31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

6 a 10 de Maio de 2013
Brasília-DF

Anais
Trilha Principal e Salão de Ferramentas

Editora
Sociedade Brasileira de Computação (SBC)

Organizadores
Carlos André Guimarães Ferraz (UFPE)
José Augusto Suruagy Monteiro (UFPE)
Jacir Luiz Bordim (UnB)
Rafael Timóteo de Sousa Júnior (UnB)
William Ferreira Giozza (UNB)

Realização
Universidade de Brasília (UnB)

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

Capa: João Paulo Andrade Lima
Produção Editorial: Eduardo Adilio Pelinson Alchieri
Marcos Fagundes Caetano

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 (31 : 2013: Brasília, DF)

Anais / 31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos; organizado por Jacir Luiz Bordim... [et al.] — Porto Alegre: SBC, c2013

1161 p.

SBRC 2013
Realização: Universidade de Brasília
ISSN: 2177-496X

1. Redes de Computadores - Congressos. 2. Sistemas Distribuídos - Congressos. I. Bordim, Jacir Luiz. 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)

Diretor Administrativo
Luciano Paschoal Gaspary (UFRGS)

Diretor de Finanças
Luci Pirmez (UFRJ)

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

Diretora de Educação
Mirella Moura Moro (UFMG)

Diretora de Publicações
Karin Koogan Breitman (PUC-Rio)

Diretora de Planejamento e Programas Especiais
Ana Carolina Brandão Salgado (UFPE)

Diretora de Secretarias Regionais
Thais Vasconcelos Batista (UFRN)

Diretor de Divulgação e Marketing
Edson Norberto Cáceres (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 Araújo Macêdo (UFBA)

Promoção

Conselho

Mandato 2011-2015

Ariadne Carvalho (UNICAMP)
Carlos Eduardo Ferreira (IME - USP)
José Carlos Maldonado (ICMC - USP)
Luiz Fernando Gomes Soares (PUC-Rio)
Marcelo Walter (UFRGS)

Mandato 2009-2013

Flávio Rech Wagner (UFRGS)
Itana Maria de Souza Gimenes (UEM)
Jacques Wainer (UNICAMP)
Silvio Romero de Lemos Meira (UFPE)
Virgílio Almeida (UFMG)

Suplentes – 2011-2013

César A. F. De Rose (PUCRS)
Maria Izabel Cavalcanti Cabral (UFCG)
Renata Mendes de Araújo (UNIRIO)
Ricardo Augusto da Luz Reis (UFRGS)

Laboratório Nacional de Redes de Computadores (LARC)

Diretoria

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 e UFS.

Realização

Comitê de Organização

Coordenação Geral

Jacir Luiz Bordim (UnB)

Rafael Timóteo de Sousa Jr (UnB)

William F. Giozza (UnB)

Coordenação do Comitê de Programa

Carlos André Guimarães Ferraz (UFPE)

José Augusto Suruagy Monteiro (UFPE)

Coordenação de Palestras e Tutoriais

Nelson Luis Saldanha da Fonseca (UNICAMP)

Coordenação de Painéis e Debates

Lisandro Zambenedetti Granville (UFRGS)

Realização

Coordenação de Minicursos

Joni da Silva Fraga (UFSC)

Coordenação de Workshops

Francisco Vilar Brasileiro (UFCG)

Coordenação do Salão de Ferramentas

Gustavo Sousa Pavani (UFABC)

Comitê Consultivo

Artur Ziviani (LNCC)
Bruno Schulze (LNCC)
Célio Vinícius Neves de Albuquerque (UFF)
Dorgival Guedes (UFMG)
Elias Procópio Duarte Jr (UFPR)
José Ferreira de Rezende (UFRJ)
Jussara Almeida (UFMG)
Ronaldo Alves Ferreira (UFMS)

Realização

Organização Local

Aletéia Patrícia Favacho de Araújo (UnB)
André Costa Drummond (UnB)
Divanilson Rodrigo de Sousa Campelo (UnB/UFPE)
Edna Dias Canedo (UnB)
Eduardo Adilio Pelinson Alchieri (UnB)
Flávio Elias Gomes de Deus (UnB)
Jacir Luiz Bordim (UnB)
Marcos Fagundes Caetano (UnB)
Maristela Terto de Holanda (UnB)
Priscila América Solís Mendez Barreto (UnB)
Rafael Timóteo de Sousa Junior (UnB)
William Ferreira Giozza (UnB)

Mensagem dos Coordenadores Gerais

Sejam bem vindos ao **31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos** (**SBRC 2013**), realizado pela primeira vez na Capital Federal, Brasília, DF.

Manter o SBRC na posição do mais importante evento científico nacional em Redes de Computadores e Sistemas Distribuídos e um dos maiores da área de Informática no país é um desafio importante assumido por nós da Universidade de Brasília. Além de estimular a troca de ideias e experiências, a discussão de temas de pesquisa avançados, como é sua tradição, o **SBRC**, em sua **trigésima primeira edição**, está organizado para proporcionar uma interação proveitosa entre estudantes, professores, pesquisadores e profissionais com interesses na área do evento.

