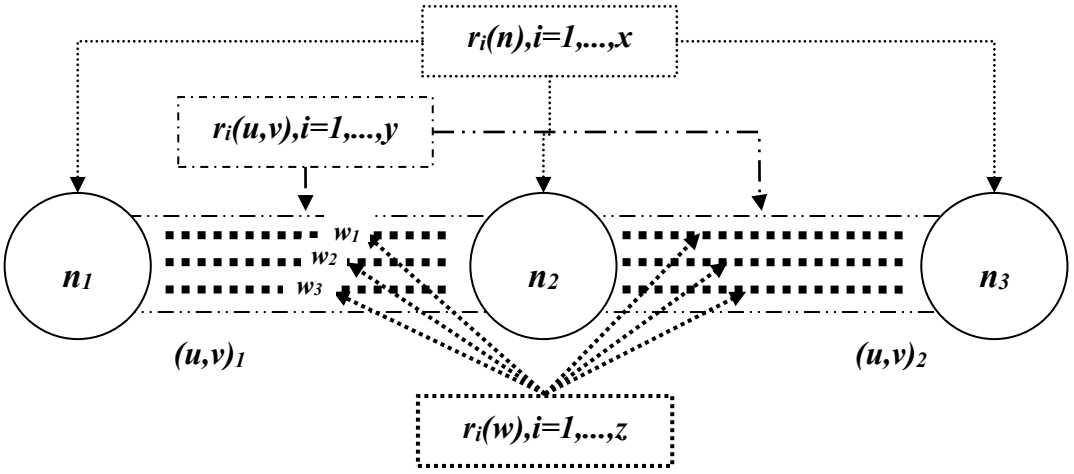



**Figura 3. Modelagem de roteamento multirrestritivo para redes ópticas translúcidas.**

Nesta abordagem, uma requisição é representada pela t-upla $(s, t, \Delta)$, onde $s, t \in N$ são os nós origem e destino da requisição de circuito óptico e $\Delta = \{\Delta_1, \ldots, \Delta_m\}$ representa os *m* valores de limites de degradações para cada uma das *m* degradações de camada física óptica.

Desta forma, é possível definir o problema de roteamento multirrestritivo ciente das degradações de camada física óptica como sendo o de encontrar uma rota com origem em *s* e destino em *t* a qual não ultrapasse nenhum dos limites $\Delta_i$, $i = 1, \ldots, m$ em nenhum dos segmentos transparentes da rota, ou seja, segmentos entre dois nós regeneradores.

Uma vez que o RWA dinâmico, incluindo o problema de roteamento com múltiplas restrições apenas no enlace já é um problema NP-Completo [12], o problema definido nesta seção de roteamento multirrestritivo, considerando, além das restrições no enlace, restrições no comprimento de onda e no nó óptico, também é um problema NP-Completo.

## 4. Estratégias Propostas

Diante da própria natureza do RWA dinâmico e da complexidade do problema de roteamento multirrestritivo identificado e apresentado na seção anterior, se faz necessário o uso de algoritmos heurísticos para determinar uma rota dinamicamente para uma requisição de circuito óptico definida por (*s, t*, Δ), segundo a Seção 3.5. Esta seção apresenta em detalhes duas estratégias de roteamento multirrestritivo incluindo uma análise comparativa da complexidade computacional de cada uma delas. Ambas as estratégias são divididas em duas fases: fase de planejamento da rede (ou fase *offline*) e fase de operação da rede (ou fase *online*).

### 4.1 Roteamento baseado em multirrestrições *offline*

A primeira estratégia (*Offline Multi-Restrictive Routing – Off-MRR)* é uma estratégia de roteamento fixo que leva em consideração todas as restrições do tipo *offline*. A seguir é descrito o *passo-a-passo* da execução do algoritmo *Off-MRR.*

Estratégia ***Off-MRR***:

---

**[Planejamento] – Nesta etapa de planejamento são conhecidos todos os recursos da rede óptica translúcida.**

1) Utilizando o algoritmo de *Dijkstra* modificado [19], calcular o conjunto $R_{(s,t)}$ de todas as possíveis rotas para cada *par(s,t).*

2) Remover de cada conjunto $R_{(s,t)}$ aquelas rotas que não satisfazem às restrições *offline* (ou por enlace, ou por comprimento de onda, ou por nó óptico).

3) Guardar a primeira rota de cada conjunto de rotas $R_{(s,t)}$, ou seja, armazenar uma rota viável para cada *par(s,t),* considerando as restrições *offline.*

**[Operação] – Quando da chegada de uma requisição (*s, t*, Δ)**

4) Retornar a rota calculada previamente na fase de planejamento.

---

### 4.2 Roteamento totalmente multirrestritivo

A segunda estratégia proposta para solucionar o problema de roteamento multirrestritivo apresentado na Seção 3.5 é chamada de *Multi-Restrictive Routing* (*MRR*). Nesta estratégia, inicialmente, na fase de planejamento, são calculadas todas as possíveis rotas viáveis para cada *par(origem,destino)*, considerando-se as restrições *offline*. Tal conduta visa ao melhor desempenho do algoritmo na fase dinâmica, potencializando a redução do tempo requerido para o estabelecimento dinâmico do circuito óptico. Na fase de operação da rede, quando da chegada de uma requisição *(s, t, Δ)* o conjunto de rotas pré-calculadas é percorrido no sentido de encontrar uma rota que satisfaça a todas as *m* restrições impostas. A seguir são apresentadas as principais etapas da estratégia proposta *MRR*.

Etapas da estratégia ***MRR***:

**[Planejamento] – Nesta etapa de planejamento são conhecidos todos os recursos da rede óptica translúcida.**

1) Utilizando o algoritmo de *Dijkstra* modificado [19], calcular todas as rotas viáveis $R_v$ para cada *par(s,t)*, considerando as restrições *offline* por enlace, por comprimento de onda e por nó.

**[Operação] – Quando da chegada de uma requisição (*s*, *t*, Δ)**

2) Para cada rota $R_v$ pré-calculada faça:
   a. Para cada restrição dinâmica por enlace $r_i(u,v)$ faça:
      $a_1$. Verificar se a rota $R_v$ é uma rota viável considerando $r_i(u,v)$.
      $a_2$. Caso contrário, voltar para o passo 2 e considerar outra rota
   b. Para cada restrição dinâmica por comprimento de onda $c_i(w)$ faça:
      $b_1$. Verificar se a rota $R_v$ é uma rota viável considerando $c_i(w)$.
      $b_2$. Caso contrário, voltar para o passo 2 e considerar outra rota
   c. Para cada restrição dinâmica por nó $x_i(n)$ faça:
      $c_1$. Verificar se a rota $R_v$ é uma rota viável considerando $x_i(n)$.
      $c_2$. Caso contrário, voltar para o passo 2 e considerar outra rota
3) Retornar a primeira rota viável encontrada, considerando todas as *m* restrições.

## 4.3 Análise da complexidade computacional das estratégias propostas

É utilizada a notação "*O"*[19] para designar a complexidade computacional dos algoritmos propostos. Para efeito desta análise de complexidade será considerada apenas a fase de operação da rede devido à característica peculiar do cenário dinâmico das redes ópticas translúcidas, onde o tempo de execução dos algoritmos na fase de planejamento não é um fator crítico porque a rede ainda não está em operação. Por exemplo, em todos os experimentos realizados neste artigo a fase de planejamento dos algoritmos *Off-MRR* e *MRR* consumiu apenas alguns minutos para calcular todas as rotas viáveis, considerando todas as restrições definidas como *offline*, o que não é um tempo significativo levando-se em conta que a rede não está em operação.

A primeira estratégia, chamada de *Offline Multi-Restrictive Routing (Off-MRR),* é um algoritmo de roteamento fixo, ou seja, ele calcula previamente uma rota levando em conta apenas as restrições *offline*. Quando da chegada da requisição este primeiro algoritmo apresenta o menor tempo possível para retornar a rota, pois a mesma já foi calculada previamente, necessitando apenas de um acesso de tempo fixo a uma estrutura de dados, a qual armazena a rota previamente calculada. Desta forma, a complexidade computacional do *Off-MRR* é de *O(1)*.

A segunda estratégia, chamada de *Multi-Restrictive Routing (MRR)* é constituída por um algoritmo que, na fase de planejamento da rede, calcula todas as rotas viáveis (considerando as restrições *offline*) e na fase de operação da rede retorna, dentre as rotas previamente calculadas, a primeira rota que satisfaz a todas as restrições dinâmicas. Para isso, o algoritmo percorre todos os nós e enlaces de cada rota pré-calculada, verificando a viabilidade de cada restrição. Com isso, considerando $R_v$ o número de rotas viáveis pré-calculadas, *m* a quantidade de restrições dinâmicas impostas, *N* a

quantidade de nós da rede e $L$ a quantidade de enlaces da rede, a complexidade computacional do *MRR* na fase de operação é $O(R_v\, m(N+L))$.

Para exemplificar, o tempo de execução dos algoritmos *Off-MRR* e *MRR* foi medido na fase de operação da rede em todos experimentos descritos na Seção 5. Esses experimentos foram realizados em um computador *desktop* Intel Core 2 Duo, 4G RAM, Sistema Operacional Windows XP. O tempo médio de execução do algoritmo *Off-MRR* para as topologias Americana e EON (Fig. 4) foi de $6{,}67\text{x}10^{-4}ms$ e $5{,}65\text{x}10^{-4}ms$, respectivamente. Por outro lado, o tempo médio de execução do algoritmo *MRR* para as mesmas topologias foi de 2,24*ms* e 2,44*ms*, respectivamente. Apesar de o algoritmo *MRR* consumir um tempo na ordem de alguns milissegundos, esse é um tempo bastante razoável para ser considerado no estabelecimento de um *ligthpath* dinâmico [20].

Desta forma, o algoritmo *MRR* despende um tempo maior na fase de operação da rede quando comparado com o algoritmo *Off-MRR*. Por outro lado, espera-se que o *MRR* apresente melhor desempenho em termos de probabilidade de bloqueio de requisições de *lightpaths*, uma vez que o *MRR* considera satisfazer qualquer tipo de restrição (*offline* ou *online*). Na seção seguinte são analisados os resultados de simulação com os algoritmos propostos.

## 5. Resultados de Simulação

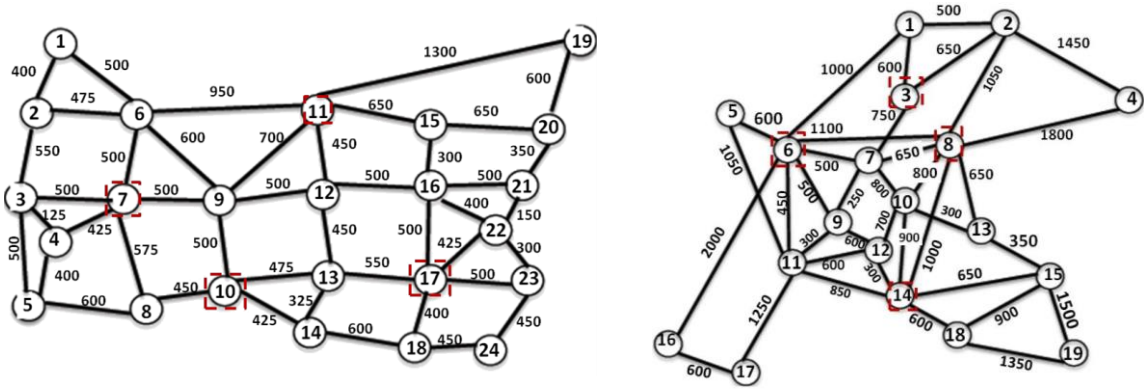
A ferramenta de simulação TONetS [21] foi estendida para suportar os estudos de avaliação de desempenho dos algoritmos de roteamento para redes ópticas translúcidas considerados neste trabalho. Na versão atual do TONetS é possível simular os seguintes algoritmos: algoritmos de posicionamento de nós opacos (considerando nós com maior número de interfaces e algoritmo aleatório, evitando nós vizinhos) e algoritmos de roteamento cientes das limitações da camada física – monorrestritivos (PIARA [12] e BSTL [15]) e os multirrestritivos propostos neste trabalho (*Off-MRR* e *MRR*).

Este estudo de avaliação de desempenho tem as seguintes características básicas. A demanda de tráfego é composta por requisições de circuitos ópticos representados por pares de nós (origem, destino). A carga de tráfego é distribuída uniformemente entre todos os $N \times (N-1)$ pares de nós (origem, destino). A geração de requisições é um processo poissoniano de taxa média $\lambda$ e o tempo de retenção dos circuitos é distribuído exponencialmente com média $1/\mu$; a intensidade de tráfego na rede em *Erlangs* é dada por $\rho=\lambda/\mu$. Todos os enlaces da rede são bidirecionais e possuem 40 comprimentos de onda em cada sentido. O algoritmo *First-Fit* [4], pela sua simplicidade e desempenho, é utilizado na alocação dos comprimentos de onda. Para cada simulação são realizadas quatro replicações com diferentes sementes de geração de valores aleatórios. São geradas duas milhões de requisições para cada replicação. Os resultados gráficos apresentam os intervalos de confiança com um nível de confiança de 95%.

Nesta avaliação, o algoritmo proposto *Off-MRR* (Seção 4.1) é comparado com o algoritmo *MRR* (Seção 4.2) considerando a topologia da rede Americana e a topologia da rede *European Optical Network* (EON) ilustradas nas Figuras 4.a e 4.b, respectivamente. Os nós opacos regeneradores são distribuídos de forma a priorizar os nós que possuem maior número de interfaces, evitando nós vizinhos.

As restrições impostas foram as seguintes: distância (1500km para a topologia Americana e 2000km para a topologia EON), *Link Crosstalk* (2 unidades) e saltos (2

saltos). O valor para a restrição de distância em cada topologia foi imposto de forma a tornar, minimamente, cada *par(s,t)* alcançável.



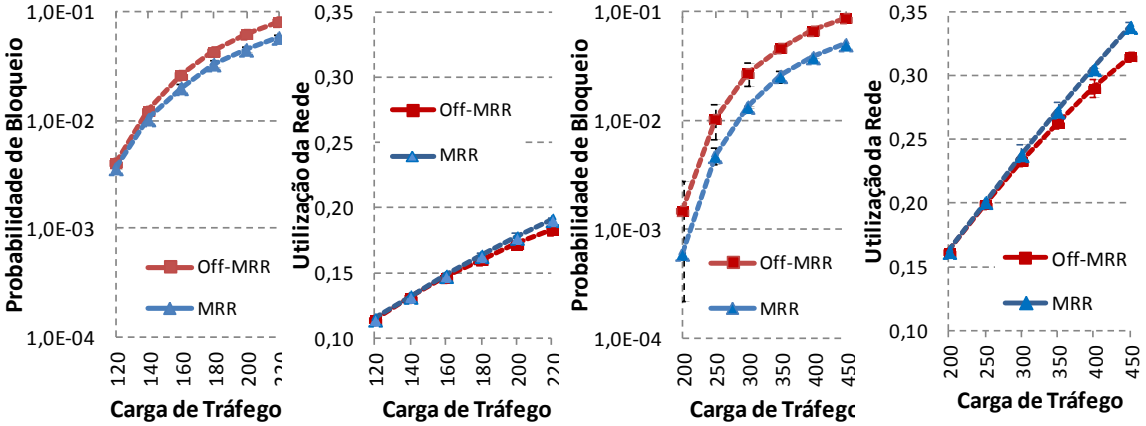

a) Topologia Americana. b) Topologia EON.

**Figura 4. Topologias de redes ópticas translúcidas avaliadas.**

Inicialmente foi realizado um estudo de avaliação de desempenho em termos de probabilidade de bloqueio de requisições de circuitos ópticos e de utilização da rede. Na segunda etapa, foi realizado um estudo de avaliação de desempenho variando o limite da restrição de *link crostalk*.

## 5.1. Probabilidade de Bloqueio

As Figuras 5.a e 5.b mostram os resultados referentes à probabilidade de bloqueio e à utilização da rede para os cenários ilustrados nas Figuras. 4.a e 4.b, respectivamente.




a) Topologia Americana b) Topologia EON

**Figura 5. Probabilidade de Bloqueio e Utilização.**

Analisando os resultados de probabilidade de bloqueio na Figura 5, observa-se que o *MRR* apresentou um desempenho superior ao *Off-MRR*. O algoritmo *Off-MRR* calcula previamente uma única rota satisfazendo as restrições *offline*, a qual, quando da chegada de uma requisição de circuito óptico, pode não ter um comprimento de onda viável para o atendimento da requisição devido às restrições *online*. Por isto, o desempenho do *MRR* foi superior ao *Off-MRR* para os dois cenários avaliados, em termos de probabilidade de bloqueio. Os gráficos de utilização da rede (Fig. 5) também evidenciam o menor uso de recursos por parte do *Off-MRR*. Isso ocorre porque o *Off-*

*MRR* disponibiliza apenas uma rota fixa para o atendimento das requisições, enquanto o *MRR* pode utilizar qualquer uma rota dentre as rotas alternativas pré-calculadas, balanceando a carga entre os enlaces dessas rotas alternativas. De acordo com os experimentos realizados, apesar de apresentar menor complexidade computacional (Seção 4.3), o algoritmo *Off-MRR* subutiliza a rede, resultando em maior probabilidade de bloqueio e menor utilização.

### 5.2. Impacto do limite da restrição de *Link Crosstalk* na probabilidade de bloqueio

Nesta segunda etapa de avaliação de desempenho, é fixado o valor de carga de tráfego e variado o limite da restrição *Link Crosstalk* (*LC Threshold*) apresentada na Seção 3.3, com o objetivo de analisar o impacto desta variação (0 a 6 unidades de *LC Threshold*) na métrica de probabilidade de bloqueio.

Para melhor compreensão deste impacto, a probabilidade de bloqueio geral é decomposta em 3 (três) tipos de Probabilidade de Bloqueio (PB): **PB por Falta de Continuidade**, **PB por Falta de Qualidade** e **PB por Interferência**. A métrica PB por Falta de Continuidade caracteriza os bloqueios por falta de comprimento de onda livre com continuidade, em algum segmento transparente. A métrica PB por Falta de Qualidade representa os bloqueios por falta de qualidade no comprimento de onda, ou seja, por inviabilidade de comprimento de onda (neste caso, todos os comprimentos de onda contínuos disponíveis ultrapassam o limite imposto pela restrição de *Link Crostalk*). Por fim, a métrica PB por Interferência contabiliza os bloqueios por interferência na qualidade de alguma conexão ativa (neste caso, existem comprimentos de onda disponíveis e com qualidade, porém, a ocupação de qualquer um desses comprimentos de onda "degradaria" uma conexão ativa, ultrapassando o limite imposto à restrição de *Link Crostalk*). As Figuras 6 e 7 mostram os gráficos de Probabilidade de Bloqueio Geral (PB Geral), PB por Falta de Continuidade, PB por Falta de Qualidade e PB por Interferência para as topologias Americana e EON, respectivamente.

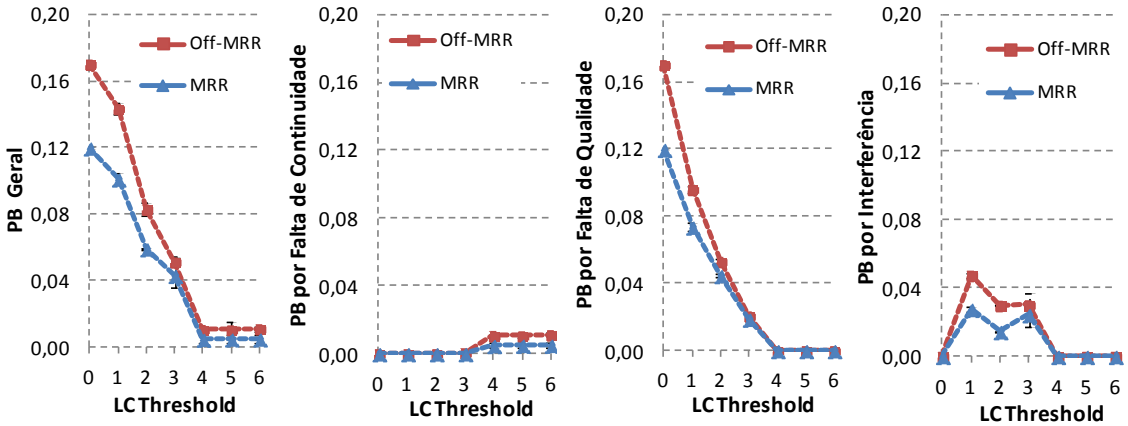



**Figura 6. PB em função da restrição de *Link Crosstalk* – Topologia Americana.**

Conforme evidenciado nas Figuras 6 e 7, independente da topologia de rede, a probabilidade de bloqueio diminui na medida em que os limites de *Link Crosstalk* vão aumentando. No entanto, a partir do limite de 4 unidades de *Link Crosstalk* (*LC Threshold*=4) não há ganho de probabilidade de bloqueio. Isto ocorre porque o máximo de acúmulo da degradação de *Link Crosstalk* é justamente igual a 4, uma vez que a cada comprimento de onda, no máximo duas unidades de *Link Crosstalk* (correspondente aos comprimentos de onda vizinhos) são contabilizados em um enlace (ver Seção 3.2) e,

além disso, para a topologia avaliada, no máximo a cada 2 enlaces o sinal óptico é regenerado e os valores das restrições são zerados. Desta forma, a partir *LC Threshold*=4 não é mais contabilizada as PBs por Falta de Qualidade e por Interferência, apenas a PB por Falta de Continuidade é contabilizada.

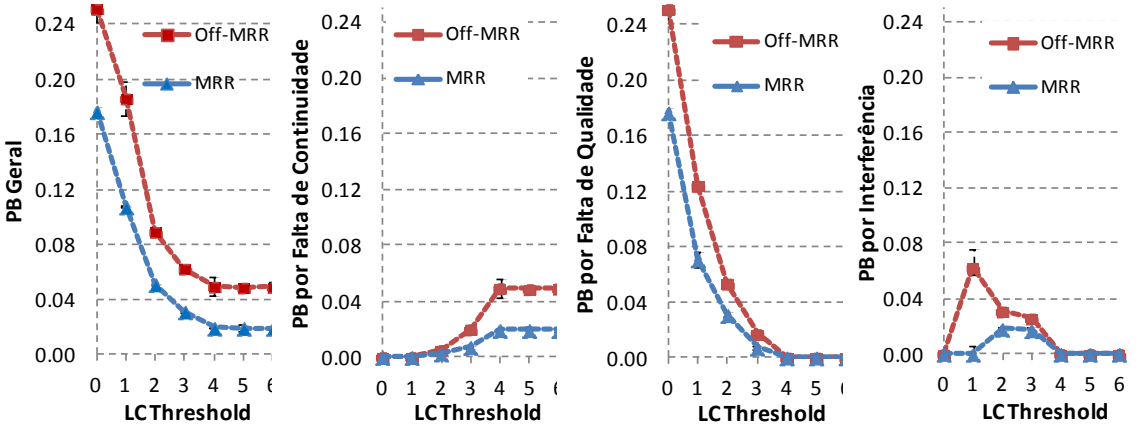



**Figura 7. PB em função da restrição de *Link Crosstalk* – Topologia EON.**

Nas Figuras 6 e 7 é possível verificar ainda que, com o *LC Threshold* = 0 (zero), a PB Geral é 100% composta pela PB por Falta de Qualidade, uma vez que o estabelecimento de um *ligthpath* inviabiliza a utilização dos comprimentos de onda adjacentes, tornando-os sempre disponíveis (PB por Falta de Continuidade = 0%), porém sem poderem ser utilizados, devido à restrição máxima (*LC Threshold* = 0).

## 6. Considerações Finais

Este trabalho introduziu uma nova abordagem de roteamento multirrestritivo para as redes ópticas translúcidas e propôs dois algoritmos de roteamento multirrestritivo que consideram ao mesmo tempo limitações da camada física óptica de diferentes tipos (enlace, nó e comprimento de onda). Os algoritmos propostos foram avaliados e comparados, possibilitando observar o desempenho superior do primeiro (*Off-MRR*) em termos de tempo de execução e do segundo (*MRR*) em termos de utilização da rede e probabilidade de bloqueio. Além disso, foi apresentada uma análise detalhada do impacto da variação de uma das métricas modeladas (*Link Crosstalk*) na probabilidade de bloqueio de conexões em redes ópticas translúcidas. Esta análise detalhada permitiu compreender melhor o comportamento das curvas de probabilidade de bloqueio em função dos limites impostos de *Link Crosstalk* aos cenários avaliados.

## Referências

[1] M. J. O'MAHONY *et al*. "Future Optical Networks", *Journal Lightwave Technologies*, v.24, p. 4684-4696, 2006.

[2] M. Gagnaire, e S. Zahr. "Impairment-Aware Routing and Wavelength Assignment in Translucent Optical Networks: State of the Art". *IEEE Communication Magazine*, v. 47, n. 5, p. 55-61, Maio, 2009.

[3] G. Shen e R. S. Tucker. "Translucent Optical Networks: The Way Forward". *IEEE Communications Magazine*, v. 45, n. 2, p. 48-54, Fevereiro, 2007.

[4] A. C. B. Soares e W. F. Giozza, "Avaliação de desempenho de algoritmos para alocação de comprimentos de onda em redes ópticas transparentes", *in Anais do 27º Simpósio Brasileiro de Redes de Computadores (SBRC),* p.661–674, 2004.

[5] G. M. Durães, *et al*. "The choice of the best among the shortest routes in transparent optical networks", *Computer Networks*, v. 54, p. 2400-2409, 2010.

[6] G. M. Durães, *et al.*, "A Escolha da Melhor entre as Menores Rotas em Redes Ópticas Transparentes", *in Anais do 27º Simpósio Brasileiro de Redes de Computadores (SBRC),* p. 1-14, 2009.

[7] B. Ramamurthy, *et al,* "Impact of Transmission Impairments on the Teletraffic Performance of Wavelength-routed Optical Networks", *Journal of Lightwave Technology (JLT),* vol. 17, no. 10, p. 1713- 1723, 1999.

[8] S. Azodolmolky, *et al*. "A survey on physical layer impairments aware routing and wavelength assignment algorithms in optical networks", *Computer Networks*, p. 926-944, Janeiro, 2009.

[9] K. Manousakis, *et al*. "Joint Online Routing, Wavelength Assignment and Regenerator Allocation in Translucent Optical Networks", "*Journal of Lightwave Technology*, v. 28, n. 8, p.1152-1163, Abril, 15, 2010.

[10] D. Awduche, *et al* , RFC 2702 – *Requirements for Traffic Engineer Over MPLS*, 1999.

[11] S. Rai, C-F Su e B. Mukherjee, "On provisioning in all-optical networks: an impairment-aware approach", *In: IEEE/ACM Transactions on Networking*, v. 17, n. 6, p. 1989-2001, 2009.

[12] F.A. Kuipers, *et al*., "Impairment-aware Path Selection and Regenerator Placement in Translucent Optical Networks," *Proc. of the 18th IEEE International Conference on Network Protocols (ICNP 2010)*, Kyoto, Japão, Outubro, 5-8, 2010.

[13] A. G. Rahbar, "Review of dynamic impairment-aware routing and wavelength assignment techniques in all-optical wavelength-routed networks", *IEEE Communications Surveys & Tutorials*, v.14, n.4, p. 1065-1089, 2012.

[14] E. A. Varvarigos e K. Christodoulopoulos, "Algorithmic Aspects in Planning Fixed and Flexible Optical Networks With Emphasis on Linear Optimization and Heuristic Techniques", *Journal of Lightwave Technology*, v.32, n.4, p. 681-693, 2014.

[15] G. M. Durães, *et al.*, "Roteamento Adaptativo de Menor Caminho para Redes Ópticas Translúcidas", *in Anais do 30º Simpósio Brasileiro de Redes de Computadores (SBRC),* p. 1-14, 2012.

[16] J. Strand e A. Chiu, "Impairments and Other Constraints on Optical Layer Routing," RFC 4054, Maio, 2005.

[17] T. Deng, S. Subramanian e J. Xu, "Crosstalk-aware wavelength assignment in dynamic wavelength-routed optical networks", *In Proceedings of the First International Conference On Broadband Networks (BroadNets),* p. 140-149, 2004.

[18] C. S. R. Murthy e M. Gurusamy, "WDM optical networks: concepts, design and algorithms", *Prentice Hall PTR*, 2002

[19] T. H. Cormen, *et al*, "Introduction to algorithms". 2ª ed., *The MIT Press and McGraw-Hill Book Company*, 2001.

[20] R. Ramaswami, N. K. Sivarajan e G. H. Sasaki, "Optical network**:** a practical perspective". 3.ed., Morgan Kaufmann Publishers, 2010.

[21] A. C. B. Soares, G. M. Durães, W. Giozza e P. Cunha, "TONetS: Ferramenta para Avaliação de Desempenho de Redes Ópticas Transparentes" *in VII Salão de Ferramentas do Simpósio Brasileiro de Redes de Computadores - SBRC*, Maio 2008.

# Algoritmo de Roteamento e Atribuição de Espectro com Minimização de Fragmentação em Redes Óticas Elásticas

**André K. Horota[1], Gustavo B. Figueiredo[1], Nelson L. S. da Fonseca[2]**

[1]Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n, Ondina – 40170-115 – Salvador – BA – Brasil

[2]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
13081-970 – Campinas – SP – Brasil.

{horota,gustavo}@dcc.ufba.br, nfonseca@ic.unicamp.br

***Abstract.*** *Elastic Optical Networks has drawn a lot of attention in the past few years because of its ability to transmit different data rates increasing ou decreasing the optical spectrum according to the necessary demand, ensuring a high spectral efficiency. However, establishing and tearing down different connections ends up segmenting the spectrum in several small fragments, making it impossible for new requests to be attended. Besides, another recurrent problem in elastic optical networks, called routing and spectrum assignment (RSA), aims to fing a path and assign to it a contiguous number of spectrum slots using the smallest possible amount of spectral resources. The main purpose of this paper is to propose an algorithm to minimize the spectrum fragmentation problem in Elastic Optical Networks by using RSA algorithms. Results obtained through simulations indicates that the proposed algorithm outperforms in terms of blocking probability and spectrum fragmentation ratio the other existing ones.*

***Resumo.*** *Redes Ópticas Elásticas tem atraído bastante atenção nos últimos anos por possuir a capacidade de transmitir dados a diferentes taxas aumentando ou diminuindo o espectro óptico de acordo com a demanda necessária, garantindo uma alta eficiência espectral. Contudo, o estabelecimento e terminação de diferentes conexões acaba resultando na divisão do espectro em diversos pequenos fragmentos, tornando impossível que novas requisições sejam atendidas. Além disso, outro recorrente problema em redes ópticas elásticas, denominado roteamento e atribuição de espectro (RSA), visa encontrar um caminho e atribuir ao mesmo uma quantidade contígua de slots de espectro utilizando a menor quantidade possível de recursos espectrais. O principal objetivo deste artigo é propor um algoritmo para minimizar o problema de fragmentação de espectro em Redes Ópticas Elásticas através de algoritmos de RSA. Resultados obtidos através de simulações indicam que o algoritmo proposto supera em termos de taxa de probabilidade de bloqueio e taxa de fragmentação de espectro os demais algoritmos comparados.*

## 1. Introdução

O sempre crescente aumento no volume de tráfego gerado por grandes provedores de conteúdo e *data centers* aliado à relação de *peering* comumente estabelecida entre os provedores de serviço terminam por aumentar a imprevisibilidade e heterogeneidade das

demandas por recursos ao longo da rede de transporte. Assim, para atender a essa demanda, os provedores tem instalado linhas com elevadíssimas taxa de bits com canais operando a *40Gbps*, ou mais recentemente a *100Gbps*, baseados na tecnologia DWDM (do inglês *Dense Wavelength Division Multiplexing*) [Gerstel et al. 2012].

Entretanto, além do provimento de canais de alta velocidade, a utilização eficiente dos recursos da rede é um requisito imperativo que deve ser atendido pelas redes de transporte sob pena de altos custos por unidade de largura de banda, além de alta probabilidade de bloqueio e consequente degradação dos serviços. Todavia, a divisão do espectro em *slots* de largura de banda fixa de *50GHz* feita nas redes DWDM, além de dificultar a transmissão em altas taxas por longas distâncias, impõe rigidez nas taxas de transmissão em cada comprimento de onda [Gerstel et al. 2012]. Isto, em última instância, reduz a eficiência na utlização dos recursos, gerando muitas vezes subutilização ou superprovisionamento dos mesmos devido às diferentes granularidades de demanda geradas nas camadas dos clientes.

As Redes Ópticas Elásticas surgiram com uma proposta capaz de diminuir a disparidade entre a granularidade das demandas das redes cliente e aquela usada nos canais de transmissão das redes de transporte. Para tal, os recursos espectrais são divididos adaptativamente para atender de forma ideal as requisições de banda, gerando canais com largura de banda variável, determinada de acordo com as necessidades dos fluxos a serem transmitidos.

Obviamente, quanto melhor a estratégia de alocação de recursos às requisições, maior a demanda que poderá ser atendida. Nas redes WDM, esse problema de alocação de recursos é chamado de RWA (do inglês *Routing and Wavelength Assignment*) e visa alocar comprimentos de onda às requisições das redes cliente. No contexto das redes ópticas elásticas este problema tem sido chamado de RSA (do inglês *Routing and Spectrum Assignment*) e seu objetivo é encontrar um caminho e atribuir ao mesmo uma quantidade contígua de *slots* de espectro utilizando a menor quantidade possível de recursos espectrais. Este, assim como o RWA, é um problema NP-Completo [Wang, Cao e Pan 2011] e, dessa maneira, diversas heurísticas tem sido propostas para a alocação de recursos em redes ópticas elásticas [Jinno et al. 2010, Christodoulopoulos, Tomkos e Varvarigos 2010, Wan et al. 2011, Zhang et al. 2013, Santos et al. 2013, Wang e Mukherjee 2013].

Além do *RSA*, outro problema associado à alocação de recursos é a fragmentação do espectro. Ela é gerada devido à instalação e remoção das reservas na rede que pode ocasionar em uma distribuição de pequenos fragmentos não-contíguos no espectro que, dessa maneira, não podem ser usados para a acomodação de nenhuma reserva [Rosa et al. 2012], mesmo que se somados possuam banda maior do que a requerida. Uma estratégia muito comum para lidar com o problema de fragmentação é a execução periódica de algoritmos de desfragmentação [Zhang et al. 2013, Ju et al. 2012, Shakya e Cao 2013, Takagi et al. 2011] que são responsáveis, a um custo computacional adicional, de realocar as reservas existentes, criando assim novos espaços contíguos de espectro.

Este custo adicional de desfragmentação pode ser eliminado caso a escolha dos caminhos e da porção do espectro a ser alocada seja realizada levando-se em conta a potencial fragmentação gerada. Neste artigo, é proposto um algoritmo de RSA que visa mi-

nimizar a fragmentação do espectro óptico através da seleção do caminho mais adequado para cada requisição. Tal seleção é feita com base no grau de fragmentação de cada caminho da rede, atribuindo uma quantidade apropriada de *slots* de espectro de acordo com a demanda das requisições. O método proposto por este artigo é original e difere de outras propostas no tocante à escolha e à quantidade dos caminhos candidatos para atender cada requisição, além de considerar o estado atual do espectro ao escolher a melhor rota. A fim de avaliar a efetividade do algoritmo proposto, o mesmo é comparado com outros algoritmos de RSA existentes através de simulações. Os resultados encontrados mostram que o algoritmo proposto produz taxas de probabilidade de bloqueio e de fragmentação de espectro inferiores às produzidas por outros algoritmos de RSA, em diferentes topologias de rede.

O resto deste trabalho é organizado como se segue. A seção 2 descreve a arquitetura das redes ópticas elásticas. Na seção 3 os problemas de fragmentação e de RSA são brevemente revistos. A seção 4 apresenta de maneira detalhada o algoritmo de *RSA* proposto. Na seção 5 são descritas informações a respeito dos experimentos e os resultados encontrados. Por fim, a seção 6 conclui o artigo.

## 2. Redes Ópticas Elásticas

De acordo com [Zhang et al. 2013], diversas tecnologias para a utilização flexível do espectro óptico foram propostas. As principais são: *SLICE* (do inglês *Spectrum-Sliced Elastic Optical Path Network*), *FWDM* (do inglês *Flexible Optical Wavelength Division Multiplexing*) e redes ópticas com taxas de dados flexíveis. A Tabela 1 apresenta uma comparação entre as três tecnologias.

**Tabela 1. Comparação entre diferentes tecnologias propostas.**

<table>
<tr><th>TECNOLOGIA</th><th>TAXA DE DADOS</th><th>LARGURA DO ESPECTRO</th><th>MODULAÇÃO</th></tr>
<tr><td>SLICE</td><td rowspan="3">Flexível</td><td rowspan="2">Flexível</td><td>Múltiplas portadoras</td></tr>
<tr><td>FWDM</td><td rowspan="2">Única ou múltiplas portadoras</td></tr>
<tr><td>Taxas de dados flexíveis</td><td>Fixa</td></tr>
</table>

As redes *SLICE* possuem a capacidade de dividir os recursos espectrais em slots de frequência na forma de sub-portadoras através da modulação *OFDM* (do inglês *Orthogonal Frequency-Division Multiplexing*), permitindo múltiplos formatos de modulação e taxas de dados e espectro de tamanhos variados [Jinno, Takara e Kozicki 2009]. As redes *FWDM*, apesar de também proverem múltiplas taxas de dados e alocação flexível dos recursos espectrais, diferem das redes *SLICE* pelo fato de serem uma evolução das antigas redes *WDM* (do inglês *Wavelength Division Multiplexing*), permitindo modulações por uma única ou múltiplas portadoras. Por fim, as redes com taxas de dados flexíveis se caracterizam por uma maior flexibilidade com relação as taxas de dados, porém não permitem tamanhos variados de espectro.

O objetivo de uma rede *SLICE* é alocar uma demanda de largura de banda óptica de tamanho apropriado a um caminho óptico fim-a-fim. Diferente da largura de banda fixa dos caminhos ópticos nas redes *DWDM*, um caminho óptico em uma *SLICE* é capaz de se expandir ou se contrair, quando necessário, de acordo com a capacidade do tráfego ou a

demanda da requisição [Jinno et al. 2009]. A fim de garantir essa flexibilidade, a arquitetura das redes de caminhos elásticos é composta por duas tecnologias fundamentais: os *BVTs* (do inglês *Bandwidth Variable Transceivers*) e os *BV-WXCs* (do inglês *Bandwidth Variable Wavelength Cross-Connects*). Os *BVTs* são responsáveis por garantir uma granularidade flexível no domínio espectral, permitindo o ajuste dos recursos ópticos de acordo com a demanda necessária. Os *BV-WXCs*, por sua vez, são encarregados de estabelecer um caminho óptico fim-a-fim com largura de banda exata para acomodar os recursos espectrais necessários [Jinno, Takara e Sone 2011]. A Figura 1 apresenta arquitetura de uma rede óptica elástica.

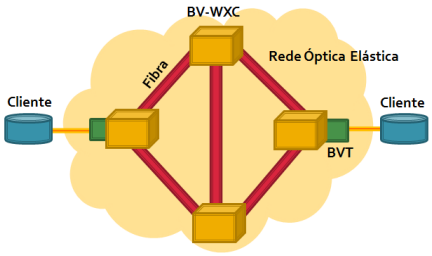



**Figura 1. Arquitetura de uma rede óptica elástica.**

Pode-se observar que os *BVTs*, responsáveis por ajustar os recursos ópticos conforme a demanda requisitada, ficam localizados nas bordas da rede e os *BV-WXCs*, responsáveis por estabelecer o caminho óptico fim-a-fim, ficam no núcleo da rede [Jinno et al. 2009].

Nas redes *SLICE*, a utilização da modulação OFDM possibilita o ajuste flexível da largura de banda requisitada ao mesmo tempo em que garante à transmissão uma alta eficiência espectral. O número de sub-portadoras e o formato de modulação utilizado pelas mesmas é ajustável de acordo com o volume do tráfego e o alcance óptico pretendido. Diferente das redes DWDM convencionais onde a largura do espectro é de *50 GHz*, nas redes *SLICE* cada slot de sub-portadora tem *12,5 GHz* [Yin et al. 2013], permitindo uma maior flexibilidade quanto aos formatos de modulação e a alocação de recursos. A escolha do formato de modulação a ser utilizado durante o estabelecimento da conexão é feita de acordo com a distância do caminho escolhido. A Figura 2 mostra como ocorre a divisão dos recursos espectrais nas redes *SLICE*, onde o espectro óptico é dividido em diversos *slots* de sub-portadoras, possibilitando a utilização de diferentes formatos de modulação.

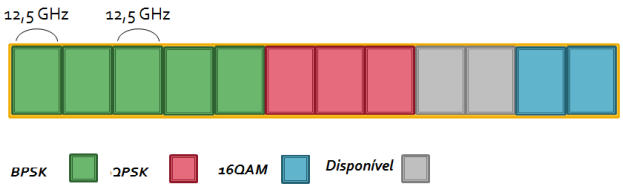



**Figura 2. Divisão dos recursos espectrais em uma rede *SLICE*.**

Nota-se que ao alocar uma determinada requisição de largura de banda, a quanti-

dade de *slots* varia conforme o formato de modulação utilizado. Na Tabela 2, são apresentados diferentes formatos de modulação e seus respectivos valores de alcance, conforme [Dahlfort et al. 2012]. É possível observar que quanto maior o alcance do formato de modulação, menor é a sua taxa de *bits* por símbolo e, consequentemente, maior é a quantidade de *slots* utilizados.

**Tabela 2. Tabela de comparação entre diferentes formatos de modulação.**

| FORMATO DE MODULAÇÃO | *BITS* POR SÍMBOLO (*bps*) | ALCANCE MÁXIMO (Km) |
|---|---|---|
| *BPSK* | 1.67 | 6,000 a 10,000 |
| *QPSK* | 3.33 | 2,000 a 5,000 |
| *16QAM* | 6.67 | Até 1,000 |

A Figura 3 apresenta uma comparação entre caminhos ópticos elásticos e caminhos ópticos rigídos. Ao contrário dos caminhos ópticos convencionais, os caminhos ópticos elásticos podem transmitir múltiplas taxas de dados ao mesmo tempo, seja através da segmentação de um único comprimento de onda (sub-comprimentos de onda), da agregação de múltiplos comprimentos de onda (super-comprimentos de onda) ou até mesmo adaptando uma conexão existente.

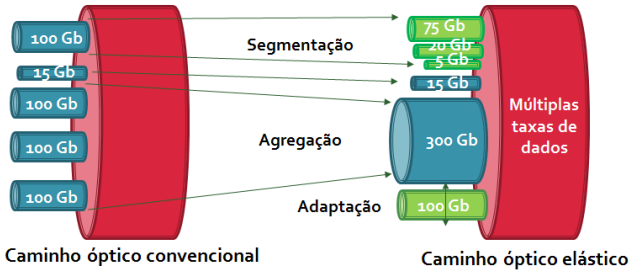



**Figura 3. Comparação entre caminhos ópticos convencionais e elásticos. Adaptada de [Jinno et al. 2009].**

Além de conseguir ajustar dinâmicamente os recursos espectrais de acordo com a demanda de largura de banda, outra grande vantagem das redes ópticas elásticas é a sua alta capacidade de restauração adaptativa em caso de sérios problemas na rede. Se por acaso ocorresse uma falha em alta escala, as rotas primárias e secundárias teriam que ser interrompidas e a utilização de rotas de desvio não seria capaz de prover os recursos espectrais suficientes para transportar a taxa de dados original e/ou o tamanho da rota de desvio poderia exceder o alcance óptico do sinal original. No entanto, a alocação adaptativa de espectro e a otimização de formatos de modulação e de largura de banda das redes ópticas elásticas garantiriam uma conexão mínima para os tráfegos de alta prioridade. [Sone et al. 2011]

## 3. Problemas de RSA e Fragmentação do Espectro

No âmbito das redes ópticas elásticas, os problemas de *RSA* e de Fragmentação de Espectro tem se tornado cada vez mais recorrentes. Diversos trabalhos a respeito de tais problemas tem sido propostos, visando, em sua maioria, encontrar heurísticas cada vez mais eficientes. A seguir, estes problemas serão brevemente revistos e alguns algoritmos de *RSA* existentes na literatura serão descritos.

### 3.1. O problema de Roteamento e Atribuição de Espectro

De acordo com [Shirazipourazad, Derakhshandeh e Sen 2013], o problema de Roteamento e Atribuição de Espectro pode ser informalmente definido como se segue: Dada uma certa topologia de rede e um conjunto de requisições com demandas variáveis (em termos de número de slots de espectro), encontrar uma rota para cada requisição e alocar uma quantidade de slots para a mesma, de acordo com a demanda requisitada, de forma que a parte utilizada do espectro seja minimizada. Existem duas diferentes versões para este problema: *RSA offline* e *RSA online*. No primeiro, todas as requisições são conhecidas antes de serem feitas a escolha do caminho e a atribuição do espectro. No segundo, as requisições chegam seguindo uma determinada ordem onde os caminhos e os recursos espectrais de cada requisição são determinados no momento da sua chegada.

Em [Rosa et al. 2012], o problema de *RSA* é tratado como uma evolução do problema de *RWA* (do inglês *Routing and Wavelength Assignment*). Ao invés de atribuir um ou mais comprimentos de onda (*RWA*), agora são atribuídos um ou mais *slots* de sub-portadoras, dependendo da demanda requisitada por uma conexão. Caso múltiplos *slots* sejam necessários, os mesmo devem respeitar o problema da restrição de contiguidade de espectro, onde *slots* de sub-portadoras de uma mesma conexão devem ser adjacentes entre si, e o problema da restrição de continuidade, onde a mesma quantidade de *slots* de sub-portadoras deve ser alocada no espectro de cada enlance pertencente ao caminho escolhido para a conexão.

#### 3.1.1. Algoritmos de RSA existentes

Na atual literatura é possível encontrar diversos trabalhos que tentam solucionar o problema de RSA. Os autores em [Wan et al. 2011], propuseram heurísticas na tentativa de solucionar o problema de RSA em um cenário dinâmico. Uma das heurísticas propostas foi o algoritmo do Menor Caminho Modificado (*MSP*), onde ao contrário do original em que é escolhido o caminho com o menor número de saltos, é computado o caminho com menor custo, da mesma forma que no algoritmo de *Dijkstra*. O funcionamento do algoritmo *MSP* pode ser observado na Figura 4 (a).

No algoritmo *MSP*, nem sempre o caminho com o menor número de saltos terá o menor custo. Na Figura 4 (a), por exemplo, o caminho com menor custo entre os nós 3 e 4 (caminho 3-5-4) possui mais saltos que o menor caminho original (caminho 3-4).

Outra heurística proposta por [Wan et al. 2011] foi o algoritmo *SCPVS* (do inglês *Spectrum Constraint Path Vector Searching*), que através de um algoritmo de busca em largura constrói uma árvore para representar os caminhos candidatos. A cada iteração, são adicionados à árvore os nós que possuem espectro disponível e os custos destas adições são computados. Ao encontrar o nó destino, é escolhido o caminho com menor custo entre os caminhos candidatos. A Figura 4 (a) mostra o funcionamento do algoritmo *SCPVS*.

O algoritmo *SCPVS*, através da busca em largura, encontra um caminho entre dois nós (por exemplo, nós 1 e 4) que possui espectro disponível para a demanda requisitada e com o menor custo entre as arestas.

Em [Wang, Cao e Pan 2011], foi proposto o algoritmo dos K-menores caminhos pré-computados (*P-SP*), onde os K-menores caminhos entre dois nós são previamente

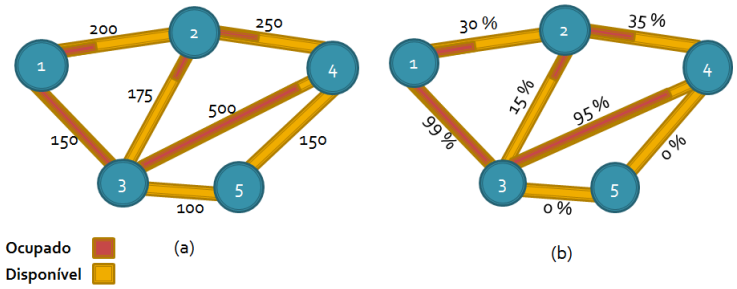



**Figura 4. Funcionamento dos algoritmos de *RSA*.**

definidos. É escolhido o caminho com menor custo entre os k-menores caminhos pré-computados que possua espectro disponível para atender a requisição. A Figura 4 (a) retrata o funcionamento do algoritmo *P-SP*. Como pode ser visto, entre os nós 1 e 4 existem 2-menores caminhos (k=2). É escolhido entre eles o caminho com menor custo entre as arestas. É importante notar que a função de custo utilizada pelos autores em [Wan et al. 2011, Wang, Cao e Pan 2011] foi a distância (em Km) entre os nós da rede.

Os autores em [Moura, Fonseca e Scaraficci 2013] propuseram dois algoritmos baseados em funções de custo que consideram o estado atual do espectro a fim de encontrar, para cada requisição, o melhor caminho entre os k-menores caminhos. O primeiro algoritmo, baseado na função de custo chamada *DF* (do inglês *Degree of Fragmentation)*, escolhe o caminho com menor grau de fragmentação externa, calculado através da Equação 1 (que será apresentada na próxima seção). O segundo algoritmo, baseado na função de custo *AP* (do inglês *Acceptance Prone*), escolhe o caminho com a maior capacidade de aceitação de requisições. A Figura 4 (b) mostra o funcionamento dos algoritmos propostos por [Moura, Fonseca e Scaraficci 2013].

No primeiro caso, como mostra a Figura 4 (b), é escolhido o caminho entre os menores caminhos que possui o menor grau de fragmentação, onde o grau de fragmentação do caminho é dado pelo somatório dos graus de fragmentação das arestas pertencentes a este caminho. No segundo, é escolhido o caminho entre os menores caminhos que possui a maior capacidade de aceitar requisições, por exemplo, se o enlace de determinado caminho tem 5 *slots* contíguos disponíveis, significa dizer que o mesmo possui a capacidade de aceitar requisições com demandas de 5, 4, 3, 2 e 1 *slots*. É importante notar que ao contrário dos outros algoritmos citados, os algoritmos *DF* e *AP* levam em consideração o estado do espectro óptico ao computar o caminho.

### 3.2. Fragmentação do Espectro

Em redes com tráfego de múltiplas taxas de dados, tais como as redes ópticas elásticas, a fragmentação do espectro é um problema que deve ser levado em consideração. Visto que o processo de estabelecimento e encerramento de conexões inevitavelmente cria pequenos fragmentos de espectro não-contíguos, grande parte das futuras requisições acabam não sendo atendidas. Dessa maneira, um dos grandes benefícios das redes elásticas, a eficiência espectral, certamente será reduzido [Wilson et al. 1995]. É bastante relevante notar que a fragmentação não está diretamente associada com a utilização do espectro, que em [Rosa et al. 2012] é definida como uma proporção do espectro utilizado pela quantidade total de recursos espectrais. O problema da fragmentação é, de fato, quando os

recursos espectrais disponíveis estão divididos em várias pequenas partes.

[Rosa et al. 2012] ainda apresenta algumas medidas de fragmentação. É sabido que existem diferentes alternativas para quantificar este problema. Uma delas é a fragmentação externa, proposta por [Wilson et al. 1995], denotada na Equação 1:

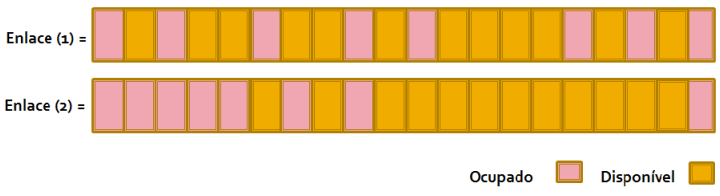
$$F_{ext} = 1 - \frac{maiorBlocoLivre}{totalLivre} \tag{1}$$

onde o *maior bloco livre* representa o número de slots do maior espaço contíguo livre, e o *total livre* é o número total de slots disponíveis. Na Equação 1, quando a fragmentação externa é próxima a um, ou seja, 100%, significa que o espaço disponível está todo dividido em pequenos fragmentos. No entanto, se o espectro está todo disponível, a fragmentação é 0%. Esta equação é válida dentro da hipótese de que sempre existirá um slot disponível no espectro óptico.

Outra alternativa de medir o grau de fragmentação do espectro leva em consideração o fato de que a fragmentação é influenciada pela demanda de largura de banda das conexões, ou seja, a fragmentação de espectro é maior quando as conexões requerem maiores recursos espectrais. Dessa forma, é possivel denotar a fragmentação como uma função do número de slots de espectro necessários para atender uma determinada requisição. Esta função é expressada na Equação 2.

$$F(c) = 1 - \frac{c \times Livre(c)}{totalLivre} \tag{2}$$

Na Equação 2, $c$ é o número de slots de espectro requisitados por uma certa conexão e *Livre(c)* é a função que retorna o número de requisições simultâneas de tamanho $c$ que podem ser satisfeitas. Para a Equação 2, qualquer tipo de conexão, representado de acordo com o número de slots requisitados, possui a sua própria medida de fragmentação. Assim como na Equação 1, a Equação 2 é válida somente dentro da hipótese de que sempre haverá um slot disponível no espectro. A Figura 5 exemplifica a utilização das Equações 1 e 2.




| | Fragmentação Externa (Eq. 1) | Fragmentação Nº de slots = 2 (Eq. 2) | Fragmentação Nº de slots = 3 (Eq. 2) |
|---|---|---|---|
| Enlace (1) | 66,7 % | 33,3 % | 75 % |
| Enlace (2) | 16,7 % | 16,7 % | 25 % |

**Figura 5. Fragmentação de acordo com as Equações 1 e 2.**

As duas formulações apresentadas nas Equações 1 e 2, apesar de simples, calculam de maneira bastante satisfatória a quantidade de espectro disponível que não está sendo utilizada de maneira eficiente devido a fragmentação.

## 4. Algoritmo de *RSA* com Minimização de Fragmentação

O algoritmo de RSA proposto neste artigo, *RSA-MF*, visa reduzir a fragmentação do espectro através da seleção de caminhos. O Algoritmo 1 detalha o funcionamento do algoritmo RSA-MF.

**Algoritmo 1** *RSA-MF*

**ENTRADA**

Uma requisição $R$ com um nó origem $O$, um nó destino $D$, uma demanda de largura de banda $L$ e um caminho com tamanho máximo $T$.

**SAÍDA**

A requisição $R$ é aceita caso seja encontrado um caminho $C$, de tamanho máximo $T$, com espectro disponível para alocar a demanda $L$. Se não houver, a requisição é bloqueada.

**RSA-MF**

***Offline***

1: Computa todos os caminhos simples (sem nós repetidos) de tamanho máximo $T$ entre cada par $O$-$D$ da rede.

***Online***

1: Verifica quais caminhos possuem espectro disponível para atender a demanda $L$.
2: Se não houver nenhum caminho com espectro disponível, a requisição é bloqueada.
3: Caso contrário, calcula o grau de fragmentação de cada caminho.
4: Escolhe o caminho com menor grau de fragmentação.
5: Atribui um formato de modulação ao caminho, de acordo com a distância do mesmo.
6: Calcula o número de *slots* e aloca o recurso.

O algoritmo *RSA-MF* é dividido em duas etapas: *Offline* e *Online*. Na fase *offline* são computados todos os caminhos simples (caminhos sem nós repetidos) entre todos os possíveis pares $O$-$D$ da rede. O tamanho máximo $T$ de um caminho é determinado com base na quantidade de nós da rede. O tamanho máximo é definido como sendo o número de saltos do maior caminho entre todos os menores caminhos da rede. Na fase *online*, com a chegada de uma nova requisição, todos os caminhos simples entre o nó $O$ e o nó $D$ da requisição, computados previamente na fase *offline*, são armazenados em um conjunto de caminhos candidatos. Então, o algoritmo verifica quais caminhos entre todos os caminhos candidatos possui espectro disponível suficiente para atender a demanda requisitada. Os caminhos que não possuem espectro suficiente são descartados. Após isso, é calculado o grau de fragmentação de cada caminho com base na Equação 2 apresentada na subseção 3.2, levando-se em conta a demanda requisitada. É escolhido o caminho com menor grau de fragmentação. Depois de computado o melhor caminho, um formato de modulação é atribuído ao mesmo baseado no seu comprimento e a quantidade de slots de sub-portadoras é calculada de acordo com a Equação 3.

$$N_{sub} = [\frac{C}{(b_m \times f_{slot})}] \quad (3)$$

onde C é o tamanho da taxa de dados requisitada, $b_m$ é a taxa de bits por símbolo do formato de modulação escolhido e $f_{slot}$ é a largura espectral do slot de frequência, respectivamente. Caso nenhum caminho entre os caminhos candidatos tenha espectro suficiente para atender a nova demanda, a requisição é bloqueada. Do contrário, o espectro é alocado e a conexão é estabelecida.

A Figura 6 mostra como é feita a escolha do melhor caminho no algoritmo *RSA-MF*. Como é possível observar, após descartar os caminhos candidatos que não possuem recursos disponíveis para atender a demanda requisitada, é escolhido o caminho com menor grau de fragmentação. Nota-se que nem sempre o caminho mais curto será escolhido, mas sim o caminho menos fragmentado. Assim como os algoritmos *DF* e *AP*, o *RSA-MF* também leva em consideração o estado atual do espectro ao escolher um caminho para uma determinada requisição. Entretanto, a quantidade de caminhos candidatos no *RSA-MF* é maior.

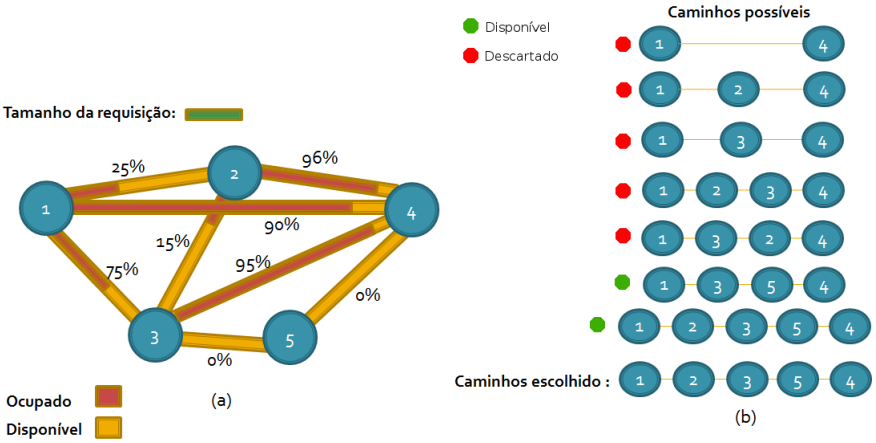



**Figura 6. Funcionamento do algoritmo *RSA-MF*.**

A complexidade computacional do algoritmo RSA-MF, em sua fase *Offline*, foi calculada como sendo da ordem O(|E| + |V| log|V|), onde E representa o número de arestas e V o número de vértices. Já na fase *Online*, a complexidade é $O(OD^2 + |S|.|P|)$, onde OD representa um determinado par origem-destino, P o conjunto de caminhos possíveis e S a quantidade de slots existentes.

## 5. Exemplos Numéricos

A fim de avaliar o desempenho do algoritmo *RSA-MF*, algumas simulações foram realizadas. Os resultados obtidos foram comparados com os dos algoritmos de *RSA* existentes citados na subseção 3.1.1. Em cada simulação, 10.000 requisições com origem e destino aleatórios foram geradas de acordo com um processo de *Poisson*. O tempo de duração de cada conexão seguiu uma distribuição exponencial média de 5 unidades de tempo. Foram utilizadas três topologias de rede nas simulações: a *NSFNET*, com 14 nós e 21 arestas, a *USA*, com 24 nós e 43 arestas e a topologia *Ring*, com 8 nós e 10 arestas, conforme mostra a Figura 7.

Os requisitos de largura de banda variaram entre 10 *Gbps*, 20 *Gbps*, 40 *Gbps*, 80 *Gbps*, 100 *Gbps*, 160 *Gbps*, 200 *Gbps*, 400 *Gbps*, 800 *Gbps* e 1 *Tbps*. O espectro de cada enlace da rede era composto por 400 *slots* de sub-portadoras, cada um com 12,5 *GHz*

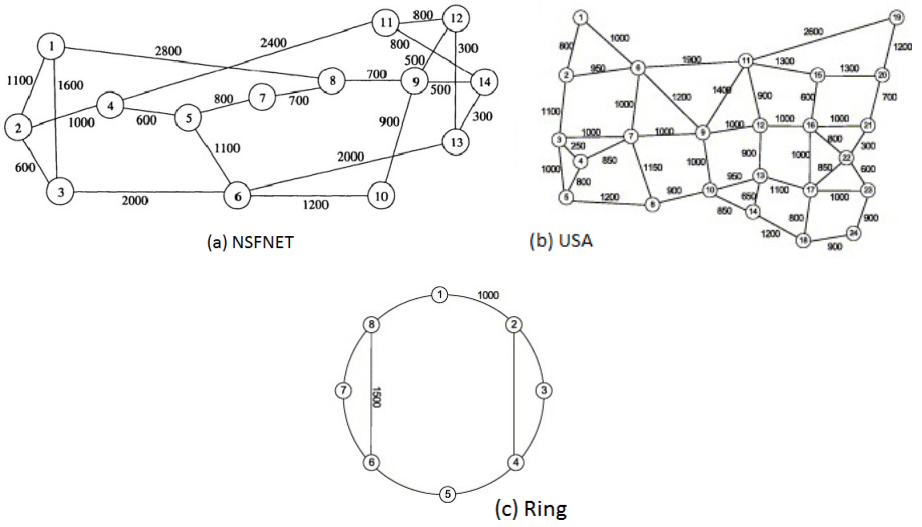



**Figura 7. Topologias utilizadas nas simulações: *NSFNET*, *USA* e *Ring*.**

[Yin et al. 2013]. Os formatos de modulação considerados e seus respectivos valores de alcance e taxas de *bits* foram baseados na Tabela 2 da seção 2. A política de atribuição de espectro utilizada nos algoritmos foi o *First-Fit*. Foram considerados para a carga da rede valores de 100 até 1000 *Erlangs*, aumentando 100 unidades em cada simulação. A métrica considerada para avaliar o desempenho do algoritmo de *RSA* proposto em comparação ao dos algoritmos existentes foi a probabilidade de bloqueio das requisições e a quantidade de espectro fragmentado. Todos os resultados obtidos foram gerados com intervalos de confiança de 95%. Devido a limitações de espaço e pelo fato de as topologias *NSFNET* e *USA* possuirem estruturas parecidas, serão apresentados somente os resultados obtidos para as topologias *NSFNET* e *Ring*. Entretanto, os resultados encontrados nas três topologias foram similares.

A Figura 8 apresenta a probabilidade de bloqueio obtida em função da carga da rede na topologias *NSFNET* e *Ring*. Para a topologia *NSFNET*, pode-se observar que apesar de quase todos os algoritmos começarem a bloquear as requisições a partir de 200 *Erlangs*, o algoritmo de *RSA* com Minimização de Fragmentação (*RSA-MF*) superou consideravelmente os outros algoritmos existentes, alcançando taxas de probabilidade de bloqueio aproximadamente quatro vezes menor que as alcançadas pelo algoritmo *AP*, que obteve o melhor desempenho entre os outros. Percebe-se que além de considerar o grau de fragmentação de cada caminho, permitir uma quantidade maior de caminhos também contribui consideravelmente para a redução da probabilidade de bloqueio.

Na topologia *Ring*, por sua vez, existe uma pequena diferença de desempenho entre os algoritmos de *RSA* existentes que consideram sempre os menores caminhos. Este fato é simples de ser observado, visto que só existem duas possíveis rotas na topologia. Desta forma, é mais uma vez provada a eficiência da função de custo utilizada pelo algoritmo *RSA-MF* e também o fato de não considerar apenas as menores rotas da rede. Os resultados obtidos pelo algoritmo *RSA-MF* indicam a importância de se analisar o estado atual do espectro e todos os caminhos candidatos. O algoritmo proposto superou os outros existentes com taxas de bloqueio aproximadamente três vezes menor.

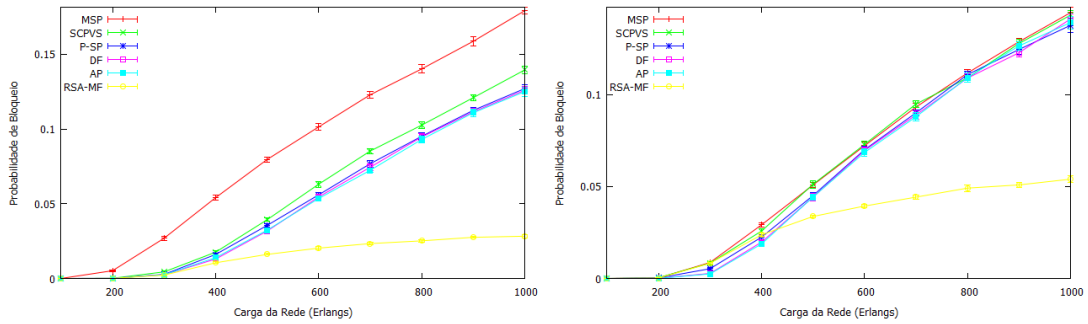



**Figura 8. Probabilidade de bloqueio nas topologias *NSFNET* e *Ring*.**

A Figura 9 apresenta a medida de fragmentação da rede ao final de cada simulação em função das diferentes cargas utilizadas. Como os algoritmos *MSP*, *SCPVS* e *P-SP* não analisam o estado do espectro ao determinar uma rota para cada requisição, eles não foram considerados.

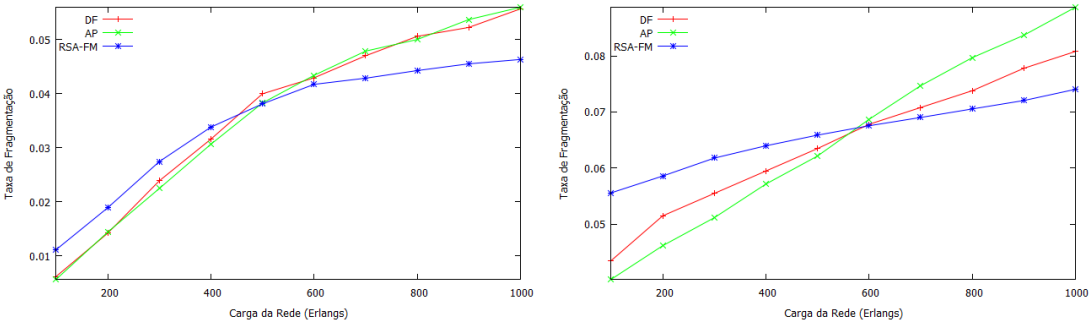



**Figura 9. Fragmentação da rede nas topologias *NSFNET* e *Ring*.**

Nota-se que o algoritmo *RSA-MF* apresenta, para cargas de rede mais baixas, desempenho inferior aos algoritmos *DF* e *AP*. Contudo, obtém melhor desempenho quando são submetidos à cargas mais elevadas (a partir de 500 *Erlangs* na topologia *NSFNET* e 600 *Erlangs* na topologia *Ring*). Isto ocorre pelo fato de que o algoritmo *RSA-MF*, por possuir um conjunto maior de caminhos candidatos, acaba distribuindo os recursos espectrais de maneira mais homogênea a medida que a carga da rede aumenta, diminuindo a quantidade de espectro fragmentado.

## 6. Conclusão

Neste artigo foram abordados dois problemas de grande relevância em Redes Ópticas Elásticas: os problemas de fragmentação e de Roteamento e Atribuição de Espectro. Discutiu-se a respeito de algoritmos de RSA existentes e foi proposto um algoritmo de RSA original com Minimização de Fragmentação. Resultados de simulações mostraram que o algoritmo proposto superou consideravelmente os algoritmos de RSA existentes em termos de probabilidade de bloqueio, onde obteve valores aproximadamente quatro vezes menores, e em relação à quantidade de espectro fragmentado quando submetido à cargas de rede mais elevadas. Desta maneira, foi demonstrada claramente a importância de se analisar o estado atual do espectro ao escolher o melhor caminho, assim como considerar um conjunto maior de caminhos candidatos para cada requisição. Como trabalhos futuros,

planeja-se desenvolver novas propostas para a minimização da fragmentação no espectro óptico juntamente com políticas de atribuição mais eficientes.

## Referências

Zhang, G. et al. (2013). *A Survey on OFDM-Based Elastic Core Optical Networking*, Communications Surveys & Tutorials, IEEE, Vol.15, Num.1, First Quarter, 2013.

Jinno, M., Takara, H. e Kozicki, B. (2009). *Concept and Enabling Technologies of Spectrum-Sliced Elastic Optical Path Network (SLICE)*. Proc. Communications and Photonics Conference and Exhibition (ACP), November 2009, Paper FO2.

Jinno, M. et al. (2009). *Spectrum-Efficient and Scalable Elastic Optical Path Network: Architecture, Benefits, and Enabling Technologies*, Communications Magazine, IEEE Vol.47 , Num. 11, 2009.

Shirazipourazad, S., Derakhshandeh, Z. e Sen, A. (2013) *Analysis of On-line Routing and Spectrum Allocation in Spectrum-sliced Optical Networks*, International Conference on Communications, 2013.

Rosa, A. et al. (2012) *Spectrum Allocation Policy Modeling for Elastic Optical Networks*, 9th International Conference on High Capacity Optical Networks and Enabling Technologies (HONET), 2012.

Wilson, P. R. et al. (1995) *Dynamic Storage Allocation: A Survey and Critical Review*, in Proccedings of the International Workshop on Memory Management, ser. IWMM'95. London, UK, UK: Springer-Verlag, 1995, pp. 1-116.

Chen, X., Zhong, Y. e Jukan, A. (2013) *Multipath Routing in Elastic Optical Networks with Distance-adaptive Modulation Formats*, in the Proceedings of IEEE ICC'2013, Budapest, Hungary, June 2013.

Christodoulopoulos, K., Tomkos, I. e Varvarigos, E. (2010) *Routing and spectrum allocation in OFDM-based optical networks with elastic bandwidth allocation*, in 2010 IEEE Global Telecommunications Conference (GLOBECOM), Dec. 2010, pp. 1-6.

Wang, Y., Cao, X. e Pan, Y. (2011) *A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks*, in Proc. IEEE INFOCOM, Apr. 2011, pp. 1503-1511.

Jinno, M. et al. (2010) *Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network*, IEEE Communications Magazine, vol. 48, no. 8, pp. 138-145, august 2010.

Zhang, M. et al. (2013) *Bandwidth Defragmentation in Dynamic Elastic Optical Networks with Minimum Traffic Disruptions*, in the Proceedings of IEEE ICC'2013, Budapest, Hungary, June 2013.

Wan, X. et al. (2011) *Dynamic routing and spectrum assignment in flexible optical path networks*, in Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference, pp. 1-3, march 2011.

Zhang, G. et al. (2012) *A survey on ofdm-based elastic core optical networking*, in IEEE COMMUNICATIONS SURVEYS & TUTORIALS, vol. Volume:PP, Issue: 99, pp. 1-23, 2012.

Yin, Y. et al. (2013) *Fragmentation-Aware Routing, Modulation and Spectrum Assignment Algorithms in Elastic Optical Networks*, in Optical Fiber Communication Conference, Anaheim, California United States, March 17-21, 2013. Defragmentation (OW3A)

Wang, R. e Mukherjee, B. (2013) *Provisioning in elastic optical networks with non-disruptive defragmentation* in Optical Network Design and Modeling (ONDM), 2013 17th International Conference on, pp. 223-228, 2013.

Moura, P. M., Fonseca, N. L. S. e Scaraficci, R. A. (2013) *Fragmentation Aware Routing and Spectrum Assignment Algorithm*, Relatório técnico. Instituto de Computação, Universidade Estadual de Campinas, outubro de 2013.

Gerstel, O. et al. (2012) *Elastic optical networking: a new dawn for the optical layer?*, Communications Magazine, IEEE (Volume:50 , Issue: 2 ), 2012.

Ju, W. et al. (2012) *Dynamic adaptive spectrum defragmentation scheme in elastic optical path networks* , in 17th Opto-Electronics and Communications Conference (OECC), 2012.

Shakya, S. e Cao, X. (2013) *Spectral Defragmentation in Elastic Optical Path Networks using Independent Sets* , in Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013.

Takagi, T. et al. (2011) *Disruption Minimized Spectrum Defragmentation in Elastic Optical Path Networks that Adopt Distance Adaptive Modulation* ,in Optical Communication (ECOC), 2011 37th European Conference and Exhibition on, 2011.

Sone, Y. et al. (2011) *Bandwidth Squeezed Restoration in Spectrum-Sliced Elastic Optical Path Networks (SLICE)* , J. Optical Communications and Networking, Vol. 3, No. 3, PP. 223-233. 2011.

Jinno, M., Takara, H. e Sone, Y. (2011) *Elastic optical path networking: Enhancing network capacity and disaster survivability toward 1 Tbps era* , in 16th OptoeElectronics and Communications Conference (OECC), 2011.

Dahlfort, S. et al. (2012) *Split Spectrum Approach to Elastic Optical Networking* in 38th European Conference and Exposition on Optical Communications, 2012, p. Tu.3.D.4.

Santos, A. et al. (2013) *Adaptação do Algoritmo BSR para Redes Ópticas SLICE.* in Simpósio Brasileiro de Redes de Computadores, SBRC, 2013.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Trilha Principal do SBRC 2014

## *Sessão Técnica 21*

## *Redes Veiculares II*

# Um Algoritmo Autônomo para Disseminação de Informações em Redes Veiculares

**Rodolfo I. Meneguette[1], Guilherme Maia[2], Edmundo R. Madeira[1], Alex R. Pinto[3], Antonio A. F. Loureiro[2], Leandro A. Villas[1]**

[1]Instituto de Computação – UNICAMP
[2]Departamento de Ciência da Computação – UFMG
[3]Departamento de Ciência da Computação – UNESP
ripolito@ic.unicamp.br, jgmm@dcc.ufmg.br, edmundo@ic.unicamp.br
arpinto@ibilce.unesp.br, loureiro@dcc.ufmg.br, leandro@ic.unicamp.br

***Abstract.*** *Vehicular Ad Hoc Networks (VANET) provide safety, traffic efficiency and user comfort for drivers and passengers. To reach these goals many envisaged applications for VANETs will rely on data dissemination, turning this task into an essential service in this network. However, due to some inherent VANET characteristics, such as intermittent connectivity, highly dynamic topology, and diverse network densities, data dissemination is a challenging activity. Although data dissemination has been extensively studied in the literature, existing solutions do not effectively address the broadcast storm and network partition problems when considered together. To deal with both problems, we propose a network partition-aware geographical data dissemination protocol for urban environments, which eliminates the broadcast storm problem and maximizes the data dissemination capability across network partitions with short delays and low overhead. The proposed algorithm is able to autonomically adjust message transmissions based on an autonomic computing approach. Simulation results show that our proposed approach provides good efficiency with respect to the coverage of the event and delay.*

***Resumo.*** *As Redes Veiculares Ad Hoc (VANETs) proporcionam segurança, maior eficiência de tráfego e conforto para os motoristas e passageiros nesse tipo de rede. Para tanto, muitas aplicações de redes veiculares utilizam o processo de disseminação de dados, transformando-o em um serviço essencial. No entanto, devido a algumas características inerentes às VANETs como conectividade intermitente, topologias altamente dinâmicas e mudanças constantes de densidade, tornam a disseminação de dados um serviço desafiador. Embora a disseminação de dados tenha sido amplamente estudada na literatura, as soluções existentes não tratam de forma efetiva e conjunta os problemas da tempestade de broadcast e partições na rede. Diante deste cenário, para lidar com ambos os problemas, propomos um algoritmo de disseminação de dados geográfico ciente de partições na rede para ambientes urbanos. Esta solução elimina o problema da tempestade de broadcast e maximiza a capacidade de disseminação de dados em redes com frequentes desconexões, ao custo de um baixo atraso e um baixo overhead. O algoritmo proposto, o qual é baseado em técnicas de computação autonômica, é capaz de auto-ajustar as transmissões de mensagens. Resultados de simulações mostram que a abordagem proposta*

*é eficiente em termos de cobertura, taxa de envio e atraso, além de manter o padrão de envio e recebimento em diferentes densidades de rede.*

## 1. Introdução

Com os avanços nas tecnologias de rede sem fio e o número crescente de veículos nas ruas e rodovias, não há apenas uma necessidade, mas também um potencial para desenvolver e implantar sistemas de transporte inteligentes (ITS) em redes veiculares *ad hoc* (VANETs). Em VANETs, os veículos são equipados com sensores embarcados, unidades de processamento e interfaces de comunicação sem fio para comunicar com outros veículos e com os *roadside units* (RSUs), permitindo a criação de uma rede "espontânea" enquanto os veículos locomovem-se [Li and Wang 2007, Toor et al. 2008, Wang and Li 2009, Maia et al. 2013a].

Existe um grande interesse no desenvolvimento de aplicações para VANETs, tanto por parte da indústria quanto da academia. As aplicações vislumbradas variam desde segurança rodoviária e de propagação de informações de trânsito até entretenimento. No entanto, VANETs enfrentam uma série de desafios em termos de disseminação de dados, devido às variações na densidade de veículos e às frequentes mudanças na topologia de rede induzidas pela alta mobilidade de veículos e comunicações de curto alcance. A disseminação é uma função vital, pois neste tipo de rede, uma mensagem pode ser de grande interesse para os veículos em uma dada região. Por exemplo, avisos para evitar colisões e pós-acidentes exigem uma disseminação de dados eficiente e confiável. Isso particularmente quando as distâncias entre o veículo emissor e os veículos receptores são maiores do que o raio de comunicação. Esses desafios limitam o uso de protocolos de disseminação de dados existentes para as redes móveis. Novos protocolos devem ser concebidos, levando-se em consideração as características intrínsecas das redes veiculares.

Protocolos para disseminação de dados em VANETS devem considerar dois problemas chave. O primeiro, conhecido como *broadcast storm*, ocorre quando vários veículos tentam transmitir simultaneamente. Isso causa alto tráfego de dados, congestionamentos na rede, colisões entre pacotes e até mesmo a interrupção de serviço na camada de controle de acesso ao meio (MAC) [Williams and Camp 2002]. O segundo, conhecido como partições na rede, ocorre quando o número de veículos na área de interesse não é suficiente para realizar a disseminação de dados entre os grupos de veículos próximos um do outro [Spyropoulos et al. 2008]. Neste cenário, se o veículo não estiver ciente de que a rede está desconectada, quando recebe uma nova mensagem, ele simplesmente irá retransmiti-la em *broadcast* e descartá-la na sequência. Uma vez que não há nenhum outro veículo dentro da área de interesse para receber essa mensagem, ela será simplesmente perdida. O problema de partições na rede é muito comum em uma rede VANET devido à distribuição esparsa e aleatória entre os veículos. Tal problema impõe grandes desafios para a disseminação de dados já que as mensagens não podem ser facilmente reencaminhadas entre as partições.

A literatura apresenta diversas soluções de disseminação de dados em VANETs [Korkmaz et al. 2004, Durresi et al. 2005, Joshi et al. 2007, Chen et al. 2009, Tonguz et al. 2010, Viriyasitavat et al. 2010, Bakhouya et al. 2011, Ros et al. 2012, Maia et al. 2013b]. Entretanto, algumas dessas soluções lidam com o problema de *broadcast storm* ou o problema de partições na rede e não os dois em conjunto. Além

disso, os algoritmos que lidam com as duas limitações não conseguem manter seu padrão de desempenho quando submetidas a condições de tráfegos distintas, ou seja não mantêm seu desempenho quando há mudança de alta densidade de tráfego para baixa ou vice-versa. Argumentamos que os algoritmos devem manter-se eficientes mesmo com flutuações na densidade de tráfego, ou seja, o algoritmo deve manter um bom desempenho mesmo em condições que proporcionem um alto grau de particionamento da rede, baixa densidade de tráfego, bem como em condições de tráfego mais intenso onde há pouca partição na rede.

Com isso em mente, propomos uma nova solução baseada em computação autonômica para realizar a disseminação de dados em ambientes urbanos com diversas condições de tráfego. A computação autonômica é inspirada em sistemas biológicos com capacidade de se auto-gerenciar. O auto-gerenciamento é composto por diversas características identificadas como auto-*: auto-configuração, auto-otimização, auto-cura e auto-proteção. O uso da computação autonômica no contexto de disseminação de dados em Redes Veiculares visa prover um sistema de decisão proativo. O mecanismo proposto oferece auto-cura durante o processo de disseminação. Além disso, oferece auto-organização no envio dos dados, uma vez que os veículos decidem se transmite ou não durante o processo de disseminação de dados. Bem como auto-ajuste com relação a realização da tarefa de disseminação dos dados. Por fim, utilizamos mecanismos para detectar e lidar com o particionamento de rede através da probabilidade do nó propagar ou não os dados que estão sendo disseminados na rede. Os principais objetivos do nosso algoritmo são diminuir o *broadcast storm* e maximizar a capacidade de disseminar dados quando a rede estiver particionada, oferecendo uma alta taxa de entrega e baixa latência.

O restante deste artigo é organizado da seguinte forma. Na próxima seção, mostramos uma visão geral das principais abordagens existentes para a disseminação de dados em VANETs. A nossa proposta de um algoritmo de disseminação é descrita na seção 3, enquanto que uma avaliação detalhada do desempenho e dos resultados de simulação são apresentados na seção 4. Finalmente, a seção 5 apresenta as conclusões e trabalhos futuros.

## 2. Trabalhos Relacionados

Muitos protocolos de disseminação de dados têm sido propostos na literatura, mas eles se concentram em redes conexas ou redes desconexas. Além disso, as soluções que abordam ambos os problemas contam com o apoio de uma infraestrutura de comunicação preestabelecida ou de uma combinação de GPS e informações geográficas.

O mecanismo mais simples para a realização de disseminação é o *flooding* (inundação), o que significa que os dados são enviados a todos os veículos vizinhos, que por sua vez irão armazenar e transmiti-los para os seus vizinhos. Essa abordagem garante um bom desempenho para redes esparsas, porém enfrenta o problema de *broadcast storm* à medida que a densidade da rede aumenta. Em vista disso, as propostas atuais para a disseminação de dados focam principalmente na forma como os pacotes serão encaminhados. Várias propostas podem ser encontradas para isso, baseadas em diferentes estratégias como posição, dados estatísticos, distância, topologia local, temporizadores e mapas.

Uma abordagem descentralizada e adaptável para a disseminação de informação em VANETs, chamada *Adaptive approach for Information Dissemination* (AID), foi pro-

posta por Bakhouya e colaboradores [Bakhouya et al. 2011]. Nessa abordagem, o veículo decide se deve ou não transmitir um pacote, dependendo do número de vezes que ele recebe o mesmo pacote de dados em um determinado período de tempo. Em redes densas, por exemplo, vários veículos podem decidir descartar um pacote, uma vez que ele já foi encaminhado por diversos veículos, e isso reduz o problema de *broadcast storm*. No entanto, o protocolo AID não trata o problema de partição da rede.

Kim e al. [Kim et al. 2008] propuseram o *Distance Based Relay Selection* (DBRS) que é uma estratégia simples e eficiente usada para a disseminação de informação em uma rede. Ao receber um pacote, o veículo o mantém por um intervalo de tempo que é proporcional ao inverso da distância até o veículo transmissor. Assim, para disseminar informações, é preferível utilizar veículos mais distantes do veículo transmissor. Os veículos que ouvem a transmissão do pacote, sendo que o mesmo já está programado para ser transmitido, cancelam a sua transmissão para evitar o *broadcast storm*. Esta abordagem é eficaz no sentido de que ela pode evitar o *broadcast storm*, mas pode trazer dois problemas: (i) o atraso pode ser elevado, pois não há garantia da existência de veículos próximos ao raio de comunicação (aqueles que transmitem com menor atraso) e (ii) a cobertura pode ser reduzida, posto que os veículos irão cancelar as suas transmissões de forma indiscriminada ao ouvir a transmissão do mesmo pacote.

Villas et al. [Villas et al. 2012] propuseram o *GEographical Data Dissemination for Alert Information* (GEDDAI) que realiza a disseminação de dados dentro de uma área de interesse (AOI). Quando ocorre um evento, o veículo que o detecta começa a disseminar as informações relevantes dentro da área de interesse. Durante esse processo, os veículos no interior da AOI continuam o processo de disseminação de dados com o intuito de diminuir o *broadcast storm*. Subsequentemente, o veículo fonte cria uma mensagem com informações sobre o evento e transmite a mesma para os seus vizinhos. Ao receber tal mensagem, cada veículo verifica se está dentro da AOI. Caso esta condição não seja cumprida, o veículo descarta a mensagem recebida. Embora o algoritmo diminua o problema de *broadcast storm* em redes menos densas, ou seja, com muitas partições na rede, esse protocolo apresenta uma baixa cobertura do evento.

A nossa proposta é utilizar computação autonômica para lidar com o particionamento na rede. Além disso, utilizamos a região da AOI para priorizar veículos situados no interior dessa região.

## 3. ALgoritmo Autônomo para Disseminação de INformações em Redes Veiculares

O algoritmo proposto, denominado ALgoritmo Autônomo para Disseminação De INformações em Redes Veiculares – ALADDIN – tem como principal objetivo diminuir o número de retransmissões, sem comprometer a cobertura. Usamos dois mecanismos para evitar o *broadcast storm* e maximizar a capacidade de disseminação de informação entre partições de rede com baixo custo, pequeno atraso e alta cobertura. No primeiro mecanismo, definimos uma região chamada de zona de preferência [Villas et al. 2013], uma região que atribui uma prioridade aos veículos para a realização de uma retransmissão, ou seja, veículos que estiverem dentro dessa região terão maior prioridade de retransmitir. O segundo mecanismo, é uma técnica autonômica que simulará a função de um *store-carry-forward* quando houver um particionamento na rede, ou seja, o veículo armazenará

a informação enquanto sua eficiência de propagação estiver baixa ou até que o número de saltos do pacote esteja além de um limiar. A eficiência da propagação além de simular um *store-carry-forward*, auxiliará na decisão de retransmitir ou não a informação. A seguir, descreveremos as características de cada mecanismo.

- *Zona de Preferência*: é uma área onde os veículos são considerados os mais adequados para continuar a disseminação de informação. Isto significa que, entre todos os veículos que receberam as informações, a transmissão de um único veículo dentro da zona de preferência é suficiente para realizar a disseminação de informação eficientemente. Os veículos localizados dentro das zonas de preferência são os mais indicados para disseminar a informação, assim como para alcançar um maior número de vizinhos não alcançados pelo último transmissor.
- *Computação Autonômica*: é um conceito usado para descrever máquinas que são capazes de realizar uma determinada atividade, sem a necessidade da intervenção humana [Huebscher et al. 2008]. No contexto da nossa proposta, o algoritmo decidirá se irá retransmitir a informação ou não baseado em sua eficiência de propagação. Essa eficiência de propagação indicará estatisticamente o quão eficiente o veículo está na atividade de disseminação da informação, assim o veículo saberá o momento de retransmitir ou manter a informação.

Portanto, as características de computação autonômica e a zona de preferência são utilizadas para garantir a entrega de informações em diversas condições de tráfego e lidar com o *broadcast storm*. Além disso, VANETs introduzem uma necessidade explícita de auto-otimização, devido à sua topologia de rede ser altamente dinâmica [Meneguette et al. 2013]. A seguir, descrevemos como incorporamos o conceito de zona de preferência com a técnica de computação autonômica, a fim de assegurar auto-otimização e adaptabilidade ao algoritmo proposto.

A zona de preferência tem por objetivo tentar eliminar o problema de *broadcast storm*, pois os veículos dentro da zona de preferência transmitem os pacotes com o menor atraso e abortam as transmissões do mesmo pacote de outros veículos que não estão dentro da zona de preferência. Dentre os veículos que estão dentro da zona de preferência, o que estiver mais distante do veículo transmissor irá transmitir primeiro e abortar as transmissões desnecessárias dos outros veículos que estão dentro da zona de preferência e mais próximos do transmissor. No caso de não haver veículos dentro da zona de preferência, o veículo mais distante de cada quadrante irá retransmitir a informação. Por uma questão de simplificação, a forma da área de comunicação é considerada circular, onde o veículo transmissor se encontra no centro do círculo. A área de comunicação é decomposta em quatro quadrantes e, em cada quadrante, uma subárea do quadrante é definida como uma zona de preferência, como podemos observar na figura **??**. Através das zonas de preferência, podemos impedir a transmissão da mesma informação dos veículos que estão próximos uns dos outros e pertençam a quadrantes diferentes, assim evitando que as mensagens alcancem áreas similares, o que seria uma redundância desnecessária.

Embora a abordagem da zona de preferência auxilie no controle do *broadcast storm*, essa abordagem não lida com o problema de partição de rede. Desta forma, para resolver esse problema, usamos uma técnica autonômica que calcula a probabilidade de um veículo disseminar ou não a informação. Esta decisão baseia-se na eficiência de propagação (equação 1) e na localização geográfica dos veículos. O principal objetivo da

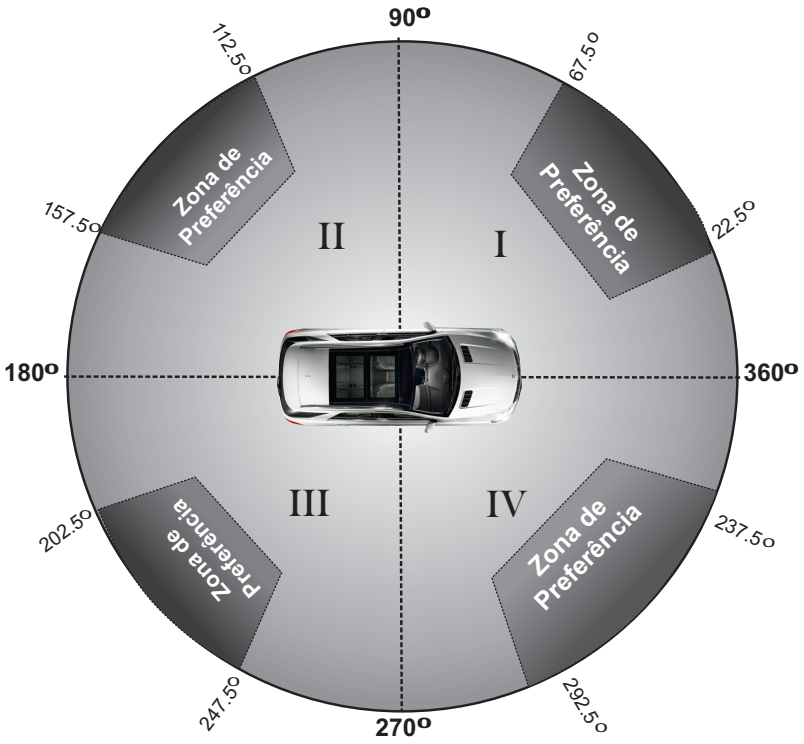



**Figura 1. Zona de preferência**

técnica autonômica é decidir se uma informação será enviada ou não. Essa decisão é baseada na atual eficiência de propagação de cada veículo (eficiência é a relação entre o número de mensagens transmitidas e o número de *beacons* recebidos).

A eficiência de propagação é usada para controlar a transmissão de cada veículo. Assim, se essa eficiência estiver abaixo de um limite aceitável, o veículo irá retransmitir a informação, por outro lado, se a eficiência estiver acima do limiar, o veículo não dissemina a informação. A eficiência é calculada em cada *round*, ou seja, ao longo do tempo o veículo verifica a eficiência da sua entrega da informação.

$$\text{Eficiência de Propagação} = \text{Transmissão}/Beacons \tag{1}$$

Além da eficiência de propagação, outro fator que pode influenciar na retransmissão da informação é o tempo de vida da informação. O algoritmo 1 mostra o cálculo da eficiência de propagação usado para decidir se o veículo deve ou não retransmitir a informação. Com base nesse mecanismo, o veículo armazena a informação até que a sua eficiência seja superior a um limiar ou até que um dos outros parâmetros sejam alcançados, permitindo assim, a entrega da informação, mesmo na presença de particionamento de rede.

No ALADDIN, quando ocorre um evento, o veículo cria uma mensagem que será enviada para todos os veículos dentro da área de interesse (AOI). Após a recepção da informação, cada veículo verifica se está dentro da AOI. Se esta condição não for atendida, o veículo descarta a informação recebida. Caso contrário, o veículo verifica se está dentro da zona de preferência do veículo transmissor, calcula o tempo de espera e agenda a

retransmissão da informação. O algoritmo 2 mostra como um veículo verifica se está dentro da zona de preferência e como é calculado o tempo de espera para agendar a retransmissão.

Sempre que um possível particionamento da rede for detectado por um veículo, que é indicado pela ausência de *beacons*, o veículo espera até receber novos *beacons*. Quando dois grupos de veículos estão desconectados, ou seja, os veículos de um determinado grupo irão receber *beacons* uns dos outros, mas não dos veículos que estão no outro grupo, o algoritmo não conseguirá detectar esse tipo de particionamento. Uma vez que o veículo detecta uma nova ligação através dos *beacons*. Porém, quando um dos veículos recebe um novo *beacon* do outro grupo, o veículo que recebeu o *beacon* irá verificar se é capaz de realizar a disseminação da informação ou não de acordo com a sua eficiência de propagação. Por exemplo, suponhamos que o veículo fonte (S) e o outro veículo (A) pertencem a uma partição (P1) e os veículos restantes da área de interesse estão numa segunda partição da rede (P2). Nesse caso, S e A, que estão em P1, não vão perceber um particionamento na rede, uma vez que S recebe *beacons* de A, e A recebe *beacons* de S, assim, tanto S como A não percebem qualquer partição na rede. No entanto, se S ou A receber um *beacon* de um veículo que estava na partição P2, o mesmo retoma o processo de disseminação e verifica se é capaz de realizar a disseminação dos dados ou não de acordo com a sua eficiência de propagação. Se o limiar de eficiência de propagação for menor do que 50%, indica que o veículo é capaz de realizar a disseminação. Além disso, o veículo que recebeu o *beacon* também verifica se o veículo transmissor está dentro da área de interesse e se já recebeu as informações que estão sendo divulgadas. Caso o veículo esteja dentro da área de interesse e ainda não recebeu a informação, o veículo que tem a informação realiza a divulgação, caso contrário a disseminação não é realizada. O ALADDIN usa o número de saltos para limitar a quantidade de disseminação de infor-

**Algorithm 1** Cálculo dos critérios de eficiência utilizados para decidir se o veículo deve ou não propagar mensagens

1: $p \leftarrow$ *getNumerodepacotes*()
2: $b \leftarrow$ *getNumerodeBeacons*()
3: $ef \leftarrow p/b$
4: **if** ($ef >$ *limiar_baixo*) **or** ($ef <$ *limiar_alto*) **then**
5: **return true**
6: **else**
7: **return false**
8: **end if**

**Algorithm 2** Verifica se o veículo receptor está dentro da zona de preferência e calcula o tempo de espera para agendar a transmissão

1: *distOrigem* $\leftarrow \sqrt{(s_x - r_x)^2 + (s_y - r_y)^2}$
2: *atrasoPadrao* $\leftarrow 0.01 \times (\frac{\textit{distparaorigem}}{\textit{raiodecomunicacao}})$
3: **if** (Está dentro da zona de preferência) **then**
4: *Atraso* $\leftarrow$ *atrasoPadrao* $+ Random(0, 0.01)$
5: **else**
6: *Atraso* $\leftarrow$ *atrasoPadrao* $+ Random(0.02, 0.05)$
7: **end if**

mação, para este trabalho. O algoritmo 3 descreve como a propagação da informação é realizada.

**Algorithm 3** Verifica se um veículo é capaz de realizar a propagação

```
1: while Não recebeu beacons do
2:     esperar
3: end while
4: if (efPropagate = true)&& (estadentrodoROI(beacon.sender)) then
5:     for cada mensagem do
6:         if (ttlMesssage ≠ 0) and (saltos < limiar) then
7:             sendMessage()
8:         else
9:             delMessage()
10:        end if
11:    end for
12: end if
```

## 4. Avaliação de Desempenho

Nesta seção, apresentamos a avaliação de desempenho do algoritmo ALADDIN, que foi comparado com quatro outros algoritmos para disseminação de dados em redes veiculares: AID, DBRS, GEDDAI e *flooding*.

### 4.1. Metodologia

A avaliação foi realizada através de simulações utilizando o OMNeT++ Network Simulation Framework v.4.1 [Varga and Hornig 2008], *vehicles in network simulation framework* (Veins) [Veins ], e SUMO-Simulation of Urban MObility [Behrisch et al. 2011]. Neste estudo, um cenário de mobilidade realístico é utilizado para realizar as simulações. O cenário gerado com o auxílio da ferramenta SUMO é uma topologia em grade de $2000 \times 2000 m^2$ com blocos de tamanho de $50 \times 100 m^2$, como representado na figura 2.

Nos cenários experimentais, o tráfego de veículos é gerado aleatoriamente. A velocidade dos veículos pode variar de 30 até 50 quilômetros por hora, as condições de tráfegos são 100, 150, 200, 250, 300 veículos/km$^2$ e a área de interesse tem um raio de 2000 m. Os veículos se movem ao longo da grade de ruas horizontais e verticais no mapa. Cada linha representa uma estrada de pista única e o movimento dos veículos ocorre nas direções norte, leste, oeste e sul. Em um cruzamento, os veículos escolhem virar ou manter se movendo na mesma direção com igual probabilidade de 50%.

O tempo de simulação utilizado é de 100 s, que é suficientemente longo para avaliar os algoritmos de disseminação através das variações da velocidade e da densidade dos veículos. Durante a simulação, um veículo arbitrário é escolhido para causar um acidente e, após a colisão, os veículos envolvidos divulgam as informações sobre o acidente para todos os veículos dentro da área de interesse. Os veículos utilizam o padrão IEEE 802.11p como protocolo de comunicação nas camadas física e de enlace. Nós usamos um modelo de propagação de rádio *two-ray ground* e o raio de transmissão é 200 m. Cada simulação foi executada 50 vezes. Em todos os resultados, as curvas representam os valores médios, enquanto que as barras de erros representam o intervalo de confiança de 95%. Os

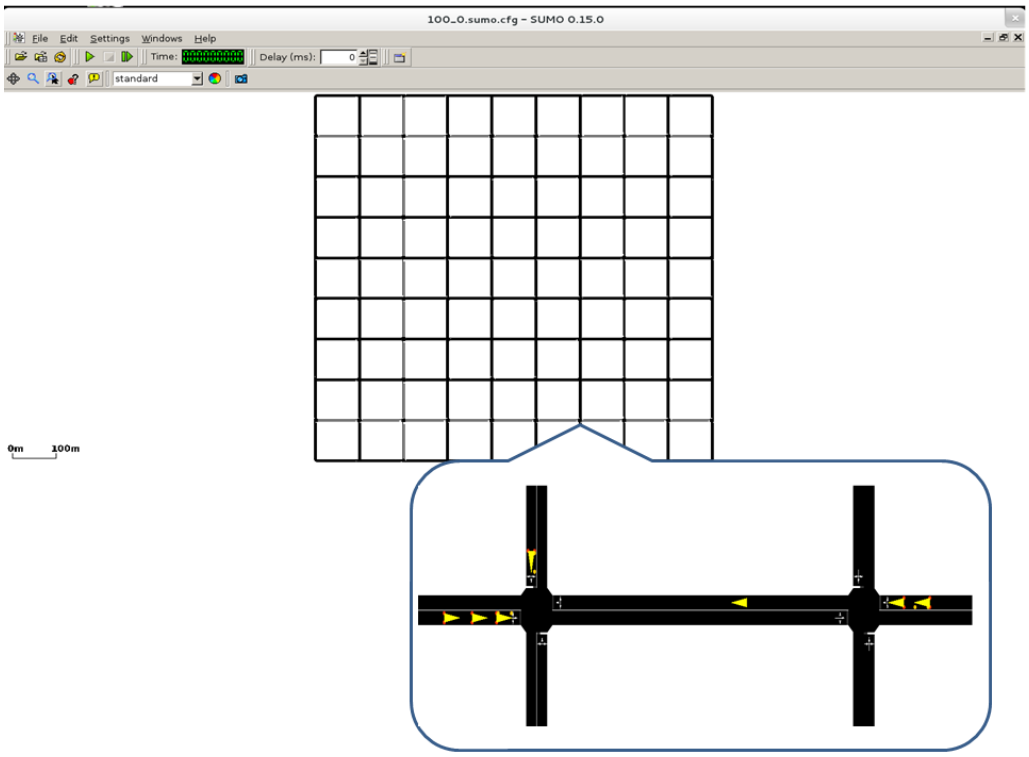



**Figura 2. Cenário de mobilidade**

**Tabela 1. Parâmetros de simulação**

| Parâmetros | Valores |
|---|---|
| Poder de transmissão | 1.6 mW |
| Alcance de transmissão | 200 m |
| Taxa de bit | 18 Mbit/s |
| Número de veículos | 100, 150, 200, 250, 300 veículos/hora |
| Tamanho do beacon | 32 bytes |
| Frequência do beacon | 1 Hz |
| Eficiência de propagação | 1 Hz |
| Ráio da área de interesse | 2 km |
| Número de execuções | 50 |
| Intervalo de confiança | 95% |

parâmetros de simulação são mostrados na tabela 1. Esse parâmetros foram abstraidos de experimentos anteriores.

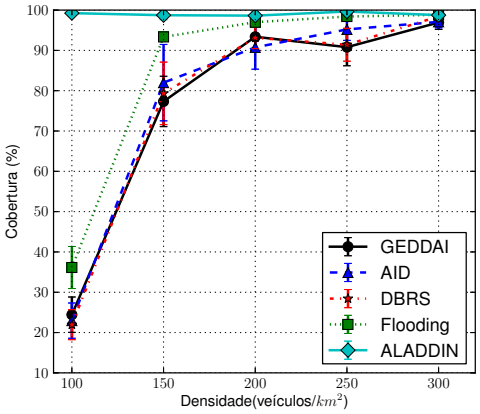
Para as métricas avaliadas, variamos a condição de tráfego, como mencionado anteriormente para analisar o desempenho do algoritmo proposto. Além disso, avaliamos cenários sem partições de rede dentro da área de interesse e cenários onde ocorrem partições de rede dentro da área de interesse. As seguintes métricas foram utilizadas na avaliação de desempenho:

- *Cobertura*: relação entre o número de veículos dentro da área de interesse quando a disseminação dos dados é realizada, e o número de veículos que receberam os dados.
- *Total de pacotes transmitidos*: número total de pacotes de dados transmitidos por todos os veículos, durante o processo de disseminação. Um elevado número de transmissões de pacotes é um forte indício de que as mensagens redundantes estão sendo disseminadas, o que pode resultar no problema de *broadcast storm*.
- *Pacotes transmitidos/entregues*: número de transmissões pela quantidade de informações recebidas, durante o processo de disseminação. Com isso podemos verificar a eficiência da retransmissão, ou seja, quanto de informação está sendo enviada por quanto realmente está chegando aos veículos. Quanto maior o valor menor será sua eficiência, pois enviará muito mais do que recebendo.
- *Atraso*: atraso médio que leva para disseminar a informação da origem para todos os veículos dentro da área de interesse. Pequenos atrasos são importantes para entrega de mensagens em situações críticas, como a disseminação de mensagem de aviso.

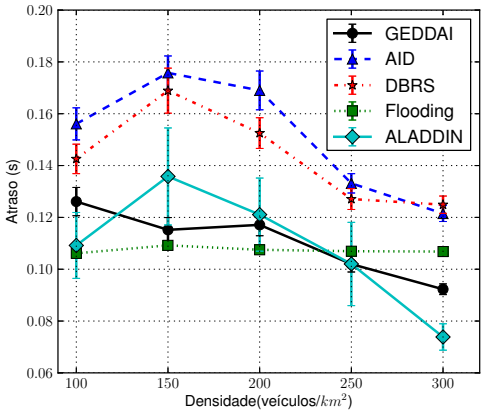
### 4.2. Resultados

A figura 3(a) mostra o resultado de cobertura para todos os algoritmos com diferentes condições de trânsito. Como pode ser observado, ALADDIN é o protocolo de melhor desempenho em todas as condições de trânsito. Quando o tráfego é esparso, por exemplo, 100 e 150 veículos/km$^2$, ALADDIN entrega a mensagem a cerca de 100% dos destinatários, enquanto os outros algoritmos podem entregar, no máximo, a 40% e 80% dos veículos, como no caso do protocolo *flooding*. Com o aumento do tráfego e a diminuição do particionamento da rede, o ALADDIN e o *flooding* podem entregar a mensagem a todos os destinatários, enquanto os demais algoritmos não atingem 100% da área de cobertura. Utilizando o mecanismo autonômico o veículo permite o armazenamento da informação até encontrar um novo veículo. A partir desse resultado, podemos concluir que ALADDIN é uma solução adequada para aplicações que requerem a entrega de dados confiável.

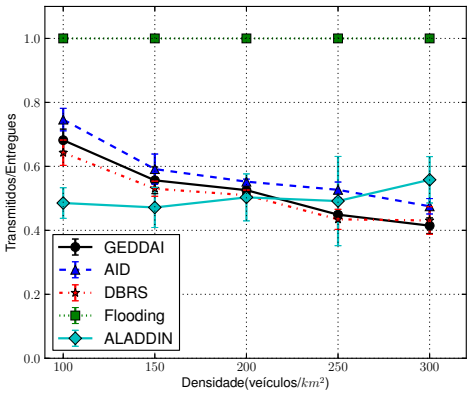
A figura 3(b) mostra o atraso médio de entrega da informação. Em densidades esparsas o protocolo AID possuiu o maior atraso, uma vez que ele deve esperar por algumas recepções para decidir se retransmite ou não a mensagem. AID é seguido de perto pelo DBRS, ALADDIN e GEDDAI, estatisticamente estão ocupando o mesmo patamar pois todos geram um valor aleatório para a sua transmissão de dados, dependendo se estão localizados em uma região de alta prioridade ou não. Portanto, se não há nenhum veículo dentro destas regiões de alta prioridade, os veículos fora dessa região são responsáveis pela retransmissão da informação, no entanto, com um atraso maior. Devido ao maior tempo e quantidade de retransmissões, como podemos ver pela figura 3(d) o



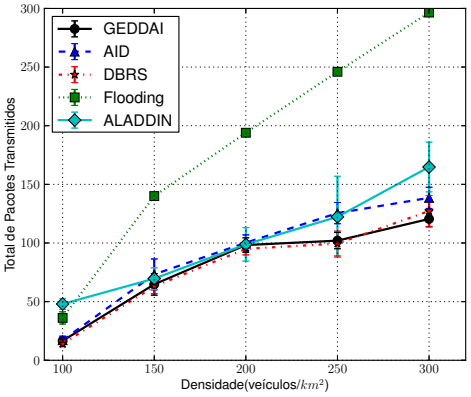

(a) Cobertura




(b) Atraso




(c) Transmitido/Entregue




(d) Transmissões

**Figura 3. Resultados**

algoritmo ALADDIN apresentou um menor atraso à medida que a densidade aumentou, 250 e 300 veículos/kmkm$^2$, pois além do tempo de transmissão a cada *beacon* que o veículo recebe, ele tentará retransmistir a mensagem consequentemente diminuindo o atraso. Esse resultado mostra que apesar de um atraso ligeiramente superior quando comparado a alguns algoritmos, por exemplo *flooding* e GEDDAI, devido ao *store-carry-forward*, o algoritmo atingiu uma cobertura de aproximadamente 100%. Portanto, o ALADDIN é uma solução adequada para aplicações que requerem que suas informações cheguem a todos os veículos dentro de uma área de cobertura com restrições temporais e tráfego esparso.

A figura 3(c) mostra a porcentagem de informações transmitidas e quantos veículos receberam essas mensagens. Como esperado o *flooding* é o protocolo com maior taxa de entrega, uma vez que não utiliza qualquer tipo de mecanismo de supressão de transmissão, simplesmente, realizando um broadcast toda vez que recebe uma informação. Observamos que o ALADDIN manteve o seu padrão de transmissão e recepção em diversas condições de tráfego variando aproximadamente 10%, diferentemente do DBRS, AID e GEDDAI que tiveram uma queda por volta de 40%. Além disso, o ALADDIN teve um melhor desempenho em rede particionadas, atingindo mais veículos com poucas retransmissões. Isso foi possível, através da técnica utilizada de *store-foward*, onde uma diferença de aproximadamente 7% com o segundo colocado, DBRS, foi atingida. Entretanto essa técnica em redes mais densas realizou mais transmissões, pois após a recepção de um *beacon*, o veículo analisa se deve ou não retransmistir a informação. Quando a eficiência estivar baixa, o veículo retransmite as informações armazenadas. Contudo o ALADDIN demonstrou o mesmo padrão de envio e recebimento em ambos cenários, denso e esparsos, ou seja, com poucas transmissões consegue atingir aproximadamente 100% dos veículos que estavam dentro da área de interesse.

Finalmente, a figura 3(d) mostra o número total de pacotes de dados transmitidos por todos os veículos durante o processo de disseminação para diferentes condições de tráfego. Como esperado, o *flooding* é o protocolo com maior sobrecarga, uma vez que não utiliza qualquer tipo de mecanismo de supressão de transmissão. Seguido pelo ALADDIN, que tem uma sobrecarga um pouco superior que os demais algoritmos, principalmente na condição de alta densidade. Isto provém do fato de que o protocolo ALADDIN retransmite muitas informações armazenadas, quando um veículo percebe que existe uma partição na rede e encontra outro veículo capaz de continuar o processo de disseminação. Os algoritmos que apresentaram menor sobrecarga foram o DBRS e GEDAI, respectivamente. No entanto, a baixa sobrecarga do DBRS pode ser atribuída ao seu fraco desempenho na entrega de pacotes. Entre os algoritmos que têm uma cobertura de 100%, ALADDIN tem uma sobrecarga de aproximadamente 30% inferior ao *flooding* enquanto AID tem uma sobrecarga 55% menor quando comparado ao *flooding*. Podemos concluir a partir desses resultados que a nossa proposta de protocolo pode entregar dados de forma confiável aos destinatários com diferentes regimes de trânsito, sem incorrer em uma grande sobrecarga na rede.

## 5. Conclusões

Neste trabalho propusemos ALADDIN, um algoritmo de disseminação de dados geográficos ciente de partição de rede que tenta eliminar o problema de *broadcast storm* e maximizar a capacidade de disseminação de dados entre partições de rede. Em redes densas, veículos dentro da zona de preferência têm uma maior prioridade para retransmitir

mensagens, com isso, diminuindo o número de veículos retransmissores, evitando assim o problema de *broadcast storm*. Em redes esparsas, ALADDIN supera o problema das partições de rede usando um critério de eficiência que consiste no número de vezes que um veículo propaga a informação dividido pelo número de informações recebidas. Por meio de simulações foram comparados ALADDIN e outras quatro soluções, que demonstraram que o método proposto tem os melhores resultados de cobertura, não obstante o fato de ter um atraso e um *overhead* um pouco maior quando comparado a algumas das soluções existentes. Como trabalho futuro, pretendemos melhorar o mecanismo de recuperação para redes particionadas, a fim de reduzir a sobrecarga e o tempo de transmissão.

## 6. Agradecimentos

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

## Referências

Bakhouya, M., Gaber, J., and Lorenz, P. (2011). An adaptive approach for information dissemination in vehicular ad hoc networks. *Journal of Network and Computer Applications*, 34(6):1971–1978.

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). SUMO - Simulation of Urban MObility: An Overview. In *International Conference on Advances in System Simulation (SIMUL '11)*.

Chen, Y.-S., Lin, Y.-W., and Lee, S.-L. (2009). A mobicast routing protocol in vehicular ad-hoc networks. In *GLOBECOM'09*, pages 1–6.

Durresi, M., Durresi, A., and Barolli, L. (2005). Emergency broadcast protocol for inter-vehicle communications. *Parallel and Distributed Systems, International Conference on*, 2:402–406.

Huebscher, C., M., and Cann, J. A. (2008). A survey of autonomic computing: degrees, models, and applications. *ACM Computer Surveys*, 40(3):1–28.

Joshi, H. P., Sichitiu, M. L., and Kihl, M. (2007). Distributed robust geocast multicast routing for inter-vehicle communication. In *First Workshop on WiMAX, Wireless and Mobility*.

Kim, T.-H., Hong, W.-K., Kim, H.-C., and Lee, Y.-D. (2008). An effective data dissemination in vehicular ad-hoc network. In VazÃ£o, T., Freire, M., and Chong, I., editors, *Information Networking. Towards Ubiquitous Networking and Services*, volume 5200 of *Lecture Notes in Computer Science*, pages 295–304. Springer Berlin Heidelberg.

Korkmaz, G., Ekici, E., Özgüner, F., and Özgüner, U. (2004). Urban multi-hop broadcast protocol for inter-vehicle communication systems. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, VANET '04, pages 76–85.

Li, F. and Wang, Y. (2007). Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular Technology Magazine*, 2(2):12–22.

Maia, G., Rezende, C., Villas, L. A., Boukerche, A., Viana, A. C., Aquino, A. L., and Loureiro, A. A. (2013a). Traffic aware video dissemination over vehicular ad hoc networks. In *Proceedings of the 16th ACM International Conference on Modeling,*

*Analysis & Simulation of Wireless and Mobile Systems*, MSWiM '13, pages 419–426, New York, NY, USA. ACM.

Maia, G., Villas, L., Boukerche, A., Viana, A., Aquino, A., and Loureiro, A. (2013b). Data dissemination in urban vehicular ad hoc networks with diverse traffic conditions. In *Computers and Communications (ISCC), 2013 IEEE Symposium on*, pages 000459–000464.

Meneguette, R. I., Bittencourt, L. F., and Madeira, E. R. M. (2013). A seamless flow mobility management architecture for vehicular communication networks. *Journal of Communications and Networks*, 15(2):207–216.

Ros, F., Ruiz, P., and Stojmenovic, I. (2012). Acknowledgment-based broadcast protocol for reliable and efficient data dissemination in vehicular ad hoc networks. *Mobile Computing, IEEE Transactions on*, 11(1):33–46.

Spyropoulos, T., Psounis, K., and Raghavendra, C. S. (2008). Efficient routing in intermittently connected mobile networks: The multiple-copy case. *IEEE/ACM Trans. Netw.*, 16(1):77–90.

Tonguz, O. K., Wisitpongphan, N., and Bai, F. (2010). Dv-cast: a distributed vehicular broadcast protocol for vehicular ad hoc networks. *Wireless Commun.*, 17:47–56.

Toor, Y., Mühlethaler, P., Laouiti, A., and de La Fortelle, A. (2008). Vehicle ad hoc networks: Applications and related technical issues. *IEEE Communications Surveys and Tutorials*, 10(1-4):74–88.

Varga, A. and Hornig, R. (2008). An overview of the OMNeT++ simulation environment. In *International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (Simutools '08)*, pages 1–10.

Veins. Veins vehicles in network simulation.

Villas, L. A., Boukerche, A., de Araujo, R. B., Loureiro, A. F., and Ueyama, J. (2013). Network partition-aware geographical data dissemination. pages 1–6. IEEE International Conference on Communications (ICC '13).

Villas, L. A., Ramos, H. S., Boukerche, A., Guidoni, D. L., Araujo, R. B., and Loureiro, A. A. (2012). An efficient and robust data dissemination protocol for vehicular ad hoc networks. In *Proceedings of the 9th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, PE-WASUN '12, pages 39–46, New York, NY, USA. ACM.

Viriyasitavat, W., Bai, F., and Tonguz, O. (2010). Uv-cast: An urban vehicular broadcast protocol. In *Vehicular Networking Conference (VNC), 2010 IEEE*, pages 25–32.

Wang, Y. and Li, F. (2009). Vehicular ad hoc networks. In *Guide to Wireless Ad Hoc Networks*, Computer Communications and Networks, pages 503–525.

Williams, B. and Camp, T. (2002). Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, MobiHoc '02, pages 194–205, New York, NY, USA. ACM.