O **SBRC 2013** está com uma programação bastante variada e com alta qualidade técnica. O número de **sessões técnicas** na trilha principal foi ampliado para 24 de modo a permitir a apresentação de **73 artigos completos**, cobrindo uma ampla lista de tópicos relevantes e atuais nas áreas de Redes de Computadores e Sistemas Distribuídos. O Salão de Ferramentas apresenta **11 ferramentas** que serão debatidas e demonstradas em um fórum específico. Este ano, os estudantes participantes do SBRC serão contemplados com a oferta de **7 minicursos** versando sobre temas atuais normalmente não abordados nas grades curriculares. A programação do SBRC 2013 inclui ainda **4 palestras convidadas** proferidas por pesquisadores de alto renome internacional e **3 painéis** abordando temas atuais e polêmicos. Além dessas atividades tradicionais, o SBRC 2013 abriga a realização de **9 workshops** que ocorrem em paralelo com o evento: XVIII Workshop de Gerência e Operação de Redes e Serviços (WGRS), XIV Workshop da Rede Nacional de Ensino e Pesquisa (WRNP), XIV Workshop de Testes e Tolerância a Falhas (WTF), XI Workshop de Computação em Grade e Aplicações (WCGA), IX Workshop de Redes P2P, Dinâmicas, Sociais e Orientadas a Conteúdo (WP2P+), IV Workshop de Pesquisa Experimental na Internet do Futuro (WPEIF), III *Workshop on Autonomic Distributed Systems* (WoSiDA), III Workshop de Redes de Acesso em Banda Larga (WRA) e o I ***Workshop of Communication in Critical Embedded Systems* (WoCCES)**. Por fim, o SBRC 2013 promove um **Fórum Governo** no intuito de reunir profissionais, acadêmicos e gestores com a finalidade de discutirem os principais desafios e as soluções avançadas de redes e sistemas distribuídos para as diversas esferas governamentais.

O **SBRC 2013** retoma a discussão sobre os caminhos da pesquisa e da inovação em redes de computadores e sistemas distribuídos no País. Em particular conta com um **painel** que visa discutir os **grandes desafios da área** com representantes da indústria e do governo. Além disso, para homenagear aqueles que contribuíram significativamente para o desenvolvimento da pesquisa e o fortalecimento da comunidade científica nas áreas de Redes de Computadores e Sistemas Distribuídos no Brasil, o SBRC 2013 mantém o **Prêmio Destaque SBRC**, nesta edição, homenageando a professora Liane Margarida Rockenbach Tarouco, da Universidade Federal do Rio Grande do Sul, em reconhecimento às suas contribuições científicas e aos serviços prestados em benefício do SBRC e de sua comunidade.

A organização de um evento do porte e da importância do SBRC só pode ser realizada se contar com a ajuda de uma equipe qualificada e dedicada.

Mensagem dos Coordenadores Gerais

Gostaríamos de agradecer aos membros dos **Comitês de Organização Geral e Local** pelo trabalho voluntário de excelente qualidade e pelo apoio incansável durante as várias etapas da organização deste evento. Somos muito gratos também ao apoio da **SBC** e do **LARC**, promotores do SBRC. Em particular agradecemos aos membros do **Conselho Consultivo do SBRC** e da coordenação da **Comissão Especial de Redes de Computadores e Sistemas Distribuídos** da SBC, pela confiança depositada e pelo suporte financeiro inicial indispensável para a realização do SBRC 2013. Gostaríamos de agradecer ainda ao **Comitê Gestor da Internet no Brasil**, às agências governamentais de fomento – **CNPq** e **CAPES**, e aos **patrocinadores** por reconhecerem e valorizarem a importância do SBRC como fórum de divulgação da pesquisa e inovação em redes de computadores e sistemas distribuídos. Por fim, nossos agradecimentos aos **Departamentos de Engenharia Elétrica** e de **Ciência da Computação** da **UnB**, por apoiarem nossa dedicação para viabilizar a realização deste evento.

Em nome do Comitê Organizador do SBRC 2013, desejamos a todos uma semana bastante produtiva e agradável.

Jacir Luiz Bordim
Rafael Timóteo de Sousa Junior
William Ferreira Giozza
Coordenadores Gerais do SBRC 2013

Mensagem dos Coordenadores do Comitê de Programa

Em sua 31ª edição, o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) se mantém como o principal evento acadêmico da área no país. Em 2013, o SBRC recebeu, mais uma vez, grande número de submissões: foram 245 artigos completos. Deste total, 73 artigos foram aceitos para publicação e apresentação no evento, resultando em uma taxa de aceitação de pouco menos de 30%. Os artigos aceitos cobrem uma grande diversidade de temas, como Internet do futuro, computação nas nuvens, redes de sensores, redes sem fio, redes veiculares, operação e gerência de redes, entre outros. Os trabalhos foram submetidos por grupos de pesquisa atuantes em todas as regiões do país, bem como do exterior.

O processo de avaliação e seleção de artigos foi composto de várias etapas. Na primeira etapa, cada artigo recebeu pelo menos 3 revisões independentes. Em seguida, houve uma etapa de discussão entre os revisores. Em uma terceira etapa, os revisores puderam considerar comentários dos autores através do mecanismo de *rebuttal*. Neste ano tivemos a experiência de, em vez de termos uma reunião presencial do Comitê de Programa (CP), demos a oportunidade a todos os membros do CP de opinar (virtualmente) sobre as faixas de status dos artigos, quais sejam: claramente aceitos, zona “cinza” e claramente rejeitados. Finalmente, um grupo de voluntários do CP (26) avaliou a coerência das revisões, das recomendações e as réplicas dos autores dos artigos da zona “cinza” para a decisão sobre a lista final dos artigos aceitos. Além disso, a avaliação dos artigos submetidos por nós (da coordenação do Comitê de Programa) foi conduzida por membros anônimos, e neste sentido gostaríamos de agradecer ao Prof. Francisco Brasileiro (UFCG) pela coordenação do processo. Este processo de avaliação e seleção rigoroso envolveu uma grande quantidade de trabalho e dedicação dos membros do Comitê de Programa (120), além dos demais revisores (165), a quem agradecemos a contribuição fundamental. Acreditamos que o programa técnico deste ano apresenta a alta qualidade tradicional do SBRC.

Agradecemos aos Coordenadores Gerais, Jacir Luiz Bordim, Rafael Timóteo de Sousa Júnior e William Ferreira Giozza (UnB), pelo convite para coordenar o Comitê de Programa do SBRC 2013 e por todo o suporte recebido durante a construção do programa.

A todos um SBRC 2013 muito proveitoso!

Carlos André Guimarães Ferraz
José Augusto Suruagy Monteiro
Coordenadores do Comitê de Programa

Mensagem do Coordenador do Salão de Ferramentas

O Salão de Ferramentas do SBRC é um fórum que congrega desenvolvedores e pesquisadores que apresentam princípios de arquitetura, implementação e avaliação prática de sistemas e ferramentas em redes e sistemas distribuídos. Em seus 12 anos de existência o evento tem cumprido um papel importante ao oferecer um espaço para apresentação da produção tecnológica e de resultados práticos derivados da pesquisa realizada pela comunidade.