# WiBus: Um Sistema de Monitoramento de Transportes Públicos Usando Redes IEEE 802.11

**Vitor Borges C. da Silva, Tatiana Sciammarella, 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,tatiana,miguel,luish}@gta.ufrj.br

***Abstract.*** *A great challenge today is to deal with the constant traffic jams related to excessive use of vehicles. It is believed that more people would adopt public transportation if these were more reliable. Thus, this paper proposes WiBus, which is a system to estimate buses times of arrival, based on information from opportunistic contacts in IEEE 802.11 networks. Estimates are provided to users through graphical interfaces of mobile devices. WiBus adjusts bus trajectories with an algorithm for dynamic creation and maintenance. The system was implemented and its performance was analyzed via emulation of a real scenario. Experimental results show that WiBus can meet demands of large cities with accumulated error on the order of a few minutes in the worst case.*

***Resumo.*** *Um grande desafio atual é lidar com os engarrafamentos relacionados ao uso excessivo de veículos. Acredita-se que mais pessoas adotariam transportes públicos se estes fossem mais confiáveis. Nesse sentido, este trabalho propõe o WiBus, um sistema para estimar os horários de chegada dos ônibus, baseado em informações de contatos oportunistas em redes IEEE 802.11. As estimativas são disponibilizadas aos usuários através de interfaces gráficas de dispositivos móveis. O WiBus ajusta as trajetórias das linhas de ônibus dinamicamente. O sistema desenvolvido foi analisado através da emulação de um cenário real. Os experimentos mostram que o WiBus pode atender demandas de grandes cidades com erro acumulado de poucos minutos no pior caso.*

## 1. Introdução

As grandes metrópoles brasileiras vêm sofrendo com engarrafamentos constantes gerados pelo aumento do volume de veículos particulares. Em apenas um ano, de setembro de 2012 até setembro de 2013, quase 3 milhões de automóveis particulares entraram em circulação no Brasil, ocasionando uma piora sensível no trânsito [DENATRAN, 2013]. A preferência pelo transporte particular é uma consequência da falta de confiança da população nos meios de transporte de massa, seja por questões de segurança, conforto ou comprometimento com os horários. Para reverter essa tendência e atrair passageiros, os sistemas de transporte públicos devem aumentar a qualidade dos seus serviços. Esse desafio é um dos principais objetivos dos recentes sistemas avançados de transporte público (*Advanced Public Transportation Systems* - APTS) que,

*Este trabalho foi parcialmente financiado pela CAPES, CNPq, FAPERJ e FINEP.

dentre outras tecnologias, usam redes de comunicação para prover mais informações aos usuários [Casey et al., 1991].

As redes de comunicação em ambientes veiculares vêm sendo investigadas com o uso de tecnologias que vão desde as de telefonia celular (3G/4G) até as de redes locais sem-fio (IEEE 802.11). Nessas redes, há dois tipos predominantes de comunicação, a V2I (*Vehicular-to-Infrastructure*) e a V2V (*Vehicular-to-Vehicular*). A primeira envolve nós veiculares e nós de rede estáticos instalados em infraestruturas ao longo das vias, já a segunda envolve somente nós móveis em veículos automotores. A arquitetura V2I pode oferecer maior conectividade de rede e por isso vem sendo utilizada com maior frequência, apesar do maior custo em infraestrutura.

O trabalho desenvolvido visa aumentar a confiabilidade dos transportes públicos fornecendo a estimativa do tempo de chegada dos ônibus aos seus pontos de parada. Entretanto, apesar de considerar a arquitetura V2I, o desafio em calcular as estimativas reside no fato do trânsito e dos passageiros influenciarem no tempo que um ônibus leva em sua trajetória (p.ex., quando congestionamentos se formam devido a acidentes). Além disso, situações como mudanças de trajetória podem acontecer de forma a perturbar as estimativas feitas pelo sistema. Um requisito, portanto, para calcular o tempo de chegada sem erros significativos é o conhecimento das posições dos ônibus de forma mais atual possível. Com esse propósito, muitos trabalhos utilizam sistemas de GPS, devido à sua grande acurácia. Contudo, o uso do GPS exige alguma outra tecnologia de comunicação para divulgação da posição do veículo, sendo os sistemas GPS normalmente apenas receptores. Alternativamente ao GPS, existem técnicas de localização que utilizam tecnologias de redes como as celulares e as sem-fio IEEE 802.11. Uma das vantagens do uso desses sistemas é que eles também podem ser utilizados para comunicações. Esse potencial é explorado mesmo em sistemas simples de localização que funcionam por proximidade. Dessa forma, a localização do veículo é dada pelo nó da rede ao qual ele está conectado. A acurácia desses sistemas de localização é inferior à dos sistemas GPS. Contudo, técnicas para aumentar a acurácia dos sistemas de redes vêm sendo propostas, chegando a obter erro médio quadrático da localização de apenas sete metros [Caceres et al., 2009].

Considerando os sistemas de localização em rede, alguns trabalhos da literatura calculam as estimativas usando métodos baseados em séries históricas coletadas em janelas de tempo que podem ser até da ordem de meses [Manolis e Kwstis, 2004]. Assim, a estimativa de localização usa a média pertencente ao mesmo período de um mesmo dia da semana, desconsiderando situações como acidentes e exigindo um volume de informações passadas muito grande. Em oposição à abordagem com séries históricas, existem também aquelas de tempo real, onde apenas as informações do próprio dia são usadas, modelando melhor situações imprevistas, como acidentes. As abordagens de tempo real assumem que o tempo que um dado veículo leva em um percurso é o mesmo gasto por um veículo anterior [Lin e Zeng, 1999] já que o histórico passado é curto. Existem ainda técnicas que utilizam modelos de regressão, que realizam a previsão através de um conjunto de variáveis independentes escolhidas para modelar o trânsito. O problema dessa premissa é que as variáveis dos sistemas de transporte são correlacionadas, o que limita a aplicabilidade dessa abordagem [Chien et al., 2002]. Por fim, existem ainda alguns métodos que usam filtros de Kalman que não possuem um bom desempenho quando os dados de localização são temporalmente esparsos [Karbassi e Barth, 2003].

Este trabalho propõe um sistema para a estimativa da chegada dos ônibus usando redes IEEE 802.11 chamado WiBus. O objetivo é manter ao mesmo tempo requisitos de baixo custo e baixa complexidade, reunindo vantagens das propostas anteriores. Para isso, o WiBus utiliza a localização por proximidade em relação a roteadores sem-fio IEEE 802.11, caracterizando um sistema baseado em redes veiculares com arquitetura V2I. Essa abordagem, ao contrário das que utilizam GPS, permite o uso de apenas um tipo de dispositivo na fase de localização do veículo e na fase de divulgação da informação, reduzindo a complexidade e o custo do sistema. Como neste trabalho os dados de localização são esparsos devido ao método de localização e como não há uma base de dados com informações de localização antigas, a técnica usada para previsão de tempo de chegada é a de tempo real. O WiBus estima o horário de chegada de ônibus e se adapta a mudanças de trajetória dinamicamente. Assim, mesmo que um ônibus fuja da sua trajetória conhecida, ele consegue se adaptar. O sistema é capaz de atender as demandas de uma cidade grande como a do Rio de Janeiro com erros da ordem de poucos minutos, como os alcançados a partir de experimentos com dados reais. Além disso, para facilitar o acesso do público ao sistema, foram desenvolvidos um aplicativo para smartphones Android, chamado BuZoom, e uma página Web que agem como interfaces para solicitação de estimativas. Com isso, pode-se dizer que o sistema atende as especificações para utilização em ambientes reais.

O presente trabalho está organizado da seguinte forma. A Seção 2 apresenta a arquitetura do WiBus. A Seção 3 fornece os detalhes da implementação do sistema e um exemplo de funcionamento para o algoritmo de criação e manutenção dinâmica de trajetórias. Na Seção 4, o cenário de testes é descrito, enquanto na Seção 5, os experimentos realizados e os resultados obtidos são apresentados. Finalmente, a Seção 6 conclui este trabalho e apresenta as direções futuras.

## 2. Arquitetura do WiBus

No sistema WiBus, ilustrado na Figura 1, existem quatro entidades: Central, Unidade de Acostamento (UA), Ônibus e Cliente. A Central é um computador que executa o programa principal do sistema. Ela possui todas as informações necessárias para o cálculo das estimativas, os mapas criados dinamicamente, a localização de cada veículo e os dados referentes aos tempos entre pontos consecutivos. Além disso, a Central é responsável por calcular as estimativas e responder as requisições dos Clientes. Já as Unidades de Acostamento (UAs) são os pontos de acesso instalados nos pontos de ônibus. Elas oferecem redes sem-fio com identificadores (*Service Set IDentifier* - SSID) pré-fixados iguais para a conexão dos roteadores nos ônibus. Para que um ônibus seja rastreado, ele deve possuir um roteador sem-fio IEEE 802.11 executando alguns programas desenvolvidos. Esses programas localizam o ônibus em relação às UAs e enviam a informação de localização à Central através das próprias UAs. O Cliente é qualquer pessoa que solicita estimativas ao sistema. A requisição do Cliente é através do aplicativo BuZoom para smartphones Android ou através da página web desenvolvidos.

**Serviços do WiBus:** O sistema desenvolvido oferece os serviços complementares de localização de veículos e a estimativa do tempo de chegada de um ônibus aos seus pontos de parada. Os dois serviços se baseiam na técnica de localização por proximidade em relação a pontos de acesso posicionados nos pontos de parada. Logo, ao se aproximar de

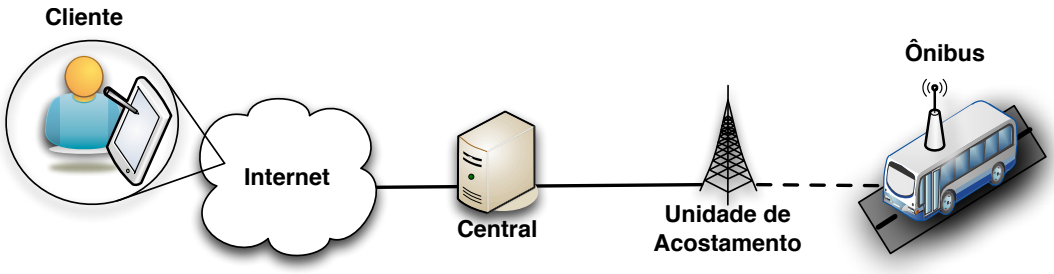



**Figura 1. Entidades do sistema na arquitetura.**

um ponto de parada, o ônibus se conecta ao ponto de acesso. Em seguida, ocorre uma troca de mensagens, onde o ônibus se identifica à UA e vice-versa. Após conectar-se à UA, o ônibus envia uma mensagem à Central informando a qual UA ele está conectado. Mesmo de posse da localização dos ônibus, ainda é necessário ter um mapa com as trajetórias das linhas de ônibus para oferecer o serviço de estimativa de horário de chegada. Esse mapa deve ser usado em conjunto com o histórico de períodos recentes que os ônibus levam entre dois pontos consecutivos da linha. A abordagem inicial para ter o mapa das linhas de ônibus é supor que as trajetórias não se alteram. Porém, essa suposição é falha já que mudanças temporárias podem ocorrer.

## 3. Implementação

O WiBus é implementado como um conjunto de seis programas, listados a seguir:

- `WiBusUAServer` – programa executado ininterruptamente na UA que a cada conexão de um ônibus envia uma mensagem com a identificação da própria UA ao ônibus conectado, recebendo, em seguida, uma mensagem com a identificação do ônibus utilizada apenas para manter um histórico de contatos na UA.
- `WiBusClientUA` – programa executado no Ônibus para a recepção das mensagens enviadas pelas UAs com a identificação delas e para envio de mensagens com a identificação do ônibus às UAs. O `WiBusClientUA` é executado toda vez que o Ônibus se conecta a uma UA.
- `WiBusClientCentral` – programa executado no Ônibus para recepção de mensagens enviadas pela Central e para envio de mensagens para a Central contendo informações de localização na rede. Esse programa é executado após o fim do programa `WiBusClientUA`.
- `WiBusCentralServer` – programa executado ininterruptamente na Central para envio de mensagens ao ônibus e recepção de mensagens enviadas pelo ônibus. A partir das mensagens recebidas, cria-se o mapa das linhas de ônibus, localizam-se os ônibus e calculam-se as estimativas de tempo de chegada dos ônibus. Além disso, este programa recebe e responde os pedidos de estimativa dos Clientes.
- `BuZoom` – programa da interface executada nos smartphones Android dos Clientes. Esse programa solicita estimativas de tempo de chegadas dos ônibus à Central.
- `BuZoom Web` – programa da interface web acessada pelos Clientes. Assim como o BuZoom, ele acessa a Central para obtenção das informações sobre os ônibus.

No WiBus, o ônibus descobre sua posição na rede com a execução do programa `WiBusClientUA`. Em seguida, o programa `WiBusClientCentral` é executado, enviando uma mensagem à Central. Essa mensagem contém as seguintes

informações: UA onde o ônibus está conectado, UA anterior na qual o ônibus esteve conectado, o identificador do ônibus e o identificador da linha de ônibus. O programa `WiBusCentralServer`, ao receber essa mensagem, realiza seu tratamento.

### 3.1. Central

No WiBus, uma linha de ônibus é caracterizada pelo seu número e seu sentido (p.ex. 913 Del Castilho), de maneira que todas as linhas de ônibus não são consideradas circulares para o sistema. Além disso, uma linha de ônibus é composta de vários trechos, onde um trecho é definido como um par de UAs, em que a primeira UA do par vem imediatamente antes da segunda na trajetória da linha de ônibus.

Ao receber uma mensagem, a Central atualiza a posição dos ônibus com as UAs especificadas na mensagem. Em seguida, a Central atualiza um mapa associativo que relaciona trechos das linhas de ônibus aos tempos gastos pelos últimos ônibus nesses trechos. Esse mapa associativo é utilizado na estimativa do tempo necessário para o ônibus mais próximo alcançar cada UA das linhas de ônibus.

Como uma linha de ônibus é caracterizada como uma sequência de trechos, para estimar o tempo necessário para que um ônibus alcance uma determinada UA, basta somar as estimativas de tempo necessárias para o ônibus percorrer cada um dos trechos que liga a UA onde o ônibus se encontra à UA alvo da estimativa. Assim, o problema se reduz a estimar o tempo necessário para que um ônibus percorra um dado trecho. No WiBus, a estimativa para um trecho é calculada como a média dos tempos gastos pelos $K$ últimos ônibus que percorreram o mesmo trecho. Em outras palavras, é uma média móvel simples de tamanho $K$ dos tempos do trecho mantidos no mapa associativo da Central. O uso da média móvel simples provê complexidade reduzida ao cálculo das estimativas.

A Central, utilizando os conceitos acima descritos, estima o tempo necessário para que o ônibus mais próximo alcance cada UA de uma dada linha de ônibus. Isso é realizado calculando para todos os ônibus da linha, as estimativas até cada UA dessa linha, armazenando apenas a menor estimativa para cada UA. Esse processo é descrito formalmente no Algoritmo 1, onde $estimativa_b$ armazena a estimativa para um dado ônibus $b$ alcançar o trecho atual a partir de sua posição inicial, e a função `estimativaTrecho` retorna a estimativa do ônibus $b$ percorrer apenas o trecho atual. Já, o algoritmo de criação e atualização dos mapas das linhas de ônibus é descrito na Seção 3.2.

**Algoritmo 1:** Cálculo da estimativa para próximo ônibus alcançar UAs de uma linha de ônibus.

**Entrada**: linha de ônibus $l$.
**Saída**: estimativa mínima para cada UA.

```
1 para todo ônibus b servindo a linha l faça
2     enquanto trecho existe faça
3         estimativa_b += estimativaTrecho( trecho);
4         se estimativa_b < estimativa da UA atual então
5             estimativa da UA atual = estimativa_b
6         fim
7         trecho = próximo trecho;
8     fim
9 fim
```

### 3.2. Algoritmo de Criação e Manutenção Dinâmica de Trajetórias

A necessidade de representar trajetórias dinamicamente emerge de situações onde imprevistos alteram as trajetórias, como quando vias são fechadas. Contudo, peculiaridades do mundo real, como um motorista de ônibus mudando a trajetória inadvertidamente, impedem que mudanças sinalizadas pelos veículos sejam assumidas como verdade instantaneamente. Uma maneira de verificar se a trajetória da linha de ônibus realmente mudou é através da observação de outros ônibus da mesma linha. A mudança é confirmada, caso outros ônibus da mesma linha repitam a mesma trajetória. Assim, o algoritmo de criação e manutenção dinâmica de trajetórias proposto utiliza um esquema de pesos para evitar mudanças prematuras na trajetória das linhas de ônibus.

Esse esquema possui dois tipos de peso: um que é usado nos arcos, que são as estruturas utilizadas para indicar as UAs próximas e anteriores de uma UA na trajetória do ônibus, e outro que é usado na própria UA. O primeiro tipo de peso rege a dinamicidade do mapa, capturando o quão rápido uma modificação na trajetória é considerada como permanente. Esse peso pode ser ajustado desde que respeite a restrição de ser no máximo duas vezes o número de ônibus servindo a linha. Essa restrição é necessária para que, em uma mudança de trajetória, os pesos dos arcos de uma UA em remoção alcancem o valor zero antes do peso da própria UA.

O segundo tipo de peso é usado para remoção de UAs que não pertençam mais a uma dada trajetória. O valor desse tipo de peso deve ser grande o suficiente para que nenhuma UA seja excluída erroneamente. Por isso, o valor inicial atribuído a uma UA é igual ao dobro do produto da quantidade de UAs da linha pela quantidade de ônibus realizando-a. Como o peso máximo depende do número de UAs cadastradas, ao acrescentar uma UA em uma linha, todas as UAs da linha têm seu peso igualado ao novo máximo, por questões de equivalência. Vale ressaltar, portanto, que o sistema trata também situações onde uma nova UA é inserida na trajetória ou uma anterior reaparece. Essa informação é obtida das mensagens recebidas pela Central.

No WiBus, o valor do peso inicial atribuído a uma UA também é o peso máximo permitido às UAs. Isso significa que é necessário que todos os ônibus de uma dada linha percorram todas as UAs da linha ao menos duas vezes, para que uma UA não utilizada seja excluída. Similarmente, o incremento dado quando uma UA é usada deve garantir que essa UA não seja excluída enquanto o ônibus percorre as outras UAs da linha. Para isso, o incremento é igual ao dobro da quantidade de UAs na linha de ônibus.

Durante o transiente de um processo de mudança de uma linha de ônibus, uma UA pode possuir mais de uma possibilidade de próxima UA ou UA anterior. Nesses casos, uma UA possui uma lista de arcos indicando as UAs próximas ou anteriores a ela, ao invés de apenas um arco. Assim, a atualização dos arcos para as UAs anteriores ou próximas é realizada da seguinte forma: subtrai-se uma unidade do peso de todos os arcos da lista, de UAs anteriores ou próximas, e somam-se duas unidades ao peso do arco, anterior ou próximo, usado. Caso o peso de algum arco se torne zero na atualização, ele é removido, indicando que uma UA não é mais usada como anterior ou próxima da atual.

Para concluir o esquema de atualização dos pesos, resta explicar como é realizada a atualização do outro tipo de peso. No processo de atualização do peso das UAs de uma linha de ônibus, o peso de todas as UAs da linha é decrementado de uma unidade. Após a

subtração, incrementa-se o peso da UA atual com o valor anteriormente especificado, isto é, duas vezes a quantidade de UAs na linha de ônibus.

Por fim, outra operação necessária é a exclusão das UAs não utilizadas da linha de ônibus. Nesse processo as UAs cadastradas que tem peso igual a zero são eliminadas. Com isso, o pseudocódigo simplificado responsável pela criação e atualização dos mapas das linhas de ônibus pode ser visto em Algoritmo 2.

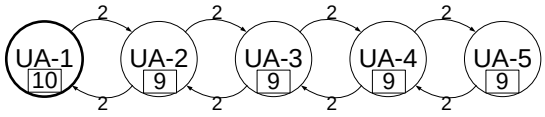
**Algoritmo 2:** Criação e atualização do mapa de linha de ônibus.

```
Entrada: mensagem m recebida do ônibus b servindo a linha l.
Saída: mapa da linha de ônibus atualizado.
se recebeu mensagem anterior então
    busca l descrita em m;
    se achou l então
        busca UA_atual de m em l;
        se achou UA_atual então
            Atualiza arcos para UAs anteriores da UA_atual;
            Atualiza peso das UAs de l;
            Atualiza arcos para próximas UAs da UA_anterior;
            Exclui UAs não utilizadas de l;
        senão
            Cria UA_atual de m;
            Insere UA_atual em l;
            Peso das UAs de l = peso_maximo;
            Cria arcos entre UA_anterior e UA_atual;
        fim
    senão
        Cria UA_atual de m;
        Cria l de m com UA_atual;
    fim
    se b mudou de linha então
        Altera quantidade de ônibus da linha atual l e da linha antiga;
    fim
    se b visto pela primeira vez então
        Soma 1 à quantidade de ônibus da linha atual l;
    fim
fim
```

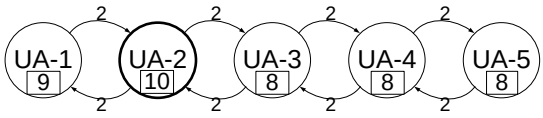
### 3.3. Exemplo de Funcionamento

O exemplo de funcionamento demonstra como o sistema lida com mudanças na trajetória de uma linha de ônibus. No exemplo, um ônibus percorre a trajetória de uma linha de ônibus fictícia que sofre modificações. Para tal, os mapas da linha de ônibus são representados na Figura 2. Nessas figuras, as circunferências representam as UAs e o número no interior do retângulo em cada UA é o peso da UA usado no critério de exclusão. Já os números nas setas representam os pesos dos arcos associados às informações de próxima UA e UA anterior da UA de origem da seta. Toda vez que um ônibus se move, o algoritmo de atualização dos mapas é executado pela Central.

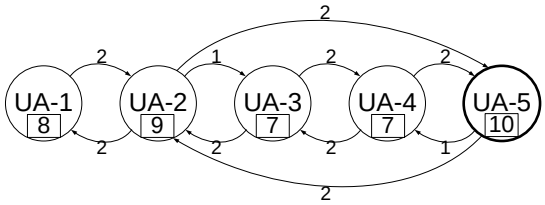
O teste se inicia com o ônibus se movendo até a UA-1. Com isso, o algoritmo de



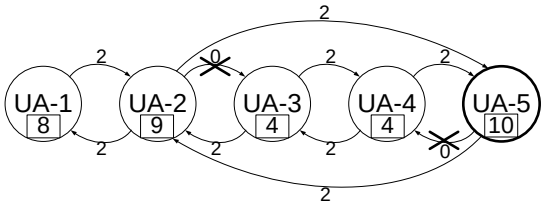

(a) Trajetória inicial da linha de ônibus.



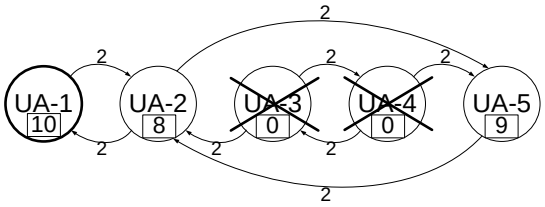

(b) Ônibus se move até a UA-2.




(c) Ônibus alcança a UA-5, sem passar pela UAs 3 e 4.




(d) A mudança na trajetória é concretizada.




(e) Exclusão das UAs 3 e 4.

**Figura 2. Exemplo de funcionamento do Algoritmo de Criação e Manutenção Dinâmica de Trajetórias.**

atualização de trajetórias atualiza o peso da própria UA para o máximo, e reduz em uma unidade o peso de todas as outras UAs da linha. Essa situação se encontra na Figura 2(a).

À medida que o ônibus se movimenta sem alterações na trajetória, os pesos dos arcos entre as UAs continuam no valor máximo permitido. Entretanto, a cada movimento do ônibus, o peso de todas as UAs da linha é decrementado de uma unidade, enquanto o peso da UA onde o ônibus se encontra é incrementado. Por exemplo, o mapa atualizado para o ônibus na UA-2 pode ser visto na Figura 2(b).

Nesse exemplo de funcionamento, supõe-se que as UAs 3 e 4 estão defeituosas. Com isso, a Central não recebe as mensagens referentes a esses dois pontos de acesso, fazendo com que a próxima UA funcionando na trajetória seja a UA-5. Ao chegar à UA-5, o ônibus envia uma mensagem à Central, informando que ele está lá e que antes esteve na UA-2. Com isso, a UA-5 é cadastrada na lista de próximas UAs da UA-2. Da mesma maneira, a UA-2 é cadastrada na lista de UAs anteriores da UA-5. Ambos os arcos são inicializados com peso máximo e as outras informações da lista de próximas UAs da UA-2 e da lista de UAs anteriores da UA-5 são decrementadas de uma unidade. O resultado desse processo pode ser visto na Figura 2(c).

Continuando o exemplo, o ônibus percorre a trajetória alcançando a UA-1 e depois a UA-2. Quando o ônibus se aproxima da UA-5 novamente, o sistema verifica que, novamente, a UA anterior a UA-5 é a UA-2. Por conseguinte, a UA seguinte a UA-2 é a UA-5. Com isso, o peso desses arcos se mantém, enquanto o peso dos outros arcos decresce de uma unidade. Assim, os arcos antigos atingem o valor zero e por isso são excluídos, o que representa a concretização da modificação da trajetória. Isso pode ser afirmado, pois apesar das UAs 3 e 4 existirem na lista de UAs da linha, elas já não são alcançáveis a

partir do ponto inicial da linha de ônibus. As modificações realizadas podem ser vistas na Figura 2(d).

Com a trajetória do ônibus atualizada, não ocorrem mais alterações nos arcos entre as UAs da trajetória. Todavia, o ônibus continua realizando a trajetória, e os pesos das UAs são atualizados de acordo. Esse processo se repete até os pesos das UAs 3 e 4 chegarem a zero. Isso faz com que o mecanismo de atualização da linha de ônibus exclua essas UAs da lista de UAs da linha, como pode ser visto na Figura 2(e).

Com a exclusão das UAs 3 e 4 da lista de UAs da linha de ônibus, o valor máximo dos pesos das UAs diminui. Porém, o peso de cada UA só é atualizado, obedecendo ao novo limite, quando o ônibus alcançar cada UA. Com esse exemplo, mostra-se que os mapas das linhas de ônibus podem ser alterados dinamicamente. Vale ressaltar que a velocidade de concretização de alterações na trajetória pode ser configurada escolhendo o valor máximo dos pesos dos arcos das UAs.

### 3.4. Interface Gráfica

A interface gráfica desempenha um papel fundamental no projeto. É através dela que o usuário faz sua requisição de forma amigável. A primeira interface gráfica criada para o sistema foi uma página web, o BuZoom Web. No entanto, tal estrutura é mais indicada quando o acesso à informação é efetuado em computadores. Acreditando-se que a maioria dos usuários deva realizar sua pesquisa pelos seus smartphones, foi desenvolvido o aplicativo BuZoom, que pode ser visto na Figura 3. Essa interface está disponível para dispositivos com sistema operacional Android 2.2 (Froyo) ou versões mais recentes.

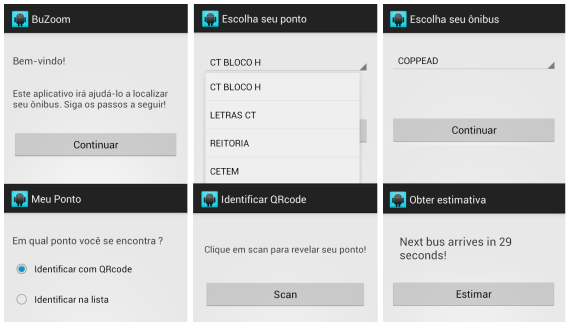



**Figura 3. Telas do BuZoom para Android.**

A consulta do usuário pode ser feita de duas formas distintas, cada uma mais apropriada para um cenário específico. Na primeira, o usuário usa um leitor de `QRcode`, que está disponível no ponto, para identificar o ponto de ônibus. Na segunda, o usuário escolhe seu ponto em uma lista suspensa. A primeira forma é mais apropriada para usuários móveis que, apesar de já estarem no ponto de ônibus, não sabem onde se localiza tal ponto. Já a segunda forma é mais apropriada para os usuários remotos que ainda não se encontram no ponto ou que desejam ir até o ponto apenas quando o ônibus estiver mais próximo. Após a escolha do ponto de ônibus, é possível escolher a linha desejada através de uma listagem. Para enviar o par (ponto escolhido, ônibus escolhido) para a Central, é preciso que o usuário tenha conexão com a Internet. Após o envio da requisição, a resposta pode ser visualizada na tela do dispositivo.

As ferramentas utilizadas no desenvolvimento do BuZoom foram o Android SDK (*Software Development Kit*) e o JDK (*Java Development Kit*). Já o reconhecimento do QRcode foi feito importando o pacote `Zxing.integration.android` e fazendo uma requisição a um aplicativo auxiliar chamado `BarcodeScanner`. A produção dos `QRcodes`, referentes aos pontos de ônibus do Campus Ilha do Fundão da UFRJ, foi feita com o aplicativo `QRdroid`.

## 4. Rede de Testes

Para testar o WiBus, foi montado um pequeno protótipo no laboratório, onde os diferentes elementos da arquitetura são emulados. A Central é um PC com sistema operacional Debian instalado, 8 GB de memória RAM e processador Intel Core i7 860. Por outro lado, as UAs e os ônibus são representados por roteadores sem-fio D-Link, modelo DIR-320. A Central e as UAs estão conectadas através de um roteador, também D-Link e modelo DIR-320, que foi configurado para funcionar como comutador. Já a conexão das UAs e da Central com esse comutador é realizada por cabo Ethernet. O modelo DIR-320 possui processador ARM de 240 MHz e 32 MB de memória RAM. Além disso, possui uma entrada USB que pode ser usada para aumentar a capacidade de armazenamento do roteador, que originalmente é de apenas 4 MB, ou para instalar uma segunda interface de rede sem-fio. O sistema operacional usado nos roteadores é o OpenWRT versão Backfire 10.03.1, uma distribuição Linux para dispositivos embarcados.

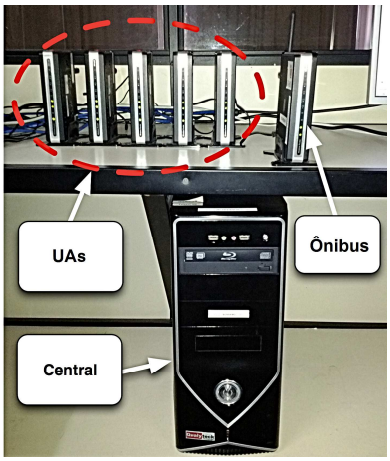



**Figura 4. Ambiente de testes.**

## 5. Avaliação

Nos experimentos deste trabalho o peso usado para os arcos das UAs é igual ao máximo possível definido (Seção 3.2). O primeiro experimento tem o objetivo de avaliar a capacidade de atendimento de ônibus do WiBus, ou seja, avaliar a quantidade máxima de ônibus sob a qual o sistema ainda é capaz de rastrear os veículos e calcular as estimativas. Já o segundo experimento, que é realizado com dados reais de um cenário universitário, tem o objetivo de avaliar os erros das estimativas calculadas pelo WiBus.

### 5.1. Capacidade de Atendimento de Ônibus

Este teste simula as condições da cidade do Rio de Janeiro e tem por objetivo avaliar a capacidade de atendimento de ônibus do WiBus. No teste, são cadastrados 8.000

ônibus distribuídos uniformemente em 800 linhas fictícias de ônibus, de forma a reproduzir um cenário similar ao do Rio de Janeiro [de Urbanismo Pereira Passos, 2011]. Esses cadastros são feitos através de um *script* que emula a passagem do ônibus pelas UAs, enviando mensagens do roteador que faz papel de ônibus à Central. Como o número de UAs inseridas em uma linha de ônibus pode variar, por exemplo, devido às diferentes quantidades de pontos de ônibus nas linhas, o número de UAs das linhas de ônibus é variado no experimento de 10 a 100 com incrementos de 10. Após a criação das trajetórias das linhas fictícias, um dos ônibus cadastrados percorre uma das trajetórias 10 vezes, enviando mensagens para a Central ao trocar de UA. A Central trata as mensagens e mede o tempo gasto no tratamento. Todo o processo é repetido três vezes para cada número de UAs diferente nas linhas de ônibus. Os resultados do teste podem ser vistos na Figura 5.

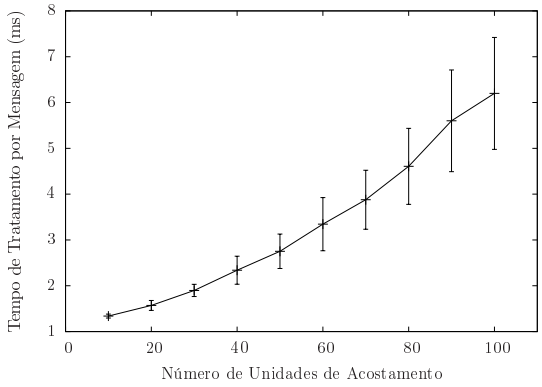


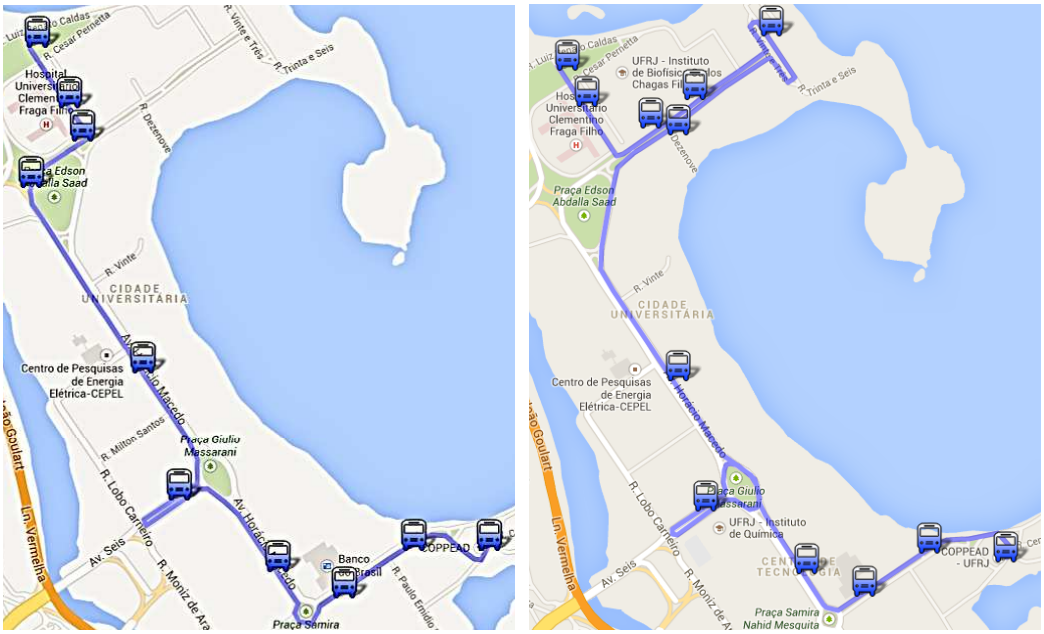

**Figura 5. Tempo de tratamento de mensagens.**

A partir do gráfico, infere-se que o sistema WiBus é capaz de monitorar e estimar toda a cidade do Rio de Janeiro com apenas um computador similar ao usado. Essa inferência é realizada observando que o tempo de tratamento por mensagem é inferior a 7 ms e supondo que os ônibus da cidade levam em média mais de um minuto para alcançar à próxima UA da trajetória. Com isso, o sistema tem pelo menos 60 segundos para tratar as 8.000 mensagens. Dividindo-se os 60 segundos pelo tempo de tratamento do pior caso, 7 ms, obtém-se uma taxa de aproximadamente 8.500 mensagens tratadas por minuto. Portanto, todas as mensagens enviadas à Central são tratadas, mesmo considerando o pior caso para todas as linhas de ônibus da cidade. Como o sistema é capaz de atender grandes demandas, no experimento seguinte é avaliada a qualidade das estimativas realizadas.

### 5.2. Experimentos em um Cenário Universitário

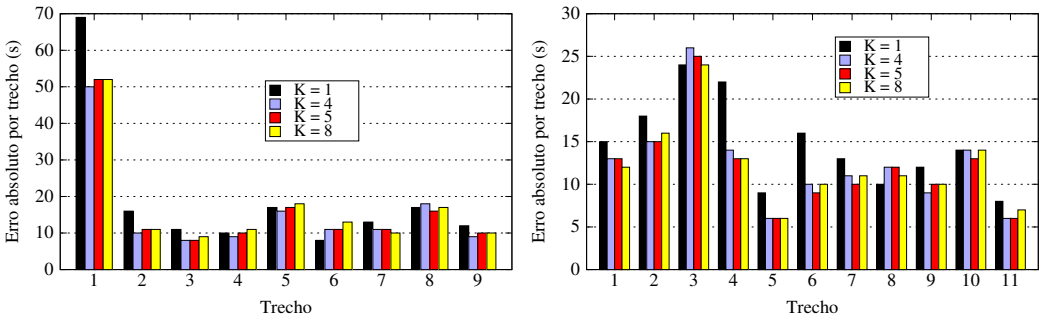
Para este teste, mediu-se nove vezes o tempo gasto em todos os trechos das linhas internas de ônibus universitários "COPPEAD" e "Estação UFRJ", mostradas na Figura 6. Esses dados foram usados para avaliar a qualidade das estimativas do sistema WiBus. As estimativas das linhas de ônibus são calculadas através de médias móveis simples de tamanho $K$. Portanto, também é necessário definir um valor para o parâmetro $K$. Para ambos os fins, as estimativas e os erros foram calculados para os dados reais medidos com $K$ variando de 1 a 8. Por questões de clareza, são apresentados os erros absolutos médios por trecho dos casos onde $K$ vale 1, 4, 5 e 8 para ambas as linhas de ônibus na Figura 7.

O valor de $K$ é escolhido de forma a minimizar o erro médio da estimativa do ônibus desde o ponto inicial até o ponto final da linha. Os valores de $K$ são os que




(a) Linha “COPPEAD”. (b) Linha “Estação UFRJ”.

**Figura 6. Trajetórias das linhas de ônibus universitárias.**

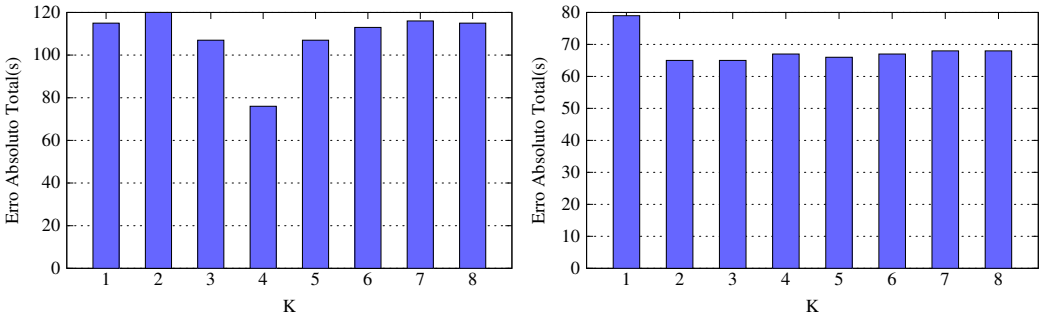



(a) Linha “COPPEAD”. (b) Linha “Estação UFRJ”.

**Figura 7. Erro absoluto médio para diferentes valores de $K$.**

produzem o menor erro absoluto segundo os gráficos da Figura 8, $K = 4$ para a linha “COPPEAD” e $K = 3$ para a linha “Estação UFRJ”.

A escolha de tamanhos diferentes para as médias móveis é justificada pelo fato de linhas de ônibus diferentes apresentarem comportamentos diferentes. As características dos trechos das linhas de ônibus estudadas são resumidas nas Tabelas 1 e 2. Nessas tabelas, observa-se que alguns trechos apresentam elevado coeficiente de variação, definido como a divisão do desvio padrão pela média. O elevado coeficiente de variação desses trechos é justificado pela presença de semáforos e travessias de pedestres. Além disso, a partir das informações da tabela, pode-se calcular o coeficiente de variação médio das duas linhas. A linha “COPPEAD” possui coeficiente de variação médio igual a 0,21, enquanto a linha “Estação UFRJ” apresenta o mesmo coeficiente igual a 0,18. Esses valores ajudam a compreender os gráficos da Figura 8, já que linhas com maiores variações tendem a apresentar um maior compromisso entre informações recentes e informações de longo




(a) Linha “COPPEAD”. (b) Linha “Estação UFRJ”.

**Figura 8. Erro médio de estimativa do ponto inicial até o final para diferentes valores de $K$.**

prazo, como é o caso da linha “COPPEAD”, enquanto linhas com menores variações não apresentam esse comportamento de forma tão nítida, como a linha “Estação UFRJ”.

| Trecho | Média (s) | Desvio Padrão | Coeficiente de Variação |
|---|---|---|---|
| 1 | 116,44 | 52,04 | 0,45 |
| 2 | 42,67 | 12,43 | 0,29 |
| 3 | 44,33 | 9,90 | 0,22 |
| 4 | 107,56 | 11,75 | 0,11 |
| 5 | 79,67 | 17,26 | 0,22 |
| 6 | 140 | 14,73 | 0,11 |
| 7 | 85,33 | 10,83 | 0,13 |
| 8 | 63,67 | 16,59 | 0,26 |
| 9 | 61,22 | 9,00 | 0,15 |

**Tabela 1. Linha “COPPEAD”.**

| Trecho | Média (s) | Desvio Padrão | Coeficiente de Variação |
|---|---|---|---|
| 1 | 58,56 | 13,62 | 0,23 |
| 2 | 42,11 | 15,31 | 0,36 |
| 3 | 85,44 | 24,69 | 0,29 |
| 4 | 122,11 | 15,14 | 0,12 |
| 5 | 173,89 | 5,92 | 0,03 |
| 6 | 128 | 9,75 | 0,08 |
| 7 | 127,56 | 10,88 | 0,09 |
| 8 | 80,44 | 11,64 | 0,14 |
| 9 | 27,33 | 8,94 | 0,33 |
| 10 | 97,11 | 13,56 | 0,14 |
| 11 | 61,56 | 7,63 | 0,12 |

**Tabela 2. Linha “Estação UFRJ”.**

Finalmente, a partir da Figura 8 verifica-se que mesmo para o maior trajeto das linhas, do ponto inicial até o ponto final, o erro médio absoluto das estimativas se mantém abaixo de 80 segundos. Com isso, conclui-se que as estimativas fornecidas pelo WiBus têm grande potencial para os usuários dos transportes públicos, já que o erro da estimativa do sistema representa o tempo que o usuário deverá esperar em média no ponto de ônibus até que um ônibus da linha desejada chegue. Nesse contexto, o erro obtido pode ser considerado desprezível frente ao tempo que um usuário sem informação aguardaria.

## 6. Conclusões e Trabalhos Futuros

Este trabalho propôs um sistema para estimativa dos horários de chegada dos ônibus nos seus respectivos pontos. O WiBus utiliza informações obtidas a partir de uma rede sem-fio IEEE 802.11, cujos dispositivos estão instalados nos ônibus e na infraestrutura ao longo das vias, para manter atualizadas as informações de localização dos ônibus. Com o intuito de evitar falhas do sistema no caso de mudanças de trajetória, um algoritmo de criação e manutenção dinâmica de mapas digitais foi proposto e avaliado. O serviço

de estimativa de chegada de ônibus nos respectivos pontos foi baseado em médias móveis simples de tamanho $K$, tendo $K$ sido calculado para duas linhas de ônibus universitárias com base em dados reais. Os resultados obtidos mostram que as estimativas realizadas pelo WiBus apresentam erro da ordem de minuto, mesmo do ponto inicial até o ponto final das linhas de ônibus avaliadas. Os resultados ainda mostram que o WiBus é capaz de atender as exigências de uma cidade grande como o Rio de Janeiro, sendo capaz de tratar mais de 8.500 mensagens por minuto no pior caso. O sistema WiBus possui adicionalmente duas possibilidades de interface com o usuário: uma via página web e a outra via aplicativo para Android. Em ambos os casos, a interface é chamada de BuZoom.

Como trabalhos futuros, pretende-se estender as medidas com mais dados práticos dos tempos gastos pelos ônibus para percorrer trechos de suas linhas. De posse de mais dados, seria possível avaliar mais precisamente a escolha do parâmetro $K$, sendo possível inclusive a criação de um algoritmo dinâmico para ajuste de $K$ em função do erro.

## Referências

Caceres, M., Sottile, F. e Spirito, M. (2009). WLAN-based real time vehicle locating system. Em *Vehicular Technology Conference. VTC Spring 2009. IEEE 69th*, p. 1–5.

Casey, R. F., Labell, L. N., Prensky, S. P. e Schweiger, C. L. (1991). Advanced public transportation systems: the state of the art. Relatório técnico, U.S. Department of Transportation.

Chien, S., Ding, Y. e Wei, C. (2002). Dynamic bus arrival time prediction with artificial neural networks. *Journal of Transportation Engineering*, 128(5):429–438.

de Urbanismo Pereira Passos, I. M. (2011). Total de linhas, frota operante, passageiros transportados, viagens realizadas, quilometragem coberta, combustível utilizado e pessoal ocupado pelo sistema de Ônibus - município do Rio de Janeiro - 1994 - 2011 (tabela nº 1736). Armazém de Dados - Informações sobre a cidade do Rio - `http://www.armazemdedados.rio.rj.gov.br/`.

DENATRAN (2013). Frota - DENATRAN - Departamento Nacional de Trânsito. `www.denatran.gov.br/frota.htm`.

Karbassi, A. e Barth, M. (2003). Vehicle route prediction and time of arrival estimation techniques for improved transportation system management. Em *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, p. 511–516.

Lin, W.-H. e Zeng, J. (1999). Experimental study of real-time bus arrival time prediction with GPS data. *Transportation Research Record: Journal of the Transportation Research Board*, 1666(1):101–109.

Manolis, K. e Kwstis, D. (2004). Intelligent transportation systems - travelers' information systems the case of a medium size city. Em *Mechatronics. ICM '04. Proceedings of the IEEE International Conference on*, p. 200–204.

# Sistema de monitoramento de veículos usando dispositivos no padrão IEEE 802.11p

**Vladimir P. Barcelos**[1], **Thiago C. Amarante**[1], **Carlos D. Drury**[1], **Luiz H. A. Correia**[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Lavras – Lavras-MG, Brasil

vpbarcelos@posgrad.ufla.br, tcamarante@posgrad.ufla.br

carlosdanieldrury@computacao.ufla.br, lcorreia@dcc.ufla.br

***Abstract.*** *In Vehicular Ad hoc Networks (VANETs), data can be transmitted between vehicles and infrastructured devices. This paper proposes the development of communication device capable of operating in the IEEE 802.11p standard, without using dedicated hardware. Due to the high cost and scarcity of dedicated equipment, most researches on VANETs addresses only simulations, and there is a lack of pratical studies in these networks. To validate the actual operation of the communication device, Android and server applications were developed, in order to collect physical data of vehicle (such as engine status, speed and GPS location) and transmit them to a server using the communication device proposed.*

***Resumo.*** *Redes veiculares (VANETs) são redes de transmissão de dados entre veículos e dispositivos infraestruturados. Este trabalho propõe o desenvolvimento de um dispositivo de comunicação capaz de operar no padrão IEEE 802.11p sem precisar utilizar hardwares dedicados. Devido ao custo elevado e a escassez de equipamentos dedicados, a maioria das pesquisas em VANETs abordam apenas simulações, sendo que estudos práticos destas redes ainda são incipientes. Para validar o funcionamento real do dispositivo de comunicação, foram desenvolvidas aplicações para Android e outra para servidor, com a finalidade de coletar dados físicos de um veículo (como estado do motor, velocidade e localização geográfica) e transmiti-los para um servidor utilizando o dispositivo de comunicação proposto.*

## 1. Introdução

Anualmente cerca de 1,24 milhões de pessoas morrem e outras 50 milhões ficam feridas em acidentes de trânsito em todo o mundo [Organization 2013]. Este número poderia ser reduzido se motoristas envolvidos nesses eventos pudessem antever situações de risco ou fossem resgatados em tempo hábil. As redes veiculares *ad hoc* (*Vehicular Ad Hoc Networks* – VANETs) têm como objetivo prover a comunicação de dados entre veículos. Nessas redes são trocadas mensagens sobre as condições de tráfego de veículos, segurança do trânsito (tais como comunicação de acidentes), e/ou mensagens de propósito geral (tais como acesso a Internet ou aplicações de entretenimento).

As VANETs são formadas por diferentes topologias para a comunicação de dados envolvendo veículos. Os nós dessas redes podem ser móveis (veículos) ou fixos

(dispositivos infraestruturados) que têm a finalidade de prover acesso a outras redes. A comunicação nas VANETs pode ser estabelecida exclusivamente entre veículos (V2V), entre veículos e dispositivos de infraestrutura (V2I) ou possibilitar comunicações híbridas (V2X), onde veículos podem se comunicar entre si ou com dispositivos de infraestrutura, conforme necessidade. O dispositivo de comunicação localizado nos veículos é denominado de *On Board Unit* (OBU). Já os dispositivos de comunicação das infraestruturas fixas são denominados *Road Side Unit* (RSU).

As VANETs, ao contrário das tradicionais redes móveis *ad hoc* (*Mobile Ad hoc Networks* – MANETs), possuem características peculiares como: mobilidade dos veículos limitada às pavimentações (ruas, estradas e avenidas), constantes mudanças de trajetória, alta velocidade dos veículos e curto tempo de contato entre os envolvidos na transmissão. Devido a tais características, essas redes possuem diversos desafios a serem explorados para manter a confiabilidade da conexão, minimizar atrasos na entrega das informações, evitar perda de pacotes e manter uma largura de banda suficiente para atender as diferentes aplicações [Cheng et al. 2011, Karagiannis et al. 2011].

Apesar do grande potencial das VANETs a maioria das pesquisas atuais são baseadas principalmente em simulações, ou no desenvolvimento de protocolos de roteamento, reforçando a necessidade de experimentos práticos [Neves et al. 2011]. A justificativa está na escassez e no alto custo dos dispositivos de comunicação veicular, além da alta complexidade envolvida em experimentos reais.

A proposta deste trabalho é desenvolver um sistema completo de monitoramento de veículos, envolvendo hardware e software. Este sistema coleta, processa e transmite informações relacionadas à localização e condições dos veículos, disponibilizando-as na Internet. O sistema de monitoramento desenvolvido é composto por um coletor de dados acoplado à central de processamento do veículo, um *smartphone* que fornece dados de localização do veículo e estabelece conexão com a VANET por meio da OBU. Para a formação da infraestrutura da VANET foram desenvolvidos dispositivos de comunicação de baixo custo, compatíveis com o padrão IEEE 802.11p [IEEE 2010] e que se comunicam com um servidor remoto. Este servidor possui uma aplicação capaz de processar os dados da VANET e apresentar a localização e as condições dos veículos em tempo real.

Resultados mostraram que os dispositivos desenvolvidos realizaram transmissões em um diâmetro de até 700 metros. A uma distância relativa de até 200 metros, a perda de pacotes ficou abaixo de 10% para as comunicações V2V e abaixo de 20% para as comunicações V2I, sendo que o atraso e a taxa de transmissão não apresentaram melhoras significativas em velocidades e distâncias menores. Diferente da taxa média de perda de pacotes, o atraso e a taxa de transmissão sofreram menor degradação ao variar a distância e velocidade dos nós. O sistema proposto foi capaz de receber em tempo real os dados transmitidos pela VANET. Os dados de posicionamento geográfico e estado do veículo foram armazenados e apresentados em um servidor remoto.

Este trabalho está organizado como descrito a seguir. A Seção 2 apresenta os trabalhos relacionados. Na Seção 3, os procedimentos utilizados no desenvolvimento do sistema são apresentados. Os resultados obtidos são expostos e discutidos na Seção 4. Por fim, na Seção 5, são apresentados as conclusões e os trabalhos futuros.

## 2. Trabalhos Relacionados

As VANETs têm como objetivo principal promover a segurança no trânsito [Kamal et al. 2012]. As aplicações relacionadas à segurança no trânsito, como os alertas de provável colisão, exigem que as mensagens transmitidas cheguem até os destinatários no máximo em 100 ms, que é o limite de atraso tolerável para a maioria das comunicações de eventos de emergência [Xu et al. 2004, Consortium 2004]. Para promover regras e padrões de operação das redes veiculares foi definida a arquitetura de comunicação WAVE (*Wireless Access Vehicular Environments*). Essa arquitetura padroniza as camadas física e de controle de acesso ao meio (IEEE 802.11p) [IEEE 2010], bem como define as especificações para as camadas superiores (IEEE1609) [Group 2013].

O padrão 802.11p define uma camada física que opera na faixa exclusiva, denominada *Dedicated Short-Range Communications* (DSRC). Nos Estados Unidos, uma faixa de frequência de 75 MHz (entre 5,850 GHz e 5,925 GHz) foi reservada pelo *Federal Communications Commission* (FCC) [Cheng et al. 2011]. Na Europa, esta faixa de comunicação foi reservada em 2008 pelo *European Telecommunications Standards Institute* (ETSI) e ocupa a faixa de frequência entre 5,860 GHz e 5,900 GHz. A faixa de frequência é dividida em canais de 10 MHz, sendo um canal exclusivo para controle (CCH) e os restantes utilizados para diferentes categorias de serviços da rede (SCH). É possível atingir taxas de transmissão de até 27 Mbps, com velocidade de descolamento dos nós de até 200 km/h. A camada controle de acesso ao meio (MAC) do protocolo 802.11p é baseado no padrão 802.11e, que utiliza o método *Enhanced Distributed Channel Access* (EDCA) com extensão de Qualidade de Serviço (QoS) [IEEE 2010]. Já o padrão IEEE 1609 é composto por quatro documentos que definem uma arquitetura, um conjunto complementar de serviços padronizados e uma interface que coletivamente viabilizam as comunicações veiculares [Gräfling et al. 2010].

O padrão IEEE 802.11p não possui autenticação e associação nas camadas MAC e física pois estes métodos do padrão 802.11 tradicional demoram um tempo grande, maior que 100ms, tornando inviável aplicá-los em redes veiculares [Booysen et al. 2011]. Portanto, o tempo limite considerado nas VANETs deve incluir o envio de uma mensagem crítica, a resposta do dispositivo veicular e a reação do motorista frente a um evento de emergência. A demanda por valores limitados de atraso sugere que redes veiculares baseadas em tecnologias GPRS, 3G ou 4G, sejam utilizadas somente para aplicações não-críticas [Chandrasekaran 2008]. Dessa forma, a utilização combinada do padrão IEEE 802.11p e redes celulares deve ser feita utilizando a rede veicular para notificação de eventos críticos (principalmente aos veículos ao seu redor), deixando a rede celular a cargo de aplicações não críticas ou como uma redundância do padrão IEEE 802.11p.

Apesar da arquitetura WAVE ser uma realidade, existem no mercado poucos dispositivos de redes veiculares prontos para o usuário final. Portanto, aplicações reais e experimentos práticos ainda são escassos. [González et al. 2008] realizaram testes de comunicação entre veículos utilizando dispositivos com cartões Wi-Fi IEEE 802.11b de alta potência e longo alcance. Os resultados foram coletados com dois veículos em movimento e mostraram que foi possível realizar comunicação entre veículos.

[Martelli et al. 2011] realizaram testes no padrão IEEE 802.11p com o hardware dedicado NEC Linkbird-MX. A conectividade de outros dispositivos ao Linkbird foi realizada somente via interface cabeada devido a limitações físicas do hardware. Os resultados

obtidos da troca de *beacons* entre dois veículos proporcionou relevante contribuição para outros experimentos práticos com este padrão.

Nos trabalhos de [Sukuvaara 2012], os resultados mostraram que as especificações propostas pelo padrão IEEE 802.11p puderam ser satisfeitas na prática, proporcionando comunicações V2I, V2V e V2X de maneira eficiente. Os autores também utilizaram o dispositivo NEC Linkbird-MX e mediram o tempo de conexão e a vazão em veículos com diferentes velocidades.

[Agafonovs et al. 2012] propôs uma solução ao elaborar um hardware de comunicação utilizando o padrão IEEE 802.11p. No entanto, foi utilizada a placa de comunicação UNEX DCMA-86P2, uma das únicas existentes no mercado com suporte nativo ao padrão IEEE 802.11p. No Brasil, o custo desta placa, somado aos impostos, eleva consideravelmente o custo, quando comparado às placas dos padrões 802.11a/b/g/n.

Em [Teixeira et al. 2013], uma avaliação prática do padrão foi realizada utilizando dois notebooks equipados com a placa UNEX DCMA-86P2. Latência, *jitter*, vazão, taxa de perda de pacotes e tempo de associação dos nós foram avaliados. Os resultados indicaram que o padrão IEEE 802.11p proporcionou adequadamente comunicações entre dois veículos.

Informações importantes para as aplicações em redes veiculares podem ser extraídas do *On Board Unit II* (OBD-II). O OBD-II é um sistema de sensores criado pela *California Air Resources Board* em 1994, que é capaz de monitorar motor, chassi, corpo e acessórios de carros e caminhões leves [ISO 2005]. Este sistema de sensores está presente em todos os carros fabricados ou importados pelo Brasil a partir de janeiro de 2011, conforme a resolução Conama número 354 de 2004 [Diniz et al. 2009]. Dispositivos coletores, como o ELM327 podem ser conectados ao OBD-II do veículo e acessar suas informações, enviando os dados coletados por meio de uma conexão USB (*Universal Serial Bus*) ou sem fio (*Bluetooth* ou Wi-Fi).

[Baek et al. 2011] desenvolveram uma aplicação para coleta de dados do veículo usando o OBD-II, mas nenhuma comunicação entre veículos foi abordada neste trabalho. [Zaldivar et al. 2011, Wideberg et al. 2012] também utilizaram o OBD-II para coletar informações dos veículos em aplicações de segurança de trânsito. No entanto estes autores utilizaram apenas redes 3G para comunicação dos veículos com um servidor remoto.

Neste trabalho foi desenvolvido um dispositivo de comunicação altamente customizável e de baixo custo, capaz de se comunicar no padrão IEEE 802.11p. O dispositivo pode atuar como OBU ou RSU. Uma aplicação para *smartphones* que coleta e transmite os dados do veículo também foi desenvolvida afim de validar a funcionalidade do dispositivo. A seção a seguir apresenta o sistema de monitoramento de veículos.

## 3. Sistema de monitoramento de veículos

A proposta deste trabalho é desenvolver um sistema completo de monitoramento de veículos que coleta, processa e transmite informações da localização e condições dos veículos, disponibilizando-as via Internet. Este sistema é composto por elementos de hardware e software. Desta forma, é possível ter o conhecimento e domínio de toda a tecnologia envolvida no sistema. A arquitetura de hardware é composta por um coletor

de dados do veículo, um *smartphone*, dispositivos OBUs/RSUs e um servidor remoto. O coletor de dados é acoplado à central de processamento do veículo que transmite os dados para um *smartphone* via *Bluetooth*. O *smartphone* atua como um intermediador, processando e retransmitindo os dados do coletor para a OBU via rede *Wi-Fi*. Essas OBUs/RSUs foram desenvolvidas para realizar comunicação compatível com o padrão IEEE 802.11p [IEEE 2010], que são conectadas ao servidor remoto. O software é composto por uma aplicação Android executada no *smartphone* e outra aplicação que é executada no servidor remoto.

### 3.1. Dispositivos de comunicação

O processo de desenvolvimento do hardware de comunicação veicular visou atender alguns requisitos: custo acessível, capacidade de customização, capacidade de operar tanto como OBU como RSU, robustez para operação em cenários extremos, tamanho compatível para instalação em veículos e executar um sistema operacional e programas de código aberto.

Para atender estes requisitos, foi adotado uma placa RouterBoard modelo RB433AH. RouterBoard é o nome dado a uma série de equipamentos de rádio ou roteadores da fabricante MikroTik. São projetadas primariamente para provedores de Internet oferecendo acesso banda larga via rede sem fios, suportando alta capacidade de tráfego. Este modelo possui três portas *ethernet*, três *slots* para cartões miniPCI, um *slot* microSD para cartões de memória e uma interface serial RS232.

Foram preparadas quatro RouterBoards: uma configurada como RSU, instalada em um ponto fixo, e as outras instaladas em veículos (OBUs). Nas que atuavam como OBU, foram instaladas dois cartões miniPCI: um para prover uma rede local, interna ao veículo, no padrão IEEE 802.11g e outro para comunicar com os veículos e dispositivos de infraestrutura no padrão IEEE 802.11p. O custo estimado para montagem de cada dispositivo OBU/RSU é de aproximadamente 150 dólares. A arquitetura geral do dispositivo de comunicação é mostrada na Figura 1.

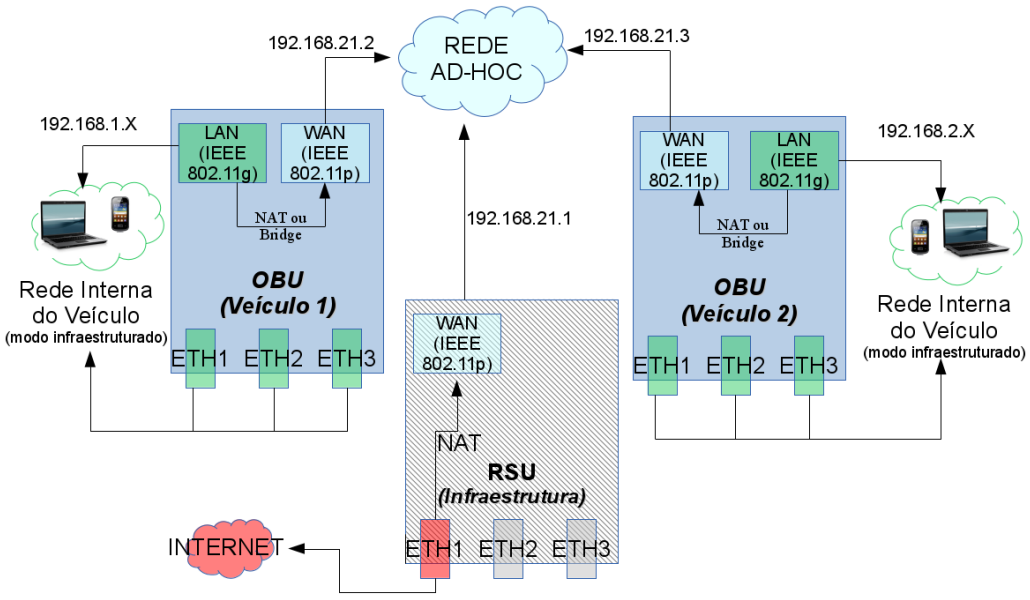



**Figura 1. Arquitetura do dispositivo de comunicação proposto.**

As especificações do cartão IEEE 802.11a/b/g utilizado na VANET, indicaram um modo de operação em frequências de até 6,1GHz. O dispositivo OBU/RSU foi configurado com as mesmas especificações do padrão IEEE 802.11p. Para isso, um *driver* de dispositivo foi modificado. A transmissão de *beacons* foi removida na camada MAC para otimizar o tempo de contato dos nós. Na arquitetura WAVE a operação multicanal é proposta pelo padrão IEEE 1609.4. A camada física do dispositivo opera a uma taxa de transmissão fixa de 6 Mbps, frequência de 5,890 GHz e banda de 10 MHz, correspondendo ao CCH.

Por padrão, a RouterBoard roda o sistema operacional proprietário da Mikrotik, o RouterOS. No entanto, este sistema operacional foi substituído por uma distribuição Linux voltada para roteadores, o OpenWRT [Team 2013].

Dois protocolos de roteamento foram avaliados para serem utilizados nos dispositivos de comunicação. O OLSR (*Optimized Link State Routing Protocol*) e o BATMAN (*Better Approach To Mobile Ad-hoc Networking*). O BATMAN foi o eleito, baseado em simulações e outros experimentos práticos no padrão IEEE 802.11p (não apresentados neste artigo). Os resultados indicaram que o protocolo BATMAN proporcionou menores taxas de perdas e atrasos. O problema encontrado no funcionamento do OLSR é a latência elevada para detectar entradas e saídas de nós da rede, o que não ocorre com o BATMAN, já que ele é um protocolo distribuído [Dias 2012].

### 3.2. Aplicação Android e Servidor

Uma aplicação prática foi desenvolvida com a finalidade de testar o dispositivo de comunicação. Foi proposto um conjunto de softwares para coletar, transmitir e tratar informações geradas pelos veículos. A aplicação Android é executada no *smartphone* e realiza as tarefas de coleta de dados do motor e de localização GPS, transmitindo-os para uma outra aplicação executada em um servidor remoto.

O aplicativo Android foi desenvolvido utilizando as linguagens de programação JAVA e XML juntamente com o kit de desenvolvimento Android SDK. A aplicação permite que o *smartphone* atue como um *middleware*, estabelecendo uma comunicação via *Bluetooth* com um dispositivo ELM327 previamente conectado na interface OBD-II do veículo. A coleta de dados do motor (temperatura, RPM, velocidade) é realizada juntamente com a posição do veículo, detectada via GPS interno do *smartphone*. Este também estabelece uma conexão com a OBU, via rede interna do veículo no padrão IEEE 802.11g. O dispositivo OBU atua como um *gateway*, estabelecendo a comunicação do *smartphone* com o servidor remoto. Uma visão do funcionamento do sistema é mostrado na Figura 2. O funcionamento da aplicação Android é apresentado no Algoritmo 1.

As informações são enviadas ao servidor por meio de mensagens JSON *(JavaScript Object Notation)*, formatadas da seguinte maneira:

```
[{"carId":1,"gpsDate":1386283974068,"latitude":"-21.227979",
"longitude":"-44.9786027","rpm":1243 RPM,"speed":18.3 km/h}]
```

O servidor executa uma aplicação WEB desenvolvida com linguagens Groovy e *framework* Grails. As informações recebidas são salvas em um banco de dados PostgreSQL, onde o histórico do veículo fica armazenado. Cada veículo é identificado por um id único na rede. Por meio das informações coletadas, é possível plotar a localização

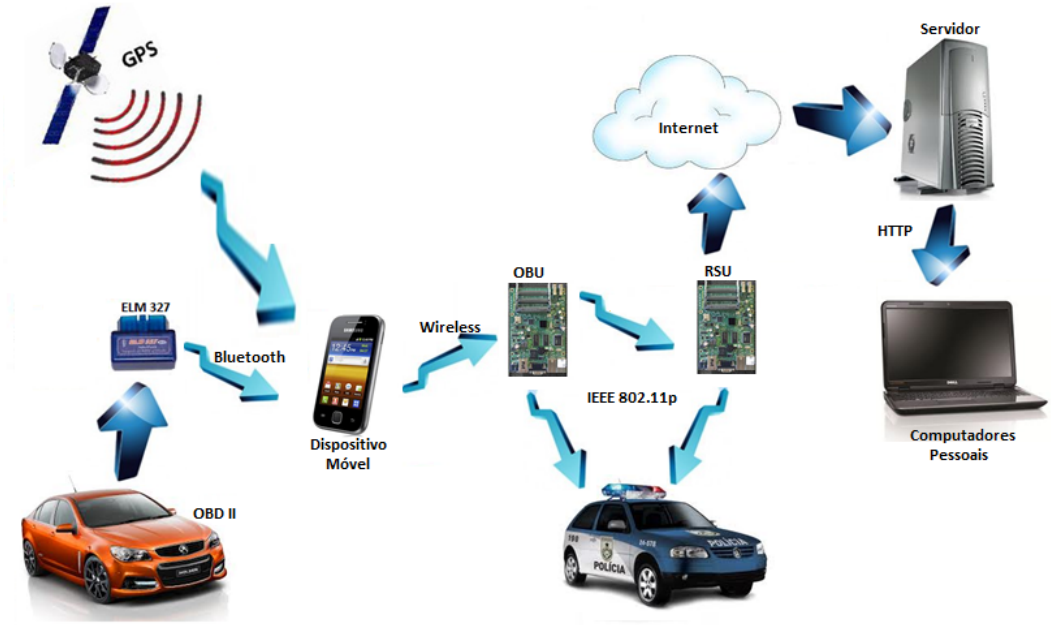



**Figura 2. Fluxo de informações geral da arquitetura.**

```
1  Procedure coletaTransmite()
2  Ativa o bluetooth;
3  Ativa o wireless;
4  Ativa o GPS;
5  Sincroniza hora do smartphone com a do sistema de GPS;
6  Estabelece conexão com o dispositivo ELM327 previamente pareado;
7  Conecta a rede Wi-Fi interna do veículo;
8  Estabelece conexão com o servidor;
9  Ativa o serviço de monitoramento;
10 Thread do serviço é iniciada;
11 while thread do serviço ativa do
12     Coleta informações solicitadas ao ELM327;
13     Coleta latitude e longitude;
14     Envia todos os dados coletados ao servidor.
```

**Algoritmo 1**: Mecanismo de coleta e transmissão de dados do veículo.

do veículo em um mapa em tempo real. Para exibição do mapa utilizou-se linguagem Javascript e a API (*Application Programming Interface*) do Google Maps. A aplicação no servidor pode ser acessado via *browser*. O servidor grava todas as informações no banco de dados, onde o histórico do veículo fica armazenado.

## 4. Resultados

Foram avaliados em cenários reais os dispositivos de comunicação da VANET (OBU/RSU) operando no padrão IEEE 802.11p. Foram consideradas as métricas de perda de pacotes, atraso de entrega e taxa de transmissão. O sistema de monitoramento de veículos obteve sucesso ao receber os dados coletados no veículo, armazená-los e disponibilizá-los via web.

### 4.1. Cenários Avaliados

O cenário utilizado para a realização dos experimentos foi a avenida sul da Universidade Federal de Lavras mostrada na Figura 3. Para avaliar a comunicação V2V, dois veículos foram utilizados, cada um posicionado em cada extremo da avenida (pontos 2 e 4) e eles iniciaram seus deslocamentos no mesmo instante, indo de encontro um do outro. Os experimentos foram realizados nas velocidades de 20km/h, 40km/h e 60km/h.

Um cenário híbrido foi adotado nos experimentos de comunicações entre veículos e dispositivo de infraestrutura. Um veículo estacionado no ponto 2 atuou como um intermediador das comunicações, visto que o prédio localizado na frente do ponto 1 prejudica a difusão do sinal da infraestrutura. Um veículo iniciou o deslocamento em uma das extremidades da avenida (ponto 4), mantendo também as velocidades de 20km/h, 40km/h e 60km/h. A avenida e uma ilustração da disposição dos nós é mostrado na Figura 3. A distância entre os pontos 3 e 4 é de aproximadamente 792 metros, sendo que do ponto 2 ao ponto 4 são 465 metros.

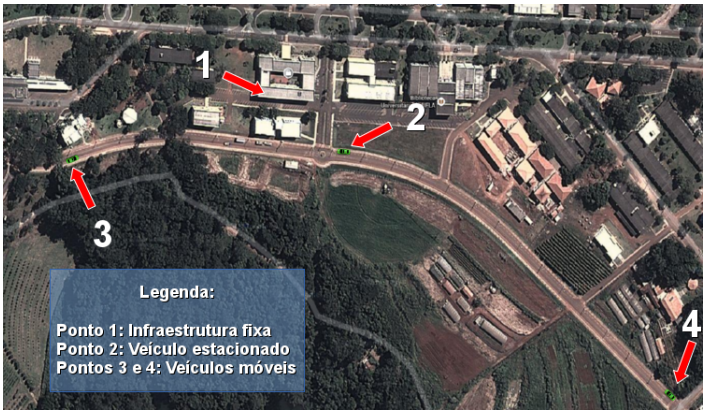



**Figura 3. Mapa de satélite da avenida utilizada nos experimentos.**

### 4.2. Métricas Avaliadas

Nos cenários mencionados, a avaliação da rede foi realizada medindo o tempo de atraso dos pacotes, a taxa de perda e taxa de transmissão. Para o atraso da transmissão, foi medido o momento em que a pacote foi transmitido até o momento em que chegou ao receptor. Em relação a taxa de perda de pacotes foi comparado o número de pacotes transmitidos com o número de pacotes efetivamente recebidos. Os dados foram obtidos utilizando uma versão modificada do software de medição bwping, que disparou pacotes UDP com 512 bytes a uma taxa de transmissão de 2048 kbps. Quatro repetições foram realizadas para cada experimento. A posição geográfica do veículo foi registrada durante a realização dos experimentos. Essas métricas são úteis para estimar o estado da rede, detectando perda de informações e atrasos nas transmissões.

### 4.3. Avaliação de desempenho da rede

Os resultados foram extraídos de quatro repetições para cada experimento, no cenário apresentado na Figura 3 e conforme descrito na seção anterior. O intervalo de confiança considerado foi de 95%, mas não são representados no gráfico para facilitar a disposição das informações.

A taxa de perda de pacotes foi avaliada para os cenários V2V e V2I nas velocidades de 20 km/h, 40 km/h e 60 km/h. A Figura 4 apresenta a taxa de perdas das comunicações V2I, e a Figura 5 para V2V. Nos gráficos, a distância negativa significa aproximação do veículo ao nó destino e a distância positiva representa seu afastamento.

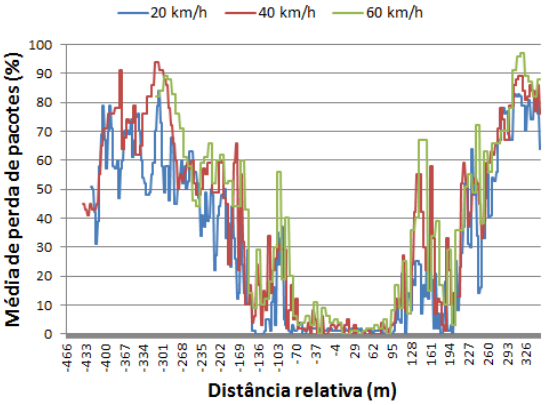



**Figura 4. Média de perda de pacotes (V2I).**

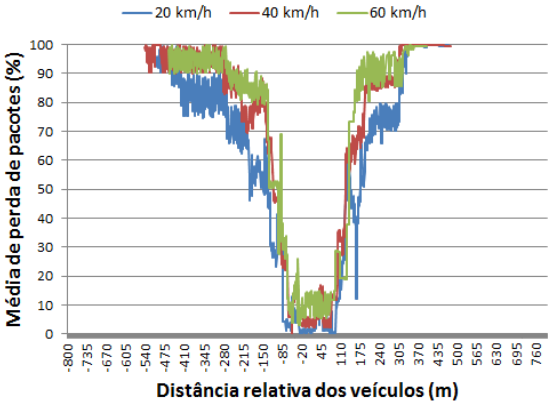



**Figura 5. Média de perda de pacotes (V2V).**

Os dados obtidos em diferentes velocidades mostram que a rede se comporta de maneira mais robusta em velocidades menores. Foi possível realizar a transmissão em um diâmetro de aproximadamente 700 metros. Quanto mais próximo o veículo está do nó receptor, menor é a perda de pacotes. Quando os nós estão a uma distância relativa de até 200 metros, a perda de pacotes ficou abaixo de 10% para as comunicações V2V e abaixo de 20% para as comunicações V2I. A velocidade influencia na perda de pacotes, diretamente proporcional a velocidade dos nós.

O atraso na transmissão dos pacotes foi medido nos cenários V2V e V2I para as velocidades de 20 km/h, 40 km/h e 60 km/h. A Figura 6 apresenta os atrasos para as comunicações V2I e a Figura 7 para V2V. O atraso foi medido considerando apenas os pacotes efetivamente transmitidos.

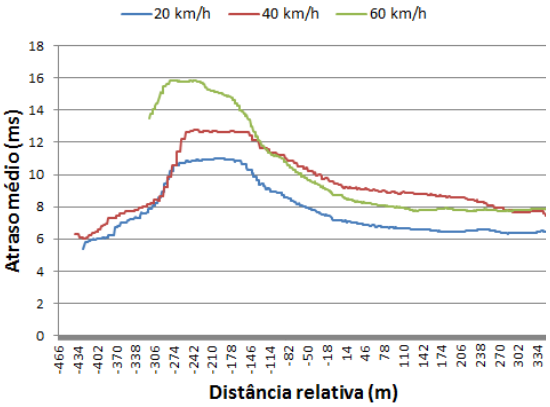



**Figura 6. Atraso médio (V2I).**

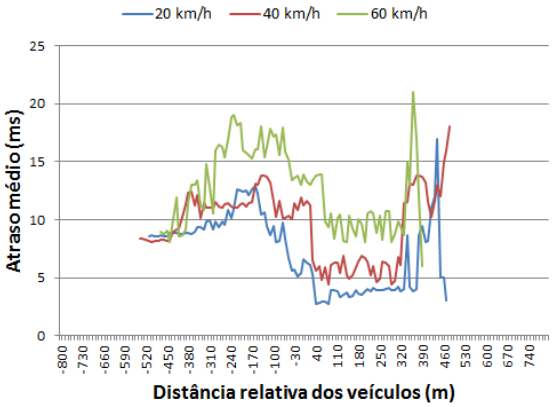



**Figura 7. Atraso médio (V2V).**

Os dados obtidos das três avaliações em diferentes velocidades mostram que nas comunicações entre veículos o atraso variou de forma mais intensa. Isto se deve ao des-

locamento de ambos os veículos que torna a comunicação mais instável se comparado as comunicações V2I. O atraso médio não variou de forma significativa em relação a distância. Os valores obtidos quando os nós estavam em pontos distantes variaram pouco em relação ao atraso obtido quando os nós estavam próximos. Em todas as repetições realizadas, a média do atraso foi inferior a 100 ms. Foi verificado que ao aumentar a velocidade, o atraso nas comunicações também sofre incremento. Nas comunicações entre veículos na velocidade mais alta, a velocidade relativa dos nós foi de 120 km/h. Neste cenário os atrasos foram maiores que nos outros.

A taxa de transmissão de dados foi avaliada para os cenários V2V e V2I nas velocidades de 20 km/h, 40 km/h e 60 km/h. A Figura 8 apresenta as taxas obtidas nas comunicações V2I, e a Figura 9 mostra os resultados obtidos para as comunicações V2V.

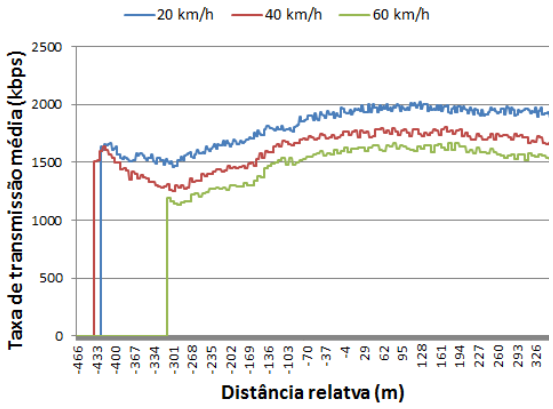



**Figura 8. Taxa de transmissão média (V2I).**

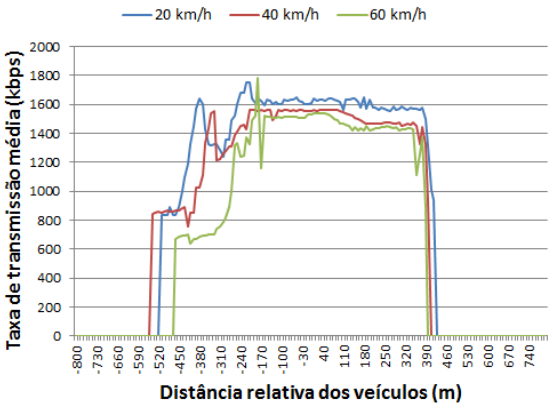



**Figura 9. Taxa de transmissão média (V2V).**

Os dados obtidos das três avaliações em diferentes velocidades mostram que nas comunicações entre veículos, a taxa de transmissão permaneceu relativamente constante, independe da distância relativa dos nós. A taxa de transferência oscilou no início, ao se estabelecer a comunicação, mas depois se estabilizou. A taxa de transmissão variou de forma mais intensa nas comunicações V2V devido ao deslocamento de ambos os veículos, o que torna a comunicação mais instável quando comparado às comunicações V2I. A taxa de transmissão média não variou de forma significativa em relação à distância.

### 4.4. Aplicação

A aplicação Android, mostrada na Figura 10 utiliza uma *thread* para atualizar os dados lidos do veículo a cada 250 ms. Posteriormente esses dados são enviados para um servidor remoto que os armazena em banco de dados. Para que o usuário utilize corretamente a aplicação, é necessário inicializá-la (*start*), conhecer o IP e a porta de comunicação do servidor. O usuário poderá ver os dados na própria tela, em que são apresentadas informações de velocidade, rotação do motor, temperatura e localização de posicionamento global em latitude e longitude.

O *smartphone* conectado a OBU transmitiu os dados corretamente ao servidor. A Figura 11 apresenta o mapa gerado no servidor com as localizações do veículo em tempo real, bem como as informações do estado do veículo.


**Figura 10. Aplicação Android em execução.**

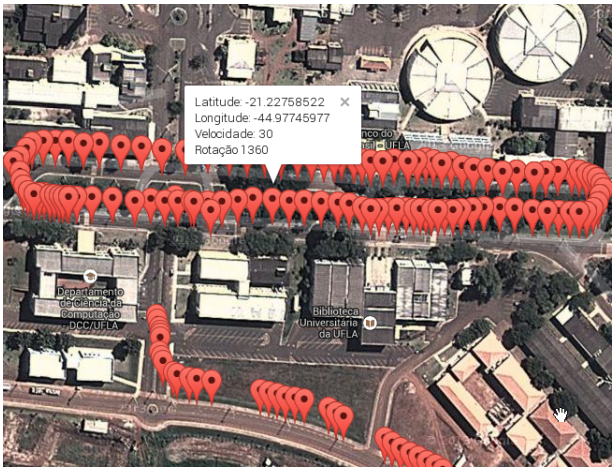



**Figura 11. Dados de trajetória coletados no servidor remoto**

## 5. Conclusões e Trabalhos Futuros

O dispositivo OBU/RSU de baixo custo desenvolvido neste trabalho permitiu com sucesso a comunicação dos veículos no padrão IEEE 802.11p em um raio de até 450m aproximadamente. A taxa de perda de pacotes manteve-se ligeiramente menor em distâncias de até 200 metros sendo que para distâncias maiores, esse valor incrementa significativamente. O atraso médio das comunicações foi inferior a 100ms, o que permite a utilização destes dispositivos em aplicações de segurança de trânsito. Apesar disso, o atraso incrementa com o aumento da velocidade. A taxa de transmissão manteve-se relativamente constante para cada uma das velocidades avaliadas. A aplicação Android coletou os dados do veículo com sucesso e os transmitiu para o servidor remoto sempre que OBU tivesse conexão. O protótipo de software está em desenvolvimento constante, visando se tornar uma ferramenta de monitoramento de veículos, identificando padrão de comportamento, histórico de condução e identificando/antecipando potenciais acidentes.

Como trabalhos futuros, além do aprimoramento da aplicação, pretende-se agregar um módulo de comunicação 3G para atuar como uma redundância do link principal IEEE 802.11p. Pretende-se estender os estudos, instalando uma OBU em vários veículos de uma frota de longa duração, com o propósito de avaliar mais profundamente o comportamento desta rede. Com o aprimoramento das pesquisas, esta tecnologia terá potencial para indústria, contribuindo para a popularização das VANETs no mercado em geral.

## 6. Agradecimentos

Os autores agradecem o apoio das agências de pesquisa CAPES, FAPEMIG e CNPq.

## Referências

Agafonovs, N., Strazdins, G., and Greitans, M. (2012). Accessible, customizable, high-performance ieee 802.11p vehicular communication solution. In *Ad Hoc Networking Workshop (Med-Hoc-Net), 2012 The 11th Annual Mediterranean*, pages 127–132.

Baek, S.-H., Kim, H.-S., Jeong, D.-W., Kim, M.-J., Park, Y.-S., and Jang, J.-W. (2011). Implementation vehicle driving state system with obd-ii, most network. In *Communications (APCC), 2011 17th Asia-Pacific Conference on*, pages 709–714.

Booysen, M., Zeadally, S., and van Rooyen, G.-J. (2011). Survey of media access control protocols for vehicular ad hoc networks. *Communications, IET*, 5(11):1619–1631.

Chandrasekaran, G. (2008). Vanets: The networking platform for future vehicular applications. *Department of Computer Science, Rutgers University*.

Cheng, H. T., Shan, H., and Zhuang, W. (2011). Infotainment and road safety service support in vehicular networking: From a communication perspective. *Mechanical Systems and Signal Processing*, 25(6):2020 – 2038. Interdisciplinary Aspects of Vehicle Dynamics.

Consortium, C. A. M. P. V. S. C. (2004). *Vehicle Safety Communications Project: Task 3 Final Report: Identify Intelligent Vehicle Safety Applications Enabled by DSRC*. National Highway Traffic Safety Administration, Office of Research and Development, Washington, D.C.

Dias, J. F. (2012). Mobilidade em comunicações veiculares.

Diniz, I. S., Colón, D., da Silva, B. N., and de Aguiar and, F. P. (2009). Scanner automotivo wireless. In *XIII Congresso Internacional e Exposição Sul-Americana de Automação - Brazil Automation ISA*.

González, V., Santos, A. L., Pinart, C., and Milagro, F. (2008). Experimental demonstration of the viability of ieee 802.11b based inter-vehicle communications. In *Proceedings of the 4th International Conference on TridentCom*, pages 1:1–1:7.

Gräfling, S., Mahonen, P., and Riihijärvi, J. (2010). Performance evaluation of ieee 1609 wave and ieee 802.11p for vehicular communications. In *Ubiquitous and Future Networks (ICUFN), 2010 Second International Conference on*, pages 344–348.

Group, I. . W. (Novembro, 2013). IEEE 1609 Working Group Public Site. http://vii.path.berkeley.edu/1609_wave/.

IEEE (2010). IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments.

ISO (2005). Std 15031-6 road vehicles - communication between vehicle and external equipment for emissions-related diagnostics - part 6: Diagnostic trouble code definitions. Iso, International Organization for Standardization, Geneva, Switzerland.

Kamal, F., Lou, E., and Zhao, V. (2012). Design and validation of a small-scale 5.9 ghz dsrc system for vehicular communication. In *Electrical Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, pages 1 –4.

Karagiannis, G., Altintas, O., Ekici, E., Heijenk, G., Jarupan, B., Lin, K., and Weil, T. (2011). Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *Communications Surveys Tutorials, IEEE*, 13(4):584 – 616.

Martelli, F., Renda, M., and Santi, P. (2011). Measuring ieee 802.11p performance for active safety applications in cooperative vehicular systems. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1–5.

Neves, F., Cardote, A., Moreira, R., and Sargento, S. (2011). Real-world evaluation of ieee 802.11p for vehicular networks. In *Proceedings of the Eighth ACM International Workshop on Vehicular Inter-networking*, VANET '11, pages 89–90, New York, NY, USA. ACM.

Organization, W. H. (2013). *Global Status Report on Road Safety 2013: Supporting a Decade of Action*. World Health Organization.

Sukuvaara, T. (2012). Field measurements of ieee 802.11p based vehicular networking entity. In *Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on*, pages 135–139.

Team, O. D. (December, 2013). Openwrt website. https://openwrt.org.

Teixeira, F. A., Silva, V. F., Leoni, J. L., Santos, G. C., Souza, A., Macedo, D. F., and Nogueira, J. M. S. (2013). Análise experimental de redes veiculares utilizando o padrão ieee 802.11p. In *Anais do V Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP)*, page 10, Maceió. SBCUP.

Wideberg, J., Luque, P., and Mantaras, D. (2012). A smartphone application to extract safety and environmental related information from the obd-ii interface of a car. *International journal of vehicle modelling and testing*, 7(1):1–11.

Xu, Q., Mak, T., Ko, J., and Sengupta, R. (2004). Vehicle-to-vehicle safety messaging in dsrc. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, VANET '04, pages 19 – 28, New York, NY, USA. ACM.

Zaldivar, J., Calafate, C., Cano, J.-C., and Manzoni, P. (2011). Providing accident detection in vehicular networks through obd-ii devices and android-based smartphones. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pages 813–819.

# A Multi-flow-driven Mechanism to Support Live Video Streaming on VANETs

**Carlos Quadros**[1], **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.*** *Vehicular Ad Hoc Networks (VANETs) promise a wide scope of real-time multimedia services, which need to deal with strict quality level requirements and highly dynamic network topology. To handle these challenges, multi-path routing approaches have been applied to bring improvements in Quality of Experience (QoE) levels and support on-road real-time video delivery. Though, in real situations, there will likely be multiple streams simultaneously, which can cause more congestion periods and packet loss rates. This paper proposes the Multi-flow-driven VIdeo DElivery (MVIDE) mechanism to select best routes for live video sequences in VANETs. MVIDE can be integrated with routing protocols to define routes considering the characteristics of multiple paths, vehicle mobility, and application requirements. This cross-layer mechanism ranks the quality of candidate paths based on the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method and, subsequently, assigns paths to different substreams. MVIDE was added to the GPSR-MA protocol, being called GPSR-MA-MVIDE. Simulating results show that GPSR-MA-MVIDE outperfoms GPSR-MA in terms of multi-flow handling and received video quality.*

## 1. Introduction

Vehicular Ad Hoc Network (VANET) is an emerging networking technology that uses moving vehicles to create a high mobility and wide range wireless network. Furthermore, with the rapid development of standards such as IEEE P1609 (WAVE), DSRC, and IEEE 802.11p, VANETs become a reality. As a result of that process, this technology has received a lot of attention in recent years and is targeted to support new services, including on-road multimedia safety and security and entertainment video flows [Felice et al. 2013].

Video stream supporting is an attractive service for VANET applications. Vehicles can cooperate among themselves to disseminate videos of dangerous situations allowing users and authorities (firefighters and paramedics) more meaningful information than scalar data, e.g., text messages [Piñol et al. 2012]. Nevertheless, transmission of such applications over VANETs still suffers from several open issues due to highly dynamic network topology and requirements of Quality of Experience (QoE) and Service (QoS) typical of multimedia applications. For instance, source, destination or relay vehicles can change direction abruptly, leading to a frequent and sometimes immediate disconnection.

To troubleshoot broken connection problems, multipath-based approaches can be employed as a promising alternative, since data path diversity is commonly abundant during peak periods of day in VANETs. Distribution of flows between two or more paths allows the network to achieve higher availability, load balancing, and resilience [Wang et al. 2012] [Mérindol et al. 2011]. In addition, a well-known technique to provide multistream coding suitable for transport over multiple paths is Multi-Description Coding (MDC) that creates two or more independent layers decodable in a stand-alone fashion [Wang et al. 2005]. MDC enables the traffic to explore the path diversity of the underlying VANET by sending alternative subtreams of the video along different routes. It guarantees better error resilience in cases of broken links, while increasing the packet delivery ratio and the human experience when watching a live video sequence [Qadri et al. 2012].

An issue that remains untouched in recent studies consists of the traffic distribution of network [Hashemi and Khorsandi 2012]. With the number of flows increasing, the probability of collision and buffer overflow in some vehicles will increase dramatically, and the network performance will degrade catastrophically [Zhuang and Ismail 2012]. Control schemes enable the network to distribute the flows between the nodes and thus avoiding these problems, as well as redirecting the traffic to other alternatives routes. The usage of underloaded nodes along the network allows end-users to receive better video quality. In addition, most of recent work have investigated only the route discovery process [Rezende et al. 2012] [Wahab et al. 2013], without considering different criteria concerning to network status, fair distribution and simultaneous multi-flows.

This paper proposes a cross-layer multi-criteria mechanism based on the vehicle mobility, current network conditions of each path, and application requirements to select multiple routes for live videos in VANETs, named Multi-flow-driven VIdeo DElivery (MVIDE). This mechanism can be integrated with a routing protocol and aims to improve the usage of scarce wireless resources, while increasing the human experience receiving video sequences. We design MVIDE as a Multiple-Criteria Decision-Making (MCDM) problem represented by the well-know Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) to conduct proper path ranking [Hwang and Yoon 1981]. TOPSIS is suitable for VANET scenarios by presenting high performance for multicriteria decisions in telecommunication and engineering environments [Naderi et al. 2013] [Taleb et al. 2012]. MVIDE divides each streaming into MDC substreams, which are forwarded to the best available routes. In case of overload in any node, MVIDE trigger the previous nodes to select a new path with available resource to keep higher video quality level from the user point-of-view. According to performance evaluation, MVIDE outperforms single-path approach and no multi-flow handling solution when number of flows increases. We measure the impact and benefits of MVIDE through simulation experiments with real video sequences in multi-path VANETs scenarios.

The remainder of this paper is organized as follows. Section 2 presents relevant related work. Section 3 introduces the MVIDE mechanism. Section 4 describes the coupling of MVIDE with the Greedy Perimeter Stateless Routing protocol with Movement Awareness (GPSR-MA). Simulation setup and evaluation results comparing GPSR-MA and GPSR-MA with MVIDE are presented in Section 5. Finally, conclusions are summarized in Section 6.

## 2. Related Work

The pervasiveness of mobile devices equipped with positioning systems (e.g., GPS) has significantly grown recently. In addition, high mobility of VANETs is shifting the attention of topology-centric to location-based protocols. Instead of their network addresses, geographic positions identify end-points of the communication [Felice et al. 2013]. Consequently, several work have investigated path discovery and dissemination of messages without concern about network status, multiple flows, and fair distribution of traffic in nodes [Xie et al. 2007].

Besides these work, in [Wang et al. 2012], authors extend the idea presented in [Rezende et al. 2012], developing a Location-aware Multipath Video Streaming scheme (LIAITHON) for video transmission in urban scenarios. This scheme uses positioning information, a route coupling prevention mechanism and a waiting time calculation to discover two relatively short paths with minimum route coupling effect. Although improvements regarding underlying single-path solution, the proposed scheme does not address the impact of distributing multiple video flows in VANETs. Therefore, coupling effects occur if there is more of one flow traversing the same communication range.

[Razzaq and Mehaoua 2010] suggest the use of Scalable Video Coding (SVC), multipath transmission, and a MCDM method called Grey Relational Analysis (GRA), to ranking and prioritizing important frames by transmitting them over the most suitable ranked routes. This work divides video stream to a base layer and enhanced layers then assigns path to different layers according to their importance aiming maximize throughput. With the usage of SVC scheme and GRA method, this approach operates well in term of packet delivery and overhead, however, a basic problem with SVC consists of complex interdependencies between the n-layers. Indeed, to decode a frame, the base layer must be obtained completely before reconstruction of enhancements layers [Xing and Cai 2012]. Thus, this work cannot guarantee the delivery of the intact base layer, which is necessary in SVC schemes. Furthermore, it does not ensure acceptable packet delivery time by using network coding at both source and intermediate nodes.

In urban scenarios, it will eventually lead to several streams being transmitted simultaneously. Once all the nodes in the network use the same routing strategy, a large amount of data will follow on more attractive nodes. Hence, it is vital to consider problems as higher congestion and loss ratio due to overhead on those nodes. [Hashemi and Khorsandi 2012] proposed a VANET Load Balanced Routing protocol (VLBR). It defines a traffic balancing between potential paths by attaining congestion feedback from the network and switching to lower congested routes by utilizing k-Shortest Paths algorithm. However, this work does not use video streaming and does not consider mobility and quality level parameters to path rankings.

In [Granelli et al. 2007] and [Granelli et al. ], authors present the Greedy Perimeter Stateless Routing protocol with Movement Awareness (GPSR-MA), which from mobility parameters, retrieves information about the state of nodes in the network, such as speed, distance from destination, and direction. This proposal cover many aspects and takes into account several parameters, nevertheless, it is interesting and advantageous to consider not only mobility but also quality level parameters in the the path selecting process. In order to show the benefits of MVIDE, The GPSR-MA will be used as case study and underlying routing protocol in this work to apply MVIDE mechanism, as discussed in

Section 4. Therefore, our objective is to combine different independent criteria, including quality level parameters, into TOPSIS method for the path ranking. Further, the proposed mechanism also allows better distribution of the concurrent flows, when overload is detected in any node or route, hence the main contribution of this paper.

## 3. Multi-flow-driven VIdeo DElivery (MVIDE)

This section describes the proposed mechanism called MVIDE to enhance the control of simultaneous video flows in VANETs. The mechanism works jointly with an underlying routing geocast protocol and considers mobility and quality level attributes to transmit multiple video flows, while exploring the multipath capabilities of VANETs. Depending on the routing protocol, other different parameters can be incorporated aiming a more robust decision process. MVIDE occurs in three stages, namely Path Ranking, MDC Forwarding Decision Process, and Multi-flow Handling. The Path Ranking stage calculates and ranks the robustness and stability of all available paths. The MDC Forwarding Decision Process stage performs the forwarding process sending MDC pairs between chosen paths. The Multi-flow Handling stage obtains warnings from the monitor of traffic level, allowing dynamic change to other paths in case of congestion periods.

We consider the usage of video delivery on a VANET environment where there are $k$ vehicles (nodes) moving over a grid area, each node has an identifier ($i \in [1, k]$), and the spatial distribution of nodes does not change very quickly in a short period of time. The combination of those nodes configures a dynamic graph $G(V, E)$, 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 a queue $(Q)$ with a maximum queue capacity $(Q_{max})$ and current queue length $(Q_{length})$. The queue policy schedules the packet transmission by using First In First Out (FIFO) and drops packets by using Drop Tail in case of buffer overflow. Each node $v_i$ has a positioning system, such as GPS, so that it is aware of location and accurately synchronized in time.

Through positioning and traffic periodic information exchanged between a subset $N(v_i)$, each node $v_i$ is able to estimate the follows parameters: 1-hop distance $(d)$, direction $(\theta)$, speed $(s)$, delay $(\tau)$, and the buffer level $(\phi)$ of its neighbors. Thus, each node $v_i$ is aware of the latest state $S(d, \theta, s, \tau, \phi)$ of its neighborhood. For a given $G(V, E)$, which represents a multi-hop and multi-flow network, there is a set of real-time video flows $F = \{f_1, f_2, ..., f_n\} | n \leq \frac{k}{2}$. Furthermore, this network is composed of a finite subset of source nodes $VS = \{vs_1, vs_2, ..., vs_n\}$ and a finite subset of destination nodes $VD = \{vd_1, vd_2, ..., vd_n\}$, where $\forall\, vs_j \in VS\ \exists!\, vd_j \in VD | (j \in [1, n])$. Thus, for each $vs_j$ there is a respective $vd_j$ resulting in $n$ source-destination pairs $(VS, VD) \subset V$, and the $vd_j$ position is known. Each stream $f_j$ has different sizes and characteristics and its transmission from $vs_j$ to its respective $vd_j$ demands a transmission request from $vd_j$ composed of the $vsj$ identifier (logical address), its location $(x_{vs_j}, y_{vs_j})$ and the $f_j$ identifier.

The node state $S(d, \theta, s, \tau, \phi)$ enables MVIDE to define and select different possible paths, taking into account the topology dynamism to adjust each flow for increasing the link resilience. Furthermore, MVIDE redirects flows to underloaded paths in case of congestion likelihoods. Thus, it supports robust real-time video distribution and, consequently, a better QoE support for video streaming in multi-flow scenarios. We will present

each stage in detail in the following subsections.

### 3.1. Path Ranking

The Path Ranking stage allows MVIDE to calculate, among available paths, routes with better link quality and stability. MVIDE considers performance metrics and location features $S(d, \theta, s, \tau, \phi)$, previously defined, for obtaining the state information of each node, as factors in the TOPSIS method. It consists of a multi-criteria method to classify among a set of alternatives the best one, so that the chosen alternative should not only have the shortest distance from 'positive ideal' reference point ($A^+$), but also the farthest distance from a 'negative ideal' reference point ($A^-$). $A^+$ has the most benefits and lowest cost of all alternatives, and $A^-$ is the one with the lowest benefits and highest cost [Hwang and Yoon 1981]. In MVIDE, the TOPSIS alternatives $A = \{A_1, A_1, ..., A_m\}$ mean all paths in which it is possible to calculate their respective factors $n = 5$, thus the number of alternatives $m$ varies according to the available paths.

The TOPSIS operation is expressed in a series of steps [Naderi et al. 2013]. Thus, given a source-destination pair $(vs_j, vd_j) \in V$ and a flow $f_j$, respectively, MVIDE firstly selects paths for the flow $f_j$. Each flow $f_j$ contains a subset of forwarding nodes $VF_j = \{vf_{j,1}, vf_{j,2}, ..., vf_{j,z}\} \subset V$ that establishes routing paths from $vs_j$ to $vd_j$ and transports a subtream MDC originated from $f_j$. Thereby, it is built the decision matrix $D$ (Eq. (1)), where $x_{pq}$ describes the value of alternative $A_p|(p \in [1, m])$ and factor $C_q|(q \in [1, n])$.

$$D = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \cdots & C_n \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \end{array} \tag{1}$$

MVIDE selects only one path for each MDC substream, as shown in Subsection 3.2. Thus, for each ranking, the best disjoint paths are chosen from matrix D by using linear combination. In addition, the raw data are normalized to eliminate differences by measurement units and scales, similarly to several MCDM problems. Decision matrix R receives the normalized elements of matrix D by using Eq. (2):

$$r_{pq} = \frac{x_{pq}}{\sqrt{\sum_{p=1}^{n} (x_{pq})^2}} \quad p = 1, 2, ..., m \quad and \quad q = 1, 2, ..., n \tag{2}$$

At this point, considering the different weight $\omega_q$ for each factor, the weighted normalized decision matrix W is calculated by multiplying the values $r_{pq}$ in R and the importance weights of evaluation criteria. As result, W is defined as Eq. (3):

$$W = [w_{pq}]_{m \times n} | w_{pq} = r_{pq} \times \omega_q, \quad p = 1, 2, ..., m \quad and \quad q = 1, 2, ..., n \tag{3}$$

Using the measures in the matrix W, $A^+$ and $A^-$ are identified by all best and worst performance scores (Eqs. (4) and (5)), respectively.

$$A^+ = \left\{ w_1^+, \cdots, w_n^+ \right\} = \left\{ \left( Max w_{pq} \mid q \in Q \right), \left( Min w_{pq} \mid q \in Q' \right) \right\} \tag{4}$$

$$A^- = \left\{w_1^-, \cdots, w_n^-\right\} = \left\{\left(Minw_{pq} \mid q \in Q\right), \left(Maxw_{pq} \mid q \in Q'\right)\right\} \quad (5)$$

For MVIDE, all factors $S(d, \theta, s, \tau, \phi)$ are considered as cost factors. This allows the definition of $A^+$ and $A^-$, and the identification of distances ($d_p^+$ and $d_p^-$) ,i.e., the relative proximity between alternatives. Thus, the distances of $A_p$ to $A^+$ and $A^-$ are calculated by using Eqs. (6) and (7), respectively:

$$d_p^+ = \sqrt{\sum_{q=1}^{n} \left(w_{pq} - w_q^+\right)^2}, \quad p = 1, 2, ..., m \quad and \quad q = 1, 2, ..., n \quad (6)$$

$$d_p^- = \sqrt{\sum_{q=1}^{n} \left(w_{pq} - w_q^-\right)^2}, \quad p = 1, 2, ..., m \quad and \quad q = 1, 2, ..., n \quad (7)$$

Since the closeness relative is computed, the ranking order of all alternatives is obtained, allowing selecting the most feasible paths. Eq. (8) expresses the closeness coefficient $S_p$ of each alternative:

$$S_p = \frac{d_p^-}{d_p^+ + d_p^-}, \quad p = 1, 2, ..., m \quad (8)$$

Lastly, the index value of $S_p$ varies between 0 and 1 and all alternatives are ordered in a descending way. Under these circumstances high $S_p$ means lower distance to $A^+$ while lower $S_p$ establishes great distance to the best solution. From use of these values for ranking all possible paths as alternatives, it is possible to obtain more suitable paths of the list.

### 3.2. MDC Forwarding Decision Process

This stage employs the forwarding task of the live video flows along the routes chosen in the Path Ranking stage. In this stage, given a source-destination pair $(vs_j, vd_j)$, the Multi-Stream Video Encoder component (from MDC architecture) divides a flow $f_j \in (VS_j, VDj)$ in $z$ independet substreams, called MDC descriptors $MD_j^c|(c \in [1, z])$. Each $MD_j$ will be transported by the routing protocol along $z$ set of disjoint paths established each one by forwarding nodes.

For simplification, MVIDE will be analysed only with two substreams (MDC descriptors), i.e., $z = 2$ ( $MD_j^1$ and $MD_j^2$). Figure 1 illustrates the architecture of the system combining MDC, multipath, and MVIDE. To start a video transmission, $vd_j$ requests a video from $vs_j$, thus $S(d, \theta, s, \tau, \phi)$ is exchanged periodically among $N(vs_j)$ and sent to $vs_j$. The information update periods follow a spatial distribution of nodes whose variation does not change rapidly in a short period of time.

Once $vs_j$ divides a multimedia stream into $MD_j^1$ and $MD_j^2$, the Traffic Allocator distributes the packets from these substreams among the two best paths. Then, the routing protocol will route the substreams along the two paths. At each update period, the routing protocol detects the available paths between the $vs_j$ and $vd_j$ and informs to $vs_j$.

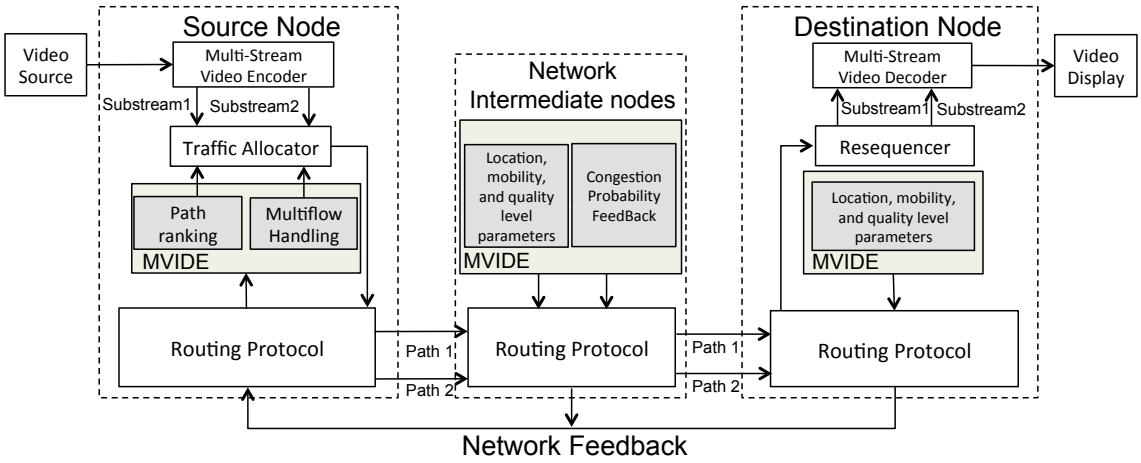



**Figure 1. MDC architecture with MVIDE.**

In the Multi-flow Handling phase, $vs_j$ receives information about the congestion likelihood of routes in activity, as detailed in the next subsection. At $vd_j$, the Resequencer component retrieves each packet arriving from the two paths and reassembles it after a predetermined time period. At last, Multi-Stream Video Decoder reconstructs the video from the received packets in each substream. The Multi-Stream Video Encoder and Decoder, as well as Traffic Allocator and Resequencer components, belong to the MDC architecture and do not overload the system [Wang et al. 2005].

MVIDE filters the Path Ranking through a *max delay* and a *forwarding zone* threshold to decrease the processing time and memory consumption. If there is no more than one available path, i.e., channel conditions cannot assure the video requirements, it is preferable that only the available path forwards both substreams [Zhuang and Ismail 2012], thus MVIDE redirects the substreams as single path strategy.

### 3.3. Multi-flow Handling

This stage controls the multi-flow handling on intermediate nodes. Figure 2 shows a logical representation of the progressive download task in a forwarding node $vf_j$, where multiple flows are received simultaneously. With the number of flows increasing, intermediate nodes can receive heavy traffic, which indicates a poor distribution of flows in the network. This can cause multiplexing, buffer overflow, delay, packet loss, leading to degradation of the video quality perceived by the user.

When the buffer of an intermediate node starts to be full, this node should not receive more flows, as shown in Figure 2. Thus, each $vf_j$ should also know the congestion probability of its neighbors. To establish better flow distribution, we define the buffer level $\phi_j$ as the number of packets stored in the buffer queue of a $vf_j$. Furthermore, each forwarding node $vf$ has knowledge about the current amount of data traversing it, i.e., the packets per second that it is receiving and transmitting $trafficLevel = rateOut - rateIn$. Hence, a given $vf_j$ is able to calculate its congestion indicator $CI_j$ according to Eq. (9).

$$CI_j = \frac{1}{1 + e^{(-c(trafficLevel_j) - Q_{max}/2)}} \tag{9}$$

We employed the sigmoid function for the exponential distribution of $CI$. The

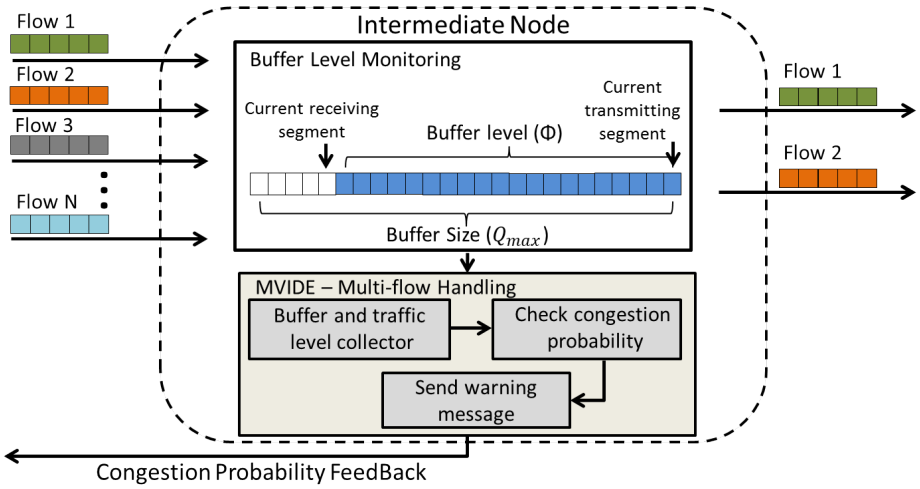



**Figure 2. MVIDE into an intermediate node.**

parameter $Q_{max}/2$ determines the center of the sigmoid output, and $c$ regulates the slope or "growth rate" of this function during its rising portion. The parameter $c$ has a negative value to enable large values of $trafficLevel$ and to generate an output closer to 0. Finally, the congestion probability $\rho_j(CI_j, \phi_j)$ of a given node $vf_j$ increases when $trafficLevel_j$ rises rapidly together with high values of $\phi_j$. It is calculated in the weighted Eq. (10):

$$\rho_j^t(CI_j, \phi_j) = \alpha \cdot CI_j^t + (1 - \alpha) \cdot (\phi_j^t) \tag{10}$$

Where $\rho_j^t(CI_j, \phi_j)$ is a congestion probability at a given instant $t$ on a $vf_j$ and $\alpha$ is a weighted factor. Hence, the greater the amount of flow traversing $vf$ nodes, the longer $\rho(CI, \phi)$. This means that MVIDE not only allows the handling of the traffic load distribution, but also deals with the transmission latency.

Through Eq. (10), we attempt to exploit, when a given $vf_j$ experiences many concurrent streams. Thus, $vf_j$ uses its $\rho(CI, \phi)$ to notify the previous nodes through a greedy message. The network feedback collaborates to make decisions aiming QoE improvements. Inspired by [Hashemi and Khorsandi 2012], we opted by readjusting a new route for the traffic allocator when a predefined congestion threshold is exceeded. Based on the path ranking, a new alternative route is selected to replace the paths without quality enough for the live video flows. This new route will be used until either the arrival of the new warning message or the selection of a new best route by path ranking scheme.

## 4. Integration of MVIDE with the GPSR-MA protocol

This section presents the MVIDE mechanism in addition to the underlying routing protocol. To assess MVIDE functionalities, we develop and adapt its three stages jointly with the geocast protocol GPSR-MA [Granelli et al. 2007] [Granelli et al. ]. GPSR-MA improves the well-know GPSR protocol by using not only parameters related to distance, but also some of the factors, which allows MVIDE to select the best routes, such as speed and direction information.

In VANET scenarios, vehicles at a given instant $t$ move along well defined segments, which form the roads. Each node performs neighbor position estimates at time

$t+1$ by using the above mentioned parameters (distance, speed, and direction). GPSR-MA creates three functions to calculate these parameters in the routing process: $g(d)$, $f(s)$, and $h(\theta)$, given by Eqs. (11a), (11b) and (11c), respectively.

$$g(d) = e^{-(t_{range}+d-d_i)} \qquad f(s) = e^{-(\frac{(x-s_i)^2}{2\sigma^2})} \qquad \text{(11a,11b)}$$

$$h(\theta) = \begin{cases} 0 & if \quad \theta \geq \frac{\pi}{2} \\ cos^2(\theta) & elsewhere \end{cases} \qquad \text{(11c)}$$

Where $t_{range}$ means transmission range and $d_i$ is the distance of current node $i$ from destination. GPSR-MA employs a Gaussian function to a given node $v_i$ calculate the difference between the speed $x$ of $N(v_i)$ in comparison with its reference speed $s_i$ within a predefined standard deviation $\sigma$. To calculate movement direction component, $\theta$ means the angle between node movement direction and the line connecting it to destination.

The MVIDE mechanism coupled with GPSR-MA protocol, called GPSR-MA-MVIDE, uses other link quality factors (namely delay and traffic level) allowing a robust Path Ranking in addition to the position-based parameters of GPSR-MA. To add the MVIDE functionality with GPSR-MA protocol, we establish criteria as input to the MCDM method without affecting the protocol performance. As defined above in Eqs. (11a), (11b), and (11c), GPSR-MA periodically exchanges messages containing location and mobility information. Thus, GPSR-MA-MVIDE also uses delay and traffic level information to determine the two best route options.

The arrival-time between messages suggests delay measurements. This information is important, since some vehicles with good mobility and position parameters may have higher transmission delay due to areas with higher probability of collisions, i.e., nodes with larger contention windows. As the number of transmissions increases, there will be more contention on the channel and back-off values will be higher. Hence, we add the delay transmission values of signaling messages for the MVIDE Path Ranking stage.

The traffic level determines whether forwarding nodes face heavy load, which may impair its flow transmissions. Thus, the buffer level $\phi$ allows MVIDE to determine the traffic level of candidate nodes. Once established the best paths, the forwarding nodes transmit the congestion probability $\rho(CI, \phi)$ only if it exceeds a given threshold. The buffer level and delay information allows source node to run a more robust path ranking.

Figure 3 illustrates the path selection process to GPSR-MA-MVIDE protocol (represented by ***ii***) in comparison to the pure GPSR-MA protocol using MDC and multipath (GPSR-MA-MDC, represented by ***i***). For a given time $t$ there is only one flow in the network from $S_1$ to $D_1$. In this case, both GPSR-MA-MDC and GPSR-MA-MVIDE have the same choices for paths, i.e., they select the paths with shortest distance to the destination node. However, for a time $t+1$, a new stream starts from $S2$ to $D_2$ and, through Path Rankin stage, GPSR-MA-MVIDE selects an alternative route ($S_3$) for a MDC substream instead of the path with shortest distance to the destination node, in this way it relieves $vf_3$ that already transmits data. The same applies for time $t+2$, where, through Multi-Flow-Handling stage, MVIDE achieves a better distribution of flows in the network. On the other hand, GPSR-MA-MDC concentrates all traffic in more attractive nodes, such as $vf_3$, which can impair the concurrent flows.

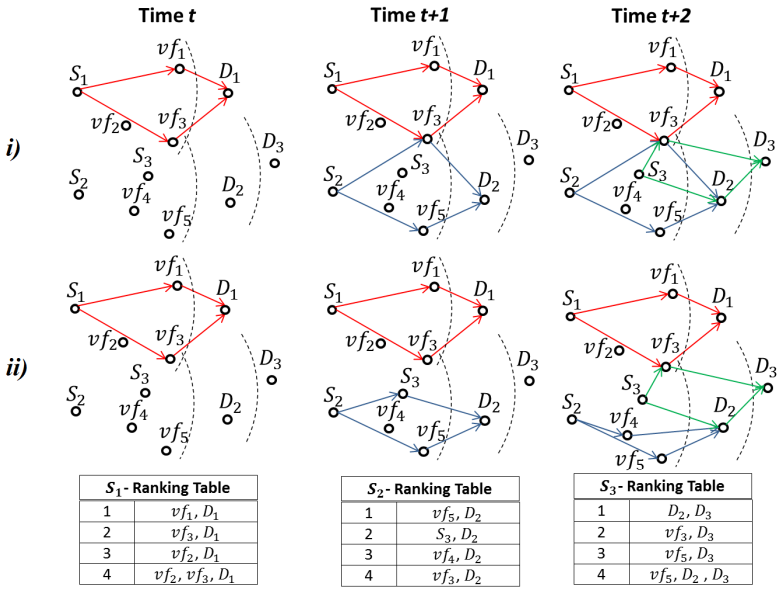



**Figure 3. *i)* GPSR-MA-MDC and *ii)* GPSR-MA-MVIDE.**

Other protocols can implement MVIDE. It might be suitable for protocols that use SNR level rather than distance [Felice et al. 2013], as well as added in multipath schemes. Depending on the routing strategy, the MVIDE steps can be easily adapted.

## 5. Performance Evaluation

We employed the Network Simulator version 2.33 (NS2) to perform the simulations and SUMO to generate the vehicular movement traces. The mobility model is the Manhattan Model. More details about the simulation parameters are shown in Table I. For simplification, in this work MVIDE produces only two substreams for each video flow and considers an area large enough to cover several live videos flows with two paths each one.