Neste ano, 19 trabalhos foram submetidos e avaliados pelo Comitê de Programa do Salão de Ferramentas formado por 31 pesquisadores associados a universidades e instituições usuárias ou produtoras de tecnologia. Agradeço a todos os membros do comitê de programa que doaram seu tempo para realizar um cuidadoso trabalho de revisão que permitiu selecionar os 11 trabalhos que apresentaram o maior nível de qualidade para apresentação no Salão de Ferramentas do SBRC 2013.

Tenho a certeza que o evento será uma grande oportunidade não apenas para a divulgação dos trabalhos selecionados, mas também para o seu próprio aprimoramento.

Parabenizo a todos os autores dos trabalhos submetidos, pois é justamente devido à excelente qualidade destas submissões que teremos um grande evento. Convido a todos que prestigiem os autores durante as apresentações dos trabalhos tanto nas sessões técnicas como na sessão de demonstração.

Agradeço ainda todo apoio e confiança do Coordenador Geral do SBRC 2013 Prof. William Ferreira Giozza (UnB), que foi um grande facilitador em todo o processo.

Finalmente deixo a minha homenagem a todos que participaram diretamente e indiretamente para a evolução e continuidade do Salão de Ferramentas durante esses 12 anos.

Desejamos a todos um ótimo Salão de Ferramentas no SBRC 2013!

Gustavo Sousa Pavani
Coordenador do Salão de Ferramentas

Comitê de Programa do SBRC 2013

Aldri dos Santos (UFPR)
Alexandre Sztajnberg (UERJ)
Alfredo Goldman (USP)
Aloysio de Castro Pinto Pedroza (UFRJ)
Altair Santin
Ana Paula Couto da Silva (UFJF)
André Soares
Andre Aquino
Anelise Munaretto (UTFPR)
Antônio Abelém (UFPA)
Antônio Tadeu Azevedo Gomes (LNCC)
Antonio Alfredo Ferreira Loureiro (UFMG)
Antonio Rocha (UFF)
Artur Ziviani (LNCC)
Bruno Schulze (LNCC)
Carlos Ferraz (UFPE)
Carlos Goulart (UFV)
Carlos Kamienski (UFABC)
Carlos Alberto Vieira Campos (UNIRIO)
Carmelo Bastos-Filho
César Melo (UFAM)
Celso Hirata
Cesar Marcondes
Christian Esteve Rothenberg (CPqD)
Cintia Margi
Cristina Murta (CEFET-MG)
Daniel Figueiredo (UFRJ)
Daniel Menasche
Danielo G. Gomes (UFC)
Dênio Mariz Sousa (IFPB)
Djamel Fawzi Hadj Sadok (UFPE)
Dorgival Guedes (UFMG)
Edmundo de Souza e Silva
Edmundo Madeira (UNICAMP)
Eduardo Bergamini (INPE)
Eduardo Cerqueira (UFPA)
Eduardo Nakamura (FUCAPI/UFAM)
Eleri Cardozo
Elias P. Duarte Jr.
Fabíola Greve (UFBA)
Fabio Costa (UFG)
Fabio Kon (USP)
Fatima Duarte-Figueiredo
Fábio Luciano Verdi
Fernando Dotti
Flavia Delicato
Francisco Brasileiro (UFCG)
Geraldo Robson Mateus (UFMG)
Gustavo Figueiredo
Gustavo Pavani
Hélio Guardia
Humberto Marques Neto (PUC Minas)
Igor Moraes (UFF)
Jacir Bordim (UnB)
Jó Ueyama (USP)
Jean-Marie Farines (UFSC)
Joaquim Celestino Júnior (UECE)
Joberto Martins (UNIFACS)
Joni da Silva Fraga (UFSC)
José Augusto Suruagy Monteiro (UNIFACS)
José De Souza
José Ferreira de Rezende
Jose Marcos Nogueira (UFMG)
Jussara Almeida (UFMG)
Keiko Fonseca (UTFPR)
Kelvin Dias (UFPE)
Kleber Cardoso (UFG)
Lau Cheuk Lung (UFSC)
Leobino Sampaio
Liane Tarouco (UFRGS)
Lisandro Zambenedetti Granville (UFRGS)
Luci Pirmez (UFRJ)
Luciano Paschoal Gaspary (UFRGS)

Comitê de Programa do SBRC 2013

Luis Henrique Costa (UFRJ)
Luiz Magalhaes
Luiz Eduardo Buzato
Magnos Martinello (UFES)
Marcelo Pellenz
Marcelo Rubinstein (UERJ)
Marco Aurélio Spohn (UFCG)
Marcos Salvador (CPqD)
Marcos José Santana
Marinho Barcellos (UFRGS)
Markus Endler (PUC-Rio)
Mauro Fonseca (PUCPR)
Michael Stanton (UFF/RNP)
Michele Nogueira (UFPR)
Michelle Wangham
Miguel Elias Mitre Campista (UFRJ)
Moises Ribeiro
Nabor Mendonça (UNIFOR)
Natalia Castro Fernandes
Nazareno Andrade (UFCG)
Nelson Fonseca (UNICAMP)
Nelson Rosa (UFPE)
Noemi Rodriguez (PUC-Rio)
Orlando Loques (UFF)
Paulo Aguiar (UFRJ)
Paulo Cunha
Paulo Pires (UFRJ)
Paulo André da Silva Gonçalves (UFPE)
Pedro Rosa
Pedro Velloso (UFF)
Rafael Sousa
Raimundo Jose de Araújo Macêdo (UFBA)
Raquel Mini (PUC Minas)
Regina Silveira
Roberto Willrich (UFSC)
Ronaldo Ferreira (UFMS)
Ronaldo Salles (IME)
Rosa Leão
Rossana Andrade (UFC)
Sidney Lucena
Silvana Rossetto (UFRJ)
Stenio Fernandes
Tereza Carvalho (USP)
Thais Vasconcelos Batista (UFRN)
Vinod Rebello (UFF)
Wagner Meira Jr.
William Giozza (UnB)