**Table 1. Simulation Parameters.**

| Parameter | Value |
|---|---|
| Area Size | 1600m x 1600m |
| Average road length | 400m |
| Speed Range | 5.5 to 15.5 m/s |
| Beacon interval | 0.3s |
| Radio Range | 250 |
| Number of Vehicles | 60 to 220 |
| Simulation Time | 500s |
| MAC Layer | IEEE 802.11p |
| Queue Capacity | 30 pkts |
| Propagation Model | Nakagami Dist. |


**Figure 4. Video sequences.**

In order to establish a relevant scenario where multiple video streams with different sizes and characteristics (bitrate and length) are requested and generated as in a real scenario, we have used EvalVid - A Video Quality Evaluation Tool-set. This framework allows evaluating the video streaming quality and getting more realistic results. In this way, we have conducted the simulation by transmitting 15 real and widely used differ-

ent MPEG sequences of natural scenes (352 x 288 pixels) with duration varying from 5s to 70s and internal GoP structure configured as two B-frames for each P-frame (IBBP pattern) [Seeling and Reisslein 2012]. Figure 4 shows one frame of each video.

To demonstrante the impact of MVIDE, we use the single path protocol, GPSR-MA, for comparison. Also, to investigate the performance gain when taking into account the handling of multi-flows caused by overhead, we have compared GPSR-MA-MVIDE with a GPSR-MA multipath-MDC version (GPSR-MA-MDC), i.e., this protocol sends one MDC substream to two different paths. Thus, the lost frames from one substream are compensated by those from the other substream, whereas lost frames from both paths cannot be reconstructed. In that case, the video decoder performs the Frame-Copy error concealment technique, which uses the last well-received frame to replace lost frames.

The impact and benefits of the proposed solutions were measured by the following metrics: Packet Delivery Rate (PDR), average number of hops per source-destination pair, and End-to-end Delay. From the received video quality point-of-view, measurements were carried out with Structural SIMilarity (SSIM). It is a well-known QoE metric, which measures the structural distortion of the video to obtain a better correlation with the user's subjective impression. These metrics determine the behavior of GPSR-MA-MVIDE regarding the GPSR-MA-MDC and pure GPSR-MA. It also presents the results with the variations in number of connections (flows), number of vehicles, overhead generated, and the quality of routes for the video delivery.

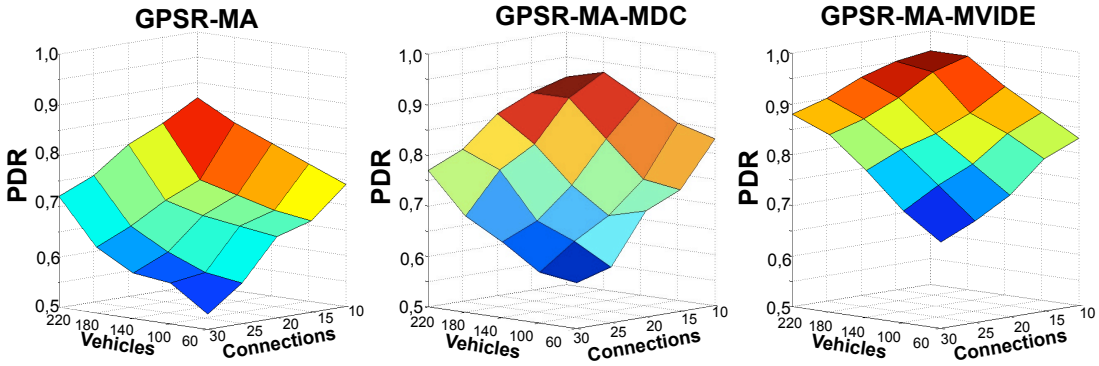
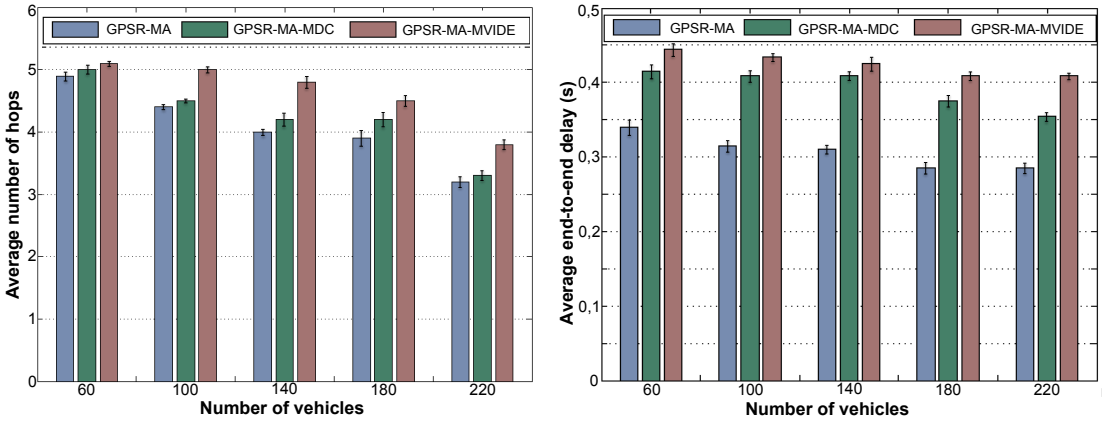



**Figure 5. PDR performance comparison among GPSR-MA, GPSR-MA-MDC, and GPSR-MA-MVIDE when number of vehicles and connections (flows) varies.**

In terms of network performance and PDR, Figure 5 shows the performance results for the three above listed cases: GPSR-MA protocol, GPSR-MA with MDC and multipath (GPSR-MA-MDC), and GPSR-MA coupled with MVIDE mechanism (GPSR-MA-MVIDE). Each plotted result in the graphs is the average of results generated from 35 different simulations. The confidence interval was calculated with 95% confidence level. As we can observe from the simulation results, GPSR-MA-MVIDE notably outperforms other solutions when number of vehicles and connections increases. The major rationale for this behavior is that GPSR-MA faces several broken link situations when there are few available vehicles. Whereas, GPSR-MA-MDC and GPSR-MA-MVIDE neatly distribute the load into two complementary paths, which reduces the broken link problem of GPSR-MA. Despite being a multipath solution, GPSR-MA-MDC does not gain any performance improvement facing multiple flows. The problem with this solution is the

increasing number of concurrent flows in the network. As all the nodes use the same routing strategy, many streams are directed to the same more attractive paths. This can cause congestion, overload, and collision in some nodes, consequently, lead to degradation in received video qualities. However by using the Path Ranking and Multi-flow Handling stages based on traffic level, GPSR-MA-MVIDE ensures better traffic distribution of the network, resulting in more PDR.




(a) (b)

**Figure 6. (a) Number of hops comparison. (b) End-to-end delay comparison.**

Figure 6a presents the average number of hops comparison per source-destination pair among the discussed approachs. This figure shows that the paths for the MDC substreams (GPSR-MA-MDC and GPSR-MA-MVIDE) exceeds ones of the single-path solution (GPSR-MA), which follows the same pattern as there are more paths and more probability over which packets can be forwarded in an MDC scheme. Moreover, due to Path Ranking, GPSR-MA-MVIDE considers other quality level metrics to classify routes. Thus, it is not restricted to shortest paths and mobility parameters such as distance.

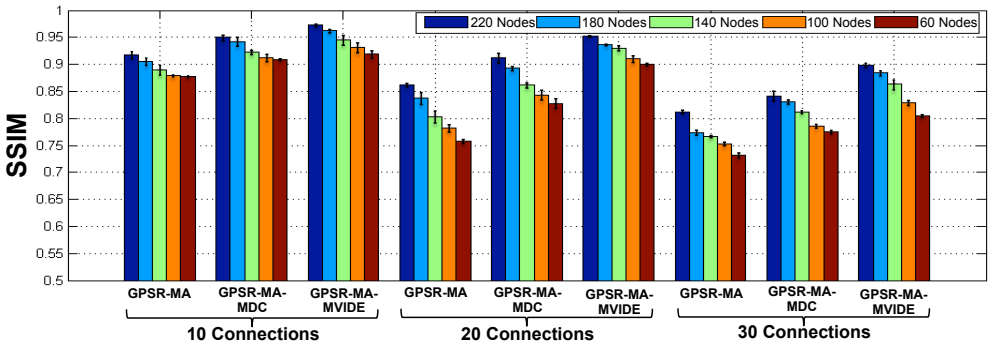



**Figure 7. Average video quality evaluation.**

In the last measure we analyze the average end-to-end delay of each frame, which is slightly lower mainly in GPSR-MA compared to GPSR-MA-MVIDE (Figure 6b). As mentioned previously, in MVIDE, according to its operating model, provides more effort to deliver flows when faces congestion, this could mean forwarding of substreams to alternatives sparse roads with increasing transmission durations. On the other hand, the other two protocols just drop packets in these situations and will not make any effort. This might result in longer delays and path lengths. However, delay levels are less than one

second, which are negligible even in video application and are significantly smaller than requirements of 4 to 5 seconds defined by CISCO [Hatting et al. 2005].

To demonstrante the impact of the proposed solution on the human perception, the results in Figure 7 present the SSIM metric. With 10, 20 and 30 connections, MVIDE keeps the video quality high, i.e., SSIM around 0.94, 0.93, and 0.86, respectively. An average increase of 15% and 9% compared to GPSR-MA and GPSR-MA-MDC. It confirm the results obtained in the Figure 5 and show similar benefits to those achieved earlier. SSIM values range from 0 to 1, where a higher value means better video quality. As we can see, in scenarios with 10 connections, the three protocols maintain a similar video quality level. However, in scenarios with 30 connections at the same time, the difference between them becomes very significant.

From the graphs aforementioned, is clear that GPSR-MA-MVIDE preserves more video quality because of its path ranking and multi-flow handling procedure in which it selects and chooses the best options as next hops and switches to other routes as soon as congestion is identified. As result, in the performance network, GPSR-MA-MVIDE will also outperform GPSR-MA and GPSR-MA-MDC in congested periods.

## 6. Conclusions

This paper introduced MVIDE to enable an efficient on-road real-time multiple videos dissemination with QoE support in VANETs. MVIDE calculates and ranks the best routes to distribute simultaneous MDC substreams in the network. It was integrated with GPSR-MA protocol and considers the characteristics of multiple paths, vehicle mobility, and application requirements. Further, MVIDE allows multi-flow handling by using information about traffic level in intermediate nodes to adjust paths without breaking connections, providing a better network traffic distribution. Simulation results highlight the performance, effectiveness and QoE support of MVIDE by measuring the video quality levels when the number of vehicles and flows varies. We measured the video quality of each transmitted video by means of QoS and QoE metrics. MVIDE provided multimedia dissemination with robustness and QoE support compared to single-path GPSR-MA protocol and GPSR-MA with MDC scheme.

## References

Felice, M. D., 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.

Granelli, F., Boato, G., and Kliazovich, D. Mora: A movement-based routing algorithm for vehicle ad hoc networks. In *IEEE Workshop on Automotive Networking and Applications (AutoNet 2006), San Francisco, USA*.

Granelli, F., Boato, G., Kliazovich, D., and Vernazza, G. (2007). Enhanced gpsr routing in multi-hop vehicular communications through movement awareness. *Communications Letters, IEEE*, 11(10):781–783.

Hashemi, H. T. and Khorsandi, S. (2012). Load balanced vanet routing in city environments. In *IEEE 75th Vehicular Technology Conference (VTC Spring'12)*, pages 1–6.

Hatting, C. et al. (2005). *End-to-end qos network design*. Cisco Press.

Hwang, C.-L. and Yoon, K. (1981). *Multiple attribute decision making*. Springer.

Mérindol, P., François, P., Bonaventure, O., Cateloin, S., and Pansiot, J.-J. (2011). An efficient algorithm to enable path diversity in link state routing networks. *Computer Networks*, 55(5):1132–1149.

Naderi, H., Shahosseini, H., and Jafari, A. (2013). Evaluation mcdm multi-disjoint paths selection algorithms using fuzzy-copeland ranking method. *International Journal of Communication Networks and Information Security (IJCNIS)*, 5(1).

Piñol, P., López, O., Martínez, M., Oliver, J., and Malumbres, M. P. (2012). Modeling video streaming over vanets. In *Proc. of the 7th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 7–14.

Qadri, N. N., Fleury, M., Rofoee, B. R., Altaf, M., and Ghanbari, M. (2012). Robust P2P multimedia exchange within a vanet. *Wireless Pers. Communications*, 63(3):561–577.

Razzaq, A. and Mehaoua, A. (2010). Video transport over vanets: Multi-stream coding with multi-path and network coding. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 32–39. IEEE.

Rezende, C., Ramos, H. S., Pazzi, R. W., Boukerche, A., Frery, A. C., and Loureiro, A. A. (2012). VIRTUS: A resilient location-aware video unicast scheme for vehicular networks. In *IEEE International Conference on Communications (ICC'12)*, pages 698–702.

Seeling, P. and Reisslein, M. (2012). Video transport evaluation with H.264 video traces. *IEEE Communications Surveys and Tutorials, in print*, 14(4):1142–1165. Traces available at `trace.eas.asu.edu`.

Taleb, T., Ksentini, A., and Filali, F. (2012). Wireless connection steering for vehicles. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 56–60. IEEE.

Wahab, O. A., Otrok, H., and Mourad, A. (2013). VANET QoS-OLSR: QoS-based clustering protocol for vehicular ad hoc networks. *Computer Communications*, 36(13):1422–1435.

Wang, R., Rezende, C., Ramos, H. S., Pazzi, R. W., Boukerche, A., and Loureiro, A. A. F. (2012). LIAITHON: A location-aware multipath video streaming scheme for urban vehicular networks. In *IEEE Symposium on Computers and Communications (ISCC'12)*, pages 000436–000441.

Wang, Y., Reibman, A. R., and Lin, S. (2005). Multiple description coding for video delivery. *Proceedings of the IEEE*, 93(1):57–70.

Xie, F., Hua, K. A., Wang, W., and Ho, Y. H. (2007). Performance study of live video streaming over highway vehicular ad hoc networks. In *Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th*, pages 2121–2125. IEEE.

Xing, M. and Cai, L. (2012). Adaptive video streaming with inter-vehicle relay for highway vanet scenario. In *IEEE International Conference on Communications (ICC'12)*, pages 5168–5172.

Zhuang, W. and Ismail, M. (2012). Cooperation in wireless communication networks. *Wireless Communications, IEEE*, 19(2):10–20.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Salão de Ferramentas do SBRC 2014

## *Sessão Técnica 1*

## *Internet do Futuro e Computação Paralela e Distribuída I*

# FIBRE - An International Testbed for Future Internet Experimentation

**Tiago Salmito[1], Leandro Ciuffo[1], Iara Machado[1], Marcos Salvador[1], Michael Stanton[1,6], Noemi Rodriguez[4], Antonio Abelem[2], Leonardo Bergesio[3], Sebastià Sallent[3], Loïc Baron[5]**

[1]RNP – National Education and Research Network, Brazil
{tiago.salmito;iara;leandro.ciuffo;michael} @rnp.br

[2]UFPA – Federal University of Pará, Brazil
{abelem} @ufpa.br

[3]i2Cat – Foundation, Research and Innovation in the Internet Area
{sebastia.sallent,leonardo.bergesio} @i2cat.net

[4]PUC-Rio – Catholic University of Rio de Janeiro
noemi@inf.puc-rio.br

[5]UPMC – Pierre and Marie Curie University
loic.baron@lip6.br

[6]on secondment from UFF – Fluminense Federal University, Brazil

***Abstract.*** *This paper describes the FIBRE testbed, a large-scale research facility for experimentation on Future Internet. The current testbed is a federation of 13 local testbeds (aka experimental islands), located in different R&E organizations. The FIBRE infrastructure combines heterogeneous physical resources and different technologies, including OpenFlow, wireless and optical communications. This paper discusses the architecture of FIBRE, which includes different Control Management Frameworks, and describes how the testbed can be used in research and education to experiment with networking and distributed systems.*

## 1. Introduction

The need for experimenting with new protocols and techniques for the Future Internet (FI) led to construction of testbeds, that is, networks totally dedicated to experiments and isolated from the production Internet. The architecture of these testbeds must take a number of issues into account. Researchers must be capable of running their experiments with the scale and heterogeneity of the real environment. Resources must be allocated elastically to different experimenters, but each researcher must be able to replicate the exact environment of previous tests. Testbeds must also offer monitoring and instrumentation tools, and must control access to their resources according to the policy designed for them.

FIBRE (Future Internet testbeds/experimentation between Brazil and Europe) is one of five projects that were approved in response to the 2010 Brazil-EU Coordinated Call in ICT, jointly funded by CNPq (the Brazilian Council for Scientific and Technological Development) and by the European Commission within its Seventh Framework Programme (FP7). FIBRE was launched in October 2011 with the goal of deploying a

new experimental facility for Future Internet research in Brazil, federated to European ones, both at the physical connectivity and control and monitoring framework levels.

With the globalization of experimental FI research, there has been considerable interest in the federation of distinct testbed facilities, in order to permit carrying out experiments that span multiple testbeds. This design, besides addressing the challenge to incorporate resources from legacy projects, enables the extension of the testbed to support new networks and include new research facilities.

Thus, FIBRE is currently deployed as a federation of 13 local experimental facilities (a.k.a. "islands"). One of the biggest challenges is to provide a unified view of resources managed by different organizations, by way of a federation of independent resources.

The management of such a testbed facility is complex. The use of a control and monitoring framework (CMF) can facilitate this task, allowing for its automation. The CMF allows a central management site in the facility to control the access to virtualized computing and network resources and to provide support for measurement of resource use.

The architecture built for FIBRE brings together different technologies, including OpenFlow [McKeown 2008] and wireless and optical communications, as well as different CMFs. In this article we describe the testbed that is available at this point and its use in experiments.

## 2. FIBRE architecture

The testbed is currently composed of ten islands located in Brazil and three in Europe. In this paper, we will concentrate on the Brazilian side of the testbed.

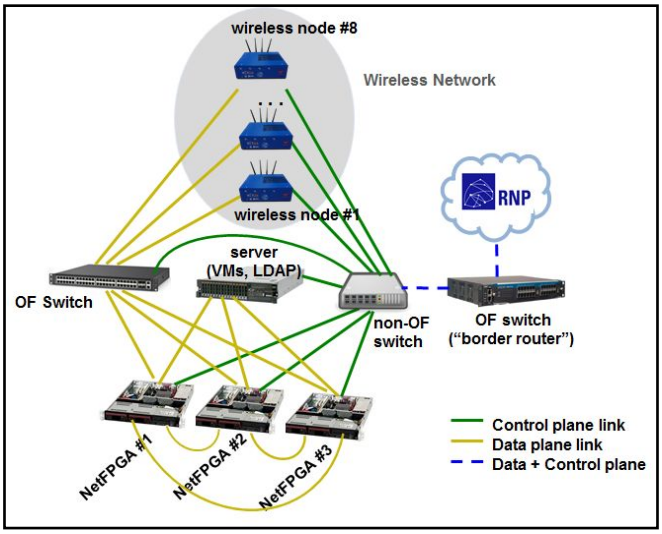



**Figure 1.** Overview of a FIBRE island.

Each island has a common nucleus of OpenFlow-capable switches, together with their controller(s), as well as a cluster of compute and storage servers, appropriately virtualised, and (usually) a cluster of virtualised wireless nodes. Each site integrates its own site-specific resources to FIBRE, such as wireless access testbeds (WiFi, WiMax, 3G/4G), OF-enabled equipment, optical networks or even more complex testbeds with

heterogeneous resources and their own control framework (e.g.: the Emulab cluster at USP). Figure 1 illustrates a typical FIBRE island, with its common facilities and external connectivity.

One of the challenges for network environments for experimentation is scale. Because experimentation networks are supported by real hardware, a large topology requires controlled multiplexing of the resources of the underlying physical system, to provide manageable virtual resources. The FIBRE testbed uses multiplexing techniques to slice control and data traffic based on a specific OF controller, the FlowVisor [Azodolmolky 2012] or channel scheduling for wireless, in order to virtualize resources like processing nodes, network devices and networks. It is worth mentioning that virtualization does not provide strict scientific fidelity, because the experiments are based on shared, rather than dedicated, physical resources. However, there are good reasons for relaxing this constraint: 1) some applications, like peer-to-peer systems, even though requiring large topologies are not resource-intensive; 2) the strict scientific fidelity requirement might be dispensable for many applications; and, 3) multiplexing allows more efficient use of limited hardware resources.

From the start, the project team decided that FIBRE should include the following CMFs: OFELIA Control Framework (OCF), OMF and ProtoGENI. The use of different CMFs represents a gain for the project as it allows the simultaneous orchestration of three complementary classes of resource: OpenFlow resources, wireless resources, and emulated resources. All CMFs were customized for use in FIBRE.

OCF [Sune 2013] was originally created in the context of the OFELIA testbed project[1], but today it is supported by a wider community in which FIBRE and GEANT participate. From the point of view of experimenters (or network researchers), the available underlying network substrate is fully controllable using explicit and dynamic configurations based on OpenFlow abstractions like FlowSpace. Once a FlowSpace is set up, the researcher can proceed with the allocation of a controller, either remotely or in a local virtual machine, to test his new Internet research project.

OMF [Rakotoarivelo 2010] is a framework with the focus on controlling and managing network devices. It was developed based on XMPP (eXtensible Messaging and Presence Protocol) in the Ruby language. The OMF suite also provides OML (OMF Monitoring Library), which allows instrumentation of applications for collecting measurements.

ProtoGENI [Duerig 2012] is a CMF from the University of Utah. It is based on an enhanced version of the Emulab testbed management software. The Emulab testbed is used to perform experimental network research on distributed systems. ProtoGENI was created to provide the integration between Emulab and other testbeds in order to build the Cluster C facility of GENI.

As aforementioned, federation is a key issue in the design of the FIBRE testbed. In fact, one of FIBRE's goals is to design a framework where all the CMFs adopted can work together complementing each other, in addition to federating different instances of the same CMF. In its first phase, the FIBRE testbed is being accessed through a simple web

[1] http://www.fp7-ofelia.eu

interface (see Section 3). An important component in FIBRE's architecture will be MySlice [Augé, 2013], a software layer that enables the creation of a federation abstraction, integrating the different FIBRE testbeds using the Slice-based Federation Architecture (SFA) [Peterson, 2010]. This interface is based on a web client that allows users to interact with the great volume of results generated by each testbed island.

The distribution of FIBRE islands is shown in Figure 2. The integration of these resources creates a large-scale network. In Europe, there are three islands: one at Fundació i2CAT (Spain), one at the University of Bristol (UK), and one at UTH (Greece). The Brazilian part of the testbed is composed of ten islands widely spread across seven Brazilian states. Each island is controlled by one or more CMFs. Figure 2 shows the set of CMFs available in each island, and the network that connects them, called the FIBRE backbone.

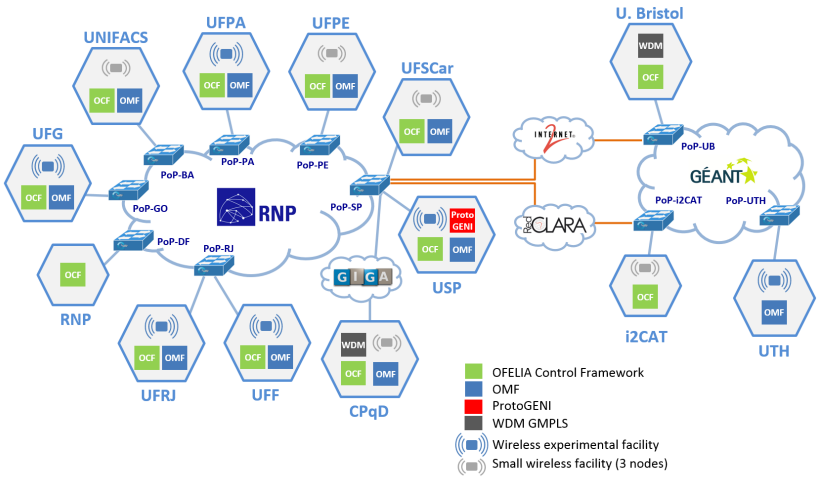



**Figure 2:** The FIBRE testbed.

Each island in FIBRE monitors its resources using Zenoss, an open-source application for network management. Zenoss provides a web interface that allows administrators to monitor availability, performance, and events. Using Zenoss, a publicly available web page with information about network and virtual machines is made available to experimenters.

## 3. Using the FIBRE testbed

Because the MySlice portal is in final stages of development, FIBRE is at the moment accessed through a simpler web page, maintained by the FIBRE NOC (Network Operation Center)[2]. The NOC is responsible for controlling and monitoring the network assets of the testbed and for monitoring the provided services. User authentication is carried out using an LDAP directory in each island synchronised with an LDAP directory at the NOC. LDAP allows authentication with all CMFs (OMF, OCF, and ProtoGENI)[3]. To obtain access to the FIBRE testbed, a user should either contact the

[2] https://portal.fibre.org.br

[3] The roadmap for authentication is to migrate to a federated model using the Brazilian Academic Network Federation, CAFe.

local island administrator, if he/she is associated to one of the institutions maintaining an island, or contact the NOC at info@fibre.org.br.

Previous to conducting an experiment, the user will have developed or selected the software he/she desires to experiment with. This may be user-level software (a peer-to-peer content distribution system or a mobile app) or system-level (modified kernels with new or modified communication protocols, or reimplementations of well-established protocols, in the case of educational experiments). Software may be completely agnostic of the environment on which it will be running or may include calls to libraries that collect measurement. OMF, for instance, includes the OML framework for collecting measurements [Singh 2005]. Once the user has the software ready, he/she must design the (set of) experiments to be conducted [Jain 1991].

Due to lack of space, we describe here the setup of an experiment using only the OCF control framework. Documentation about experimentation with the other frameworks is available online at the FIBRE wiki[4].

Figure 3 illustrates the lifecycle of an experiment in the testbed. Initially, the user must provide an "experiment description", which includes the desired nodes and the links between them, as well as the desired timeslot for the experiment. Given this description, the testbed portal will handle the reservation of resources. At the arranged time, the user will initiate the experiment, uploading the desired code and configuration to the reserved nodes. During the execution of the experiment, both the user and the framework can monitor and control it, generating data that can be used either interactively or off-line.

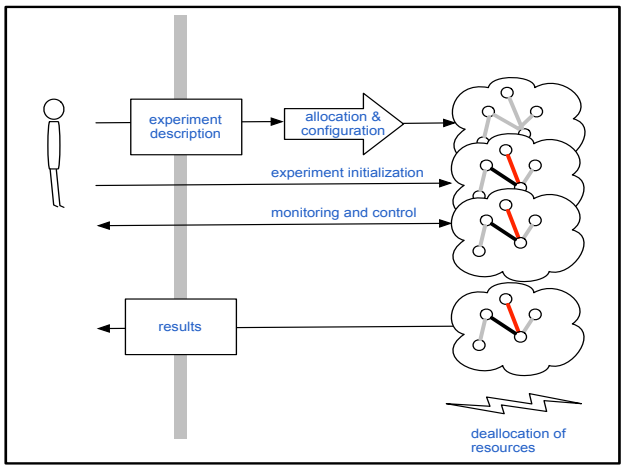



**Figure 3**: The lifecycle of an experiment.

## Setting up an experiment in OCF

Resources in OCF are managed by so-called Aggregate Managers (AMs). Aggregates encompass one or more instances of a single type of resource (OpenFlow resources, Virtual Machines, etc.). They are created by the island managers and are local to their respective island. Each island runs local instances of OCF subcomponents, which

[4] wiki.rnp.br/display/fibre/

manage its local resources and externalize their AMs to the OCF instance that runs on the NOC site.

To set up an experiment, users must first request access to a project in the OCF interface. Users can see and modify configurations in projects for which they have been granted permission. These permissions are usually granted by the project owners. For new projects, permission is usually granted by the island manager.

The user must begin the setup of the experiment by making a reservation of the desired resources on the appropriate OCF instance. For cross-island experiments, users must access the OCF instance running at the NOC and add to their project the necessary aggregates from all desired islands; experiments configured by an OCF instance running at an island may use only its own local resources.

OCF uses the concept of *slice* to describe an experiment. Slices describe the resources to be used and the associated configurations, and encompass the state of the experiment. Once the user has permission to participate in a project, he can either use existing slices associated to this project, or create new ones. When creating a new slice, information such as name, description and expiration date (the slice life-time) must be provided. After slice expiry, the island manager may deallocate the reserved resources.

On the Slice Management page (Figure 4), the Topology panel shows the physical topology of the resources in the aggregates available to the slice. These resources may comprise the available OpenFlow switches, virtualization servers, and the connections between switches and servers.

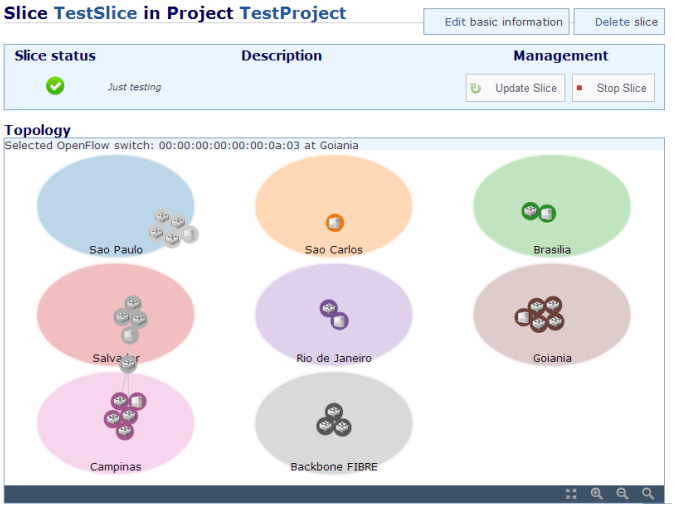



**Figure 4:** Topology of a slice.

VMs for experiments are created in a Computational Resources area on the Slice Management page. During the creation process, the VM will be granted an IP, which will be displayed in the Topology Panel and the Computational Resources area. This IP is only reachable through FIBRE's VPN. VMs may be started, stopped, rebooted or deleted by clicking on the respective action link in the Computational Resources area. Users can create as many VMs as needed for their experiment.

To use OpenFlow resources, users are required to add OpenFlow resources to their slice and specify an OpenFlow Controller for the experiment. On the Slice Management page in the OpenFlow Aggregate area (Figure 5), data paths (consisting of *id* and port

number) that are available in the slice can be selected to define a FlowSpace for the experiment. Once the FlowSpace is selected and the VMs are created, the user must set the controller IP address to match to the VM that will host the OpenFlow controller.

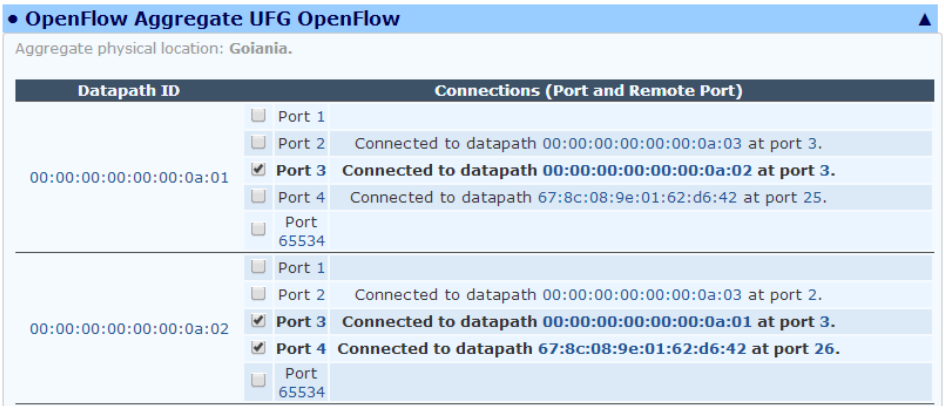



**Figure 5:** OpenFlow datapath

To effectively run the experiment, the user starts the slice on the Slice Management page. This will trigger a FlowSpace request to the island manager and make sure all the VMs in the slice are active. Until the user receives the FlowSpace approval by the island manager, their FlowSpace has not yet been granted and cannot be used. Only granted FlowSpaces are installed/known in the FlowVisor that handles the slicing of the overall FIBRE OpenFlow resources.

Within the slice, the user can use the VMs as end-hosts and the FlowSpaces (allocated on the OpenFlow switch fabric) as the network data-plane. Users can access the running VMs through ssh, using their FIBRE username and password over FIBRE's VPN. Users can then install on the VMs any OpenFlow controllers or arbitrary software that is needed for their experiment (A set of OpenFlow controllers are pre-installed in the VM images, but users can also provide their own controller implementation.).

In the demonstration at the tool session, we will go through the procedures described in this Section in a step-by-step fashion, setting up an experiment that spans multiple islands and going through the configuration steps that are necessary to access the virtual resources and to run an experiment.

## 4. Final Remarks

The implementation of new experimental facilities in Brazil, as well as the integration with European facilities, offer a valuable infrastructure for research and education. The infrastructure provided by FIBRE will allow researchers to evaluate and benchmark innovative algorithms, techniques and approaches for the Future Internet. The collapsing of network and compute resources, and the ability to instantiate virtual infrastructures dynamically using such resources, allow for experiments of interest not only to the academia (e.g., new Internet architectures where the intelligence is at the edge, in cloud datacenter VMs), but also to the industry (e.g., network functions virtualization composition and service chaining). On the other hand, students can experiment with well-known protocols and algorithms, implementing them from scracth if so desired.

The FIBRE project was recently extended to last until September 2014. The Brazilian partners are organizing themselves to keep the testbed operational beyond the project lifetime. At the moment, the project team is working on further support for instrumentation and monitoring (I&M). The basic requirement for the FIBRE I&M Architecture is the capability to configure, monitor, collect, and display both infrastructure and experiment-specific data for distinct federated or individual CMF aggregates. Additionally, with the conclusion of MySlice deployment, FIBRE users will be able to access both the experimental and control planes, with a common access interface to the different underlying CMFs. MySlice will also allow the I&M activities to be carried out through this same interface.

We welcome new institutions willing to join the testbed. For further information, please go to the project website: http://www.fibre.org.br

## References

Augé, J., Parmentelat, T., Turro, N., Avakian, S., Baron, L., Larabi, M., Rahman, M., Friedman, T., Fdida, S.. Tools to foster a global federation of testbeds. *Computer Networks (special issue on Future Internet Testbeds)*. 2013.

Azodolmolky, S., Nejabati, R., Peng, S., Hammad, A., Channegowda, M.P., Efstathiou, N., Autenrieth, A., Kaczmarek, P. and Simeonidou, D.. "Optical FlowVisor: An OpenFlow-based optical network virtualization approach," Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference , 4-8, March 2012.

Duerig, J., Ricci, R., Stoller, L., Strum, M., Wong, G., Carpenter, C., Fei, Z., Griffioen, J., Nasir, H., Reed, J. and Wu, X.. Getting Started with GENI: A User Tutorial. *ACM SIGCOMM Computer Communication Review* (CCR). 42, 1 (Jan 2012), 72-77.

Jain, R. (1991) The Art of Computer Systems Performance Analysis. Wiley.

McKeown, N.,and others. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*. 38, 2 (March 2008), 69-74.

Peterson, L., Ricci, R., Falk, A., and Chase, J.. Slice-based federation architecture (SFA). working draft (2010). Version 2.0.

Rakotoarivelo, T., Ott, M., Jourjon, G., and Seskar, I.. 2010. OMF: a control and management framework for networking testbeds. *SIGOPS Operating Systems Review*. 43, 4 (January 2010), 54-59.

Singh, M., Ott, M., Seskar, I. and Kamat, P.. ORBIT Measurements Framework and Library (OML): Motivations, Design, Implementation, and Features. proceedings of IEEE Tridentcom 2005 (2005), Trento, Italy.

Sune, M., Bergesio, L., Woesner, H., Rothe, T., Kopsel, A., Colle, Puype, B., Simeonidou, D., Nejabati, R., Channegowda, M., Kind, M., Dietz, T., Autenrieth, A., Kotronis, V., Salvadori, E., Salsano, S., Krner, M., and Sharma, S., "Design and implementation of the OFELIA FP7 facility: The european openflow testbed", *Computer Networks (special issue on Future Internet Testbeds)* (2013), pp. 1389-1286.

# SIMMYCLOUD, simulando o gerenciamento de recursos virtualizados em plataformas de Computação em Nuvem

**Cássio P. Alkmin, Daniel Cordeiro***

[1]Instituto de Matemática e Estatística
Universidade de São Paulo (IME-USP)
Rua do Matão, 1010 – Cidade Universitária
São Paulo – SP – Brazil – 05508-090

{cassiop,danielc}@ime.usp.br

***Abstract.*** *Cloud Computing platforms have changed the Information Technology industry, allowing on-demand rent of computational resources. There are, nevertheless, several infrastructure challenges that need to be tackled by Cloud Computing providers in order to offer an efficient service, while guaranteeing quality of service for their users. The management of computational resources is one of their main challenges, mainly because of the variety of users' performance goals and due to the constant changes on the demand for resources. Simulation tools are essential to evaluate different strategies on large scale scenarios. In this work, we present the* SIMMYCLOUD*, a framework to simulate Cloud Computing environments, that allows the use of real-world workload traces to allow the evaluation of new strategies to manage computational resources on a Cloud Computing platform.*

***Resumo.*** *Plataformas de Computação em Nuvem revolucionaram a indústria de Tecnologia da Informação ao permitir a locação sob demanda de grandes quantidades de recursos computacionais. Porém, ainda há muitos desafios infraestruturais para que tais recursos sejam oferecidos de modo mais eficiente, e sem comprometer a qualidade do serviço contratado pelos usuários. Dentre tais desafios, o gerenciamento dos recursos computacionais se destaca, frente à diversidade de usos que os usuários farão dos recursos e à demanda volátil dos mesmos. Ferramentas de simulação tornam-se, assim, importantes aliadas na avaliação de diferentes estratégias de gestão de recursos computacionais em cenários de grande escala. Neste trabalho apresentamos o* SIMMYCLOUD*, um arcabouço para a simulação de ambientes de Computação em Nuvem que permite o uso de traços de execução reais para a avaliação e comparação de novas estratégias de gestão de recursos em plataformas de Computação em Nuvem.*

## 1. Introdução e motivação

Plataformas de Computação em Nuvem (*Cloud Computing*) revolucionaram a indústria de Tecnologia da Informação ao permitir que grandes quantidades de recursos computacionais fossem oferecidos aos usuários como um serviço sob demanda.

*Este trabalho foi parcialmente financiado pela CAPES e pela FAPESP (proc. #2012/03778-0).

Tais plataformas proveem diversas vantagens competitivas para a indústria. Por exemplo, uma empresa *startup* pode colocar uma nova aplicação em produção sem se preocupar com os custos de aquisição e manutenção de servidores. Outra vantagem de sua utilização é a escalabilidade que pode ser atingida com tais plataformas. Por exemplo, um pesquisador com uma simulação composta por diversas tarefas do tipo *batch* poderia executá-la em menos tempo em uma plataforma de Computação em Nuvem. Em tais plataformas, o custo de utilizar 1.000 servidores durante uma hora é, tipicamente, o mesmo que utilizar um servidor durante 1.000 horas.

A Computação em Nuvem não é classificada como um novo paradigma em termos de pesquisa em Ciência da Computação. Computação em Nuvem é, na verdade, a evolução e a convergência de pesquisas realizadas em diferentes áreas da Computação, tais como aglomerados (*clusters*) e grades (*grids*) de computadores, computação autônoma, computação utilitária, etc. Entretanto, seu sucesso com aplicações comerciais em grande escala indica seu potencial para a execução de aplicações científicas de alto desempenho.

Para tal, é necessário que a gestão de recursos computacionais seja controlada pelo desenvolvedor e realizada de tal forma que a plataforma forneça garantias sobre a qualidade do desempenho que pode ser obtido. Contudo, o problema de gestão de recursos em plataformas de Computação em Nuvem— diretamente relacionado ao problema clássico de combinatória conhecido como *Vector Bin Packing* — é sabidamente NP-difícil e pouco se conhece sobre os métodos de gestão utilizados em plataformas comerciais (que são mantidos como segredo industrial).

Plataformas de Computação em Nuvem podem utilizar desde estratégias de escalonamento estático (onde máquinas virtuais são designadas às máquinas físicas que as executarão no início da execução), quanto dinâmicas (onde máquinas virtuais são migradas à medida que novas informações sobre a plataforma de execução são obtidas e analisadas. Para estudar novas estratégias de gestão de recursos, desenvolvemos o simulador SIMMYCLOUD.

O SIMMYCLOUD foi projetado para que novos algoritmos de gestão de recursos sejam avaliados utilizando-se traços de execução de plataformas de Computação em Nuvem reais. Abstraímos os tipos de eventos (lógicos e físicos) presentes em tais traços e fornecemos uma API que permite não só a criação de novas heurísticas de escalonamento, mas também acesso à estatísticas de desempenho em tempo de execução e métricas consolidadas de desempenho ao final da simulação.

O restante deste artigo é organizado como se segue. A Seção 2 discute os trabalhos relacionados ao estudo da gestão de recursos e consolidação de servidores em plataformas de Computação em Nuvem. Nossa proposta para a simulação de novos algoritmos de gestão utilizando-se traços de execução reais é apresentada na Seção 3, e sua arquitetura e detalhes de implementação na Seção 4. Na Seção 5 apresentamos experimentos realizados com nosso simulador para a avaliação de algoritmos clássicos de *Vector Bin Packing* aplicados no problema da gestão de recursos de uma nuvem. Detalhes sobre a documentação e demonstração de uso do simulador são apresentados na Seção 6. Finalmente, apresentamos nossas considerações finais na Seção 7.

## 2. Trabalhos relacionados

Sabe-se que a consolidação de servidores é uma estratégia efetiva para maximizar a utilização dos recursos, ao mesmo tempo que diminui o consumo energético dos *datacenters* em que é empregada [Zhang et al. 2010]. A migração de máquinas virtuais é outra estratégia que tem sido adotada em ambientes virtualizados. Com o uso de *live migration*, sistemas operacionais completos podem ser migrados de uma máquina física para outra, com interrupção de disponibilidade do serviço por intervalo menor que um segundo.

A alocação das máquinas virtuais na menor quantidade possível de máquinas físicas, considerando mais de um recurso computacional, pode ser modelado como o problema de otimização combinatória conhecido como *Vector Bin Packing*, que é NP-difícil [Panigrahy et al. 2011]. É necessário o uso de heurísticas para encontrar soluções em tempo viável para este problema. Avaliamos algumas ferramentas que propõem simular um ambiente de infraestrutura de Computação em Nuvem, de modo a permitir comparações dos efeitos de se empregar diferentes heurísticas para alocação de máquinas virtuais, bem como estratégias para predição de desempenho e migração de máquinas virtuais.

A ferramenta CloudSched [Tian et al. 2013] simula uma plataforma de computação considerando quatro recursos: *rede*, *disco*, *memória* e *CPU*. A ferramenta permite realizar simulações com *workloads* gerados artificialmente ou com traços de execução preexistentes. A simulação considera que a execução é realizada em um ambiente virtualizado, e a alocação de máquinas virtuais ocorre dinamicamente. A ferramenta não foi projetada para analisar a migração de máquinas virtuais entre máquinas físicas.

Um arcabouço de simulação de ambientes de Computação em Nuvem precursor do CloudSched é o CloudSim [Calheiros et al. 2011]. Ele permite a modelagem de fatores tanto técnicos quanto de negócios, inerentes a todo provedor de computação como serviço. Com a arquitetura direcionada à modelagem de comportamentos, a ferramenta não permite a utilização de um traço de execução preparado previamente. O CloudAnalyst [Wickremasinghe et al. 2010] provê uma interface gráfica e outras extensões para o CloudSim, de modo a facilitar a visualização dos resultados de suas simulações.

O SIMMYCLOUD foi idealizado como um *framework* capaz de simular plataformas de Computação em Nuvem, com foco em avaliar diferentes soluções para desafios de infraestrutura desses ambientes, e quais os efeitos dessas soluções em conjuntos de dados previamente coletados. O fluxo da simulação permite avaliar efeitos de diferentes estratégias para alocação dinâmica de máquinas virtuais, predição de desempenho das mesmas, migração de máquinas virtuais e critérios para desligar máquinas físicas ou colocá-las em estado de baixo consumo de energia.

## 3. SIMMYCLOUD e suas funcionalidades

Neste trabalho propomos a ferramenta SIMMYCLOUD. O SIMMYCLOUD provê um arcabouço para a simulação e análise de diferentes estratégias de gerenciamento de recursos computacionais em ambientes virtualizados. O SIMMYCLOUD foi modelado como um *sistema a eventos discretos* [Medina and Chwif 2010], onde a simulação ocorre de acordo com o processamento de uma sequência de eventos ordenados de acordo com um relógio lógico. O estado do sistema só pode ser alterado durante o processamento de um evento.

A alocação dos recursos computacionais ocorre em tempo de execução e utiliza

dados como a quantidade de recursos necessários à execução da máquina virtual, o tempo pelo qual ela deverá ser executada, etc. Os dados necessários em cada requisição são apresentados em maior detalhe na Seção 5.

Com o objetivo de comparar os efeitos no uso de diferentes estratégias de gerenciamento de recursos computacionais, quatro classes de estratégias podem ser implementadas e substituídas em arquivo de configuração:

**Predição de desempenho** algoritmo que determina (utilizando-se de alguma estratégia de predição) a quantidade estimada de recursos computacionais que uma dada máquina virtual irá utilizar no próximo intervalo de tempo da simulação;

**Alocação de máquinas virtuais** estratégia que determina qual máquina física será designada para a execução de uma máquina virtual que ainda não iniciou sua execução;

**Migração de máquinas virtuais** estratégia que utiliza-se de dados do preditor de desempenho para decidir se uma dada máquina virtual deve continuar sua execução em uma máquina física ou se deve ser migrada para uma máquina com mais recursos.

**Desligamento (ou suspensão) de máquinas físicas** determina quais máquinas físicas ociosas podem ser desligadas ou configuradas para um estado de menor consumo de energia.

Módulos de estatísticas são os responsáveis por computar dados relevantes para a comparação do emprego de diferentes estratégias. Novos módulos de estatísticas podem ser adicionados para medir e analisar novas métricas de desempenho. O SIMMYCLOUD atualmente permite a análise das seguintes métricas:

**# falhas de *QoS*** número de máquinas virtuais que executaram em uma máquina física sobrecarregada.

**# máquinas físicas ativas** quantidade de máquinas físicas ativas, ou seja, não estão em um estado de baixo consumo de energia.

**# migrações** quantidade de migrações de máquinas virtuais que ocorreram no último intervalo de tempo.

**# máquinas virtuais em espera** quantidade de máquinas virtuais que não foram executadas por falta de recursos disponíveis [1].

**Capacidade residual total** soma da capacidade residual das máquinas físicas ativas. Veja definição a seguir.

**% capacidade residual em cada máquina física** porcentagem da capacidade residual de cada máquina física.

***Stretch* de cada máquina virtual** – razão entre o tempo que a máquina virtual demora para ser executada (contado a partir de sua submissão no sistema) pelo tempo realmente utilizado para a sua execução.

Cabe aqui elucidar o que é considerado uma falha de *QoS* e a capacidade residual de uma máquina física. Uma falha de *QoS* (*Quality of Service*) ocorre quando uma máquina virtual está alocada em uma máquina física sobrecarregada, ou seja, a quantidade de recursos disponíveis na máquina física é inferior à soma das demandas das máquinas virtuais alocadas nessa máquina física. A *capacidade residual* [Panigrahy et al. 2011] de uma máquina física é definida como a norma do vetor residual $\vec{r}$, cujas componentes são determinadas pela quantidade ainda disponível de cada recurso na máquina física, como ilustrado na Figura 1.

[1] Uma máquina virtual em espera não é considerada uma falha de *QoS*.

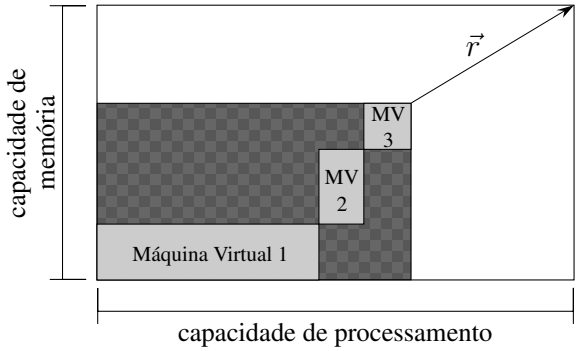



**Figura 1. Vetor residual de uma máquina física**

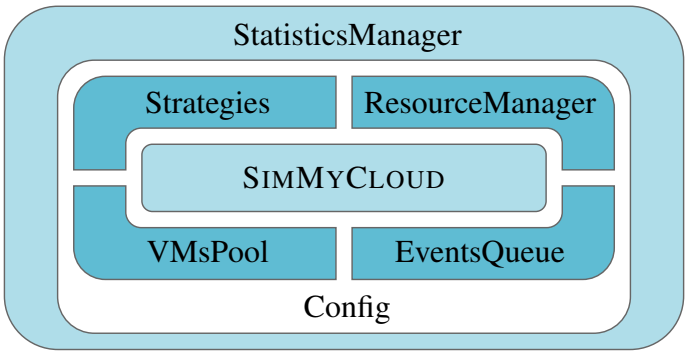



**Figura 2. Arquitetura do SIMMYCLOUD**

## 4. Arquitetura do SIMMYCLOUD

O SIMMYCLOUD foi desenvolvido seguindo alguns princípios, a saber:

- *Extensibilidade* – Regras e fluxos que podem variar entre simulações devem ser componentizados, de forma a permitir fácil adição ou substituição de regras de negócio.
- *Softcoding* – Variáveis e módulos não devem ser *hard-coded*; para executar simulações diferentes não devem ser necessárias alterações no código fonte.
- *Duck typing* – A fim de evitar uso de computação para verificar constantemente se estratégias e módulos em uso são de um tipo específico, assume-se que o objeto possui os métodos e propriedades necessários de acordo com sua semântica.
- *Agnosticismo perante componentes* – Cada componente tem sua responsabilidade e sabe de que dados ou outros componentes da simulação precisa, assim, a injeção de dependências é minimizada e todos os componentes são, de modo padronizado, acessíveis.

Tais princípios foram úteis para que a arquitetura ficasse enxuta e permitisse que simulações com diferentes módulos e variáveis fossem facilmente configuradas.

A Figura 2 ilustra os componentes principais do SIMMYCLOUD, cujas responsabilidades são:

- SIMMYCLOUD – executar o fluxo principal do simulador, coordenando o acesso aos demais componentes a fim de que executar o comportamento esperado a cada evento.

- *Strategies* – tomar decisões quanto ao gerenciamento dos recursos computacionais, quando necessário.
- *ResourceManager* – gerenciar os recursos computacionais em si, desde ligar e desligar máquinas físicas a alocar e liberar recursos dessas máquinas.
- *VMsPool* – gerenciar as máquinas virtuais que não estão alocadas, o que geralmente ocorre quando nenhuma máquina física é capaz de prover os recursos requisitados por uma ou mais máquinas virtuais.
- *EventsQueue* – gerenciar os eventos de simulação agendados, priorizando-os para execução.
- *StatisticsManager* – gerenciar os módulos de estatísticas ativos na simulação, notificando-os quando ocorre eventos no gerenciamento dos recursos computacionais.
- *Config* – gerenciar os componentes configurados para a simulação atual, sendo uma interface para acesso a todos os componentes.

## 5. Avaliação experimental

Para avaliar o simulador, utilizamos um traço de execução disponibilizado publicamente pelo Google, o `ClusterData2011_1` [Wilkes and Reiss 2013], que possui informações detalhadas sobre todos os eventos que ocorreram em uma "célula"[2] de um *datacenter*, com aproximadamente 12 mil máquinas físicas. Os dados coletados representam um período de 29 dias do mês de maio de 2011, anonimizados para fins de confidencialidade.

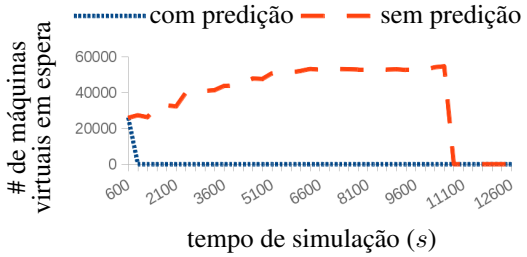
Para essa avaliação experimental, utilizamos os dados referentes às primeiras duas horas desse traço de execução, período suficiente para uma comparação preliminar das estratégias. Importamos os dados do traço para o simulador utilizando *scripts* implementados no SIMMYCLOUD. A entrada do simulador é composta pelos seguintes dados para cada máquina virtual:

- *momento da submissão* – tempo de simulação em é requisitada a execução da máquina virtual;
- *identificador* – identificador da máquina virtual, que pode conter letras, números e pontuação, excetuando a "," (vírgula);
- *CPU e memória solicitadas* – recursos de processamento e memória RAM solicitadas para a execução desta máquina virtual;
- *tempo de execução* – tempo estimado para a execução completa da máquina virtual.

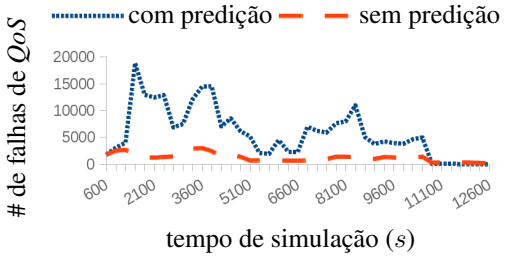
A simulação foi configurada para execução com as estratégias de gerenciamento de recursos computacionais descritas a seguir. A estratégia `FirstFit` foi escolhida para realizar alocação das máquinas virtuais, a qual as aloca na primeira máquina física capaz de prover a quantidade de recursos requisitados. O algoritmo de predição de desempenho estima o uso de recursos necessários baseado na média das últimas cinco medições de cada máquina virtual. A estratégia de migração `MigrateIfOverloaded` migra máquinas virtuais de máquinas físicas sobrecarregadas para outras. Caso uma máquina física fique ociosa, a estratégia `PowerOffIfEmpty` a desliga (ou a suspende).

Utilizamos o SIMMYCLOUD para comparar o desempenho da estratégia `FirstFit` com e sem o uso de um algoritmo de predição de desempenho para amparar a gestão de

[2]Uma célula é um conjunto de máquinas que compartilham um mesmo sistema de gerenciamento de *clusters*, geralmente presentes em um único *cluster*.




(a) Número de máquinas virtuais em espera em função do tempo




(b) Número de falhas de *QoS* em função do tempo

**Figura 3. Número de falhas de *QoS* e quantidade de máquinas virtuais ociosas obtidas com e sem o uso de informações de predição de desempenho.**

recursos. A não utilização de um algoritmo de predição de desempenho implica que as decisões de gerenciamento de recursos serão baseadas apenas na estimativa dada pelo desenvolvedor. A comparação é apresentada na Figura 3.

Como pode ser observado na Figura 3(a), o uso de um algoritmo de predição de desempenho é capaz de evitar a presença de máquinas virtuais na fila de espera por uma máquina física na maior parte do tempo. Sem o uso de predição, elas precisam esperar o término de outras máquinas virtuais antes de serem alocadas a uma máquina física, o que só ocorre a partir do tempo $t = 11.100$ segundos.

Uma análise aprofundada (Figura 3(b)) mostra, entretanto, que a diminuição da fila de espera implica em um aumento significativo do número no número de falhas de *QoS*. Isso ocorreu pois a estratégia de predição de desempenho subestimou, em alguns casos, a quantidade de recursos que as máquinas virtuais necessitariam nos intervalos de tempo seguintes, o que levou à sobrecarga de algumas máquinas físicas.

A partir dessa análise, podemos planejar, por exemplo, nova simulação com a mesma estratégia de predição, porém com uma estratégia de alocação de máquinas virtuais que não busque utilizar 100% da capacidade de recursos computacionais disponíveis nas máquinas físicas. Desse modo, teríamos uma tolerância a erros da estratégia de predição, ainda que com possível aumento na quantidade de máquinas virtuais em espera.

## 6. Documentação e demonstração no SBRC

Na página da ferramenta, disponibilizada publicamente em `http://www.ime.usp.br/~cassiop/simmycloud/`, é possível encontrar uma breve descrição do SIMMYCLOUD, sua documentação e seu código fonte (escrito na linguagem Python e disponibilizado como software livre). O Guia do Usuário apresenta com mais detalhes o funcionamento da ferramenta, seus componentes e meios para extendê-los.

A demonstração da ferramenta no SBRC mostrará todos os passos necessários para sua utilização, desde o *download* do traço de execução real disponibilizado pelo Google até a simulação e análise das métricas de desempenho calculadas automaticamente pela ferramenta. A demonstração apresentará todas as estratégias implementadas pelo SIMMYCLOUD e realizará, ao vivo, uma comparação de duas execuções configuradas com estratégias de gestão de recursos distintas. Apresentaremos como analisar graficamente os dados gerados pelo SIMMYCLOUD.

## 7. Considerações finais

Neste trabalho apresentamos o SIMMYCLOUD, um novo simulador para amparar a criação e análise de métodos de gerenciamento de recursos em plataformas de Computação em Nuvem.

O SIMMYCLOUD implementa diversas estratégias para a consolidação de servidores baseadas no problema clássico conhecido como *Vector Bin Packing*. Além disso, sua arquitetura foi projetada para armazenar e fornecer informações e métricas de desempenho relevantes, que podem ser facilmente utilizadas para o desenvolvimento de novos algoritmos.

Neste artigo ilustramos a utilização da ferramenta utilizando como exemplo a análise comparativa entre o uso ou não de um algoritmo simples para predição de desempenho. A análise foi realizada utilizando-se um traço de execução de um provedor de Computação em Nuvem comercial bastante conhecido.

O código fonte da ferramenta e instruções para sua utilização estão disponíveis publicamente em `http://www.ime.usp.br/~cassiop/simmycloud/`. O código é disponibilizado sob a licença permissiva MIT, e seu guia do usuário sob a licença Creative Commons - Atribuição 4.0 Internacional.

## Referências

[Calheiros et al. 2011] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50.

[Medina and Chwif 2010] Medina, A. C. and Chwif, L. (2010). *Modelagem e Simulação de Eventos Discretos*. Editora Campus, 3 edition.

[Panigrahy et al. 2011] Panigrahy, R., Talwar, K., Uyeda, L., and Wieder, U. (2011). Heuristics for vector bin packing. Technical report, Microsoft Research. Disponível em: `http://research.microsoft.com/apps/pubs/default.aspx?id=147927`.

[Tian et al. 2013] Tian, W., Zhao, Y., Xu, M., Zhong, Y., and Sun, X. (2013). A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center. *IEEE Transactions on Automation Science and Engineering*.

[Wickremasinghe et al. 2010] Wickremasinghe, B., Calheiros, R. N., and Buyya, R. (2010). CloudAnalyst: A CloudSim-based visual modeller for analysing cloud computing environments and applications. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 446–452. IEEE.

[Wilkes and Reiss 2013] Wilkes, J. and Reiss, C. (2013). ClusterData2011_1 - Details of the ClusterData-2011-1 trace. Google Project Hosting – `http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1`. [Online; acessado em 23/02/2014].

[Zhang et al. 2010] 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.

# HighFrame: Uma Ferramenta para Desenvolvimento em Alto Nível e Deployment Automático de Sistemas Distribuídos Baseados em Componentes

**Saulo Eduardo Galilleo Souza dos Santos [1,2], Artêmio Oliveira de Andrade Junior[1], Victor Machado Pasqualino [1], Tarcísio da Rocha[1], Felipe Oliveira Carvalho [1]**

[1]Departamento de Computação – Universidade Federal de Sergipe (UFS)
Caixa Postal 353 – 49100-000 – São Cristovão – SE – Brasil

[2]Instituto Federal de Educação, Ciênc. e Tec. de Sergipe (IFS) - Campus São Cristóvão
Caixa Postal 11 – 49100-000 – São Cristovão – SE – Brasil

{saulogalilleo,victorpasqualino}@yahoo.com.br

{tarcisiorocha,ocfelipe}@gmail.com, artemiojr182@hotmail.com

***Abstract.*** *This paper presents HighFrame, an integrated solution for high-level development and deployment that aims to reduce the complexity of developing heterogeneous component-based distributed systems. The developer keeps the focus on the business of the system (generic components in POJO style + Fraclet annotations) and uses a graph model to define the distributed system architecture; HighFrame performs the deployment process of the system automatically, generating technical code of the component models and the remote bindings, distributing, instantiating and making the the system ready for use.*

***Resumo.*** *Este trabalho apresenta o HighFrame, uma solução integrada para desenvolvimento e deployment em alto nível que tem como propósito diminuir a complexidade do desenvolvimento de sistemas distribuídos heterogêneos baseados em componentes. O desenvolvedor mantém o foco no desenvolvimento do negócio do sistema (componentes genéricos tipo POJO + anotações Fraclet) e usa um modelo gráfico para definir a arquitetura distribuída do sistema; o HighFrame realiza o processo de implantação do sistema de forma automática, gerando o código técnico dos modelos componentes e dos bindings remotos, distribuindo e instanciando o sistema, deixando-o pronto para o uso.*

## 1. Descrição e motivação do problema tratado pela ferramenta

Os sistemas distribuídos tem se mostrado cada vez mais complexos sendo compostos por partes heterogêneas interconectadas dinamicamente na formação de sistemas ainda mais ricos[Blair et al. 2011]. Esse tem sido um dos grandes desafios da nova geração de projetistas de middlewares e outras soluções para o desenvolvimento de sistemas distribuídos.

Uma abordagem promissora que tem sido adotada no desenvolvimento de sistemas distribuídos complexos é a engenharia de software baseada em componentes. O uso de um modelo de componentes permite o desenvolvimento de componentes reutilizáveis, que expõe operações de dependência (serviços requeridos e fornecidos), promovendo também a modularidade e configurabilidade, o que facilita a construção de sistemas

dinâmicos. Atualmente, diversos modelos de componentes para sistemas distribuídos já disponibilizam recursos para reconfiguração de *middlewares* e sistemas distribuídos (ex: Fractal, SCA, OSGi) [Bennour et al. 2009]. Porém, apesar desses benefícios, a adoção deste modelo de desenvolvimento pode representar a introdução de mais complexidade no processo de desenvolvimento de *software* uma vez que o desenvolvedor precisa destinar esforços para o desenvolvimento de código técnico do modelo de componentes específico que foi adotado na implementação da aplicação [Rouvoy and Merle 2009].

Além desse desafio, o processo de desenvolvimento de sistemas distribuídos baseados em componentes envolve outros obstáculos como: *(i)* a comunicação remota entre os componentes distribuídos; *(ii)* a implantação dos componentes em nós distribuídos; e *(iii)* a interconexão de componentes desenvolvidos em modelos heterogêneos. Essas dificuldades podem tornar complexa a tarefa de desenvolver um sistema distribuído, mesmo quando esse possui requisitos funcionais simples. Dado esse nicho emergente dos sistemas distribuídos baseados em componentes, um dos desafios relevantes é o de diminuir a complexidade do desenvolvimento desses tipos de sistemas distribuídos.

Neste trabalho é apresentada uma ferramenta gráfica para planejamento e deployment em alto nível de arquiteturas de sistemas distribuídos baseados em componentes – a HighFrame. Ela é uma solução integrada que reduz a complexidade do desenvolvimento desses tipos de sistemas, oferecendo soluções que lidam diretamente com os seus obstáculos. Com ela o desenvolvedor volta a preocupação para o desenvolvimento dos requisitos funcionais do sistema e da sua arquitetura em um modelo gráfico de alto nível. Outras preocupações complexas envolvidas são tratadas automaticamente pela HighFrame como, *(i)* a criação do código técnico dos modelos de componentes (ex: OpenCom, Fractal) e do código técnico da comunicação remota entre os componentes distribuídos (ex: Web Services, RMI), *(ii)* a implantação dos componentes nos nós distribuídos e *(iii)* a interconexão entre componentes heterogêneos (com modelos de componentes diferentes).

Algumas ferramentas publicadas na literatura também identificam a necessidade de soluções no sentido de facilitar o processo de desenvolvimento de sistemas baseados em componentes. Por exemplo, [Sentilles et al. 2009] apresenta uma solução para o desenvolvimento de sistemas embarcados com geração automática de componentes específicos no modelo SaveCCM; [Riba and Cervantes 2007] apresentam uma ferramenta MDA para a construção de aplicações orientadas a serviços baseadas em componentes onde modelos de alto nível são transformados em componentes específicos no modelo OSGi. Porém, além destes trabalhos se limitarem apenas a uma tecnologia de componentes específica, eles não consideram os problemas dos cenários que envolvem distribuição. Além disso, esses trabalhos não retiram do desenvolvedor a complexidade de desenvolvimento de código técnico para comunicação remota, como também não provê comunicação remota entre componentes heterogêneos.

## 2. HighFrame

A HighFrame tem como objetivo simplificar o desenvolvimento de sistemas distribuídos baseados em componentes, permitindo ao desenvolvedor manter o foco no negócio do sistema. Ela propõe uma solução integrada que inclui: *(i)* desenvolvimento de componentes baseado em implementações genéricas focada no negócio do sistema; *(ii)* definição

da arquitetura do sistema baseado em um modelo gráfico de alto nível; *(iii) deployment* automático da arquitetura nos nós distribuídos disponibilizando o sistema na sua forma funcional. A Arquitetura do HighFrame framework é apresentada na figura 1.

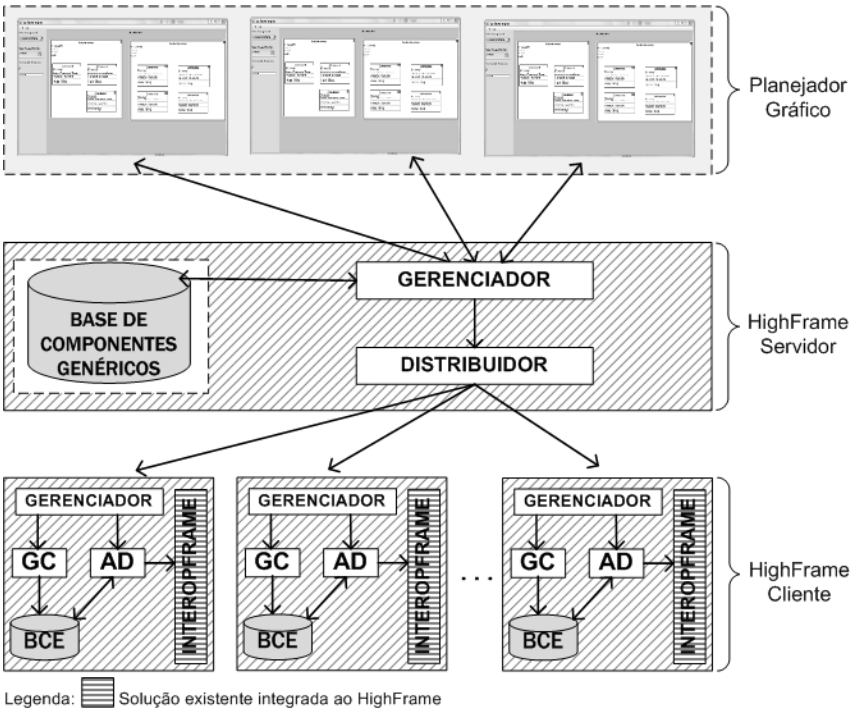



**Figura 1. Arquitetura do HighFrame framework**

Para que o desenvolvedor possa manter o foco no negócio do sistema a ser desenvolvido, a HighFrame fornece uma camada de abstração para o desenvolvimento de componentes. Esta camada é baseada no modelo de anotações Fraclet [Rouvoy and Merle 2009], que representa preocupações do desenvolvimento baseado em componentes independente de tecnologia de componentes. Por exemplo, desenvolver um componente Java com esse modelo seria equivalente a desenvolver uma classe no estilo POJO incluindo as anotações Fraclet no código fonte como: @*Component* (marca um componente), @*Interface* (marca uma interface do negócio do sistema), @*Requires* (marca uma interface requerida) e @*Provides* (marca uma interface provida). O código fonte anotado pode representar um componente e passa a ser inserido em uma base de componentes genéricos do *HighFrame*, esta base centraliza todos componentes genéricos em um servidor para que sejam disponibilizados para reutilização em qualquer nova arquitetura independente de um dado modelo de componentes específico, distribuição ou formas de comunicação.

O HighFrame Servidor disponibiliza através do módulo gerenciador um catálogo de componentes genéricos existentes na base de componentes genéricos, isto para que o desenvolvedor possa compor a arquitetura do sistema e o plano de *deployment* distribuído do sistema. O processo de composição de um sistema pode ser feito graficamente com a HighFrame Designer (planejador gráfico) – com ele o desenvolvedor, define os nós distribuídos, arrasta os componentes para os respectivos nós e cria as interconexões entre os mesmos. Além disso, o desenvolvedor também define qual(quais) modelo(s) de componentes específico(s) a serem usado(s) no *deployment* e as formas de *binding* remoto entre

eles (ex: Web Services SOAP/REST, RMI...). Para representar internamente a arquitetura projetada pelo desenvolvedor, a HighFrame usa uma ADL específica chamada HighADL. Um plano de *deployment* é também criado automaticamente com a HighADL que é posteriormente usado pela HighFrame para a realização do *deployment* distribuído. A arquitetura e o plano de *deployment* em HighADL são submetidos através do HighFrame Designer para o HighFrame Servidor. O módulo gerenciador do HighFrame Servidor processa as ADL´s e seleciona cada componente genérico referenciado para realizar a distribuição para os seus respectivos nós distribuídos.

Em cada nó distribuído reside o HighFrame Cliente que é o responsável por receber o plano de *deployment* e os componentes genéricos para realizar *deployment* da subarquitetura do nó (um projeto pode envolver vários nós, cada um com sua subarquitetura). Depois disso, o HighFrame Cliente usa o seu Gerador de Componentes (GC) para gerar os componentes específicos em um dado modelo (ex: OSGi) com base no componente genérico e armazená-los na Base de Componentes Específicos (BCE). Em seguida é acionado o Agente de *Deployment* (AD) que faz a instalação, ativação e interconexão (*binding* local) dos componentes como também realiza a interligação (*binding* remoto) entre os componentes que se conectam a componentes de outros nós. Para o processo de *binding* remoto é utilizada uma solução chamada InteropFrame [Nascimento et al. 2013] que foi desenvolvida para prover interconexão automática entre componentes distribuídos possivelmente heterogêneos. O InteropFrame gera automaticamente os *proxies* que realizam o *binding* remoto. A Arquitetura foi implementada baseada em plugins, de forma a ser extensível para novos modelos de componentes através de *templates*.

## 3. HighFrame Designer

O HighFrame Designer é o planejador gráfico que pertence ao HighFrame. Ele tem o objetivo de fornecer um ambiente para composição da arquitetura de sistemas distribuídos e criação de plano de *deployment* baseado em um modelo gráfico com as facilidades do modo "arrastar e soltar". O HighFrame Designer possibilita que o usuário desenvolvedor crie a arquitetura do sistema distribuído através de um catálogo de componentes genéricos. A ferramenta disponibiliza ao usuário desenvolvedor os componentes genéricos para utilização no modo *design* da ferramenta. Estes componentes genéricos são desenvolvidos previamente independentemente de tecnologia de componentes, como descrito anteriormente. Com isso, ele retira do desenvolvedor a necessidade de conhecer o código técnico para desenvolvimento de componentes, comunicação remota distribuída e *deployment* distribuído. Ao invés disso, o usuário foca no planejamento da arquitetura e na definição dos parâmetros para o *deployment* do sistema como, o modelo de componentes específico a ser usado, os tipos de *binding* remoto e os endereços dos nós distribuídos.

### 3.1. Funcionamento

A figura 2 apresenta um exemplo no HighFrame Designer de uma arquitetura composta por dois componentes, cada um em um nó distinto. Na caixa "Select Component" do ambiente de design estão disponíveis os componentes genéricos que foram armazenados no HighFrame Servidor. O HighFrame Designer permite criação de uma arquitetura completa como também de uma subarquitetura independente. Subarquiteturas independentes podem ser criadas e salvas para posterior reutilização. Elas ficam disponíveis na ferramenta na caixa "Select SubArquitecture".

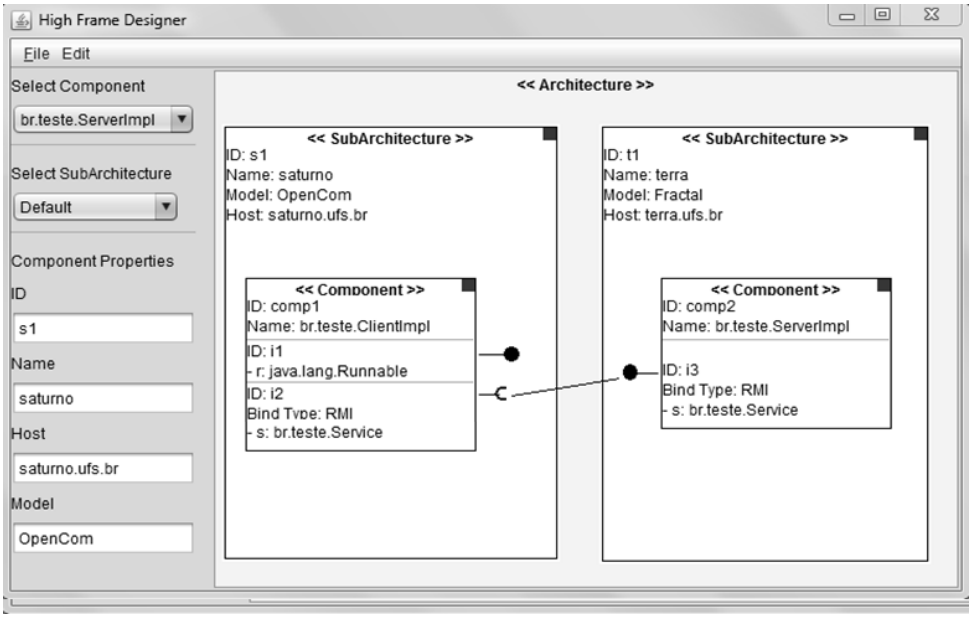



**Figura 2. Ambiente da IDE**

Uma subarquitetura refere-se a um subsistema que irá ser executado em um nó distribuído. A subarquitetura pode ser vista como um componente composto que exporta interfaces para conexão com outras subarquiteturas distribuídas. No exemplo apresentado temos duas subarquiteturas, uma denominada "saturno", que possui o endereço saturno.ufs.br como nó de destino. A outra é denominada "terra", que possui como nó de destino terra.ufs.br. Na subarquitetura é definido também o modelo de componentes para o qual os componentes presentes na mesma serão transformados. Cada nó distribuído deverá conter o HighFrame Cliente, pois este módulo distribuído irá proporcionar o suporte ao processo de *deployment* e à comunicação entre componentes distribuídos.

O uso do HighFrame Designer como ferramenta para composição do sistema e criação do plano de *deployment* tem como produto final dois arquivos para submissão ao HighFrame Servidor. O Planejador possui uma função chamada "Execute Deployment", esta função faz a transformação do modelo gráfico em alto nível para dois arquivos XML em HighADL: *data.xml* e *plan.xml* que consistem na representação da arquitetura distribuída e no plano de *deployment*, respectivamente. O código fonte 1 apresenta a HighADL gerada para o exemplo apresentado na figura 2.

**Código Fonte 1. HighADL - arquivo data.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<architecture>
 <subarchitecture idsubarchitecture="s1">
  <component id="comp1" name="clientImpl">
   <provideinterface idinterface="i1" name="r" signature="java.lang.Runnable"/>
   <requiredinterface idinterface="i2" name="s" signature="Service"/>
  </component>
  <exportinterface idinterface="i2"/>
 </subarchitecture>
 <subarchitecture idsubarchitecture="t1">
  <component id="comp2" name="serverImpl">
   <provideinterface idinterface="i3" name="s" signature="Service"/>
  </component>
  <exportinterface idinterface="i3"/>
 </subarchitecture>
```

```
</architecture>
```

Na HighADL são definidas as subarquiteturas que compõem o sistema e quais interfaces são exportadas. Este XML refere-se a descrição da arquitetura do sistema baseado em componentes. O segundo arquivo é o plano de *deployment*, que possui informações necessárias para que o HighFrame execute o *deployment* distribuído. O quadro 2 apresenta o plano de *deployment* gerado para o exemplo apresentado.

**Código Fonte 2. Plano de deployment - arquivo plan.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<plan>
 <deployment idsubarchitecture="s1" componentmodel="OpenCom" host="saturno.ufs.br"/>
 <deployment idsubarchitecture="t1" componentmodel="Fractal" host="terra.ufs.br"/>
 <remotebindind idexportinterfaceclient="i2" idexportinterfaceserver="i3" typebinding="
     RMI"/>
</plan>
```

O plano de *deployment* contém informações para que o HighFrame execute o *deployment* distribuído. No código apresentado, temos a definição de *deployment* das duas arquiteturas correspondentes. A primeira com a subarquitetura 's1' destinada ao nó saturno.ufs.br com a transformação dos seus componentes genéricos em componentes do modelo OpenCom. A segunda, 't1', destinada ao nó terra.ufs.br com a transformação dos componentes genéricos em componentes do modelo Fractal. O elemento *remotebinding* define as interfaces exportadas de cada subarquitetura destinada a nós distintos e também o tipo de conexão entre os nós distribuídos. Para o exemplo acima, a comunicação remota definida foi RMI.

Apesar de, por motivo de espaço, o exemplo apresentado acima ter sido o mais trivial possível, a HighFrame é flexível o suficiente para permitir a definição uma arquitetura com várias subarquiteturas, cada uma delas com vários componentes interconectados. Cada subarquitetura pode usar um modelo de componentes diferente, se for o caso. Isso é útil tanto pelo fato de permitir a reutilização de componentes já existentes quanto pelo fato de permitir que o desenvolvedor explore as diferentes potencialidades e características de cada modelo de componentes na composição de subarquiteturas diferentes.

Com a HighFrame é possível também que as subarquiteturas de uma mesma arquitetura possam ser interconectadas usando *bindings* remotos diferentes. Essa flexibilidade permite também explorar a melhor forma de *binding* para cada caso. Por exemplo, pode-se escolher RMI como *binding* entre componentes de duas subarquiteturas localizadas em nós de uma rede local para favorecer o desempenho da comunicação, enquanto que usar WebServices REST para o binding entre duas subarquiteturas localizadas em redes distintas pelo fato do RMI não oferecer suporte padrão para redes distintas. A escolha é feita pelo desenvolvedor, porém a geração do código técnico do *binding* é feita automaticamente pelo HighFrame através do InteropFrame.

## 4. Demonstração planejada para o Salão de Ferramentas-SBRC

A demonstração que foi preparada para o SBRC usará um cenário simples com dois nós (que podem ser dois notebooks pessoais pré-configurados em uma rede local básica). O primeiro nó conterá uma instância do HighFrame Server e uma instância do HighFrame-Client; o segundo nó conterá uma instância do HighFrame Client e uma instância do

HighFrame Designer (apesar de que essas instâncias podem ser totalmente distribuídas em nós distintos). Ela consistirá no processo de planejamento e implantação de um servidor web distribuído chamado Comanche WebServer [Rouvoy and Merle 2009]. A arquitetura baseada em componentes do Comanche WebServer a ser usada é apresentada na figura 3. Ela é distribuída em dois nós: o "No1" com o *Frontend* e o "No2" com o *Backend* do exemplo.

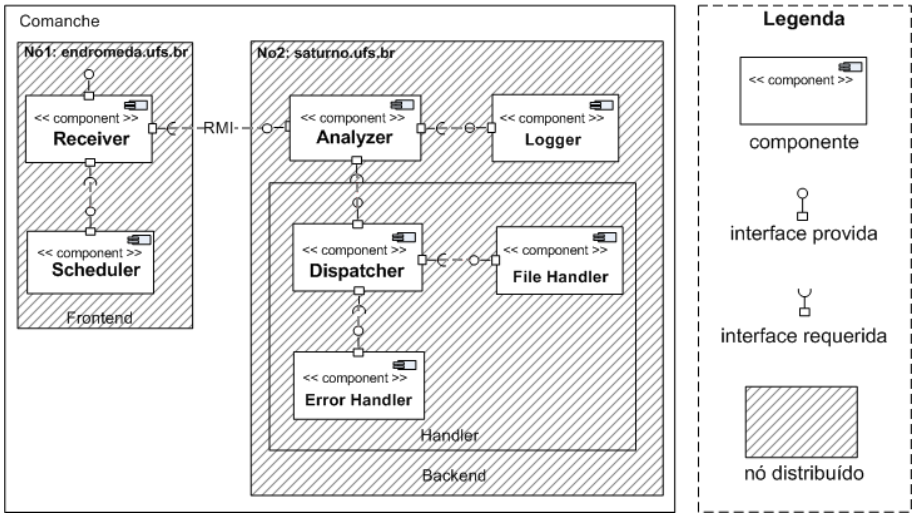



**Figura 3. Modelo do comanche WebServer**

Em particular, o servidor web comanche é composto pelos seguintes componentes: *Receiver*, *Scheduler*, *Analyzer*, *Logger*, *Dispatcher*, *FileHandler* e *ErrorHandler*. Mais detalhes sobre o papel de cada um desses componentes podem ser encontrados na documentação do HighFrame disponibilizada no link abaixo. Para o modelo proposto, teremos transformações para o modelo OpenCom no *Frontend* e para o modelo Fractal no *Backend*. O componente *Receiver* localizado no nó "endromeda.ufs.br" conecta-se através de RMI ao componente *Analyzer* localizado no nó "saturno.ufs.br" ligando assim o *Frontend* ao *Backend*.

A demonstração será iniciada mostrando brevemente como um componente genérico é desenvolvido com o foco único do negócio do sistema. Para isso mostraremos trechos do componente "*Analyzer*" destacando a sua característica POJO e a simplicidade do uso das anotações Fraclet. Em seguida será mostrado como a arquitetura do Comanche Web Server pode ser criada de forma rápida e simples com o uso do HighFrame Designer. A próxima etapa, será o processo de *deployment* do Web Server mostrando como o HighFrame realiza automaticamente o *deployment* do *Frontend* no "No1" (primeiro notebook) e do *Backend* no "No2" (segundo notebook) usando somente os componentes genéricos e o modelo em alto nível criado no planejador. Por fim, será realizado um teste simples de funcionamento com o Servidor Web distribuído que, neste momento, já terá sido ativado e já estará pronto para ser requisitado.

Apesar da arquitetura desse exemplo ser composta por vários componentes, todo esse processo de demostração levará poucos minutos, enfatizando a capacidade da ferramenta de agilizar o processo de desenvolvimento e *deployment* de sistemas distribuídos baseados em componentes, considerando que os componentes genéricos da composição

já estarão implementados e armazenados na base do HighFrame.

O **HighFrame**, o seu **manual de uso** e sua **documentação** estão disponíveis em `https://github.com/saulogalilleo/HighFrame/wiki/HighFrame`

## 5. Conclusão e Trabalhos Futuros

Neste artigo apresentamos o HighFrame, uma solução integrada para o desenvolvimento e *deployment* em alto nível de sistemas distribuídos baseados em componentes genéricos visando a redução da complexidade do desenvolvimento dos mesmos. Suas principais características: *(i)* permite ao desenvolvedor manter o foco no negócio do sistema (implementação dos requisitos funcionais na forma de componentes genéricos) e especificação da arquitetura distribuída através de um modelo gráfico de alto nível; *(ii)* automatização do processo de *deployment* envolvendo a geração do código técnico dos componentes específicos, distribuição dos componentes entre os nós distribuídos, instanciação e interligação entre os mesmos; *(iii)* geração automática dos *proxies* para realização do *binding* remoto. Um outro ponto relevante é que a arquitetura do HighFrame é flexível para dar suporte a novos modelos de componentes ou novas formas de *bindings* remotos na forma de *plugins*, ou seja, sem precisar alterar ou recompilar as partes existentes ferramenta.

Os trabalhos futuros incluem: *(i)* o desenvolvimento de plugins para outros modelos de componentes como o CCM e o SCA e novas formas de *binding* como WebServices REST (atualmente são suportados os modelos de componentes OpenCom e Fractal e os *bindings* do tipo RMI e WebServices SOAP); *(ii)* suporte a componentes desenvolvidos em outras linguagens (atualmente somente componentes em Java são suportados); *(iii)* desenvolvimento de uma versão do planejador integrada à IDE Eclipse.

## References

Bennour, B., Henrio, L., and Rivera, M. (2009). A reconfiguration framework for distributed components. In *Proceedings of the 2009 ESEC/FSE Workshop on Software Integration and Evolution Runtime*, SINTER'09, pages 49–56, NY, USA. ACM.

Blair, G., Paolucci, M., Grace, P., and Georgantas, N. (2011). Interoperability in Complex Distributed Systems. In *11th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Connectors for Eternal Networked Software Systems*. Springer.

Nascimento, S. C., Carvalho, F. O., and da Rocha, T. (2013). Towards interoperability between heterogeneous distributed components. In *12th International Workshop on Adaptive and Reflective Middleware*, ARM '13, pages 3:1–3:7, NY, USA. ACM.

Riba, N. and Cervantes, H. (2007). A mda tool for the development of service-oriented component-based applications. In *ENC*, pages 149–156. IEEE Computer Society.

Rouvoy, R. and Merle, P. (2009). Leveraging component-based software engineering with fraclet. *annals of telecommunications*, 64:65–79.

Sentilles, S., Pettersson, A., Nystrom, D., Nolte, T., Pettersson, P., and Crnkovic, I. (2009). Save-ide - a tool for design, analysis and implementation of component-based embedded systems. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 607–610, Washington, USA. IEEE.


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Salão de Ferramentas do SBRC 2014

## *Sessão Técnica 2*

## *Sessões Computação Paralela e Distribuída II e Tolerância a Falhas*

# Naegling: Um Sistema para Implementação e Administração de *Clusters* e Submissão de Tarefas

**Daniel Yokoyama[2], Henrique Klôh[3], Bruno Schulze[1], Antonio Mury[1]**

[1]ComCiDiS - Laboratório Nacional de Computação Científica - LNCC
Petrópolis-RJ - Brasil

[2]Laboratório Nacional de Computação Científica - LNCC
Petrópolis-RJ - Brasil

[3]Faculdade de Educação Tecnológica do Estado do Rio de Janeiro - FAETERJ
Petrópolis-RJ - Brasil

***Abstract.*** *The configuration process and deployment of a cluster can be time consuming and technically complex. Allied to this problem, it is common for computational resources to be dedicated to running applications that do not use all the available resource in the best possible way rendering it underutilized. Given these assumptions, this paper presents a tool to create and manage virtual clusters to facilitate its configuration and administration, as well as provide its users with a simplified interface for submitting jobs. This proposal seeks to automate the process of creating and managing clusters, simplify the process of submitting applications and enable the creation of virtual clusters on demand for different purposes, maximizing the efficiency of resource usage.*

***Resumo.*** *O processo de configuração e disponibilização de um cluster pode ser demorado e tecnicamente complexo. Aliado a este problema, é comum que recursos computacionais fiquem dedicados para execução de aplicações que não utilizam o recurso da melhor forma tornando-o subutilizado. Considerando essas premissas, este trabalho apresenta uma ferramenta capaz de criar e gerenciar clusters virtuais visando facilitar a sua configuração e administração, bem como fornecer aos seus usuários uma interface simplificada para a submissão de tarefas. Esta proposta procura automatizar o processo de criação e gerência de clusters, simplificar o processo de submissão de aplicações e possibilitar a criação de clusters virtuais sob-demanda para diferentes fins, maximizando a eficiência da utilização dos recursos.*

## 1. Introdução

Um *cluster* computacional consiste na agregação de vários servidores com funcionamento e capacidade de processamento independente em uma única entidade. Caracteriza-se por possuir capacidade de processamento e disponibilidade de recursos muito superiores aos módulos que a formam.

O seu uso para a solução de problemas complexos, torna-se cada dia mais comum e necessário, principalmente no cenário de CMPD (Computação Massivamente Paralela e Distribuída), permitindo a resolução de tarefas que anteriormente seriam inviáveis devido à grande quantidade de cálculos necessários para obter o resultado no tempo desejado.

Por este motivo, diversas áreas, tanto do setor acadêmico como produtivo, cada vez mais empenham-se na modelagem e paralelização de seus algoritmos. Isso permite que o processamento de uma tarefa ocorra em diversos servidores, tornando-se uma ferramenta para setores como biomedicina, a meteorologia, a astronomia, entre outros.

Entretanto, o processo de configuração e disponibilização de um *cluster* pode ser tecnicamente demorado e complexo. Necessita de pessoal capacitado para mantê-lo e configurá-lo para atender os requisitos do usuário[Pacheco 2011]. As restrições anteriormente citadas aliadas ao fato de que único ambiente não é capaz de atender a todas as necessidades dos usuários, poderá acarretar no retardo ou mesmo na impossibilidade da execução dos seus trabalhos.

Este trabalho tem como objetivos principais: 1) criação, configuração e gerência de *clusters* virtuais; 2) auxiliar os usuários na submissão e gerencia de suas aplicações nos clusters, de modo a garantir a transparência da utilização dos recursos; e 3) gerenciar a criação e configuração de *clusters* virtuais permitindo ainda a submissão de aplicações de forma a auxiliar o usuário menos experiente na utilização dos recursos.

O trabalho a seguir está organizado da seguinte forma: a seção 2 apresenta trabalhos relacionados com o processo de criação, configuração e gerencia de *clusters*; a seção 3 descreve as funcionalidades e a arquitetura da ferramenta apresentada neste trabalho e a seção 5 apresenta as conclusões e os trabalhos futuros.

## 2. Trabalhos Relacionados

A criação e configuração de *clusters* é, muitas vezes, uma tarefa onerosa devido a mão-de-obra e tempo necessários para configuração de todos os nós do *cluster*. Para auxiliar e facilitar esta tarefa, dentro do contexto deste trabalho, foram criadas algumas ferramentas que contribuem para gerência e utilização do *cluster*. A seguir serão apresentadas algumas destas ferramentas.

O *PelicanHPC* [PELICANHPC 2012] fornece uma imagem em ISO para CD e USB com um sistema operacional GNU/Linux, permitindo a rápida configuração de um ambiente de *cluster* voltado para o processamento de tarefas desenvolvidas com a biblioteca MPI. Funciona utilizando pelo menos um computador que irá iniciar utilizando a imagem ISO disponível. Para adicionar nós a este *cluster* o *PelicanHPC* utiliza o PXE (*Preboot Execution Environment*). Desta forma, a máquina que é iniciada com a imagem ISO atua como um servidor PXE, através do qual os outros nós do *cluster* receberão o sistema operacional.

O *Rocks Cluster* [ROCKSCLUSTER 2012] é uma distribuição GNU/Linux voltada para a criação *clusters*. Para seu uso o *Rocks* é instalado em um servidor, que se tornará o nó master, e a partir deste todos os nós de trabalho serão criados automaticamente utilizando o PXE, a exemplo do PelicanHPC. O *Rocks Cluster* pode ser utilizado tanto com máquinas reais como virtuais.

*Scyld ClusterWare* [SCYLD 2012] é uma opção comercial para a criação e gerência de ambientes de *cluster*. Permite a execução em ambiente de *cluster* de programas com MPI e programas PVM (*Parallel Virtual Machine*), que é uma API para programação de aplicações paralelas. A criação dos nós é executada com o auxilio do PXE e o sistema *Scyld ClusterWare* possui uma interface Web para as tarefas de controle

do *cluster*.

O *PelicanHPC* e o *Rocks Cluster* se diferenciam do trabalho proposto na medida em que se limitam a criação do ambiente. Ambos os sistemas não auxiliam na utilização dos *clusters* criados e, principalmente, têm por objetivo a criação de um único ambiente de *cluster* por conjunto de máquinas, dedicando este ambiente a apenas um tipo de utilização.

O *Scyld ClusterWare* é, dos trabalhos abordados, o que mais se aproxima, quanto ao funcionamento do sistema proposto. Porém vale ressaltar que para seu uso é necessário dedicar toda a infraestrutura na qual pretende-se criar o ambiente de *cluster*. O presente trabalho se diferencia ao permitir que até mesmo máquinas que estejam sendo utilizadas para outros fins, como por exemplo os *desktops* de uso pessoal, sejam utilizados em um ou mais *clusters*. As máquinas já existentes em outros projetos podem ser incluídas sem que suas configurações sejam alteradas e uma máquina que esteja dedicada ao sistema proposto pode ser utilizada para outros fins quando necessário. Outra diferença entre o sistema criado e o *Scyld ClusterWare*, é que neste último a submissão é feita manualmente por meio de comandos em terminal, já o sistema criado é capaz de submeter e gerenciar as tarefas através da interface gráfica.

Apesar de sua relevância no cenário de HPC, a criação de ambientes de *clusters* e sua utilização não são tarefas simples . Este trabalho procura contribuir para esta tarefa ao fornecer uma ferramenta capaz de fornecer as funcionalidades apresentadas na próxima seção.

## 3. Funcionalidades e Arquitetura da Ferramenta

O sistema aqui apresentado visa atender a necessidade de criação de *clusters* sob demanda. O *cluster* criado é capaz de atender os requisitos específicos de cada pesquisador ou projeto, eliminando assim a necessidade destes adaptarem-se a um ambiente genérico. **A ferramenta Naegling está disponível para ser acessada através das instruções disponíveis em `http://comcidis.lncc.br/salaodeferramentas.php`.**

### 3.1. Visão Geral

O sistema proposto por este trabalho está baseado em três módulos principais: Programa Servidor; Interface Gráfica do Usuário - GUI (*Graphical User Interface*); e Rotinas do *Cluster*. Na figura 1 pode-se observar a arquitetura aqui apresentada com suas relações de troca de mensagens e transferência de arquivos.

O programa servidor, entre outros atributos, é encarregado de serviços como a criação e o controle das máquinas virtuais, o gerenciamento local dos *templates* dos *clusters* utilizados e pela comunicação entre o usuário e o *cluster* propriamente dito. Este é o módulo responsável por todo o trabalho de gerenciamento e criação dos *clusters* do sistema.

A GUI ou interface gráfica é o meio pelo qual o usuário irá interagir com o sistema. Através da GUI será possível criar os *clusters*, gerenciar os *templates* disponíveis, submeter os trabalhos a fim de serem processados, receber os resultados destes trabalhos e ainda a interface fornecerá um acesso gráfico ao ambiente de *clusters*, permitindo assim, uma forma para controlar diretamente o *cluster* e corrigir erros.

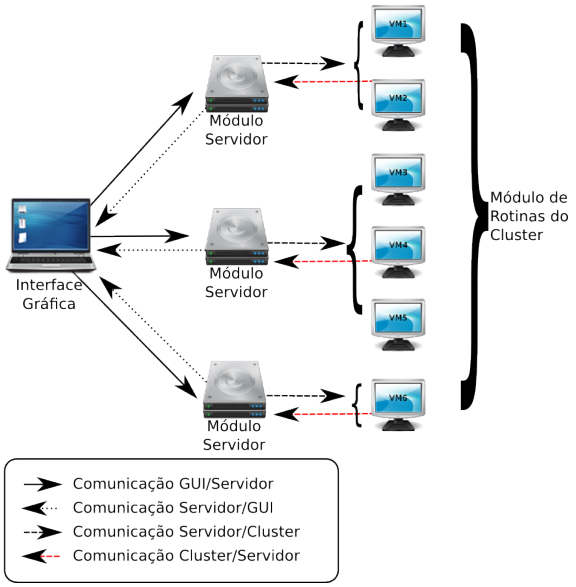



**Figura 1. Arquitetura do Sistema Proposto**

Para que o servidor seja capaz de executar os trabalhos nos *clusters*, é necessário o módulo de rotinas do *cluster*. Este é composto por funções e métodos que controlam o fluxo do trabalho sendo executado desde a configuração dos nós necessários ao processo de clusterização até o retorno dos resultados obtidos pelo processamento das tarefas. Através deste sistema o usuário será capaz de trabalhar com a maior quantidade possível de tipos de *clusters* diferentes, ao desenvolver um pacote de rotinas específicas para cada tipo suportado. O presente trabalho possui estas rotinas, inicialmente, implementadas para *clusters* de MPI, usando a implementação gratuita e de código aberto MPICH2.

### 3.2. Aplicação

O módulo servidor foi desenvolvido tendo em mente sua utilização em sistemas que atendam as especificações POSIX (*Portable Operating System Interface* - Interface Portável entre Sistemas Operacionais) em especial sistemas operacionais do tipo GNU/Linux. Isto é justificado pela dominância deste tipo de sistemas operacionais na área de computação de alto desempenho, sua segurança, código aberto, escalabilidade, desempenho, estabilidade e flexibilidade.

Apesar do módulo servidor estar focado em sistemas operacionais Linux, por ser a plataforma ideal para os *clusters*, nada impede que o controle seja feito em plataformas diferentes, nas quais o usuário esteja mais familiarizado ou tenha maior facilidade de acesso. Partindo deste princípio, a interface gráfica responsável pela interação entre o usuário e os *clusters*, pode ser instalada e utilizada em uma gama de sistemas diferentes, sendo a máquina virtual Java™ o único pré-requisito para sua utilização.

### 3.3. Procedimentos Para a Criação de *Clusters*

Para iniciar o processo de criação do *cluster* é necessário apenas escolher um nome para o novo *cluster*. Ao ser criado, este inicialmente não possui nenhum nó. O passo seguinte é a criação dos nós que são divididos em dois tipos básicos: *Master nodes*

(nós mestres) e *working nodes* (nós de trabalho). Um *cluster* neste sistema é composto por um nó mestre e $n$ $(n \geq 0)$ nós de trabalho.

Para a criação de um nó do *cluster*, seja ele o nó mestre ou os nós de trabalho, é necessário primeiramente que haja pelo menos uma máquina com o módulo servidor. As máquinas que possuem este módulo são denominadas de *hosts* ou hospedeiras. Na figura 2 está representado o fluxo necessário para a criação de um nó do *cluster*.

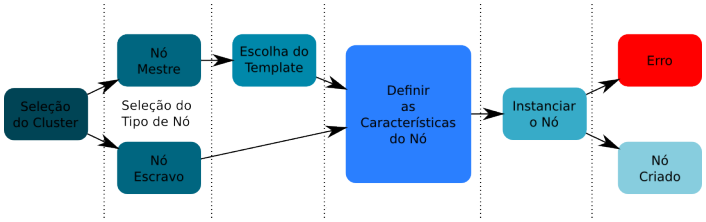



**Figura 2. Fluxograma da Criação de um Nó do *Cluster***

O nó mestre é o encarregado do gerenciamento interno do *cluster*, divisão e submissão do trabalho que será executado e pela consolidação dos resultados. No caso do sistema aqui proposto, o nó mestre é a máquina do *cluster* que possui instalado o módulo de rotinas do sistema. Para cada tipo de *cluster* existe um módulo diferenciado.

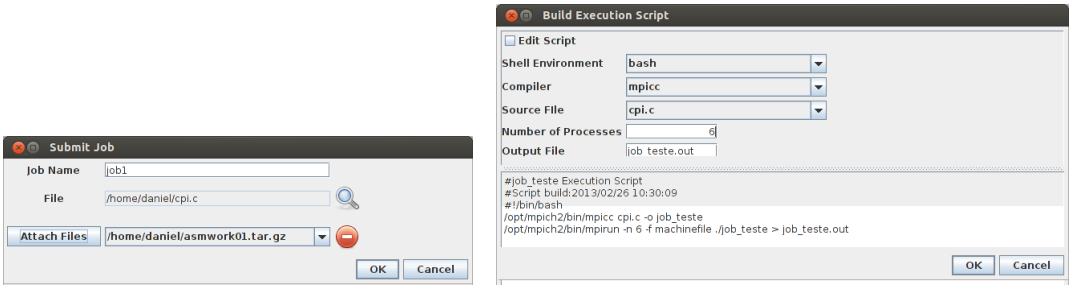
Uma vez selecionado *Master* como sendo o tipo de nó, o usuário terá que selecionar o *template* do *cluster*. O *template* é um modelo que será disponibilizado, ou poderá ser criado e personalizado. No escopo deste trabalho foi criado um *template* para *clusters* do tipo MPI utilizando o MPICH2. No sistema aqui proposto os nós de trabalho são máquinas virtuais *diskless*, que não possuem disco de armazenamento permanente.

### 3.4. Procedimentos Para a Submissão de Tarefas

As tarefas (*jobs*) são criadas para um determinado *cluster* através do botão para criação de *jobs* na barra de ferramentas da interface gráfica. Cada tarefa é composta de pelo menos um arquivo (fonte ou binário), podendo anexar quantos arquivos auxiliares forem necessários à correta execução do trabalho, e um *script* de execução. Após escolher a opção de criação de *jobs* na barra de ferramentas, aparecerá uma janela na qual é possível nomear e escolher os arquivos da tarefa. Ao terminar de escolher os arquivos necessários uma segunda janela permitirá a criação do *script* de execução de forma iterativa, sendo possível escolher através de menus como a tarefa será executada. Caso seja necessário o usuário pode editar manualmente o *script* nesta mesma janela. As figuras 3(a) e 3(b) mostram as janelas da interface gráfica responsáveis pela criação de um *Job*.

Através da janela de visualização de *jobs*, figura 4, é possível gerenciar a tarefa. Nela é possível enviar os arquivos de um determinado *job* para as máquinas do *cluster*, iniciar a execução da tarefa e recuperar os resultados do processamento.

Para executar uma determinada tarefa, os arquivos têm que estar nas máquinas do *cluster*. A transferência é realizada por um procedimento de cópia e redirecionamento desenvolvida especificamente para isto. É possível acompanhar o andamento da execução por meio do acesso gráfico fornecido. Ao término pode-se ainda realizar o *download* dos resultados através da barra de ferramentas. Na figura 5 é apresentado o processo para




(a) Janela inicial (b) Criação do *Script* de Execução de um *Job*

**Figura 3. Janelas do processo de criação de *Job***

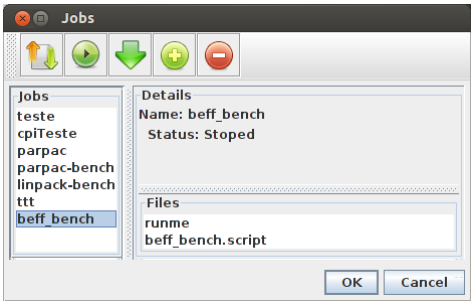



**Figura 4. Janela Para Visualização e Execução de um *Job***

a criação, submissão, execução e reexecução de um determinado *job* de um *cluster*. O módulo do servidor funciona como a interface de comunicação entre o nó mestre de um *cluster* e a interface gráfica acessada pelo usuário, por isso ele efetua troca de mensagens tanto entre o módulo de rotinas do *cluster* como com o módulo da interface gráfica, para isso foi desenvolvido um protocolo próprio.

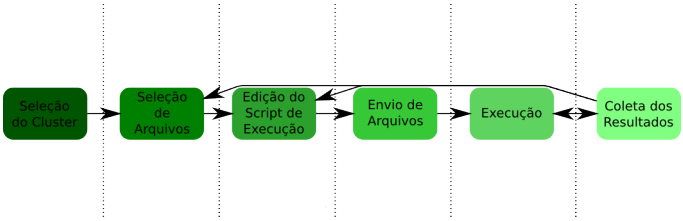



**Figura 5. Fluxograma que Descreve a Criação e Submissão de um *Job***

## 4. Componentes e Funcionalidades da GUI

O módulo da interface gráfica está desenvolvido em Java™ para viabilizar sua portabilidade e facilidade de execução. Este é o módulo responsável pela persistência dos dados do usuário, pela criação de *clusters* pelos usuários, pela submissão de tarefas, pelo acesso gráfico e pela interação do usuário com o sistema. Na figura 6 é exibida a janela principal da interface gráfica, através da qual o usuário realiza todas as operações necessárias ao controle do *cluster* e a submissão e o controle das tarefas.

Este módulo pode ser dividido de acordo com suas funcionalidades em cinco partes principais: controle do *cluster*, submissão de trabalhos, comunicação entre a interface e o servidor, acesso gráfico ao ambiente dos *clusters* e persistência de dados.

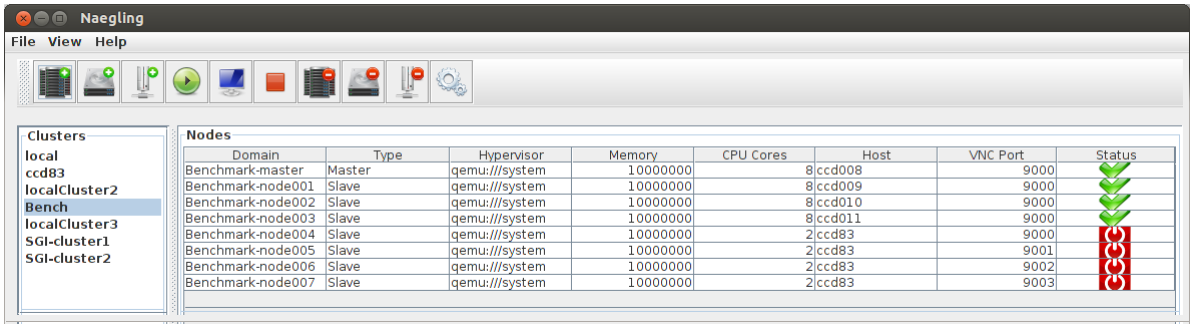



**Figura 6. Janela Principal da Interface Gráfica**

1. Controle do *Cluster* - É responsável por permitir ao usuário criar o *cluster* propriamente dito bem como seus nós, ligá-los e verificar o estado de cada *cluster*. Para facilitar estas tarefas foram criados menus, barras de ferramentas e janelas, tornando o processo mais intuitivo.
2. Acesso ao Ambiente - Este sistema foi desenvolvido de forma que o usuário não tenha que acessar o ambiente de *cluster* da forma tradicional, que é geralmente feito por acesso remoto via SSH (*Secure Shell*). Todo o processo, de criação do *cluster*, submissão de tarefas e verificação do resultado, foi desenvolvido de forma que possa ser executado diretamente através da interface gráfica.

Caso o usuário do sistema tenha necessidade de acessar o ambiente de alto desempenho, seja por motivo de um erro indeterminado que esteja ocorrendo ou ele queira acompanhar o processamento do trabalho, a interface gráfica possui esta capacidade através do cliente VNC. A figura 8 é uma captura de tela realizada durante um acesso gráfico ao ambiente de um *cluster*.

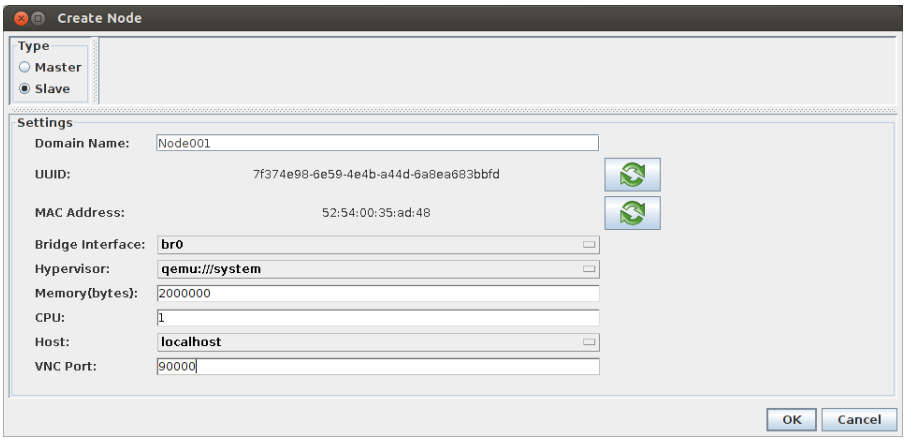



**Figura 7. Janela de Criação de Nó**

## 5. Conclusões e Trabalhos Futuros

O presente trabalho foi desenvolvido para suprir uma lacuna existente no atual cenário de HPC, facilitando tanto a criação de *clusters* sob demanda, permitindo que o mesmo seja efetuado a partir de uma interface gráfica, sem complicações para o usuário final, como também, auxiliar o usuário durante todo o processo de submissões de tarefas, acompanhamento do trabalho e recuperação de resultados.

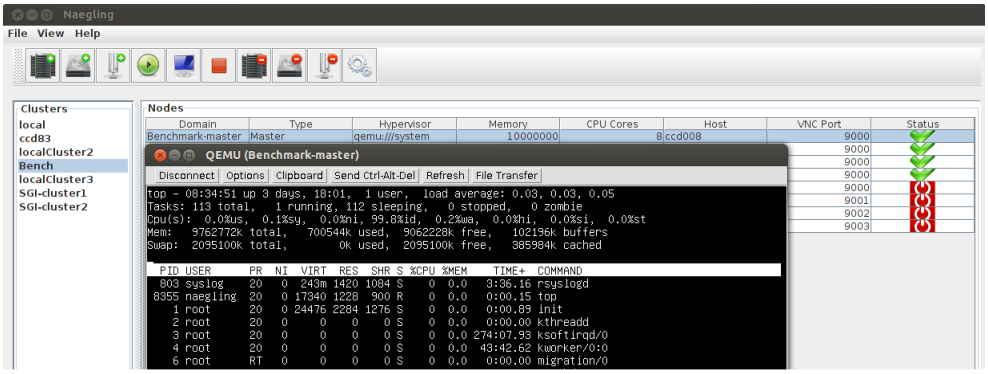



**Figura 8. Acesso Gráfico ao Ambiente**

Com o sistema apresentado, atualmente é possível criar *clusters* de centenas de nós em questão de minutos, enquanto normalmente esta tarefa demoraria dias, semanas ou até meses. Além disto, todo o processo de submissão de tarefas também foi simplificado, podendo ser realizado através de uma interface gráfica instalada no sistema operacional de preferência do usuário do *cluster*. O sistema possibilita que todo o acesso seja realizado através da interface criada, não mais necessitando acesso remoto para realizar a tarefa.

Apesar destas melhorias fornecidas, ainda há melhorias e capacidades a serem adicionadas. A seguir será listado alguns itens importantes que serão estudados e implementados:

- Ampliar os tipos de *Templates* suportados pelo sistema.
- Ampliar as capacidades de criações dos nós (criação de máquinas virtuais com discos, criação dos nós em maquinas reais).
- Ampliar os tipos de *Hypervisors* suportados.
- Aperfeiçoar a interface de submissão de tarefas para novos modelos de aplicações.
- Implementar um modelo de segurança em cada nível do ambiente.
- Portar a interface gráfica, de uma aplicação tipo *desktops*, para um ambiente *web*.

## Agradecimentos

Os autores agradecem o apoio recebido do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, por meio do projeto - Programa de Capacitação Institucional - LNCC/MCT.

## Referências

[Pacheco 2011] Pacheco, P. (2011). *An Introduction to Parallel Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

[PELICANHPC 2012] PELICANHPC (2012). Pelicanhpc: A gnu/linux distribution to create a hpc cluster for mpi based parallel computing. [Online; acessado em 05-fevereiro-2014 URL:http://pareto.uab.es/mcreel/PelicanHPC/].

[ROCKSCLUSTER 2012] ROCKSCLUSTER (2012). www.rocksclusters.org — rocks website. [Online; acessado em 05-fevereiro-2014 URL:http://www.rocksclusters.org/wordpress/].

[SCYLD 2012] SCYLD (2012). Scyld clusterware hpc. [Online; acessado em 05-fevereiro-2014 URL:http://gilgamesh.cheme.cmu.edu/scyld-doc/users-guide/users.html].

# N-Clusters: Ferramenta para a gerência de ambientes de Computação Massivamente Paralela e Distribuída

**Jonathan Barbosa**[1,3], **Victor Oliveira**[1], **Matheus Bandini**[2,3], **Bruno Schulze**[1], **Antonio Mury**[1]

[1]Laboratório Nacional de Computação Científica (LNCC)
Av. Getúlio Vargas, 333 – Quitandinha – 25651-075 – Petrópolis – RJ – Brasil

[2]Instituto de Computação - Universidade Federal Fluminense (IC/UFF)
Rua Passo da Pátria 156 – São Domingos – 24210-240 – Niterói – RJ – Brasil

[3]Faculdades de Educação Tecnológica do Estado do RJ (FAETERJ/Petrópolis)
Av. Getúlio Vargas, 335 – Quitandinha – 25651-075 – Petrópolis – RJ – Brasil

***Abstract.*** *This paper presents a tool that has the goal of providing on demand infrastructure to comply with user specific configuration needs. N-Clusters provides task submission environments for Massive Parallel and Distributed Computing, to help users through task submission process, to improve resource management and to allow the usage of underutilized infrastructure for knowledge dissemination.*

***Resumo.*** *Este artigo apresenta uma ferramenta que tem por finalidade, disponibilizar infraestruturas sob demanda que atendam a necessidade de configurações específicas de usuários. O N-Clusters provê ambientes para a execução de tarefas de Computação Massivamente Paralela e Distribuída (CMPD), para auxiliar o usuário através do processo de submissão de tarefas, aprimorar a gerência dos recursos e permitir o uso de infraestruturas subutilizadas e legadas, que podem ser utilizadas para a disseminação do conhecimento.*

## 1. Introdução

A complexidade dos problemas atuais cresceu significativamente por haver um aumento na quantidade de informações disponíveis e no número de variáveis a elas relacionadas. Com isso, a dificuldade em obter resultados e o tempo de experimentos passam a ser fatores impactantes em diversas áreas na pesquisa científica e no desenvolvimento tecnológico.

O uso de *clusters* associado à utilização de tecnologias com arquiteturas diferenciadas, como *multi/many cores*, visam resolver problemas complexos em tempos menores. Com isso, passam a ser necessárias aplicações voltadas para essas arquiteturas, para que possam usufruir ao máximo destes recursos.

Este avanço, porém, acarreta também na complexidade da gestão dessa infraestrutura e na capacidade de uso por seus usuários finais, em consequência de maiores quantidades de recursos sob a gerência dos administradores e a quantidade de etapas para a submissão e execução das tarefas, além de conhecimento técnico sobre o assunto. Com base nesses elementos, verifica-se que, para melhor utilizar esses recursos, surge a necessidade de: elaborar um sistema de gestão simples e unificado, contendo informações

sintéticas e com uma inteligência embutida tratando as informações coletadas; criar um ambiente para assessoramento dos usuários; e melhor aproveitar as infraestruturas disponíveis/legadas e otimizar o seu uso.

O foco da ferramenta desenvolvida, chamada N-Clusters, é apresentar e disponibilizar uma infraestrutura sob demanda (*Infraestructure as a Service - IaaS*), atendendo a características e configurações específicas de usuários, além de permitir a criação de ambientes virtuais para processamento massivamente paralelo e distribuído, por meio de uma interface via navegador. Desta forma, pretende-se: prover ambientes para a submissão de tarefas de Computação Massivamente Paralela e Distribuída (CMPD); facilitar o processo de submissão de tarefas, guiando o usuário através de uma interface simples e intuitiva; obter flexibilidade e melhor gerência dos recursos; e permitir que a infraestrutura subutilizada e legada possa ser utilizada para a disseminação do conhecimento.

Para isto, técnicas como virtualização são usadas para atender tais necessidades individuais de pesquisadores, professores e alunos da área acadêmica, bem como funcionários de pequenas e médias empresas do setor industrial.

A seção 2 deste artigo apresenta a revisão da literatura e trabalhos que, de alguma forma, estão relacionados com a ferramenta N-Clusters. A seção 3 apresenta a descrição do N-Clusters, composta pela arquitetura da ferramenta e por suas funcionalidades. Por fim, a seção 4 apresenta as considerações finais, contendo as conclusões obtidas e os possíveis trabalhos futuros identificados.

## 2. Trabalhos Relacionados

A capacidade de proporcionar ambientes mais amigáveis, com grande poder computacional e com o uso colaborativo de recursos torna cada vez mais necessário o desenvolvimento de aplicações para apoio às atividades de pesquisa, sobretudo no que tange à infraestrutura computacional. Dentro deste contexto, o uso de ambientes virtualizados e de nuvens computacionais podem agregar uma capacidade computacional valiosa em apoio à aplicações que necessitam de computação massivamente paralela e distribuída.

Dentro desse cenário, a tecnologia de virtualização de recursos contribui para a disseminação do paradigma de Nuvem. Isto se dá através de suas propriedades, tais como: isolamento entre as plataformas virtualizadas, permitindo que coexistam sistemas distintos hospedados em um único recurso real; a virtualização dos dispositivos de entrada e saída; a possibilidade de que essas plataformas virtualizadas possam ser migradas entre plataformas reais, permitindo a consolidação do seu uso; e o melhor gerenciamento do consumo de energia. Dessa forma, segundo [Matthews et al. 2007], [Ongaro et al. 2008] e [Yang et al. 2009], traz como benefícios:

1. A possibilidade de otimizar a utilização da infraestrutura computacional por meio do seu compartilhamento;
2. O isolamento entre ambientes criados e em execução em um mesmo recurso;
3. A possibilidade do balanceamento de carga;
4. A consolidação dos serviços; e
5. A capacidade de provisão de um ambiente dedicado para os seus usuários, no nível de plataforma e infraestrutura virtualizada.