Revisores do SBRC 2013

Adriana Centurion
Adriano Branco
Alberto Schaeffer-Filho
Alcides Calsavara
Aldelir Fernando Luiz
Aldri dos Santos
Alejandro Frery
Alexandre Passito
Alexandre Sztajnberg
Alfredo Goldman
Alírio Santos Sá
Allan Edgard Freitas
Aloysio de Castro Pinto Pedroza
Altair Santin
Alyson de Jesus dos Santos
Americo Sampaio
Amilcar César
Ana Carolina Riekstin
Ana Cristina B. Kochem Vendramin
Ana Paula Couto da Silva
Anderson Nascimento
André Cardoso
André Drummond
André Lage
André Soares
André Aquino
André Mendes
Anelise Munaretto
Antônio Jorge Gomes Abelém
Antônio Tadeu Azevedo Gomes
Antonio Alfredo Ferreira Loureiro
Antonio Augusto Teixeira R. Coutinho
Antonio Mury
Antonio Rocha
Arlindo Conceição
Arlindo L. Marcon Jr.
Arthur Callado
Artur Ziviani
Atslands Rocha
Ayla Débora Dantas S. Rebouças
Bruno Schulze
Eduardo Cerqueira
Bruno T. Kuehne
Carina Teixeira de Oliveira
Carlos Ferraz
Carlos Goulart
Carlos Kamienski
Carlos Senna
Carlos Alberto Vieira Campos
Carlos Eduardo Santos
Carlos Henrique Nicodemus
Carmelo Bastos-Filho
César Melo
Celso Alberto Saibel Santos
Celso Hirata
Cesar Marcondes
Charles Miers
Christian Alonso
Christian Esteve Rothenberg
Cintia Borges Margi
Claudio de Farias
Cleber Kiel Olivo
Cristina Murta
Daniel Cason
Daniel Chaves
Daniel Cordeiro
Daniel Figueiredo
Daniel Menasche
Daniel Sperry
Daniel Stefani
Danielo G. Gomes
Davi Böger
David Moura
Dênio Mariz Sousa
Djamel Fawzi Hadj Sadok
Dorgival Guedes
Eder Leão Fernandes
Edhelmira Lima Medina
Edison Albuquerque

Revisores do SBRC 2013

Edmundo de Souza e Silva
Edmundo Madeira
Edson Tavares de Camargo
Eduardo Alchieri
Eduardo Bergamini
Eduardo Nakamura
Eleri Cardozo
Elias P. Duarte Jr.
Elisa Mannes
Esteban Clua
Ewerton Salvador
Fabíola Greve
Fabiano Oliveira
Fabio Costa
Fabio Kon
Fabio Luiz Albini
Fabio Oliveira Lima
Fabrício Barbosa de Carvalho
Fabrício Jorge Lopes Ribeiro
Fatima Duarte-Figueiredo
Fábio Albini
Fábio Favarim
Fábio Luciano Verdi
Felipe Henriques
Felipe Prado
Fernanda G. Oliveira
Fernando Dotti
Fernando Gielow
Flavia Delicato
Flavio Deus
Flávio Roberto Santos
Francisco Brasileiro
Frank Siqueira
Frederico Lopes
Geisa Negrao Alves
Geraldo Robson Mateus
Giacomo Mc Evoy
Gilmar Luiz Vassoler
Guilherme Maia
Gustavo Figueiredo
Gustavo Giménez-Lugo
Gustavo Pavani
Hélio Guardia
Heitor Ramos
Hilio Holz
Humberto Marques
Hylson Netto
Igor Moraes
Luciano Paschoal Gaspary
Iguatemi E. Fonseca
Irineu Sotoma
Islene Garcia
Jacir Bordim
Jauberth Abijaude
Jó Ueyama
Júlio Estrella
Jean-Marie Farines
Jeroen van de Graaf
Jesus Talavera Portocarrero
Jim Lau
João Pereira
Joaquim Celestino Júnior
Joberto Martins
Jorge Lima de Oliveira Filho
José Antônio Oliveira
José Augusto Suruagy Monteiro
José Ferreira de Rezende
José Geraldo Ribeiro Junior
José Gonçalves
José Marcos Nogueira
José Neuman de Souza
José Roberto A. Amazonas
José Souza de Jesus
Julian Monteiro
Juliano Naves
Jussara Almeida
Keiko Fonseca
Kelly Rosa Braghetto
Kelvin Dias
Kleber Cardoso

Revisores do SBRC 2013

Laerte Peotta
Leandro Resendo
Leandro Sales Pinto
Leobino Sampaio
Leonardo Bays
Leonardo Carlos da Cruz
Leonardo Leite
Liane Margarida Rockenbach Tarouco
Lisandro Zambenedetti Granville
Luci Pirmez
Luciana Rech
Luciano Barreto
Luciano Chaves
Luis Nakamura
Luis Henrique Costa
Luiz Magalhaes
Luiz Eduardo Buzato
Luiz Fernando Bittencourt
Luiz Filipe Vieira
Magnos Martinello
Maicon Stihler
Manoel Camillo de Oliveira Penna Neto
Marcel William Rocha da Silva
Marcelo Carvalho
Marcelo Pellenz
Marcelo Ribeiro Nascimento
Marcelo Rubinstein
Marcelo Segatto
Marcelo Luiz Drumond Lanza
Marcia Paiva
Marco Rojas
Marco Antonio Marinho
Marco Aurélio Spohn
Marcos Kakitani
Marcos Paulo Moro
Marcos Salvador
Marcos Vieira
Marcus de Lima Braga
Marcus Sandri
Maria Claudia Cavalcanti
Marinho Barcellos
Mario Zancanaro
Markus Endler
Matheus Bandini
Matheus Lehmann
Mauro Fonseca
Maxwell Monteiro
Michael Stanton
Michele Nogueira
Michelle Wangham
Miguel Elias Mitre Campista
Moisés Ribeiro
Nabor Mendonca
Natália Castro Fernandes
Nazareno Andrade
Nelson Fonseca
Nelson Lago
Nelson Rosa
Nivia Cruz Quental
Noemi Rodriguez
Odorico Mendizabal
Olga Goussevskaia
Orlando Loques
Paulo Aguiar
Paulo Cunha
Paulo Mafra
Paulo Moura
Paulo Pires
Paulo Rego
Paulo Sausen
Paulo André da Silva Gonçalves
Pedro do Prado
Pedro Rosa
Pedro Velloso
Petronio Júnior
Rafael Amaral
Rafael Antonello
Rafael Lopes Gomes