O ambiente de aplicações científicas, além de exigir requisitos de configuração fora do escopo de conhecimento da maioria dos pesquisadores, também necessita de

conhecimentos específicos sobre como essa camada de virtualização interage com as aplicações. Surge assim a ideia da criação de ambientes virtualizados pré-configurados e dedicados à solução de problemas científicos, atendendo a estes requisitos, mitigando essas restrições e que possam se adequar as características das aplicações [Yokoyama et al. 2012b], [Yokoyama et al. 2012a], [Yokoyama et al. 2011].

O trabalho apresentado por [Huang et al. 2006] tratou de determinar quais elementos presentes na camada de virtualização impactavam na perda de desempenho para ambientes de computação massivamente paralela e distribuída e propôs soluções, muitas das quais hoje já implementadas nas novas versões de *hypervisors*.

Apesar de não ser nova a ideia da criação de ambientes dedicados ao uso de aplicações científicas em plataformas virtuais, quando associada aos novos recursos dos equipamentos agora disponíveis, ela abre perspectivas de novas pesquisas e desenvolvimento de soluções. O trabalho apresentado por [Shoop et al. 2012] procura verificar a possibilidade do uso de máquinas virtuais em laboratórios de escolas e universidades para a criação de *clusters* virtuais, voltados para o ensino de processos, procedimentos e desenvolvimento de aplicações paralelas (as máquinas virtuais possuiam Ubuntu Linux, OpenMPI, TORQUE, Hadoop e WebMapReduce) nestas instituições. O trabalho de [Hacker 2011], também apresenta uma análise sobre o uso de ambientes virtualizados, na forma de *clusters* e como estes ambientes podem ser utilizados para o ensino, desde aspectos relacionados à camada de virtualização, até a criação e uso de um *cluster*.

[Marshall et al. 2010] apresentam uma proposta de um sistema capaz de disponibilizar ambientes para o ensino da configuração de um *cluster*, sua rede e seu gerenciamento. O processo era complementado por um tutorial e um conjunto de exercícios voltados para a fixação do conhecimento. Este processo foi implementado em uma nuvem computacional, com custo orçado em U$120,00 para 8 horas de uso.

No caso do uso de nuvens computacionais, o trabalho apresentado por [Juve and Deelman 2011] teve como objetivo disponibilizar um sistema de provisionamento de *clusters* em nuvens computacionais, para a execução de *workflows* científicos. Neste trabalho os autores avaliam os efeitos que este sistema proposto causa no tempo de provisionamento dos recursos nas nuvens públicas utilizadas. O trabalho traz, além da avaliação da proposta, uma informação importante para o caso do provisionamento em nuvem: as falhas que ocorreram em função da capacidade de administração, as dependências dos nós em função da complexidade do *cluster* configurado e da mudança das características dos nós virtuais em função as mudanças dos requisitos de uma aplicação.

Nos dois trabalhos anteriores os autores utilizaram de recursos de uma nuvem computacional para a implementação de um *cluster* virtual. Para o caso das nuvens computacionais, elas trazem à tona a possibilidade de criação e o uso de ambientes dedicados voltados para atender as necessidades imediatas dos ambientes de alto desempenho. As nuvens computacionais agregam a possibilidade de instanciação imediata de recursos virtuais, além de possibilitar que seus usuários possam executar remotamente aplicações, independente da sua plataforma, podendo utilizá-las a qualquer momento, assim como armazená-las para uso futuro, otimizando o uso dos recursos, possibilitando a estes usuários, acesso a um ambiente de alto desempenho, voltado para solução de problemas e muitas vezes proibitivos para os mesmos.

## 3. Descrição da Ferramenta N-Clusters

O foco da ferramenta N-Clusters é disponibilizar infraestrutura sob demanda (*Infraestructure as a Service* - IaaS), atendendo a demandas específicas de usuários.

Para isto, técnicas como virtualização são usadas para atender tais necessidades individuais de pesquisadores, professores e alunos da área acadêmica, bem como funcionários de pequenas e médias empresas do setor industrial. Para garantir que todos tenham acesso ao N-Clusters, ele é regido pela *GNU General Public License* (GNU GPL), uma licença para *softwares* de código aberto, que permite o acesso destes usuários às tecnologias citadas neste trabalho.

### 3.1. Arquitetura do N-Clusters

A figura 1 mostra como o ambiente para a utilização do N-Clusters foi elaborado, podendo esta também variar de acordo com as necessidades ou objetivos dos usuários. Para utilizar a ferramenta proposta, é preciso um conjunto de máquinas para que os *containers* possam ser criados e uma máquina para hospedar a interface *web* contida no *Live CD*. Desta forma, um conjunto de 4 máquinas permite que pesquisadores, professores e alunos possam utilizar *clusters* e exercitar suas habilidades de programação paralela.

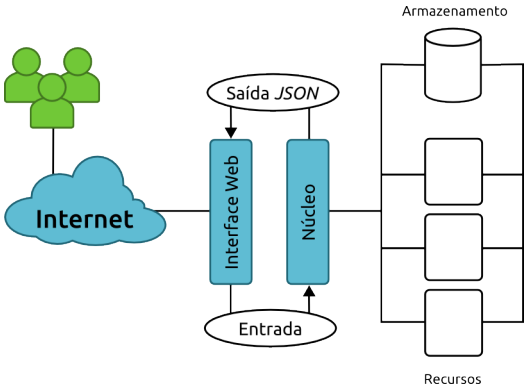



**Figura 1. Iteração da interface web com o núcleo do N-Clusters**

Ainda sobre a arquitetura do N-Clusters, a Interface *Web* é responsável por prover um ambiente gráfico intuitivo por meio de um *browser* (atendendo também a dispositivos móveis), a fim de guiar os usuários na criação de seus ambientes virtuais, na submissão de suas tarefas e na recuperação dos resultados. No caso dos administradores, prover auxílio no suporte à gerência da infraestrutura sob sua responsabilidade. Por outro lado, o núcleo do N-Clusters recebe as chamadas da interface gráfica e as executa. Ele cria, configura e inicia os *clusters* virtuais, aloca os arquivos nos dispositivos de armazenamento dos usuários, configura a infraestrutura e provê um conjuntos de comandos que os administradores podem utilizar para realizar essas tarefas através de linhas de comando.

### 3.2. Funcionalidades do N-Clusters

A ferramenta N-Clusters oferece um conjunto extenso de funcionalidades que envolvem questões como a gerência da infraestrutura, submissão de tarefas, gestão dos dados dos usuários e escalonamento, entre outras. Com o objetivo de melhor apresentar estas funcionalidades, as mesmas serão divididas de acordo com o módulo da arquitetura responsável por mantê-las.

### 3.2.1. Funcionalidades da Interface *Web* do N-Clusters

A interface com o usuário permite a gerência da infraestrutura através do menu *Hosts*. Este menu é dividido em três guias: listar (*List*), adicionar (*Add*) e Tecnologia da Informação Verde (*Green IT*). Com elas é possível listar e adicionar recursos disponíveis para os usuários e configurar cada recurso individualmente para que seja desligado automaticamente se não estiver em uso.

Através da guia *List* é possível que os recursos sejam ligados, desligados, colocados em modo de espera ou removidos da lista de recursos de forma simples e intuitiva. O administrador encontra nessa lista informações que fornecem uma visão de características e estados dos recursos cadastrados no N-Clusters. Dentre as informações exibidas, estão o *hostname* de um recurso, seu *status* para identificar se está ligado, desligado ou em modo de espera, o nome do usuário, além de informações sobre os recursos. A figura 2 exibe a lista dos recursos e das funções apresentadas ao administrador do N-Clusters.

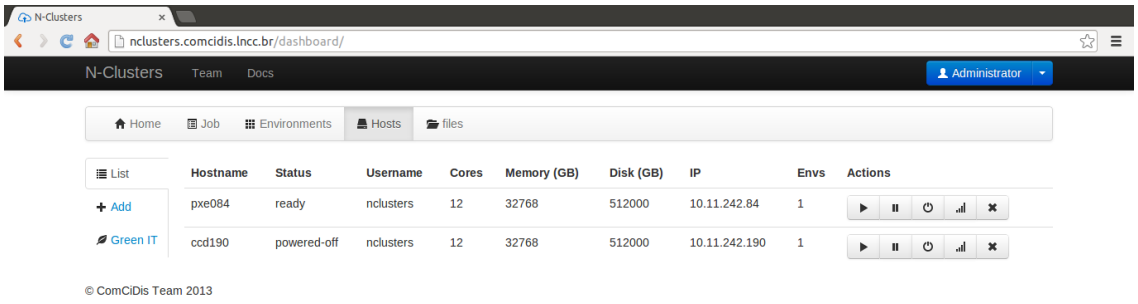



**Figura 2. Lista de recursos (navegador de um *Desktop*)**

O processo de criação de um *cluster* virtual se dá por meio de um formulário, que, após preenchido, aguarda a confirmação do administrador a fim de avaliar a real necessidade do usuário e garantir a utilização adequada da infraestrutura. Quando a autorização ocorre, o ambiente está apto a receber submissões. Este formulário contém os seguintes itens: i) nome do ambiente para facilitar a associação por parte do usuário; ii) quantidade de nós que o *cluster* virtual terá, sem contar com o nó mestre; e iii) descrição do ambiente, que é utilizado pelo usuário para informar ao administrador a finalidade do *cluster* virtual a ser criado (figura 3).

Quando o usuário submete uma tarefa, e somente neste momento, o N-Clusters cria um *cluster*. Caso essa tarefa não seja solicitada, existirá apenas um registro com as características do ambiente que o usuário requisitou. Desta forma, não há desperdício de infraestrutura enquanto uma tarefa não é solicitada, promovendo o chamado TI Verde, que sugere a redução dos gastos de energia ao poupar recursos que não são utilizados.

Para a submissão de tarefas, é necessário que o usuário possua um ambiente virtual e que este esteja apto a receber tarefas. Na especificação da submissão, além dos arquivos de código do usuário serem enviados em um formato compactado (.zip ou .tar.gz) é necessário que seja enviado e selecionado o *script* de execução para que, a partir deste, o N-Clusters execute a tarefa.

Uma tarefa não está diretamente associada a um ambiente, o que é feito no momento da execução. Quando uma tarefa é enviada, ela passa a ter um registro na base

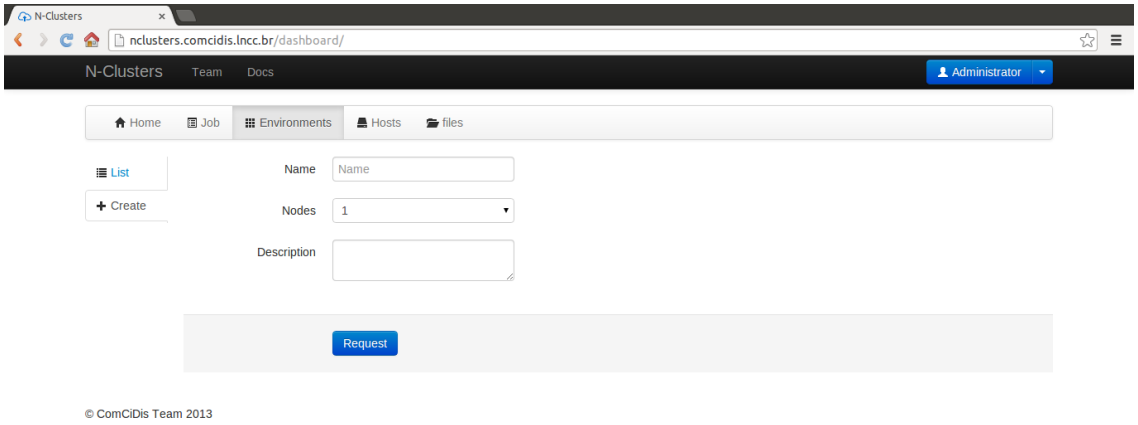



**Figura 3. Especificação do ambiente virtual**

de dados do N-Clusters. Os arquivos são armazenados dentro da pasta do usuário com o nome da tarefa e são mantidos no servidor até o momento de sua execução. A submissão de tarefas através da interface do N-Clusters é dividida em dois passos:

- Passo 1: nome da tarefa; linguagem utilizada; ambiente em que será processada; e descrição.
- Passo 2: envio dos arquivos e *script* execução; e execução da tarefa.

### 3.2.2. Funcionalidades do Núcleo do N-Clusters

O núcleo do N-Clusters é responsável por concentrar e realizar todas as suas funcionalidades. Nele é possível efetuar as operações contidas na interface com o usuário, além de outras acessíveis apenas diretamente em seu núcleo. Para isto, é disponibilizado para os administradores o acesso às funções por meio de linhas de comando.

Para configurar o novo recurso adicionado, o N-Clusters realiza *Secure Shell* (SSH) para o novo recurso com o nome de usuário e requisita ao administrador a senha para efetuar o login. Após estabelecer a comunicação, o N-Clusters instala o LXC, pacote necessário para criação de containers, além de criar o diretório `cgroup` na raiz do sistema e de adicionar entradas no arquivo de `sudoers`, para que tenha privilégios na execução de comandos. Ao final dessa configuração, o N-Clusters realiza um *secure copy* (scp) para enviar ao novo *host* o arquivo com o *template* da imagem utilizada para a criação dos containers que formarão os *cluters*. Com isso o novo recurso está apto a hospedar os *clusters* virtuais e as chamadas do núcleo do N-Clusters.

Embora o foco deste trabalho não seja desenvolver um complexo sistema para escalonar as tarefas e onde melhor alocar os *containers* na infraestrutura, este é um aspecto importante para a obtenção de um bom desempenho no processamento realizado pelos *clusters* virtuais, já que se estiverem mal dispostos eles poderão aumentar o tempo de processamento das tarefas. Foi, então, desenvolvido um escalonador simples, baseado no algoritmo *Round-Robin* que tem a função manter equilibrada a criação de *clusters* virtuais na infraestrutura que o usuário dispõe.

### 3.3. *Cluster in a Box*

O *Live CD* da ferramenta N-Clusters traz uma funcionalidade chamada *cluster in a box*. Com ela, o administrador pode poupar tempo de configuração e manter inalteradas as configurações de recursos, permitindo que o Sistema Operacional (SO) do *host* não sofra mudanças. Essa funcionalidade permite adicionar *hosts* sem a necessidade de formatar o Sistema Operacional atual para instalar um sistema Linux e configurar e instalar os pacotes necessários.

### 3.4. Descrição da Demonstração da Ferramenta

A demonstração da ferramenta será conduzida de forma a utilizar a ferramenta N-Clusters para criar um ambiente de *cluster* virtual e remoto e para submeter tarefas no ambiente de alto desempenho criado. Para tal, será utilizada uma máquina cliente que acessará, através de um navegador *web*, os recursos virtuais remotos que estarão localizados no LNCC (Laboratório Nacional de Computação Científica).

Por razões de limitação de espaço, algumas figuras e explicações mais detalhadas sobre as funcionalidades do N-Clusters tiveram que ser omitidas deste artigo. Estas informações, no entanto, se encontram de forma completa na página *web* da documentação da ferramenta, no endereço **http://comcidis.lncc.br/nclusters** junto com o manual da ferramenta. O N-Cluster será disponibilizado para avaliação através da sua interface *web*, com o objetivo de minimizar esforços necessários relativos à instalação e configuração do ambiente local.

## 4. Conclusões e Trabalhos Futuros

A ferramenta apresentada neste artigo permite disponibilizar ambientes virtuais sob demanda, de acordo com as especificações de usuários, a fim de atender suas necessidades. Isto possibilita que, a partir desse processo, seja possível a utilização desses ambientes para submissão de tarefas, testes e simulações por parte dos usuários. A ferramenta também permite que a infraestrutura subutilizada/legada seja aproveitada como recurso para a disseminação do conhecimento e, através da administração simplificada, os usuários possam obter flexibilidade e melhor gerência dos recursos.

Como trabalhos futuros, pretende-se aperfeiçoar a capacidade de gerência da infraestrutura, de modo a se obter um maior detalhamento do monitoramento dos recursos. Planeja-se atingir essa melhora coletando informações de temperatura e rede, criando uma opção para instalar a imagem utilizada pelo servidor PXE nos recursos adicionados e agregando novos algoritmos de escalonamento para melhor alocar e distribuir os ambientes virtuais nos recursos disponíveis, permitindo: aumentar, por meio do desenvolvimento de novos algoritmos de controle, a capacidade de assessoramento no uso da infraestrutura; melhorar o algoritmo que promove a TI Verde, adicionando mais variáveis coletadas no recurso à sua avaliação, visando garantir uma melhor economia de energia e a real utilização da infraestrutura; e Agregar não somente *clusters* virtuais, mas também diversas outras formas de arquiteturas, configurações e suportes a aplicações.

A incorporação dessas novas funcionalidades à ferramenta aqui apresentada tornaria possível a utilização de GP/GPU e coprocessadores, atendendo a mais grupos de usuários, além de proporcionar as vantagens associadas à utilização dessas arquiteturas.

## Agradecimentos

Este trabalho teve o apoio do Ministério de Ciência, Tecnologia e Inovação (MCTI), do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq - PCI/LNCC) e da Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ).

## Referências

Hacker, T. J. (2011). Exploring the use of virtual machines and virtual clusters for high performance computing education. In *American Society for Engineering Education*, Washington DC, USA. ASEE.

Huang, W., Liu, J., Abali, B., and Panda, D. K. (2006). A case for high performance computing with virtual machines. In *Proceedings of the 20th annual international conference on Supercomputing*, ICS '06, pages 125–134, New York, NY, USA. ACM.

Juve, G. and Deelman, E. (2011). Wrangler: virtual cluster provisioning for the cloud. In Maccabe, A. B. and Thain, D., editors, *HPDC*, pages 277–278. ACM.

Marshall, P., Oberg, M., Rini, N., Voran, T., and Woitaszek, M. (2010). Virtual clusters for hands-on Linux cluster construction education. In *11th LCI International Conference on High-Performance Clustered Computing*, Pittsburgh, Pennsylvania, USA.

Matthews, J. N., Hu, W., Hapuarachchi, M., Deshane, T., Dimatos, D., Hamilton, G., and McCabe, M. (2007). Quantifying the performance isolation properties of virtualization systems. In *Experimental computer science on Experimental computer science*, ecs'07, pages 5–5, Berkeley, CA, USA. USENIX Association.

Ongaro, D., Cox, A. L., and Rixner, S. (2008). Scheduling i/o in virtual machine monitors. In *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '08, pages 1–10, New York, NY, USA. ACM.

Shoop, E., Brown, R., Biggers, E., Kane, M., Lin, D., and Warner, M. (2012). Virtual clusters for parallel and distributed education. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 517–522, New York, NY, USA. ACM.

Yang, C.-T., Tseng, C.-H., Chou, K.-Y., and Tsaur, S.-C. (2009). A virtualized hpc cluster computing environment on xen with web-based user interface. In Zhang, W., Chen, Z., Douglas, C. C., and Tong, W., editors, *HPCA (China)*, volume 5938 of *Lecture Notes in Computer Science*, pages 503–508. Springer.

Yokoyama, D., Dias, V., Kloh, H., Bandini, M., Porto, F., Schulze, B., and Mury, A. (2012a). The impact of hypervisor layer on database applications. In *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*, pages 247–254.

Yokoyama, D., Kloh, H., Schulze, B., and Mury, A. (2011). A hybrid virtualized environment for clouds. In *Proceedings of the Workshop on Posters and Demos Track*, PDT '11, pages 14:1–14:2, New York, NY, USA. ACM.

Yokoyama, D., Kloh, H., Schulze, B., and Mury, A. (2012b). Virtualization strategies evaluation based on applications sets for scientific clouds. In *SBRC Workshop of Clouds and Grids Applications 2012*. SBC.

# BFT-SMaRt: Uma Ferramenta Robusta para Replicação Máquina de Estados

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

[1]LaSIGE, Universidade de Lisboa, Lisboa - Portugal
[2]CIC, Universidade de Brasília, Brasília - Brasil

***Abstract.*** *The last fifteen years have seen an impressive amount of work on protocols for Byzantine fault-tolerant (BFT) state machine replication (SMR). However, there is still a lack of practical and reliable software libraries implementing this technique.* BFT-SMaRt *is an open-source Java-based library implementing robust BFT state machine replication. When compared to other SMR libraries,* BFT-SMaRt *achieves better performance.*

***Resumo.*** *Nos últimos anos surgiram muitos protocolos para replicação Máquina de Estados (SMR) tolerante a falhas bizantinas (BFT). No entanto, ainda faltava uma implementação confiável desta técnica. Visando suprir esta deficiência, o* BFT-SMaRt *é uma biblioteca Java de código aberto que implementa uma ferramenta robusta para BFT SMR. Além disso, o* BFT-SMaRt *apresenta um desempenho melhor do que outras ferramentas similares.*

**URL do sistema (*download*, manuais, documentação e aplicação):**
`http://code.google.com/p/bft-smart/`

## 1. Introdução

Nos últimos anos surgiram muitos protocolos para replicação Máquina de Estados (SMR) tolerante a falhas bizantinas (BFT) (ex., [Castro and Liskov 2002, Guerraoui et al. 2010, Veronese et al. 2013]), no entanto poucas implementações completas desta técnica têm sido desenvolvidas. O PBFT [Castro and Liskov 2002] e o UpRight [Clement et al. 2009] são gratas exceções, que infelizmente não são mais mantidas. Desta forma, as organizações que necessitam deste serviço precisam desenvolver sua própria implementação (ex., [Chandra et al. 2007]).

Neste artigo descrevemos o BFT-SMaRt, uma implementação Java robusta para BFT SMR que utiliza um protocolo similar ao PBFT. O BFT-SMaRt foi desenvolvido buscando-se um alto desempenho em execuções livres de falhas e corretude nos casos em que replicas faltosas apresentem um comportamento arbitrário. Além disso, o BFT-SMaRt é a primeira implementação de BFT SMR que permite reconfigurações no conjunto de replicas [Lamport et al. 2010] e fornece um suporte eficiente e transparente para serviços duráveis [Bessani et al. 2013a].

A principal contribuição deste trabalho é a documentação da implementação e da forma de utilização de um sistema para BFT SMR, preenchendo esta lacuna encontrada na literatura. Adicionalmente, apresentamos uma avaliação experimental do BFT-SMaRt, a qual compara este sistema com outras propostas similares e discute alguns *tradeoffs* relacionados com a tolerância a falhas por parada (*crash*) vs. falhas bizantinas.

## 2. Projeto do BFT-SMaRt

O desenvolvimento do BFT-SMaRt iniciou-se no ano de 2007 mas somente em 2011 tivemos uma implementação completa e robusta. Além de beneficiar-se de arquiteturas *multi-cores*, não utilizar otimizações complexas e possuir uma API simples (Seção 5), o projeto do BFT-SMaRt segue os seguintes princípios [Bessani et al. 2013b]:

**Modelo de falhas configurável.** Por padrão, o BFT-SMaRt tolera (1) atraso, perda e corrupção de mensagens e (2) comportamento arbitrário de processos. No entanto, este sistema ainda fornece assinaturas criptográficas (para tolerar falhas bizantinas) e um protocolo de SMR simplificado (para tolerar apenas *crashes*). A menos que seja explicitado no texto, focamos nossa discussão na configuração para falhas bizantinas.

**Modularidade.** O BFT-SMaRt implementa um protocolo de SMR modular, que utiliza uma primitiva de consenso bem definida [Sousa and Bessani 2012]. Além disso, existem módulos para comunicação ponto a ponto confiável, ordenação de requisições, transferência de estado e reconfiguração. Esta abordagem facilita a manutenção e o aprimoramento do sistema. Outros sistemas, como o PBFT, implementam um protocolo monolítico onde não existe uma separação entre os algoritmos.

### 2.1. Modelo de Sistema

O BFT-SMaRt assume o modelo de sistema usual para BFT SMR [Castro and Liskov 2002, Guerraoui et al. 2010]: $n = 3f + 1$ replicas para tolerar até $f$ falhas Bizantinas; um número ilimitado de clientes sujeitos a falhas Bizantinas e sincronia parcial para garantir terminação. Como o sistema permite reconfigurações, $n$ e $f$ podem ser modificados durante a execução (Seção 2.2). Além disso, o sistema também pode ser configurado para usar somente $n = 2f + 1$ replicas e tolerar até $f$ faltas por *crash* (Seção 4).

Independente da configuração, o sistema necessita de canais ponto a ponto confiáveis, os quais são implementados usando MACs sobre TCP/IP. As chaves simétricas necessárias nestes canais para comunicação entre as replicas são geradas usando o protocolo *Signed Diffie-Helman*, usando um par de chaves RSA para cada replica. Já as chaves necessárias para comunicação entre clientes e replicas são geradas baseadas nos seus identificadores, não necessitando um par de chaves por cliente. Também é possível configurar o sistema para garantir a autenticidade das requisições dos clientes, neste caso assinaturas são habilitadas e cada cliente precisará de um par de chaves (Seção 4).

### 2.2. Módulos Principais

Esta seção discute brevemente os principais protocolos implementados no BFT-SMaRt.

***Multicast* com Ordem Total.** *Multicast* com ordem total é implementado através da utilização de um protocolo de consenso subjacente [Sousa and Bessani 2012]. Em uma execução normal, os clientes enviam suas requisições para todas as replicas e aguardam as respostas. A ordem total é conseguida através de uma sequência de instâncias de consenso, cada uma decidindo a ordem de entrega de um lote de requisições. Cada instância necessita de três passos de comunicação para terminar. Primeiro a replica líder do consenso envia uma proposta (*PROPOSE*) para as outras replicas, seguindo-se mais dois passos de comunicação (*WRITE* e *ACCEPT*) onde cada replica envia uma mensagem para todas as outras replicas. As mensagens *PROPOSE* contêm um lote de requisições enquanto que *WRITE* e *ACCEPT* contêm apenas um *hash* criptográfico destes lotes.

A descrição acima refere-se ao comportamento deste módulo em uma execução normal, que ocorre quando não ocorrem falhas e na presença de sincronia. Quando estas condições não são satisfeitas, este módulo chaveia para uma *fase de sincronização* onde um novo líder é escolhido e todas as replicas "saltam" para a mesma instância do consenso. Este salto pode fazer com que alguma replica inicie o protocolo de transferência de estado, abaixo descrito.

**Transferência de Estado.** Uma implementação de SMR completa (que possa ser usada na prática) deve possibilitar a recuperação e reintegração de replicas, sem ser necessário reiniciar o sistema todo. Além disso, memória estável deve ser utilizada para recuperar o sistema como um todo, caso possíveis falhas correlatas comprometam mais do que $f$ replicas. O BFT-SMART implementa técnicas eficientes de durabilidade, como descrito em [Bessani et al. 2013a], para recuperar replicas ou todo o sistema. As idéias principais destas técnicas são: (1) armazenar os lotes de requisições em um disco enquanto estas requisições são executadas; (2) obter *snapshots* em diferentes pontos da execução em diferentes replicas para não precisar parar o sistema; e (3) transferir o estado de forma colaborativa, com cada replica enviando diferentes partes do estado para a replica sendo recuperada. Todas estas técnicas são implementadas em uma camada bem definida entre o protocolo de replicação e a aplicação, sem influenciar no protocolo de consenso.

**Reconfiguração.** Diferente de outros sistemas, o BFT-SMART fornece um protocolo que possibilita a adição e remoção de replicas durante a execução do sistema. Para isso, utiliza um tipo especial de cliente, chamado *View Manager*, que deve ser confiável e usado por administradores do sistema. O protocolo de reconfiguração funciona da seguinte forma: o *View Manager* envia uma requisição especial de reconfiguração para o sistema, informando a replica que deve ser adicionada ou removida do sistema. Como estas requisições são ordenadas (como qualquer outra requisição), todas as replicas corretas adotam a mesma visão (configuração) atual do sistema em qualquer ponto da execução das requisições de clientes. As requisições de reconfiguração são assinadas para garantir que o emissor é o *View Manager* (administrador) e não um cliente qualquer do sistema.

Após a requisição de reconfiguração ser ordenada, a mesma não é entregue para a aplicação como acontece com as requisições normais. Ao invés disso, as replicas atualizam suas visões atuais do sistema de acordo com as solicitações contidas na requisição e enviam uma resposta para o *View Manager* informando sobre o sucesso no processamento das atualizações. Caso positivo, o *View Manager* envia uma mensagem especial para a(s) replica(s) que está(ão) aguardando para entrar (sair) no sistema, informando que as mesmas já podem iniciar (deixar) a execução. Finalmente, as replicas adicionadas no sistema iniciam o protocolo de transferência de estado para suas atualizações.

## 3. Arquitetura e Implementação do BFT-SMART

O BFT-SMART contém menos do que 13.5K linhas de código Java divididos em pouco mais de 90 arquivos. Outros sistemas possuem um código muito maior: o PBFT [Castro and Liskov 2002] contém 20K linhas de código C, o UpRight [Clement et al. 2009] contém 22K linhas de código Java e o JPaxos [Santos and Schiper 2013] possui mais do que 22k linhas de código Java.

O principal desafio no projeto e implementação de um sistema de replicação com alto desempenho está na organização das várias tarefas dos protocolos em uma arquitetura robusta e eficiente. No caso de sistemas para BFT SMR, temos dois outros requisitos: o

sistema deve suportar centenas de clientes e resistir a comportamentos maliciosos tanto de replicas quanto de clientes. A Figura 1 apresenta a arquitetura do BFT-SMART com as *threads* utilizadas no processamento das requisições. Em nossa arquitetura, todas as *threads* comunicam-se através de filas (*bounded queues*). A figura mostra quais *threads* produzem/consomem dados em cada fila.

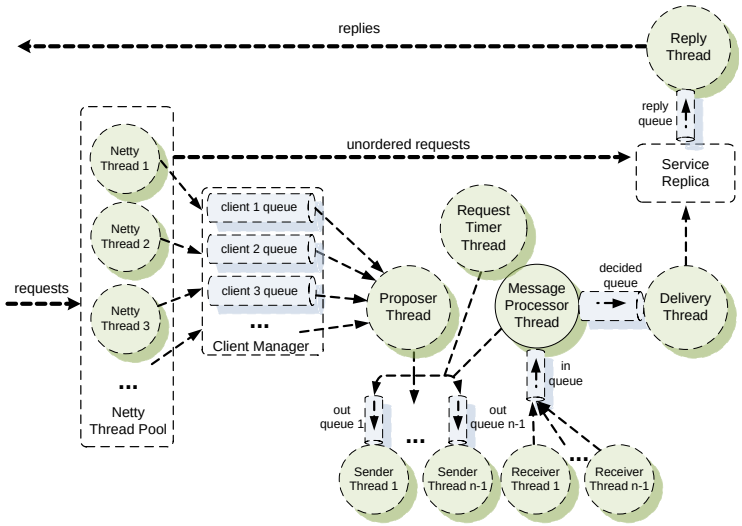



**Figura 1. Processamento de requisições em uma replica do BFT-SMART.**

As requisições de clientes são recebidas através de um *pool* de *threads* implementadas pelo *framework* de comunicação *Netty*, fazendo com que o BFT-SMART forneça um suporte eficiente para centenas de conexões entre clientes e replicas. Quando uma requisição é recebida, primeiramente é verificado se a mesma precisa ser ordenada (o BFT-SMART suporta requisições, geralmente de apenas leitura, que não precisam de ordenação). Requisições não ordenadas são diretamente entregues para a aplicação implementada (replica). As outras requisições são encaminhadas ao *client manager* que verifica a integridade das mesmas e as adiciona nas respectivas filas de cada cliente.

A *thread proposer* é responsável por criar os lotes de requisições e enviar a mensagem de proposta (*PROPOSE*) do protocolo de consenso. O BFT-SMART adiciona requisições em um lote até que (1) o número de requisições no lote atinja um limite especificado em um arquivo de configuração ou (2) não existam mais requisições pendentes. Esta *thread* apenas é ativada na replica líder do consenso.

Toda mensagem $m$ que será enviada para outra replica é adicionada em uma fila de saída (*out queue*). Após isso, uma *thread* de envio (*sender thread*) remove $m$ desta fila, serializa $m$, produz o MAC a ser adicionado na mensagem e a envia usando *sockets* TCP. Na replica receptora, uma *thread* de recebimento (*receiver thread*) para o dado emissor recebe $m$, autentica $m$ (valida seu MAC), deserializa $m$ e adiciona esta mensagem em uma fila de entrada (*in queue*) que armazena as mensagens recebidas de todas as replicas que ainda não foram processadas.

A *thread* de processamento de mensagens (*message processor thread*) é responsável pelo processamento das mensagens do protocolo BFT SMR. Esta *thread* recupera mensagens da fila de entrada e as processa caso sejam referentes à instância do consenso sendo executada. As mensagens relacionadas com uma instância seguinte do consenso são processadas apenas quando tal instância é inicializada.

Quando uma instância do consenso termina em uma replica, o lote decidido é adicionado na fila de decididos (*decided queue*). Uma *thread* de entrega (*delivery thread*) é responsável por retirar os lotes desta fila, deserializar todas as requisições do lote, remover as respectivas requisições das filas dos respectivos clientes e marcar a instância corrente do consenso como finalizada. Após isso, esta *thread* executa a aplicação implementada (*service replica*), executando as requisições e gerando as respectivas respostas que são adicionadas em uma fila de respostas (*reply queue*). Por fim, uma *thread* de respostas (*reply thread*) remove as respostas desta fila e as envia para os respectivos clientes.

Outra *thread* que merece destaque é a *request timer thread*, que é ativada periodicamente para verificar se alguma requisição permaneceu mais do que um tempo pré-definido na fila de requisições pendentes (de clientes). O primeiro *timeout* para uma requisição faz com que a mesma seja encaminhada para o líder corrente. Já o segundo *timeout* para a mesma requisição faz com que a instância corrente do consenso seja parada e a fase de sincronização seja executada (Seção 2.2). A ideia por traz destes *timeouts* é a seguinte: em condições normais da rede, um *timeout* é causado por um cliente que não enviou a requisição para o líder ou pelo líder que não ordenou a requisição de um cliente. Desta forma, primeiramente assumimos que o problema está no cliente e o líder é suspeito apenas se o problema persistir.

## 4. Configurações Alternativas

Como mencionado, o BFT-SMaRt pode ser configurado para tolerar tanto falhas por *crash* quanto falhas maliciosas.

**Tolerância a Falhas por Crash (CFT).** Um parâmetro de configuração é utilizado para indicar que o sistema deve ser CFT. Nesta configuração, o sistema tolera $f < n/2$ falhas (minoria simples), implicando em algumas mudanças nos protocolos (ex.: as mensagens *WRITE* do protocolo de consenso não são mais necessárias).

**Tolerância a Falhas Bizantinas.** O BFT-SMaRt pode ser configurado para utilizar criptografia de chave pública, fazendo cada cliente assinar suas requisições. Isso evita que clientes realizem ataques ao sistema para forçar a troca de líder [Amir et al. 2011].

## 5. API e Modelo de Programação

Para implementar uma aplicação baseada no BFT-SMaRt, duas classes principais são utilizadas. A classe `ServiceReplica` é utilizada no lado do servidor para instanciar uma replica da aplicação enquanto que a classe `ServiceProxy` é utilizada no lado do cliente para acessar a aplicação replicada. Para instanciar uma `ServiceReplica`, é necessário fornecer um identificador numérico (que é mapeado para um IP e uma porta através de um arquivo de configuração) e implementar as interfaces `Executable` (que define os métodos acessados pelo BFT-SMaRt quando a aplicação deve executar uma requisição) e `Recoverable` (que define os métodos acessados pelo BFT-SMaRt para gerenciamento do estado da aplicação). Usualmente, estas duas interfaces são implementadas por uma única classe (veja abaixo). No lado do cliente, para instanciar um `ServiceProxy` apenas é necessário o identificador numérico do cliente. O restante desta seção discute brevemente a API do BFT-SMaRt, que é descrita em detalhes na página da ferramenta [BFT-SMART , Bessani et al. 2013b].

**Lado do Servidor.** A forma mais simples de se implementar uma aplicação replicada usando o BFT-SMaRt é através da classe abstrata `DefaultSingleRecoverable`,

que implementa as interfaces `Executable` e `Recoverable` acima descritas. Para isso, o desenvolvedor precisa estender esta classe abstrata e implementar os seguintes métodos abstratos:

```
public byte[] executeOrdered(byte[] cmd, MsgContext ctx);
public byte[] executeUnordered(byte[] cmd, MsgContext ctx);
public byte[] getSnapshot();
public void installSnapshot(byte[] state);
```

O BFT-SMART invoca os métodos `executeOrdered` e `executeUnordered` para entregar as requisições (comandos) dos clientes para a aplicação. O primeiro método é executado quando o cliente solicita requisições que devem ser ordenadas, enquanto que o segundo é utilizado para requisições que não são ordenadas (usualmente requisições que apenas fazem leituras no estado da aplicação). Estes métodos devem implementar a parte principal da aplicação (o código da aplicação deve estar dentro destes métodos) e retornar respostas que serão enviadas aos clientes pelo BFT-SMART. O argumento `cmd` representa a requisição do cliente serializada e `ctx` contém meta-dados como o identificador do cliente, o identificador da instância do consenso que ordenou a requisição, a latência do consenso, etc. Além disso, para o gerenciamento do estado da aplicação, os desenvolvedores devem implementar os métodos `getSnapshot` e `installSnapshot` para criar e instalar *snapshots* serializados do estado da aplicação, respectivamente.

A classe `DefaultSingleRecoverable` geralmente é suficiente para implementação de aplicações baseadas no BFT-SMART. No entanto, uma aplicação pode utilizar implementações customizadas das interfaces `Executable` e `Recoverable` (veja [Bessani et al. 2013b] para mais detalhes).

**Lado do Cliente.** No lado do cliente, cada instância da classe `ServiceProxy` representa um cliente diferente com um identificador distinto. Esta classe fornece os seguintes métodos para enviar requisições (comandos) para a aplicação:

```
public byte[] invokeOrdered(byte[] request);
public byte[] invokeUnordered(byte[] request);
public void invokeAsynchronous(byte[] request,
       ReplyListener listener, int[] receivers, MsgType type);
```

Em todos estes métodos, requisições e respostas devem ser serializadas em um *array* de *bytes*, permitindo que os mesmos sejam genéricos e suportem qualquer aplicação. Os métodos `invokeOrdered` e `invokeUnordered` são utilizados para enviar requisições que serão e não serão ordenadas, respectivamente. O método `invokeAsynchronous` pode ser utilizado para envio de requisições (tanto ordenadas quanto não ordenadas – definido em `MsgType`) de forma não bloqueante, i.e., este método retorna sem aguardar pelas respostas das replicas. Através deste método programadores podem criar aplicações que continuam executando enquanto que o BFT-SMART coleta as respostas em *background*. Para utilizar este artifício, os programadores devem fornecer uma classe para *callback* definida pela interface `ReplyListener`, a qual deve gerenciar o recebimento das respostas.

A classe `ServiceProxy` geralmente é suficiente para implementação de aplicações baseadas no BFT-SMART. No entanto, clientes também podem sofrer customizações como por exemplo na forma de contar as respostas [Bessani et al. 2013b].

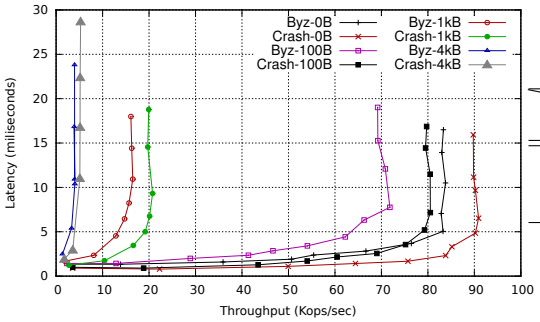
## 6. Avaliação de Desempenho

Esta seção traz algumas avaliações sobre o *throughput* e a latência apresentada pelo BFT-SMART, além de compará-lo com outros sistemas similares.

**Configuração.** Os experimentos foram executados com três (CFT) e quatro (BFT) replicas hospedadas em diferentes máquinas e até 1600 clientes foram distribuídos uniformemente em outras quatro máquinas. As máquinas são servidores Dell PowerEdge R410, configurados com JRE 1.7.0_21 e o Ubuntu Linux 10.04. Cada máquina possui 32GB de memória e dois processadores *quad-core* 2.27 GHz Intel Xeon E5520 com *hyper-threading*, i.e., suportando 16 *threads* em *hardware*. Todas máquinas se comunicam através de uma rede Ethernet *gigabit*.

***Micro-benchmarks*.** Usamos um *micro-benchmark* comumente utilizado para avaliar sistemas SMR, que consiste na implementação de um serviço "vazio" com o BFT-SMART para calcular o *throughput* no lado dos servidores e a latência no lado dos clientes. Os resultados do *throughput* foram obtidos na replica líder e a latência foi calculada em um dos clientes (sempre o mesmo). O desvio padrão sempre foi menor do que $3\%$.

**Resultados.** A Figura 2 apresenta os resultados para o BFT-SMART configurado tanto como CFT quanto BFT, considerando diferentes tamanhos para requisições/respostas: 0/0, 100/100, 1024/1024 e 4096/4096 *bytes*. Nesta figura podemos perceber que o BFT-SMART configurado como CFT possui um desempenho melhor do que quando configurado como BFT. Isso pode ser explicado pelo menor número de mensagens utilizadas na configuração CFT, o que resulta em menos trabalho para executar cada requisição.




| *Requisições* \ *Respostas* | 0 *bytes* | 100 *bytes* | 1024 *bytes* |
|---|---|---|---|
| 0 *bytes* | 83337 | 75138 | 37320 |
| 100 *bytes* | 78711 | 72879 | 36948 |
| 1024 *bytes* | 16309 | 16284 | 15878 |

**Figura 2. Latência vs. *Throughput* para $f = 1$ (esquerda) e *throughput* (op./seg.) para diferentes tamanhos de requisições/respostas e $f = 1$ (direita).**

Estes resultados são completados com a tabela da direita da Figura 2, que mostra como as diferentes combinações de tamanhos de requisições e respostas afetam o *throughput*. Estes resultados foram obtidos com 1600 clientes acessando o sistema configurado como BFT. Podemos perceber que o tamanho da requisição tem uma maior influência no *throughput* do que o tamanho da resposta. Isso pode ser explicado pelo aumento no tamanho do lote de requisições a ser proposto no consenso.

**Comparação com outros.** Comparamos o BFT-SMART (BFT) com dois sistemas que implementam BFT SMR: o PBFT [Castro and Liskov 2002] e o UpRight [Clement et al. 2009]. Já o BFT-SMART (CFT) foi comparado com o JPaxos [Santos and Schiper 2013], que implementa um CFT SMR. O *benchmark* 0/0 foi utilizado nestes experimentos. A Tabela 1 apresenta o *throughput* máximo obtido para estes sistemas e o número de clientes requeridos para isso. Também apresentamos valores para o mesmo número de clientes, onde utilizamos 200 clientes (*Throughput 200*), que foi o número máximo suportado pelo PBFT. Os resultados mostram que o BFT-SMART apresentou um *throughput* maior do que os outros sistemas em nosso ambiente.

| *Sistema* | *Throughput* | *Clientes* | *Throughput 200* |
|---|---|---|---|
| BFT-SMART (BFT) | 83801 | 1000 | 66665 |
| PBFT | 78765 | 100 | 65603 |
| UpRight | 5160 | 600 | 3355 |
| BFT-SMART (CFT) | 90909 | 600 | 83834 |
| JPaxos | 62847 | 800 | 45407 |

**Tabela 1. *Throughput* máximo para o *benchmark* 0/0 e $f = 1$.**

## 7. Descrição da Demonstração Planejada

O potencial do BFT-SMART será explorado através da execução de um serviço de armazenamento tipo chave-valor replicado (descrito em [Bessani et al. 2013b]). Este serviço implementa a interface *Map* da API do Java e usa o BFT-SMART para replicação dos dados armazenados. As replicas podem ou não executar em máquinas diferentes, sendo que a demonstração deve abordar a ocorrência de falhas e de reconfigurações, visando a exploração de todos os mecanismos fornecidos pelo BFT-SMART.

## 8. Conclusões

Este artigo apresentou o BFT-SMART, uma ferramenta robusta e eficiente para implementação de BFT SMR. Os experimentos apresentados mostram que a implementação atual do BFT-SMART fornece um *throughput* maior do que outras propostas similares. O sistema aqui descrito é disponível como *software* de código aberto na página do projeto e atualmente existem vários grupos de pesquisa usando e modificando esta ferramenta de acordo com suas necessidades. Finalmente, uma descrição mais detalhada do BFT-SMART pode ser encontrada em [BFT-SMART , Bessani et al. 2013b].

## Referências


Amir, Y., Coan, B., Kirsch, J., and Lane, J. (2011). Prime: Byzantine replication under attack. *IEEE Transactions on Dependable and Secure Computing*, 8(4):564–577.

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

Bessani, A., Sousa, J., and Alchieri, E. (2013b). State machine replication for the masses with bft-smart. DI-FCUL-TR-2013-07, Department of Computer Science, University of Lisbon.

BFT-SMART. High-performance byzantine fault-tolerant state machine replication. `http://code.google.com/p/bft-smart/`.

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

Chandra, T., Griesemer, R., and Redstone, J. (2007). Paxos made live - an engineering perspective (2006 invited talk). In *Proc. of the PODC'07*.

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

Guerraoui, R., Knežević, N., Quéma, V., and Vukolić, M. (2010). The next 700 BFT protocols. In *Proc. of ACM EuroSys'10*.

Lamport, L., Malkhi, D., and Zhou, L. (2010). Reconfiguring a state machine. *SIGACT News*, 41(1).

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

Sousa, J. and Bessani, A. (2012). From Byzantine consensus to BFT state machine replication: A latency-optimal transformation. In *Proc. of EDCC'12*.

Veronese, G., Correia, M., Bessani, A., Lung, L., and Verissimo, P. (2013). Efficient Byzantine fault tolerance. *IEEE Transactions on Computers*, 62(1).


32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Florianópolis - SC

# Salão de Ferramentas do SBRC 2014

## *Sessão Técnica 3*

## *Redes*

# OpenFlow 1.3 Software Switch

**Eder Leão Fernandes**[1], **Christian Esteve Rothenberg**[2]

[1]Diretoria de Redes Convergentes
Fundação CPqD – Campinas – SP – Brasil

[2]Departamento de Engenharia de Computação e Automação Industrial
Universidade de Campinas (UNICAMP) – Campinas – SP – Brasil

ederlf@cpqd.com.br, chesteve@dca.fee.unicamp.br

***Abstract.*** *OpenFlow is a new and fast evolving technology that enables the implementation of Software Defined Networks. This papers describes a tool designed for rapid and inexpensive experimentation of OpenFlow networks. The software switch is intended to be easy to prototype new features due to the implementation in a traditional language and for his simple code. These characteristics contributed to the adoption of the tool by an active community and has become a good choice for both OpenFlow newcomers as well as more advanced experimenters.*

***Resumo.*** *OpenFlow é uma tecnologia nova e de rápida evolução que habilita a implementação de Redes Definidas por Software. Este artigo apresenta uma ferramenta desenvolvida para experimentação rápida e de baixo custo. O software switch é voltado para a prototipagem de novas funcionalidades e novas versões do protocolo pelo seu código simples implementado em uma linguagem tradicional. Essas características ajudaram a criar uma comunidade ativa e fizeram da ferramenta uma boa escolha tanto para o primeiro contato com OpenFlow como para experimentações mais avançadas.*

## 1. Introdução

O modelo atual das redes de computadores, embora um sucesso, ao se pensar na Internet e nos serviços disponíveis hoje, sofre com vários problemas para se adaptar às novas demandas e cenários da Internet do Futuro [Shenker 1995] [Feldmann 2007]. Por isso, nos últimos anos, a área de redes de computadores tem visto a ascensão do modelo de Redes Definidas por Software. A separação do plano de controle do plano de dados da rede, apesar de não ser uma proposta realmente nova, ganhou força após a criação do protocolo *OpenFlow*, com o qual é possível programar o encaminhamento dos pacotes através de um agente externo, além de poder controlar todos os nós da rede a partir de um único ponto [McKeown et al. 2008].

Apesar de existirem outras tecnologias habilitadoras das Redes Definidas por Software, o protocolo *OpenFlow* segue como uma das principais apostas, principalmente pela indústria, visto a criação de uma organização para desenvolver o padrão [ONF 2013]. Na academia, apesar de mais tímidos, vários trabalhos começam a utilizar o *OpenFlow* para o avanço da pesquisa em redes.

Para rápida experimentação e desenvolvimento de aplicações *OpenFlow* foram desenvolvidas várias ferramentas, como *software switch*, controlador [Gude et al. 2008]

e plugin do *Wireshark* para capturar as mensagens trocadas no canal de controle. Porém todas essas ferramentas estavam disponíveis apenas para OpenFlow 1.0. Com o cenário em mente e a rápida evolução das versões do protocolo pela *Open Networking Foundation*, foi identificada a necessidade de evolução das ferramentas para a versão mais atual, pois as novas funcionalidades habilitam um número ainda maior de experimentos, permitindo a antecipação de soluções para equipamentos não existentes à epoca.

O resto do artigo está organizado da seguinte forma: A Seção 2 fala sobre o breve histórico do trabalho e a motivação para o desenvolvimento da ferramenta. A Seção 3 descreve a arquitetura em alto nível. Na Seção 4 são apresentadas as funcionalidades do *OpenFlow* 1.3 implementadas no *software switch*. A Seção 5 traz os resultados obtidos no projeto. A Seção 6 apresenta a documentação e a proposta de demonstração no Salão de Ferramentas do SBRC 2014. A Seção 7 discute trabalhos futuros e a Seção 8 é a conclusão do artigo. A última Seção traz os e agradecimentos aos colaboradores desse trabalho.

## 2. Histórico, Objetivos e Motivação

Na época em que o trabalho de implementação do *software switch* foi iniciado, não havia outras opções disponíveis. As ferramentas existentes suportavam até a versão 1.1 do *OpenFlow* [Kis 2011], num momento em que haviam vários esforços para adição de novas funcionalidades. Um exemplo é o protocolo *IPv6*, de grande interesse aos primeiros usuários, devido à situação de esgotamento do *IPv4*. O primeiro trabalho, então, foi adicionar ao *switch* de referência da versão 1.1 suporte à *IPv6* e aos seus cabeçalhos de extensão [Denicol et al. 2011]. O resultado desse primeiro trabalho foi muito importante para a sequência do *software switch*, pois a implementação de umas das principais mudanças da versão 1.2, o *OpenFlow Extended Match* (OXM), foi facilitada devido à implementação anterior utilizar o modelo do *Nicira Extended Match* do *Open vSwitch* (OVS) [Pfaff et al. ], o qual foi a base do OXM. Logo após a divulgação da especificação da versão 1.3, o trabalho continou até culminar na implementação descrita nesse artigo.

Para desenvolver a ferramenta foram consideradas duas características essenciais: (i) usabilidade simples, com integração ao Mininet [Lantz et al. 2010] e (ii) fácil de estender (linguagem e código não devem ser de difícil modificação). Esses requisitos foram identificados no *software switch* de referência da versão 1.1, o qual possui um código simples e é uma das opções de simulação no Mininet.

### 2.1. Análise das ferramentas existentes

Além desse trabalho também há outros *software switches OpenFlow* em constante desenvolvimento. Do ponto de vista da implementação e experimentação todos possuem vantagens e desvantagens.

***Open vSwitch.*** É o *software switch* mais utilizado pela comunidade. Possui grande desempenho, pois disponibiliza, além do modo usuário, um modo *kernel*, sendo propício até para ambientes de produção. O suporte à *OpenFlow* está sendo feito de modo que um único programa suporte todas as versões, por isso ainda tem muitas funcionalidades parciais de cada versão. O fato de ser mais difícil programar no nível do *kernel* e também por não ser um *switch* com foco apenas em *OpenFlow*, tornam o OVS uma escolha mais difícil para rápida prototipagem de novas funcionalidades do protocolo.

**LINC.** O LINC é um *software switch* escrito em *Erlang*. É um programa de nível do usuário e suporta *OpenFlow* 1.2 e 1.3, com integração ao Mininet. Uma grande vantagem do LINC para os outros é o suporte ao protocolo de gerência dos *switches OpenFlow* chamado OF-CONFIG. O maior desafio para o LINC está na linguagem, pois exige bom conhecimento em Erlang.

**Trema Edge Switch.** Este *software switch* faz parte do código do controlador Trema, desenvolvido pela NEC. O interessante desse *switch* é poder criar topologias virtuais utilizando apenas uma ferramenta, diferente das outras que usam o *Mininet*. O código ainda está em fase de transição para o repositório principal do Trema.

## 3. Arquitetura

A arquitetura do *software switch* segue a especificação do *OpenFlow 1.3*. A Figura 1 mostra os principais componentes da ferramenta.

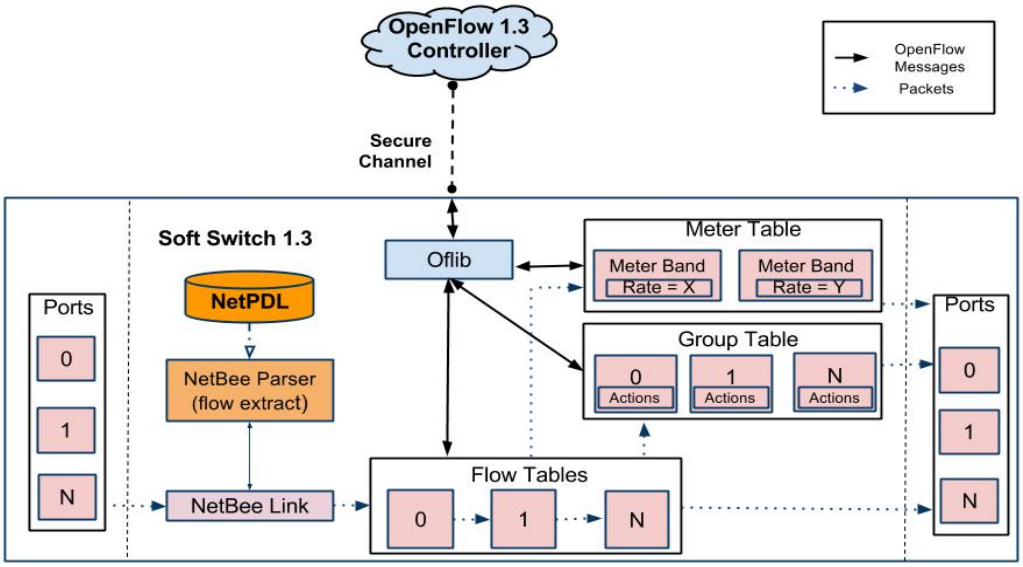



Figura 1. Arquitetura do Software Switch 1.3.

***Ports:*** São as portas do *software switch*, nas quais os pacotes são recebidos e enviados. A ferramenta pode utilizar portas físicas e virtuais do equipamento em que está sendo executado. Na Figura 1 as portas estão refletidas para demonstrar o fluxo dos pacotes.

***NetPDL:*** É uma linguagem definida em XML para descrever protocolos de rede [Risso and Baldi 2006]. A ferramenta utiliza um arquivo com os protocolos suportados pela especificação do *OpenFlow* 1.3, o qual é utilizado na decodificação dos pacotes pela biblioteca *NetBee*.

***NetBee Parser:*** Biblioteca desenvolvida para vários tipos de processamento em pacotes, como decodificação e filtragem, por exemplo [NetGroup 2004]. A biblioteca constrói uma árvore em memória a partir do arquivo *NetPDL*, a qual é usada pelo componente NetBee Link para a decodificação.

***NetBee Link:*** Utiliza a biblioteca NetBee para decodificar os pacotes que chegam pelas interfaces do *software switch*. Os campos de protocolo do pacote são convertidos na estrutura de *match* a qual será enviada para as tabelas de fluxos.

***Flow Tables:*** As múltiplas tabelas, presentes desde a versão 1.1, é o componente no qual os fluxos são instalados e verifica-se se nenhum pacote possui uma entrada correspondente. Se existir uma entrada, o pacote é processado por alguma das instruções descritas na especificação, caso contrário, o mesmo é descartado.

***Group Table:*** A tabela de grupos foi adicionada ao OpenFlow 1.1 para proporcionar novas formas de encaminhamento. Cada grupo possui um conjunto de ações chamados *Actions Buckets*. Os pacotes chegam à tabela de grupos depois de um pacote ser processado por uma ação de grupo.

***Meter Table:*** Funcionalidade adicionada na versão 1.3 do protocolo, a *Meter Table* é uma tabela utilizada para limitar a banda por fluxo. Os pacotes são enviados para ela através de uma instrução contida em algum fluxo das tabelas de fluxos, para serem processados por estruturas chamadas *Band*. Um *Band* possui uma taxa de limite e é acionado no caso do fluxo de pacotes ultrapassar esse valor. A *Meter Table* pode ter múltiplos *Band*, sendo que no caso do fluxo ultrapassar o limite de vários é escolhido aquele com a a maior taxa próxima da banda passante atual.

***Oflib:*** É a biblioteca utilizada para converter as mensagens *OpenFlow* do formato de rede para um formato interno, mais eficiente e fácil de trabalhar (geralmente sem o *padding* explicíto nas estruturas apresentadas na especificação) , e do formato interno para o de rede.

***Secure Channel:*** Reponsável pela conexão entre o *software switch* e o controlador. Apesar do nome, no momento é possível realizar apenas conexões TCP não seguras.

## 4. Funcionalidades do *OpenFlow* 1.3 implementadas

A especificação 1.3 do *OpenFlow* adicionou significativas funcionalidades ao protocolo. Quase todas foram implementadas, com exceção apenas de conexões auxiliares utilizando o protocolo UDP, pois a especificação não é clara a respeito de como o *switch* deve lidar com mensagens perdidas, uma vez que no UDP não há reenvio de mensagens. Abaixo, segue a lista de modificações e comentários a respeito de como foram implementadas no software switch.

***Table miss.*** No *OpenFlow 1.3* não há mais opções para determinar o comportamento das tabelas, no caso de não existir um fluxo instalado para algum tipo de pacote. A implementação segue o padrão de descartar esses pacotes, deixando a responsabilidade para o usuário do *switch* instalar o fluxo especial, chamado de *table-miss*, para evitar o descarte.

**Cabeçalhos estendidos do IPv6.** O trabalho realizado no trabalho para estender o *OpenFlow 1.1* para IPv6 contava com os cabeçalhos estendidos, porém a versão 1.3 verifica apenas a presença desses campos. Portanto foi necessário reescrever o código antigo do parsing para atender esse novo cenário, principalmente a verificação da ordem dos cabeçalhos.

***Flow Meter.*** Foram implementados os dois tipos descritos na especificação: *Drop*, no qual a limitação é feita com o descarte dos pacotes que ultrapassem o limite de banda, e *DSCP Remark*, em que a precedência de descarte do campo DSCP do pacote é aumentada, possibilitando a criação de um serviço diferenciado.