Revisores do SBRC 2013

Rafael Pasquini
Rafael Sousa
Raimundo José de Araújo Macêdo
Ramon S. Schwartz
Raphael Guedes
Raphael Machado
Raquel Mini
Rasha Hasan
Raul Almeida Jr.
Regina Silveira
Reinaldo Braga
Renato de Moraes
Ricardo Custódio
Ricardo Nabhen
Ricardo Puttini
Ricardo Tombesi Macedo
Rick Lopes de Souza
Roberto Willrich
Robson Albuquerque
Robson Gomes de Melo
Rodolfo Oliveira
Rodolfo S. Antunes
Rodolfo Villaça
Rodrigo de Souza Couto
Ronaldo Ferreira
Ronaldo Salles
Roni Shigueta
Rosa Leão
Rossana Andrade
Roverli Pereira Ziwich
Sand Correa
Sandro Santos Andrade
Saulo Jorge Beltrão Queiroz
Sidney Lucena
Silvana Rossetto
Stenio Fernandes
Tania Monteiro
Taniro Chacon Rodrigues
Tereza Carvalho
Thais Vasconcelos Batista
Thiago Genez
Tiago Salmito
Vinícius Cunha M Borges
Vinícius Mota
Vinícius Petrucci
Vinod Rebello
Wagner Meira Jr.
Wanderley Lopes de Souza
Wellington Albano
Weverton Luis da Costa Cordeiro
William Giozza
Yanik Ngoko

Comitê de Avaliação do Salão de Ferramentas

Aldri dos Santos (UFPR)
Alvaro Medeiros (UFJR)
Antonio Alfredo Ferreira Loureiro (UFMG)
Antonio Mury (LNCC)
Antonio Rocha (IC/UFF)
Bruno Schulze(LNCC)
Carlos Kamienski (UFABC)
Carlos Alberto Vieira Campos (UNIRIO)
Daniel Batista (IME – USP)
Daniel Cordeiro (IME – USP)
Divanilson Campelo (UFPE)
Fabio Costa (UFG)
Fabricio Benevenuto (UFMG)
Flavia Delicato (UFRJ)
Gustavo Figueiredo (UFBA)
Humberto Marques (PUC Minas)
Igor Moraes (UFF)
Jussara Almeida (DCC-UFMG)
Leobino Sampaio (UFBa)
Lisandro Zambenedetti Granville (UFRGS)
Marcio Miranda (UFRRJ)
Nazareno Andrade (UFCG)
Nelson Rosa (UFPE)
Olga Goussevskaia (UFMG)
Paulo Pires (UFRJ)
Paulo André da Silva Gonçalves (UFPE)
Raphael Camargo (UFABC)
Raquel Lopes (UFCG)
Raul Almeida Jr. (UFPE)
Ricardo R. Oliveira (UFOP)
Ronaldo Ferreira (UFMS)

Sumário

Artigos Completos do SBRC 2013

Sumário

Sumário

Sumário

Sumário

Sumário

Sumário

Sumário

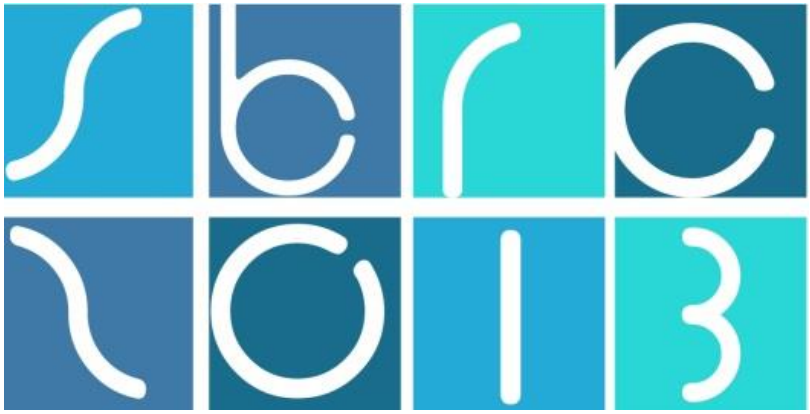
Sumário

Sumário

Sumário

Salão de Ferramentas do SBRC 2013

Sumário



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 1

Redes de Sensores sem Fio

Uma Nova Abordagem para Acesso ao Meio em Redes de Sensores Sem Fio

Felipe D. Cunha[1], **Ítalo Cunha**[1], **Antonio A. F. Loureiro**[1], **Leonardo B. Oliveira**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte – MG – 31270–901 – Brasil

{fdcunha, cunha, loureiro, leob}@dcc.ufmg.br

Abstract. *Wireless sensor networks are designed to be deployed in the physical environment to monitor a variety of real-world phenomena. These networks are composed of various small sensor nodes deposited in a given area. In these networks, battery replacement is impractical and energy is a finite resource whose consumption must be minimized. MAC-layer protocols use techniques to turn off sensors' radio and save energy. One way of doing this is by scheduling, which incurs communication overhead to share and update the schedule. This paper proposes a new protocol able to non-interactively derive the instants of waking up and falling asleep based on node identifiers and, in turn, get rid of the scheduling overhead. Our results indicate that this approach is able to prolong WSN lifetime without degrading network performance.*

Resumo. *Redes de Sensores Sem Fio são instaladas em um ambiente físico para monitorar fenômenos do mundo real. Elas são compostas de vários sensores diminutos depositados numa dada área. Nestas redes, a troca de bateria é inviável e a energia é um recurso finito cujo consumo deve ser minimizado. Protocolos desenvolvidos para a camada MAC implementam técnicas de ligar e desligar o rádio dos sensores para economizar energia. Nos protocolos síncronos, as operações de liga/desliga seguem uma agenda e sensores precisam divulgá-la/atualizá-la, o que acarreta sobrecarga de comunicação. Este trabalho propõe um novo protocolo MAC capaz de derivar de forma não interativa os instantes de "dormir" e "acordar" sensores baseado nos seus identificadores, ou seja, remove a sobrecarga de comunicação para sincronização. Os resultados indicam que essa estratégia é capaz de prolongar o tempo de vida útil da rede sem degradar seu desempenho.*

1. Introdução

Redes de Sensores Sem Fio (RSSFs) [Akyildiz et al. 2002, Loureiro et al. 2003] são um tipo particular de Redes Móveis Ad hoc (*Mobile Ad hoc Networks* – MANETs). Elas são compostas em sua maioria por pequenos nós (*nodes*) sensores cujos recursos (energia, largura de banda, processamento etc.) são extremamente limitados. Estes sensores, por sua vez, se conectam ao mundo externo por meio de dispositivos bem provisionados chamados de sorvedouros (*sink*) ou Estações Rádio Base (ERBs). RSSFs são utilizadas com o intuito de monitorar regiões, oferecendo dados sobre a área monitorada, também chamada de área de interesse (*interest area*), para o resto do sistema.

Dentre a vasta gama de aplicações de RSSFs existem operações de resgate em áreas de desastre e/ou conflito e monitoramento de recursos naturais [Akyildiz et al.

2002, Loureiro et al. 2003]. Muitas vezes, aplicações de RSSFs possuem como área de interesse ambientes inóspitos que dificultam, senão impedem, a recuperação dos sensores. Em outras palavras, os sensores são "descartáveis". Se são descartáveis, os sensores devem ser baratos; e para torná-los baratos, são dotados de poucos recursos computacionais [Akyildiz et al. 2002, Loureiro et al. 2003]. Numa aplicação ao ar livre (*outdoor*), a energia é comumente o recurso mais limitado de um sensor [Akyildiz et al. 2002,Loureiro et al. 2003].

A comunicação, por sua vez, é a principal responsável pelo dispêndio energético em RSSFs. Ao contrário das redes cabeadas, a energia dissipada na transmissão de uma informação em RSSFs é ordens de grandeza maior que a dissipada durante o seu processamento [Akyildiz et al. 2002, Loureiro et al. 2003]. Assim, é natural que se repense o projeto de protocolos considerando os requisitos exclusivos de RSSFs para que se prolongue a vida útil da rede [Demirkol et al. 2006, Langendoen and Meier 2010]

No que tange à arquitetura de RSSFs, a camada de Controle de Acesso ao Meio (*medium access control* – MAC), em particular, desempenha papel chave no consumo de energia por parte da comunicação [Demirkol et al. 2006]. É essa camada que governa diversos fenômenos que impactam substancialmente o dispêndio energético, a saber: "escuta-ociosa" (*idle-listening*), "escuta-desnecessária" (*overhearing*), "transmissão-em-vão" (*overemitting*), colisões (*collisions*), e sobrecarga de comunicação (*communication overhead*) [Demirkol et al. 2006, Langendoen and Meier 2010]. A escuta-ociosa ocorre quando um sensor está pronto para receber quadros (*frames*), mas não está de fato recebendo. A escuta-desnecessária ocorre quando um sensor decodifica um quadro que não foi destinado a ele. A transmissão-em-vão ocorre quando um sensor transmite um quadro e o destinatário não está pronto para recebê-lo. E por fim, as mais comuns colisões e sobrecargas de comunicação.

Os protocolos MAC desenvolvidos para RSSFs adotam duas principais abordagens para combater tais fenômenos: síncrona e assíncrona. Nos protocolos síncronos (como [Ye et al. 2004, Van Dam and Langendoen 2003]), ou do tipo TDMA, sensores definem e compartilham *a priori* uma agenda de transmissão e recebimento de dados [Demirkol et al. 2006]. Por um lado, isso diminui a escuta-ociosa, mas por outro é pouco flexível e acarreta sobrecarga para o alinhamento quanto à agenda [Demirkol et al. 2006]. Nos protocolos assíncronos (como [Polastre et al. 2004, Lu et al. 2004]), os sensores remetentes precedem quadros com preâmbulos ligeiramente maiores que o período de dormência (*sleeping period*) dos seus respectivos destinatários. A vantagem dessa abordagem é que além de mitigar o escuta-ociosa, ela não requer quadros acerca de agendas [Demirkol et al. 2006]. A abordagem assíncrona, no entanto, sofre tanto de escuta-desnecessária quanto de transmissão-em-vão, ambos causados pelos preâmbulos por ela adicionados [Demirkol et al. 2006]. Cumpre lembrar que há abordagens que tentam unir os prós das abordagens síncronas e assíncronas. Essa abordagem é chamada de híbrida (por exemplo [Rhee et al. 2008]).

Neste trabalho adotamos uma abordagem inovadora e propusemos um protocolo MAC baseado em identidade. O ID-MAC, como o protocolo foi batizado, é síncrono na medida em que os sensores participantes da comunicação conhecem quando cada um estará pronto para enviar/receber; no entanto, o protocolo não requer que quadros de controle acerca do agendamento de enviar/receber sejam trocados. Isso porque no ID-

MAC tais agendas podem ser derivadas a partir das identidades dos próprios sensores – por exemplo, seu endereço físico (*MAC address*).