O mecanismo de medida e limitação da banda por fluxo foi implementado utilizando o algoritmo de *Token Bucket*.

**Filtragem de eventos por conexão.** Essa funcionalidade permite ao controlador configurar o *switch* para não enviar determinados tipos de mensagem assíncrona. Foi implementada a filtragem tanto para os papéis de mestre quanto para escravo, habilitando ao desenvolvedor da aplicação especializar o controlador para receber ou não mensagens específicas.

**Conexões auxiliares.** O *switch* pode abrir conexões auxiliares com os controladores para melhor explorar o parelelismo de algumas operações. Com um identificador auxiliar o *switch* identifica a conexão principal e pode ser configurado para receber tipos especificos de mensagens, como *packet-in*.

**Adição dos campos *MPLS BOS* e *PBB*.** Foram adicionados os campos de *MPLS Bottom of the Stack* e o *I-SID* do protocolo *Provider Backbone Bridging*. O suporte à PBB também exigiu a implementação das operações de *Push* e *Pop*, para adicionar e remover o cabeçalho respectivamente.

Há outras mudanças menores, como desabilitar os contadores dos fluxos e a adição do campo cookies no packet-in, mas, pela simplicidade, não serão destacadas aqui.

## 5. Resultados e Casos de Uso

Vários resultados podem ser contabilizados até o momento.

**Desenvolvimento de comunidade open source.** A popularidade na comunidade é considerada um desses resultados. A Figura 2 apresenta a quantidade de visitas e visitantes no período de 1 a 14 de Janeiro e mostra também o número de pessoas acompanhando o repositório (42) e *forks* (56) do código. Até o momento foram realizados 219 commits, de 12 contribuintes diferentes [1] e já foram postados 82 *Issues* com informação de *bugs*, sugestões e perguntas. Estes números demonstram constante interesse da comunidade em utilizar e contribuir para o projeto.

**Integração no Mininet.** Com a popularidade, a ferramenta acabou sendo integrada ao *script* de instalação do emulador *Mininet*, alcançando um número ainda maior de usuários.

**Roteador Wireless.** Outro resultado relevante é a adaptação do código para o sistema operacional *OpenWrt*, permitindo utilizar *OpenFlow 1.3* em roteadores com esse sistema, habilitando testes em um ambiente mais real e também com interfaces sem fio.

**Outros resultados.** Como resultado indireto considera-se outros artefatos gerados, devido à necessidade de testar o *software switch*. A adaptação do controlador NOX, do *plugin* do *Wireshark* e do *framework* de testes OFtest para *OpenFlow* 1.3 fazem parte de um ambiente com todos os recursos necessários para experimentar o *OpenFlow* 1.3.

### 5.1. Casos de uso

Dentre os resultados, consideram-se os casos de uso conhecidos, pois como um projeto aberto e com foco na experimentação, o emprego da ferramenta em variados cenários são fatores do grau de sucesso alcançado.

[1] A verdade é que há mais, os 12 usuários são contribuições diretas via *Github*.

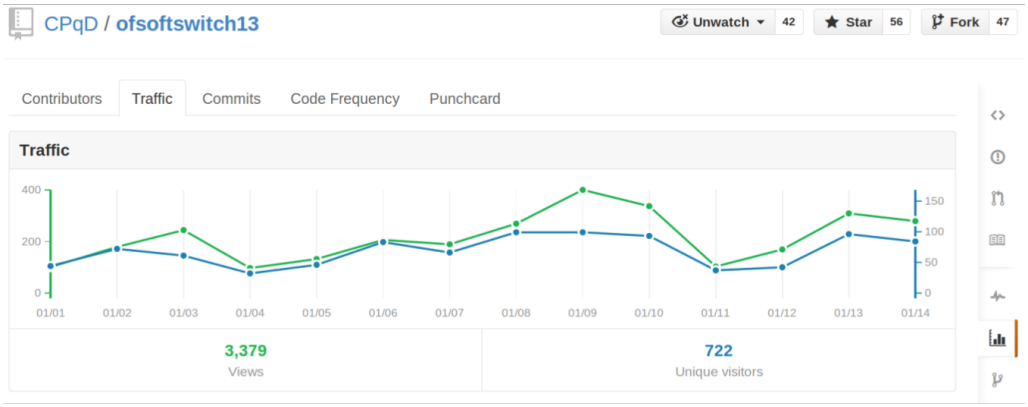



Figura 2. Estatísticas de acesso do of13softswitch no repositório *Github*.

**Plataforma para incubação e padronização de novas funcionalidades.** Membros da ONF têm empregado o *software switch* para prototipar em código aberto e validar experimentalmente novas propostas funcionalidades do protocolo *OpenFlow* [Tourrilhes 2013], requisitos necessários para novas extensões e funcionalidades serem aceitas na especificação do padrão *OpenFlow*.

**Acadêmico.** A ferramenta tem sido utilizada para ministrar cursos de introdução à *OpenFlow* e *SDN*. Além disso, aparece em dissertações de mestrado seja para adicionar novas funcionalidades [Shahmir Shourmasti 2013] ou como ferramenta de apoio [Arora 2013].

**Indústrial.** Empresas estão utilizando a ferramenta para testes de interoperabilidade com controladores, provas de conceito e até criando portes comerciais para implementações *hardware*, principalmente na linha de *Network Processors*.

## 6. Documentação, Código e Demonstração

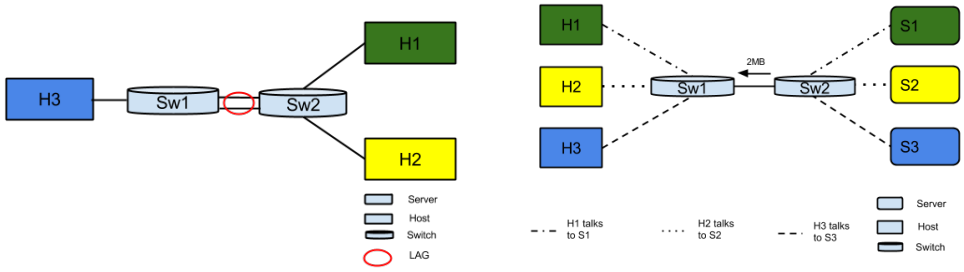
O *software* é de código aberto, licença BSD, e está disponível no *Github*, na página https://github.com/CPqD/ofsoftswitch13. O uso da ferramenta de issues do *Github*, no endereço *https://github.com/CPqD/ofsoftswitch13/issues* é encorajado para reportar *bugs* ou discutir sobre as funcionalidades do *software switch*. Contribuições podem ser submetidas realizando um *fork* do código e utilizando o recurso de *pull request* do *Github*.

A documentação está no endereço *https://github.com/CPqD/ofsoftswitch13/wiki* e inclui além de tutorial do *OpenFlow* 1.3, os comandos para utilizar o *Dpctl* e como compilar o *switch* para um roteador com o sistema operacional *OpenWrt*.

A demonstração focará em funcionalidades menos exploradas das versões mais novas do *OpenFlow*. Para isso serão criadas duas aplicações para o controlador Ryu [Open Source Software Computing Group 2012], as quais controlarão diferentes topologias no *Mininet*:

***Link Aggregation (LAG)*** Esse cenário demonstra uma das aplicações do uso de tabelas de grupos do *OpenFlow*. O cenário da Figura 3(a) mostra os *switches* Sw1 e Sw2 conectados através de uma agregação de *links*. A demonstração focará na redundância

criada pelo *LAG*, mostrando que, ao derrubar um dos *links*, os *hosts* H1 e H2 ainda serão capazes de se comunicar com H3.




(a) Link Aggregation (b) Limitação da banda por fluxo

Figura 3. Topologias da demonstração

**Limitação de Banda por fluxo.** Essa aplicação demonstrará como pode ser realizada a limitação da banda no cenário de uma rede doméstica. A topologia dessa demonstração é a da Figura 3(b), em que há seis *hosts*: H1, H2 e H3 são os clientes da rede, enquanto S1, S2, S3 representam os servidores de onde os usuários obtêm conteúdo. Os clientes estão ligados a Sw1 enquanto os servidores estão conectados a Sw2. Entre os dois *switches OpenFlow* há um *link* com velocidade de 2MB. Inicialmente todos os usuários compartilham essa mesma banda sem reserva alguma, demonstrado utilizando o programa IPERF entre os clientes e servidores. Em seguida, após a instalação de alguns fluxos com instrução de *Meter*, o trafêgo para H2 e H3 será limitado para 500KB/s, garantindo a H1 o total de 1MB do *link* entre Sw1 e Sw2.

## 7. Trabalhos Futuros

A atualização do código para as revisões, já foram lançadas até a versão 1.3.3, é uma das prioridades para o futuro próximo. Além disso, testes além do OFTest, como os de certificação do *Ryu* e da plataforma de testes *Twister*, mostram falhas em algumas funcionalidades, portanto a correção desses problemas também deve ser priorizada.

No longo prazo, o objetivo é transformar o *software switch* em uma plataforma de prototipação mais modular, de modo que uma nova funcionalidade possa ser adicionada sem a necessidade de modificações em vários pontos do código. Para isso será necessário remodelar toda a arquitetura, buscando um *design* fácilmente extensível. Outra mudança é a remoção da biblioteca *Netbee* e utilizar outro mecanismo para decodificar os pacotes, pois vários usuários relataram problemas devido a velocidade de atualização não acompanhar a dos sistemas operacionais. Por fim, novas versões do protocolo sempre estão em pauta, e a implementação da versão 1.4 seria um passo natural para manter o compasso com a versão mais recente.

## 8. Conclusão

O *software switch* da versão 1.3 do *OpenFlow* é uma ferramenta bem estabelecida como projeto *Open Source*, com uma boa base de usuários e contribuidores. Com características que o tornam a opção mais fácil para prototipar novas funcionalidades, como código mais simples em linguagem C e a facilidade de uso, tem sido a escolha da ONF para prototipação e validação de novos recursos. Os experimentos na academia, utilizado em

mestrados e aulas, e na indústria reforçam a utilidade da ferramenta no desenvolvimento do protocolo *OpenFlow* e na experimentação de aplicações com requisitos cobertos apenas pela versão 1.3 em diante.

## 9. Agradecimentos

Agradecemos ao Centro de Inovação da Ericsson Brasil, pelo suporte financeiro e a manutenção do projeto com o código aberto, e a Ericsson *Research* pela colaboração técnica. E a comunidade de usuários do projeto, pelas contribuições.

## Referências

Arora, D. (2013). Proactive routing in scalable data centers with paris.

Denicol, R. R., Fernandes, E. L., Rothenberg, C. E. R., and Kis, Z. L. (2011). On ipv6 support in openflow via flexible match structures. *OFELIA/CHANGE Summer School*.

Feldmann, A. (2007). Internet clean-slate design: What and why? *SIGCOMM Comput. Commun. Rev.*, 37(3):59–64.

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). Nox: Towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110.

Kis, Z. L. (2011). OpenFlow 1.1 software switch. `https://github.com/TrafficLab/of11softswitch`. [Online; acessado 14-Janeiro-2014].

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA. ACM.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

NetGroup (2004). The NetBee Library. `http://www.nbee.org/`. [Online; acessado 14-Janeiro-2014].

ONF (2013). Open Network Foundation Website. `https://www.opennetworking.org/`. [Online; acessado 14-Janeiro-2014].

Open Source Software Computing Group (2012). Ryu SDN Framework. `http://osrg.github.io/ryu/`. [Online; acessado 15-Janeiro-2014].

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M., and Shenker, S. e.a.: Extending networking into the virtualization layer. In *In: 8th ACM Workshop on Hot Topics inNetworks (HotNets-VIII).New YorkCity,NY(October 2009*.

Risso, F. and Baldi, M. (2006). Netpdl: An extensible xml-based language for packet header description. *Comput. Netw.*, 50(5):688–706.

Shahmir Shourmasti, K. (2013). Stochastic switching using openflow.

Shenker, S. (1995). Fundamental design issues for the future internet. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 13:1176–1188.

Tourrilhes, J. (2013). Prototyping on SoftSwitch. `https://github.com/jean2/ofsoftswitch13`. [Online; acessado 15-Janeiro-2014].

# The libfluid OpenFlow Driver Implementation

**Allan Vidal**[1], **Christian Esteve Rothenberg** [2], **Fábio Luciano Verdi**[1]

[1] Programa de Pós-graduação em Ciência da Computação, UFSCar, Sorocaba

[2]Faculdade de Engenharia Elétrica e Computação - FEEC, Unicamp

alnvdl@gmail.com, chesteve@dca.fee.unicamp.br, verdi@ufscar.br

***Abstract.*** *Implementations of network control protocols (such as OpenFlow) are usually divided in two parts: connectivity layer and message layer. The former manages the establishment of a control channel, while the latter provides message handling tools. Current implementations tie them together and are too specific regarding programming language, platform and protocol versions. Providing a clear, separate implementation of the connectivity layer based on formal software engineering techniques can benefit controller, switch and application developers by making development easier. Towards these goals we present the design and implementation of* `libfluid`*, the OpenFlow driver winner of the ONF competition. The tool aims at becoming the default open-source artifact to build OpenFlow switches and controllers by virtue of the design and implementation choices that have led to a developer-friendly and high-performance protocol library, as we will show in our early results evaluation.*

## 1. Introduction

Software-defined networking (SDN) emerged as a way to control networks programmatically, as a solution to several management problems common to networks worldwide [Feamster et al. 2013]. The OpenFlow protocol [McKeown et al. 2008] was proposed as a solution to the lack of programmability in networks. Introducing new features to networks had become a challenge due to the difficulty in testing new ideas in real-world scenarios.

OpenFlow takes advantage of the fact that most Ethernet networking equipment implement features such as NAT, QoS, firewalls, etc. as programmable structures, but do not expose them to an external interface. A protocol that exposes these interfaces to the hardware could benefit researchers interested in testing new ideas. The OpenFlow protocol provides an abstraction to this hardware features, and exposes them to applications running in external agents. These external agents would be responsible for instructing the hardware on how to behave, thus introducing programmability into the network, at device level.

In this new approach, most of the complexity lies in the external agent and the hardware becomes a slave to external software. This was not a completely new idea by itself; it comes from a long tradition of networking research, which started to gain a clearer focus on central control in the early 2000s, based on the needs of network administrators and traffic engineers who were dealing with increasingly complex and larger networks. Most of this research breaks free from current architectures in use, but they innovate in providing a clearer separation of the control and data forwarding planes.

The introduction of SDN and OpenFlow thus created two new pieces in a network: the switch protocol agent and the controller. Typically, an SDN controller abstraction provides a southbound interface (the protocols that effectively control the network) and a northbound interface (the API exposed to applications).

The southbound part of the paradigm can be understood as being divided in two parts: connectivity layer and message layer. The former manages the establishment of a control channel, while the latter provides message handling tools. Current implementations tie them together and are too specific regarding programming language, platform and protocol versions. Existing OpenFlow protocol implementations are a recent remarkable example of said limitations in practice.

Our work aims to create a low-overhead, portable and configurable SDN control connectivity layer that serves different use cases with a very basic set of tools (using the OpenFlow protocol as a mean).

## 2. Identified issues

In this session we will present two categories of issues: those that can affect SDN protocols in general (we will use as examples implementations of the OpenFlow protocol), and those which are specific to OpenFlow. We will present an overview of the problem and how current implementations are lacking regarding them.

### 2.1. SDN protocol implementations issues

**Issue 1: a unified connectivity layer for both controllers and equipment agents.** The connectivity layer on network control protocols can sometimes be understood as a server-client architecture: a control software (server) listens to connections from equipment (client) asking for a control service; this is the OpenFlow model. Alternatively, a control software (client) actively connects to equipment accepting control orientation (server); this happens in some scenarios in which a controller connects through an available interface in the equipment, such as a Telnet session with configuration automated by the control software.

For OpenFlow, this means that, given proper abstractions, a single solution can be built which will manage both sides of this architecture. Current OpenFlow protocol implementations target only one side of this issues. We want to create simple abstractions so that code can be reused for both purposes.

**Issue 2: minimal overhead on applications and no performance compromise.** The protocol implementation should not be the core of an SDN controller: it is the applications and network abstractions that are its main purpose. Current protocol implementations (in both controllers and switches) vary a lot regarding this issue: some are very fast and lean, others provide higher-level features at the expense of performance. The introduction of OpenFlow in equipments can also cause an impact on performance due to the limited CPUs present in most current networking hardware. Therefore, any improvements in a protocol agent will be beneficial.

This means that our implementation will have to be as lean as possible, reducing memory and CPU usage.

**Issue 3: requirement assumptions of controller applications.** Requirements vary wildly in network control applications. The implementation of a controller of networking equipment must allow the user to define which parameters are most important to the applications it will run on the network. It must be possible to build abstractions that reduce development time, or perform well regarding throughput, latency or fairness. It is impossible to provide all these guarantees at the same time, but the architecture must be able to provide them all to the developer with minimal impact.

Current OpenFlow controllers provide different abstractions, but most of them restrict developers to the same pattern of an event handling API. Answering the requirements of an application means optimizing the application for a given controller, since there is not a platform flexible enough for all tasks.

### 2.2. OpenFlow-specific issues

**Issue 4: a single, portable and lightweight implementation.** Current OpenFlow protocol implementations are restricted to one or just a handful of programming languages and architectures. While this is fine for general-purpose controllers, we want to provide a lower-level substrate for implementing the protocol while being agnostic to programming language and architectural choices.

**Issue 5: protocol version agnosticism.** Current implementations of the OpenFlow protocol tie version handling and negotiation in their core. This means that, for every new version, support has to be implemented in both the connectivity layers and the messaging layers, even though there are ways to leave most (if not all) of this task to the message layer implementation.

**Issue 6: make it possible to build stand-alone applications.** Sometimes, due to hardware constraints or requirements, it might be needed to develop a single, stand-alone application that performs a few functions without the overhead of a full controller. This paves the way for building embeddable applications in limited hardware or simply ad-hoc applications, which can be simpler to deploy and reason about. This might even be suitable in a teaching environment, in which students need to dive straight into the protocol, without having to learn controller-specific abstractions. Most controllers are oriented towards running several applications at the same time, and there is no bare-bones solution that allows programmers to build single-purpose, stand-alone applications. Examples of these applications can be: learning switches, simple routers implementing a custom engine or firewall applications.

## 3. The libfluid Architecture

libfluid was built in response to the Open Networking Foundation competition [Foundation 2014] and has been chosen as the winner entry (9 were submitted, worldwide), being awarded a cash prize and highlighted as an important implementation of an OpenFlow driver. This section describes the proposed architecture and the rationale behind the design and implementation choices to meet the identified issues in related work while addressing the requirements of the ONF competition.

### 3.1. Design principles and implementation choices

For each of the issues mentioned above, this is how we plan to tackle them, providing better approaches than those available in existing work.

**Issue 1: a unified connectivity layer for both controllers and equipment agents.** We will focus on the OpenFlow model and see how it can be implemented so that a single code base can serve for both servers (controllers) and clients (switches), tackling this issue from a formal software-engineering point of view: using inheritance in object orientation, so that common properties and functions are abstracted and can be reused for both servers and clients. How to make this is part of our research.

**Issue 2: minimal overhead on applications and no performance compromise.** Solving this issue means choosing a side: higher-level abstractions with more features at the cost of performance, or lower-level abstractions with less guarantees, leaving more work for the developer, but also leaving more room for optimization. We have chosen the latter, since we are building a platform, and not a full-fledged solution (such as a controller). This means that our implementation will have to be as lean as possible, reducing memory and CPU usage.

**Issue 3: requirement assumptions of controller applications.** In our architecture, we plan to leave the task of providing abstraction layers to the user. This means that building upon a single architecture, different requisites for latency, throughput or fairness can be achieved. The Maestro controller outlines different approaches to this problem, and we want to further investigate how their results can be applied to our work [Cai et al. 2011]. We will also explore how different requirements for applications can be implemented using abstractions such as message queues and event/message prioritization in order to validate our architecture.

**Issue 4: a single, portable and lightweight implementation.** Our solution should be written using low-level tools (i.e.: using C/C++ and dealing with OS abstractions as closely as possible and reasonable). This will help it works on different platforms and reduce the learning curve, since few higher level abstractions will have to be learned and maintained.

**Issue 5: protocol version agnosticism.** We have devised a solution for this problem using unchanging OpenFlow features to implement version negotiation, meaning that the only requirement for implementing new versions become an implementation of the messaging library. That is: if version X.Y of OpenFlow protocol specification is released, implementing it will be a matter of dealing with raw protocol messages. The underlying control channel will be agnostic to this, since it supports all OpenFlow versions, as long as basic concepts are kept (message header structure). Solving this issue requires us to consider the fundamental guarantees of the OpenFlow specification and implement them in a flexible manner.

**Issue 6: make it possible to build stand-alone applications.** Solving this issue means allowing the code to be used as part of a program, effectively making our architecture behave like a software library that can be embedded rather than an executable solution. This means that our solution will not have a single entry-point of execution, and different parts of it must be easy to embed and reuse as needed.

### 3.2. General architecture overview

We arrived at the following practical guidelines for solving the issues regarding both general SDN implementation issues and OpenFlow specific ones:

1. Our work must be implemented as a library that can be bundled into other software, answering issues 1, 4 and 6.
2. We will use multi-threading in order to make use of the parallelism available in modern CPUs, improving performance. By changing the way multi-threading is deployed, we can also work on answering issue 3.
3. An event-loop library will be used to provide an optimized way of dealing with sockets and other operating system abstractions, making the code easier to port to other platforms, making our work on issues 2 and 4 easier.
4. We want to use object-oriented abstractions in order to allow for easier reasoning for most developers, helping us tackle issues 1 and 3.
5. By completely separating protocol and connectivity layers, issue 5 can be solved.

Figure 1 shows a general outline of how these decisions can be implemented on a real solution.

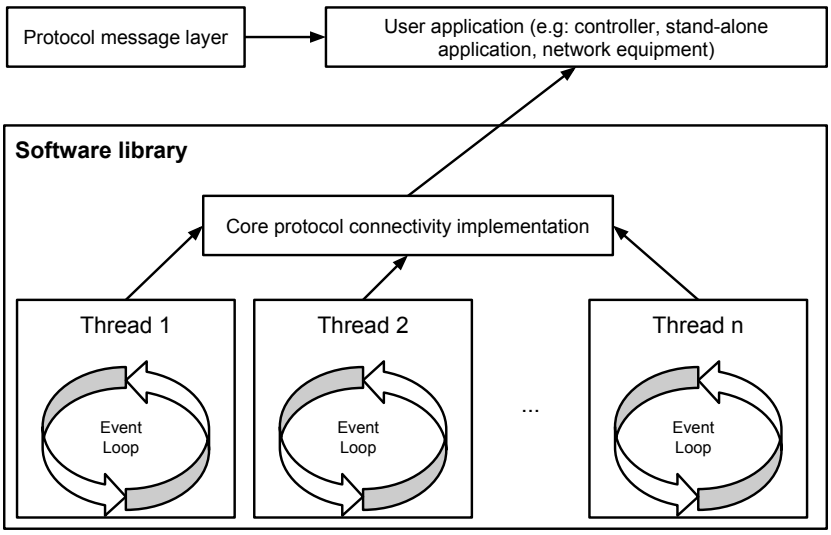



**Figure 1. Arcthiecture overview**

### 3.3. OpenFlow specific design choices

The problem of introducing OpenFlow support into different pieces of software and hardware can be roughly divided in two parts. Figure 1 illustrates how these pieces fit together.

1. The OpenFlow control channel requires the establishment of a network session (in the traditional sense): a transport layer session (TCP/UDP, optionally SSL-protected). This abstraction usually is implemented in the controller itself, and sometimes directly exposed to the application. It involves the connection establishment and protocol handshake, and most importantly, an event abstraction layer that can benefit several types of applications.

2. After this session is established, OpenFlow messages need to be exchanged, and building and parsing them is necessary for the use of the protocol. Different implementations of this are provided in a modular and easily exportable manner in libraries such as OpenFlowJ (written in Java and used in Beacon, OpenDaylight and Floodlight) and libopenflow (used by NOX and POX).

Our architecture manages the connectivity layer necessary for implementing the OpenFlow protocol and can be associated with any OpenFlow message parsing/building library. Our focus is thus on the first part of the problem. Even though our work is oriented towards the OpenFlow protocol, its results can be applied to any protocol that exposes network programmability through a control channel.

Additionally, this connectivity layer will be flexible enough to allow the implementation of different platforms: controllers running several applications, switches connecting to a controller or simply a network application that uses the OpenFlow protocol to perform its functions.

## 4. Implementation Details

The libfluid library as a client-server architecture, in which a controller is a server and a switch is a client. The library is divided in two parts, taking the direction we outlined in Section 3: libfluid_base and libfluid_msg. The first one is that one of most interest to our research, and the second one will be mentioned and used when appropriate.

An overview of the implemented architecture is illustrated in Figure 2, showing the overall architecture of a system that uses libfluid for a controller implementation (YourController). Different clients (OpenFlow switches) will be assigned to multiple threads managed by libfluid_base using a round-robin distribution algorithm (improving this attribution scheme will be part of our future research). Each of these threads will be responsible for reading and writing data to sockets. Whenever data is read, a callback with user-defined behavior is called. Upon this very simple abstraction, all message processing and application workflow has to be implemented. In Figure 2, libfluid_msg is being employed for dealing with protocol specific messaging, and the YourController class implements application workflow.

Another important part in our solution and a core aspect in our research is providing the support for different OpenFlow versions. We decided to leave the responsibility for dealing with each version of the protocol to libfluid_msg. libfluid_base is completely agnostic to OpenFlow versions and message parsing. The user specifies the OpenFlow versions its application or controller will support, and the driver performs the version negotiation, assuming only a very simple OpenFlow header (guaranteed by the specification to never change) to negotiate the version upon connection establishment. The user is then responsible for parsing the messages accordingly (using libfluid_msg or another library).

### 4.1. Event model and IO

A library is used for providing an event loop that runs in each thread, dealing with socket management and IO (libevent [Mathewson and Provos 2011]). A connection's event callbacks are guaranteed to run in a single thread during its lifecycle. A user specifies how many threads it wants to use when initializing the driver, and these threads will be responsible for listening to connection events. The number of threads will be defined by the user, but to avoid scheduling overhead, it should typically match the number of available CPUs. The first thread also has the task of listening for new connections. The user must implement two methods for handling major types of events: message and connection events. Message events happen whenever a message arrives, and connection

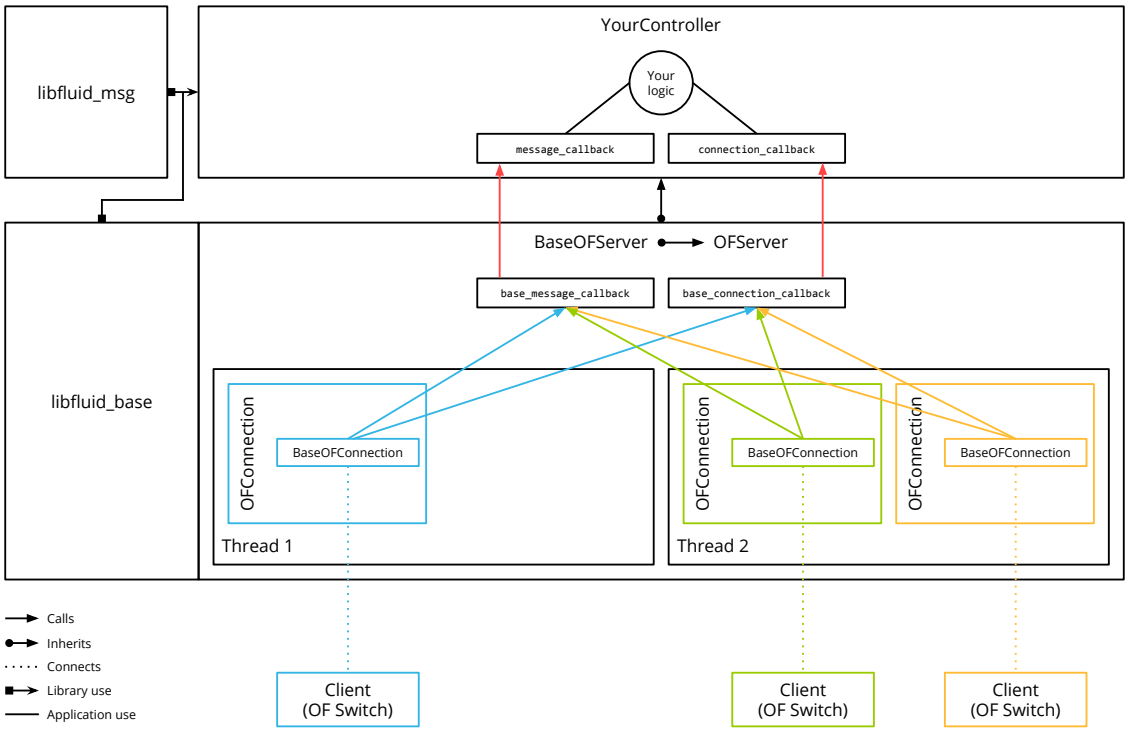



**Figure 2. libfluid architecture**

events when the connection is interrupted, fails or becomes stale. The user needs to implement the two callback methods and call a start method to have a bare-bones OpenFlow controller running.

The way of deploying threading is a core issue to our research and it impacts on design decisions on several of the issues mentioned in Section 3. Instead of the current model, an event queue could be provided to the user. However, this would impose an impact on the performance, since this structure needs to be synchronized, a result similar to the one achieved in Beacon [Erickson 2013]. Since different users of the driver will have different requirements, we did not want to limit the ways it can be used. A synchronized queue of events can still be implemented on top of this multithreaded architecture, so we delegate this responsibility to application developers. We are performing investigations on how this can be improved by using wait-free synchronization queues for lower priority messages [Kogan and Petrank 2011], while still allowing the user to have a low latency, dedicated event treatment directly in the connections threads. This makes it possible to have different applications dealing with events with different priorities.

### 4.2. Programming language choice

The submission requirements listed C/C++ as the preferred languages. We chose to follow this recommendation, since the team was used to these languages. We chose C++ due to its higher level abstractions which make it easier (and clearer) to introduce software-engineering concepts, while keeping a low-level implementation. However, we avoided using advanced C++ features (such as templates, smart pointers, etc.) in order to keep the solution fast, small and portable, reducing the number of classes and the overall complexity of the architecture.

## 5. Demonstration at SBRC 2014

The demonstration will be divided in 5 parts. These parts will be presented directly in the stand to small groups of interested people.

1. We will show libfluid's website, and the resources available.
2. Using the website resources, we will give a quick overview of the architecture, highlighting key points of interest to developers (events, callbacks and message building and parsing). Examples of applications and uses for the driver will be briefly discussed.
3. A quick overview of the installation steps will be shown on a Linux PC (it is a simple and quick installation process).
4. After installing libfluid, we'll show how to write and compile a dummy compiler that doesn't do anything, but keeps connections to OpenFlow switches alive. This example will be coded live, and it's composed of just a few lines of code.
5. Finally, we will talk about a previously written learning switch application and show it running with Mininet. We will quickly talk about how the sample controller and application were built.

Code, documentation, and initial performance benchmarks are available at: `http://opennetworkingfoundation.github.io/libfluid/`

## 6. Conclusion

By recognizing the distinct nature of the different parts in an SDN control protocol, we have identified a few issues in available solutions and possible improvements to advance the state of the art in OpenFlow driver implementations. Our solution, `libfluid`, shows promising results and has been selected as the winning choice by the ONF. We intend to improve the tool by applying a research-oriented approach to solving issues in the current implementation (e.g., event model, protocol version agnosticism and a low-overhead implementation) and also to future work (e.g., formalization of the client-server architecture, higher-level abstractions).

## References

[Cai et al. 2011] Cai, T. Z., Cox, A. L., and Ng, T. E. (2011). Maestro: balancing fairness, latency and throughput in the OpenFlow control plane. Technical report, Rice University Technical Report TR11-07.

[Erickson 2013] Erickson, D. (2013). The Beacon OpenFlow Controller. In *HotSDN*. ACM.

[Feamster et al. 2013] Feamster, N., Rexford, J., and Zegura, E. (2013). The Road to SDN. *Queue*, 11(12):20.

[Foundation 2014] Foundation, O. N. (2014). Open Networking Foundation. `https://www.opennetworking.org/`. [Access: 15-Feb-2014].

[Kogan and Petrank 2011] Kogan, A. and Petrank, E. (2011). Wait-free queues with multiple enqueuers and dequeuers. *ACM SIGPLAN Notices*, 46(8):223–234.

[Mathewson and Provos 2011] Mathewson, N. and Provos, N. (2011). libevent. `http://libevent.org/`. [Access: 15-Feb-2014].

[McKeown et al. 2008] 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*, 38(2):69–74.

# Web2Compile: Uma Web IDE para Redes de Sensores sem Fio

**Augusto Santos[1], Márcio Farias[1], Gabriel Rocha[1], Marina V. Pereira[1], Claudio M. de Farias[1], Tiago C. França[1] e Rafael O. Costa[1]**

[1]Departamento de Ciência da Computação - Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro - RJ - Brazil

***Abstract.*** *This paper presents the Web2Compile WebIDE for Wireless Sensor Networks Application development using TinyOS and NesC. Using Web2Compile, an user is able to generate a sensor image and to simulate their behaviour through TOSSIM simulator only using an updated browser. Web2Compile intends to eliminate the need of complex install procedures required nowadays by Wireless Sensor Networks development environments.*

***Resumo.*** *Este trabalho apresenta a WebIDE Web2Compile para o desenvolvimento de aplicações para redes de sensores sem fio usando TinyOS e NesC. Usando Web2Compile, um usuário é capaz de gerar uma imagem do sensor e simular o seu comportamento através do simulador TOSSIM usando apenas um navegador atualizado. Web2Compile pretende eliminar necessidade de procedimentos de instalação complexos utilizados atualmente por ambientes de desenvolvimento para Redes de Sensores sem fio.*

## 1. Introdução

Os recentes avanços na tecnologia de sistemas micro-eletro-mecânicos, nas comunicações sem fio e na eletrônica digital possibilitaram a construção de sensores de tamanho reduzido, os quais, em grande número, permitem monitorar variáveis físicas e ambientais como temperatura, umidade, níveis de ruído e movimento de objetos com elevado grau de precisão [Culler et al. 2004]. Ao conjunto desses dispositivos dá-se o nome Redes de Sensores sem Fio (RSSFs). Três necessidades para o crescimento das RSSFs são: (i) a produção de sensores em larga escala, reduzindo o seu custo, (ii) maiores investimentos no desenvolvimento tecnológico desses dispositivos, levando a novas melhorias e capacidades e (iii) a falta de mão de obra capaz de produzir aplicações para esses dispositivos [Loureiro et al. 2003].

Apesar de ser um campo promissor, existem algumas dificuldades para o ensino de desenvolvimento de aplicações para RSSF. Pode-se destacar, principalmente: (i) o alto custo dos sensores sem fio atualmente (*hardware*) e (ii) complexidade da instalação e manutenção dos ambientes de desenvolvimento. Por exemplo, os sensores MICAz da Memsic [Su and Alzaghal 2008] necessitam de um sistema operacional rudimentar, como o TinyOS [Hill et al. 2000, Delicato 2005]. Para instalação do TinyOS é necessária a configuração do ambiente de desenvolvimento na máquina do usuário, que é um procedimento complexo.

O objetivo deste trabalho é prover uma solução onde desenvolvedores evitem executar complexas instalações e atualizações em seus computadores pessoais ou em laboratórios. Com isso, acelerar o aprendizado do desenvolvimento de aplicações para RSSF.

Nesse contexto é apresentado a Web2Compile, uma WebIDE que permite o desenvolvimento de aplicações para RSSF através de um navegador de internet. O objetivo de uma WebIDE é transformar a IDE (*Integrated Development Environment*) tradicional em um conjunto de serviços acessíveis integrados, por intermédio de um navegador web, e enriquecido com as tecnologias da Web 2.0 [Murugesan 2007].

O diferencial da Web2Compile é permitir a todos os usuários desenvolver aplicações para RSSF com uma interface simples bastando dispor de um navegador de internet atualizado. A ferramenta desenvolvida destina-se a usuários que: (i) não tem tempo ou experiência para realizar uma configuração de ambiente (como por exemplo, em sala de aula), (ii) não tem um bom enlace de dados para realizar o download do sistema operacional, (iii) não possui um dispositivo sensor, como o *hardware* sensor MICAz.

O restante deste artigo está organizado da seguinte forma. Na seção 2 são destacados os principais trabalhos relacionados. Na seção 3 é apresentada a ferramenta proposta (Web2Compile) e a sua arquitetura de software. Na seção 4 são apresentados os experimentos realizados e a análise dos resultados obtidos nos testes. Na seção 5 é apresentada uma breve descrição do demo que será apresentado. Finalmente, na seção 6 são tecidas as considerações finais e as propostas de trabalhos futuros.

## 2. Trabalhos Relacionados

Aplicações desenvolvidas e que seguem o paradigma da web 2.0, como é o caso das WebIDEs, já vem sendo desenvolvidas a algum tempo. A grade maioria das WebIDEs possui o único objetivo de facilitar a vida do desenvolvedor e dentre as particularidades mais interessantes, propõem-se a disponibilizar: (i) IDE, com destaque na sintaxe da linguagem particular; (ii) compilador interno para execução do código online; (ii) servidor com autenticação para usuários armazenarem seus programas para uso posterior. Dentre estas ferramentas usufruídas através de navegadores de internet, podemos destacar Arvue [Aho et al. 2011], WWWorkspace [Ryan 2007] , eLuaproject e eXo Cloud IDE.

No Arvue [Aho et al. 2011] permite-se o desenvolvimento simples, publicação de aplicações destinadas à web e a hospedagem como um serviço. Implementações feitas com a aplicação são criadas no navegador utilizando uma interface de projeto (*interface designer*) e um editor de código integrado. Os desenvolvimentos são armazenados em um sistema de controle de versões fornecidas pelo próprio Arvue e que podem ser publicadas na nuvem (*cloud*). A ferramenta, foi desenvolvida utilizando-se para implementação um *framework* de código aberto para o desenvolvimento de Java Servlet baseado em aplicações web, denominado Vaadin. Segundo [Aho et al. 2011], sua intenção foi apresentar um editor útil para uma única tarefa: criação e publicação de aplicativos Vaadin de pequeno porte para *iPhone* baseado na web. Diferentemente da estrutura de desenvolvimento Arvue, este trabalho não tem o perfil de prover uma interface de desenvolvimento, mas semelhantemente, geram-se para o download no sensor (*hardware*), programas codificados obtidos através da saída da compilação. Vale ressaltar que Arvue é uma implementação não direcionada para RSSF.

WWWorkspace [Ryan 2007] é baseado em um ambiente de desenvolvimento integrado Java voltado para web construído em cima do Eclipse. A IDE do aplicativo permite que seus usuários carreguem seu espaço de trabalho e de código a partir de qualquer computador tradicional com um navegador de internet disponível. Vale notar que diferente das

outras ferramentas, para utilização do WWWorkspace, o usuário deve fazer o download do programa no site do desenvolvedor, que contém todos os arquivos necessários para executar o aplicativo, incluindo um servidor Jetty, plug-ins Eclipse e biblioteca DOJO javascript. Diferentemente, neste trabalho, ao contrário do WWWorkspace, possui-se um servidor atuando diretamente como compilador das implementações e este exime o usuário de fazer downloads de plug-ins da internet. Apoiado na mesma filosofia da ferramenta WWWorskpace, este trabalho também se diferencia por estar relacionado estritamente ao desenvolvimento de aplicações para RSSF.

O Projeto eLua (eLua *Project*) tem como objetivo introduzir a linguagem de programação Lua para o mundo do desenvolvimento de software embarcado. Lua é o exemplo perfeito de uma linguagem minimalista, mas totalmente funcional. Embora geralmente anunciada como uma "linguagem de script"e usada em conformidade especialmente na indústria do jogo, ela também é plenamente capaz de executar programas independentes. Sua pequena exigência de recursos torna a linguagem Lua adequada para um grande número de famílias de microcontroladores. Assim como outras WebIDE o objetivo do projeto é ter um ambiente de desenvolvimento totalmente funcional no próprio microcontrolador, sem a necessidade de instalar um conjunto de ferramentas específicas e configurações de ambiente de programação. Diferentemente, este trabalho não tem o objetivo de desenvolver aplicações para aplicativos embarcados, somente para RSSF.

eXo Cloud IDE é um ambiente de desenvolvimento baseado na web que permite o desenvolvimento colaborativo de aplicações que podem ser implantadas diretamente em um Ambiente Heroku PaaS (*Plataform as a Service*) [Beimborn et al. 2011]. Heroku é uma plataforma baseada na Web, muito poderosa para as linguagens Ruby, JavaScript, incluindo o Node.js e recentemente, foi incluído o apoio a aplicações Java voltadas para web. A implantação diretamente dentro de um ambiente PaaS permite migração rápida a partir da fase de desenvolvimento para a implantação. eXo Cloud IDE também contém um editor em tempo real de colaboração para utilização com até cinco pessoas. A eXo Cloud não possui o objetivo de desenvolver aplicações para RSSF.

## 3. Web2Compile

A Web2Compile é uma WebIDE (código disponível com documentação em http://www.labnet.nce.ufrj.br/web2compile.html), voltada para o desenvolvimento de aplicações para RSSFs. A arquitetura física deste projeto pode ser vista na Figura 1. Seguindo a arquitetura cliente-servidor e baseado na web como visto em [Chen and Teng 2002] o esquema arquitetural consiste nos seguintes componentes: (i) computadores clientes; e (ii) e um servidor web.

No servidor Web está hospedado o sistema operacional para RSSF. No servidor web, o ambiente de desenvolvimento para RSSF está configurado para executar simulações e compilações. O usuário terá opção de efetuar o desenvolvimento da sua aplicação na máquina cliente ou na própria página do sistema, como pode ser visto na Figura 1. Optando-se pelo desenvolvimento na máquina cliente, o usuário poderá escolher por fazer o *upload* dos arquivos necessários dependendo da aplicação que deseja.

Os dispositivos clientes tem a principal função de fazer o carregamento (*upload*) do código para o servidor web dedicado. Nesta primeira parte do processo, como pode ser visto na Figura 2, o cliente tem a possibilidade de escolher dentre duas possibilidade

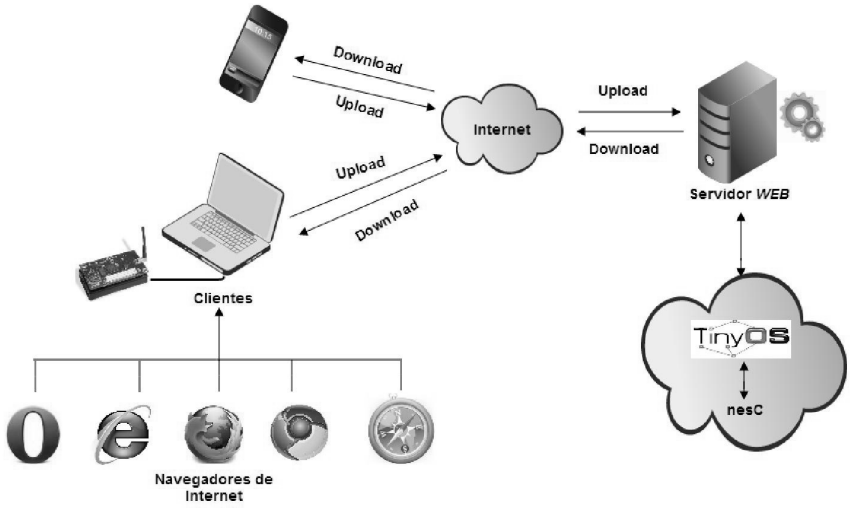



**Figura 1. Arquitetura Física**

de resposta para sua compilação: (i) somente simulação; ou (ii) download do código compilado.

O código será executado logo após que todos os arquivos necessários à compilação, sejam enviados (*upload*) ao compilador localizado no servidor web. Após isto, gerar-se-ão as imagens necessárias para que seja possível realizar o envio para o *hardware* do sensor.

A sequência de funcionamento da Web2Compile se dá da seguinte maneira: a partir da página carregada em seu navegador de internet, o usuário poderá fazer a seleção dos arquivos de acordo com as duas opções disponibilizadas pela aplicação, que são: (i) Somente a simulação; ou (ii) Somente o *deploy*.

Optando-se pela simulação, deverão ser selecionados os arquivos necessários e posteriormente, estes deverão ser carregados (*upload*) para o compilador hospedado no servidor. No momento da compilação, se todos os arquivos selecionados estiverem corretamente implementados de acordo com seu desenvolvimento, haverá sucesso na compilação e estes posteriormente, deverão ser disponibilizados para download pela própria aplicação, a partir de um botão indicador de download. Em contrapartida, se não houver sucesso na compilação, novamente, será necessário revisar a implementação desenvolvida para que seja a feita a correção dos erros encontrados pelo compilador e, por conseguinte, realizar a seleção dos arquivos a serem compilados. Optando-se pelo *deploy*, deve-se fazer a seleção de arquivos necessários ao mesmo. Neste caso, os arquivos selecionados passarão pelo mesmo processo supracitado no caso da Simulação. No entanto, neste caso com o sucesso na compilação, gerar-se-á a construção (*build*) da imagem dos arquivos. Estes arquivos gerados, poderão se descarregados (*download*), assim como a simulação, para o computador. A única diferença é que a imagem gerada a partir da construção do compilador poderá ser carregada para o sensor, a partir da configuração correta do *hardware* no computador.

No que diz respeito a arquitetura lógica, o Web2Compile é composto de 3 classes. A classe Gerente se que controla a página principal e recebe todas as requisições feitas

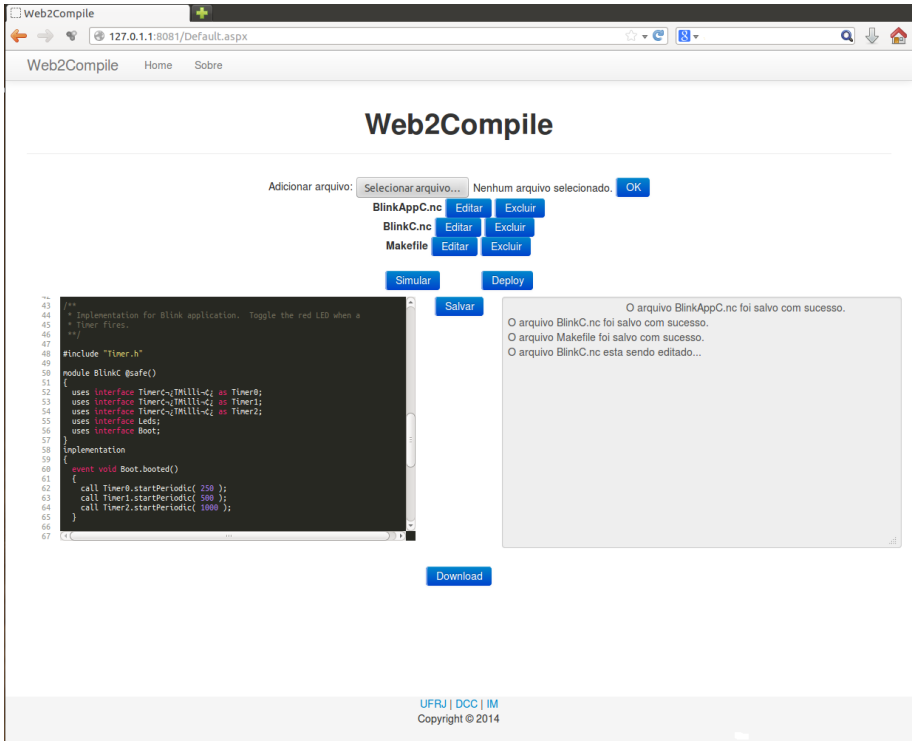



**Figura 2. Interface Web2Compile**

por ela. A classe ArquivoBO é a classe responsável por ações de pastas e arquivos. Por fim a classe Sensor é a responsável por interagir com o sistema Operacional TinyOS.

## 4. Implementação e Testes

Nesta seção são descritos: (i) as tecnologias utilizadas, assim como as ferramentas de desenvolvimento da WebIDE Web2Compile e os *frameworks*; (ii) as métricas utilizadas para avaliar o sistema e (iii) finalmente descrevem-se os testes de escalabilidade.

### 4.1. Tecnologias utilizadas na Implementação do Web2Compile

O objetivo do sistema Web2Compile consiste no provimento de um sistema baseado na web, de modo que sua arquitetura é baseada no modelo cliente-servidor. Optou-se pela linguagem de programação C#, utilizando a IDE MonoDevelop, versão 2.8.0, e o sistema Operacional Debian 7.0, versão *Wheezy*. Os testes foram realizados com execução a partir de um servidor com processador Intel i5-2500k, quatro núcleos (*quad-core*) a 3.3 Ghz e 8Gb de memória RAM.

Atualmente no protótipo desenvolvido, o Web2Compile suporta aplicações desenvolvidas na linguagem de programação nesC [David et al. 2003], para o sistema operacional TinyOS [Hill et al. 2000]. Para a edição de código dentro da plataforma, utilizou-se a ferramenta ACE (http://ace.c9.io/#nav=about).

Para a simulação, utilizamos o TOSSIM [David et al. 2003], um simulador de eventos discretos, desenvolvido para a simulação do TinyOS cuja arquitetura é baseada em componentes para RSSF. TOSSIM vêm com as características do microcontrolador AVR ATmega128 pré-programadas no simulador para atuar como uma abstração de *hard-*

*ware* virtual para o sensor MICAZ, utilizado neste projeto. O TOSSIM realiza a construção diretamente dos componentes do TinyOS e tem a capacidade de fazer a ligação de componentes, utilizando todas as interfaces necessárias.

### 4.2. Métricas

Foram utilizadas as seguintes métricas de medição: (i) Número de conexões simultâneas: quantidade de acessos recebidos pelo servidor simultaneamente para a execução de um processo (simulação ou compilação); (ii) Tempo médio, a média do tempo que leva para executar as requisições de um número "X" de conexões simultâneas; (iii) Memória consumida: quantidade de memória RAM alocada para a execução de "X" conexões simultâneas e (iv) Taxa de processamento: porcentagem disponibilizada pela CPU para a execução de um ou mais processos simultâneos.

### 4.3. Testes de Escabilidade

Os Testes de Escalabilidade foram utilizados com o intuito de verificar se o Web2compile mantém o desempenho dado o aumento da quantidade de conexões simultâneas de simulações da aplicação Blink. Foram utilizadas 1, 3, 5, 7 e 10 conexões simultâneas, onde cada computador está conectado ao servidor através de um switch.

Em relação ao tempo médio, em segundos, de uma execução em cada situação anteriormente apresentada, podemos observar na Figura 3, abaixo que, embora quando exista apenas uma execução, o tempo médio foi bem abaixo dos demais. Em relação à execução de um processo com três execuções simultâneas, foram 2,5 segundos mais rápido. No geral, os tempos de execução com 3, 5, 7 e 10 conexões simultâneas apresentaram, em média, uma aumento de 0,5 segundos (no tempo de execução, de um para outro). Com isso, pode-se concluir que, mesmo com um o número de conexões simultâneas elevando-se, o tempo de conclusão da simulação não teve uma variação significativa.

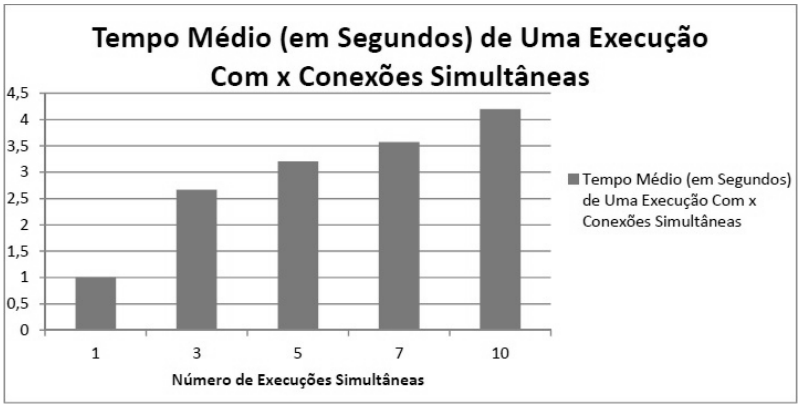



**Figura 3. Tempo Médio (em segundos) de uma execução**

Agora, em relação a porcentagem de utilização da CPU em relação ao mesmo número de conexões simultâneas, observou-se que a utilização da CPU aumentou de forma linear, como esperado, de acordo com o aumento das simulações simultâneas. Vale salientar que todas as *threads* foram criadas em um mesmo núcleo do processador, por isso, a elevada taxa de utilização do núcleo do processador. Este fato decorre da não sobrecarga do servidor se, em algum momento, caso ocorra um elevado número de conexões

simultâneas, faça-se com que o núcleo utilizado para processamento das threads fique alternando entre as mesmas. Em decorrência disso, observou-se que, com uma conexão apenas, o processador alocando requisito apenas para uma conexão, a taxa de processamento chegou a 40% da capacidade de um núcleo e, em decorrência da elevação das conexões simultâneas, até 10 conexões, a taxa de processamento alcançou 100% do núcleo, conforme pode ser visto na Figura 4.

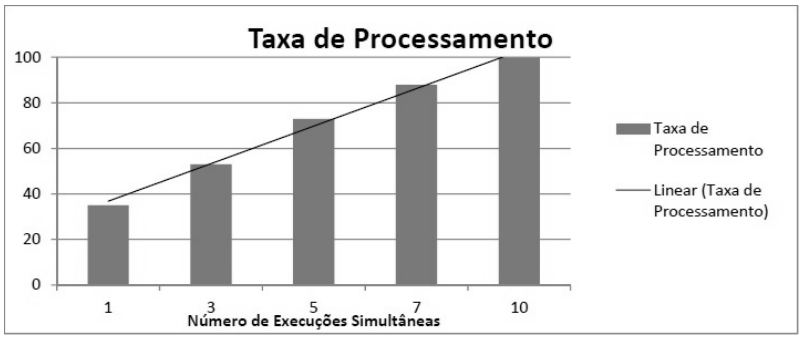



**Figura 4. Taxa de Processamento**

Com relação ao consumo de memória, pode-se observar que no teste realizado com uma única conexão apenas, o consumo alcançou 10 Mb de memória RAM para a execução da simulação pelo TOSSIM, devido aos processos necessários para realização da mesma. Quase linearmente, como esperado, a quantidade de memória consumida foi aumentando, até o momento que, em 10 execuções simultâneas, alcançou-se a quantidade de 122 Mb de memória RAM.

## 5. Descrição da Demonstração

A demonstração do sistema será feita com a construção de uma aplicação de monitoramento da temperatura do ambiente formada por dez dispositivos de comunicação Micaz, ligados a uma estação base. Os sensores coletarão os dados a cada 5 segundos e enviarão para a estação base onde será feita a média aritmética dos dados coletados. Essa aplicação será construída dentro da própria WebIDE. Será realizada então a implantação em Sensores reais (nós micaz) e no ambiente simulado com o TOSSIM.

Além disso, também estarão disponíveis outras aplicações padrões do TinyOS para a instalação e teste da plataforma. Ainda estará disponível uma máquina para que os conferencistas possam desenvolver sua própria aplicação dentro da IDE.

## 6. Conclusão

Este trabalho apresentou uma proposta de fomento ao desenvolvimento de aplicações relacionadas às RSSF e visa incentivar novos desenvolvedores e entusiastas ao desenvolvimento de aplicações para a tecnologia. No sistema Web2Compile, é estritamente desnecessário o *download* de aplicações para instalação do ambiente de desenvolvimento, muitas vezes complexas, além de configurações de ambientes em sistemas operacionais, onde se demanda experiência para que se tenha uma ferramenta apta ao desenvolvimento de aplicações para RSSF. Vale ressaltar também que este trabalho, exime ao usuário, fazer a compilação de seus códigos em IDEs locais.

Quanto aos trabalhos futuros, pode-se estudar mais profundamente aplicações de sistemas vislumbradas a partir da temática das RSSF e das WebIDEs. Algumas aplicações inerentes ao tema que podem vir a ser fomentadas como estudo são: (i) implementar o *Remote Flashing*, que possibilita que a imagem seja carregada em um sensor conectado a uma porta usb de um computador tradicional através do browser; (ii) utilizar outros sistemas operacionais para RSSF como o Contiki [Dunkels et al. 2006] e (iii) inserir outros tipos de ambiente de simulação que não o TOSSIM.

## Referências

Aho, T., Ashraf, A., Englund, M., Katajamäki, J., Koskinen, J., Lautamäki, J., Nieminen, A., Porres, I., and Turunen, I. (2011). Designing ide as a service. *Communications of Cloud Software*, 1(1).

Beimborn, D., Miletzki, T., and Wenzel, S. (2011). Platform as a service (paas). *Business & Information Systems Engineering*, 3(6):381–384.

Chen, P. M. and Teng, W.-G. (2002). Collaborative client-server architectures in the web-based viewing scheme. *Sheraton Waikiki Hotel Honolulu, Hawaii, USA*.

Culler, D., Estrin, D., and Srivastava, M. (2004). Guest editors' introduction: overview of sensor networks. *Computer*, 37(8):41–49.

David, G., Philip, L., David, C., and Eric, B. (2003). Nesc 1.1 language reference manual. *Chicago, Mayo*.

Delicato, F. C. (2005). Middleware baseado em serviços para redes de sensores sem fio. *PhD Thesis, Federal University of Rio de Janeiro, Brazil*.

Dunkels, A. et al. (2006). The contiki operating system. *Web page. Visited Oct*, 24.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. (2000). System architecture directions for networked sensors. In *ACM SIGOPS operating systems review*, volume 34, pages 93–104. ACM.

Loureiro, A. A., Nogueira, J. M. S., Ruiz, L. B., de Freitas Mini, R. A., Nakamura, E. F., and Figueiredo, C. M. S. (2003). Redes de sensores sem fio. In *Simpósio Brasileiro de Redes de Computadores (SBRC)*, pages 179–226.

Murugesan, S. (2007). Understanding web 2.0. *IT professional*, 9(4):34–41.

Ryan, W. (2007). Web-based java integrated development environment. *BEng thesis, University of Edinburgh*.

Su, W. and Alzaghal, M. (2008). Channel propagation measurement and simulation of micaz mote. *W. Trans. on Comp.*, 7(4):259–264.

# Índice por Autor

# Índice por Autor

# Índice por Autor

# Índice por Autor

Promoção

Organização

Patrocínio

Microsoft Research

Apoio

ISSN 2177-496X