Contribuições: As contribuições do trabalho em questão são as seguintes:

1. concepção de uma abordagem inovadora de protocolos MAC para RSSFs que se baseia na identidade dos sensores;
2. apresentação do ID-MAC, um protocolo MAC concreto baseado nesta nova abordagem;
3. e avaliação de desempenho do ID-MAC frente ao protocolo S-MAC [Ye et al. 2004], comumente utilizado em trabalhos de MAC para RSSFs como base de comparação em avaliações de desempenho.

Organização: Este documento está organizado da seguinte forma. Primeiramente, apresentamos uma visão geral de protocolos MAC para RSSFs (seção 2). A seguir apresentamos nossa proposta de protocolo MAC (seção 3). As avaliações e seus respectivos resultados são apresentados na seção 4. Em seguida, discutimos os trabalhos relacionados (seção 5). Por fim, na seção 6, apresentamos as considerações finais acerca do trabalho.

2. MACs para RSSFs: Visão Geral

Em muitas aplicações de RSSFs é inviável trocar a fonte energética (por exemplo, a bateria ou pilha) dos sensores. Isso faz com que uma vez finda tal fonte, seu sensor torne-se inoperante. Logo, um objetivo central em uma RSSF é minorar o consumo de energia por parte dos seus sensores, prolongando assim o tempo de vida útil da rede [Akyildiz et al. 2002, Loureiro et al. 2003].

O padrão de comunicação de uma RSSFs é um fator chave para o seu desempenho do ponto de vista energético [Akyildiz et al. 2002, Loureiro et al. 2003]. Existem basicamente três formas de comunicação: (i) difusão (*broadcast*), (ii) comunicação convergente (*convergecast*) e (iii) comunicação local (*local gossip*). Respectivamente, elas ocorrem quando (i) o sorvedouro deseja disseminar uma informação para toda a rede[1]; (ii) quando os sensores desejam reportar ao sorvedouro; e (iii) quando há uma comunicação local entre sensores de uma mesma vizinhança ou agrupamento (*cluster*).

Diversos fenômenos relacionados à comunicação são governados pelo protocolo MAC utilizado. Por exemplo, estar a postos para receber quadros (isto é, ativo e com o rádio ligado) consome muita energia mesmo se não ocorrer uma recepção [Demirkol et al. 2006, Langendoen and Meier 2010]. Assim, para poupar energia em RSSFs, os protocolos MAC buscam colocar os sensores em modo dormente (*sleeping mode*), no qual o rádio é desligado, sempre que possível. Em geral, os períodos ativos têm duração fixa enquanto que os períodos inativos dependem do ciclo de trabalho[2] (*duty cycle*) pré-definido.

Além dos problemas comuns a protocolos MAC como colisões e sobrecarga de comunicação, essa alternância de modos em RSSFs ocasionam os seguintes problemas [Demirkol et al. 2006, Langendoen and Meier 2010]:

[1]Note-se que neste contexto o termo difusão denota algo diferente do usual difusão de pacote (*packet broadcast*), em que elementos da rede enviam um quadro para todos a seu alcance.

[2]O ciclo de trabalho é a fração do tempo total em que um sensor permanece ativo, ou seja, com o rádio ligado.

1. escuta-ociosa: acontece quando potenciais destinatários estão ativos, ou seja, prontos para receber quadros e consumindo energia, sem que de fato estejam recebendo.
2. escuta-desnecessária: ocorre quando sensores recebem quadros gratuitamente, isto é, quando no intuito de verificar se um quadro é destinado a eles, recebem e decodificam quadros endereçados a outrem.
3. transmissão-em-vão: é causada quando quadros são transmitidos sem que, no entanto, seu destinatário esteja pronto para recebê-las.

Assim sendo, um projeto eficiente e eficaz de protocolos MAC deve levar em consideração os fenômenos supracitados. Além deles, outros requisitos de projeto são escalabilidade e adaptabilidade a mudanças [Demirkol et al. 2006]. Alterações na topologia, tamanho da rede e densidade de sensores devem impactar o mínimo a eficácia e a eficiência do protocolo. Alguns requisitos centrais em redes tradicionais são secundários em RSSFs. Neste contexto, vazão e atraso são deveras importantes, mas o consumo energético é ainda mais premente. Por fim, o requisito de justiça (*fairness*) não é tão relevante como nas demais redes; os sensores que compõe uma RSSF possuem usualmente um objetivo comum, não importando quem mais colaborou para alcançá-lo.

Grosso modo, são duas as abordagens de protocolos MAC em RSSFs: (i) síncrona e (ii) assíncrona [Demirkol et al. 2006, Langendoen and Meier 2010].

Os protocolos síncronos (como [Ye et al. 2004, Van Dam and Langendoen 2003]) são baseados em agendas (*schedule-based*) em que cada participante da comunicação possui um intervalo de tempo agendado exclusivo para sua transmissão. De posse dessa agenda, os potenciais destinatários ficam ativos para receber transmissões nos períodos pré-agendados. Demais remetentes também respeitam a agenda e esperam sua vez para transmitir. Logo, protocolos síncronos são capazes de tanto mitigar o problema de transmissão-em-vão, como o de colisões. Por outro lado, eles precisam trocar quadros de controle para "acordar" e/ou anunciar uma agenda comum, o que resulta em sobrecarga de comunicação. Ademais, as agendas são usualmente fixas e essa inflexibilidade pode acarretar maior atraso.

Os protocolos assíncronos são baseados na disputa do meio de transmissão (*contention-based*) (como [Polastre et al. 2004, Lu et al. 2004]). Tais protocolos também tiveram que empregar novos mecanismos para lidar com a alternância de modos de RSSFs. Em diversos protocolos, por exemplo, o remetente precede o envio de um quadro com um preâmbulo ligeiramente mais longo que o período de dormência (*sleeping period*) do destinatário. Este último, ao "acordar", nota pelo preâmbulo a intenção do remetente de enviar um quadro e se mantém ativo. Por outro lado, caso não escute preâmbulos, o potencial destinatário volta rapidamente a "dormir". Tais protocolos possuem a vantagem de minorar o problema de escuta-ociosa e de não precisarem de sincronização, o que diminui a sobrecarga de comunicação. Contudo, eles sofrem tanto de transmissão-em-vão como de escuta-desnecessária, já que todos os sensores devem esperar o preâmbulo para que possam verificar se o quadro é ou não destinado a eles.

Dentre todos os protocolos, o mais popular e utilizado como base de comparação com os demais talvez seja o S-MAC [Ye et al. 2004]. O S-MAC opera em *rodadas de transmissão* (*transmission rounds*). Cada rodada alterna entre dois períodos, ativos e dormentes (vide figura 1). Nos períodos dormentes os sensores desligam o rádio e o

consumo energético é reduzido. Findo esse período, os sensores ligam o rádio e estão prontos para trocar informações. A duração desses períodos varia com a o valor do ciclo de trabalho.

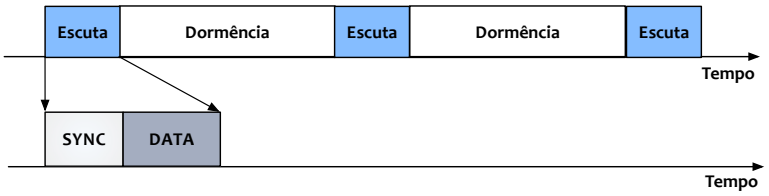


Figura 1. Períodos de escuta e dormência no S-MAC.

No S-MAC é necessário que cada sensor alinhe com seus vizinhos quanto ao início da rodada. Tal sincronização é feita na fase inicial, na qual cada sensor escolhe aleatoriamente um instante para "adormecer" e o difunde para os vizinhos. Assim, cria-se um agrupamento virtual (*virtual cluster*) de sensores que compartilham de uma mesma agenda. Mais precisamente, cada período de escuta pode ser dividido em duas etapas no S-MAC. São elas SYNC e DATA (vide figura 1). A etapa SYNC é destinada para sincronização, na qual os sensores podem enviar atualizações de agendamentos para vizinhos. Já a etapa DATA é destinada à comunicação por difusão e por ponto-a-ponto por parte dos sensores. Vale lembrar, que o S-MAC também emprega o mecanismo (RTS-CTS-DATA-ACK) [Tanenbaum 2002] para evitar colisões. Cumpre lembrar que o S-MAC, assim como os demais protocolos síncronos, gasta uma parte considerável da energia dos sensores com quadros de controle. Como veremos a seguir, tais quadros não são necessários no ID-MAC.

3. ID-MAC

A principal vantagem do ID-MAC é não ter que enviar quadros de controle e, assim, poupar energia. Isso é possível porque no ID-MAC sensores não precisam receber a agenda acerca de quando seus interlocutores estarão prontos enviar e receber dados. Ao invés disso, os sensores derivam tal informação de forma não-interativa. Para tal, a rede emprega uma função pseudo-aleatória pública, isto é, comum a todos os sensores.[3] Os sensores remetentes derivam sua agenda semeando a função com seus próprios identificadores. E os destinatários, analogamente, derivam sua agenda semeando a função com os identificadores dos seus respectivos rementes. Desta forma, remetentes e destinatários são capazes de construir de forma não interativa uma agenda comum. Assim o ID-MAC consegue reunir as vantagens dos protocolos síncronos – ou seja, mitigar o transmissão-em-vão e escuta-desnecessária sem, no entanto, "pagar o alto preço" dessas vantagens, isto é, enviar e receber quadros de controle.

A seguir, primeiro apresentamos uma visão geral do ID-MAC (seção 3.1), depois formalizamos o funcionamento do protocolo (seções 3.2 e 3.3) e por último discutimos seus detalhes de implementação (seção 3.4).

[3] A função pseudo-aleatória pode ser carregada nos sensores antes dos mesmos serem dispostos na área de interesse.

Tabela 1. Notação

NOTAÇÃO	DESCRIÇÃO
c	Contador de rodadas de transmissão
d	Duração das rodadas de transmissão
q	Duração da transmissão de um quadro
$t(c)$	Instante onde começa a rodada c
$t(s, c)$	Instante em que começa a transmissão de s na rodada c
f	Função pseudo-aleatória
$\mathcal{V}_s$	Conjunto de vizinhos conhecidos pelo sensor s

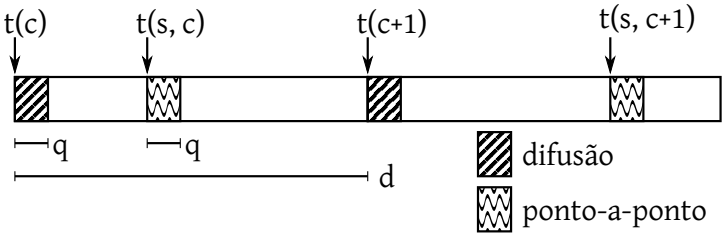


Figura 2. Ilustração das rodadas de transmissão c e $c + 1$ do sensor s.

3.1. Visão geral

Cada sensor s utilizando ID-MAC tem um identificador unívoco, por exemplo, seu endereço físico. A figura 2 mostra rodadas de transmissão de um sensor s. Sensores executando o ID-MAC compartilham um contador de rodada c, o qual é incrementado após cada rodada de transmissão. A duração d destas rodadas é fixa. O instante de tempo em que a rodada c começa é denominado $t(c)$ e a próxima rodada de trabalho começa imediatamente após a anterior, $t(c + 1) = t(c) + d$. Os sensores são sincronizados e iniciam a rodada de transmissão no mesmo instante $t(c)$.

O início de cada rodada de transmissão é reservado para transmissão de quadros de difusão. Todos os sensores "acordam" no início de uma rodada para verificar se um quadro de difusão será transmitido e recebê-lo. A seção 3.2 detalha o algoritmo que seleciona qual sensor deve transmitir quadros de difusão em uma rodada. Seja q a duração da transmissão de um quadro de tamanho máximo e sua possível confirmação, então o período entre $t(c)$ e $t(c) + q$ é reservado para difusão.

Um sensor pode enviar um quadro ponto-a-ponto por rodada de transmissão. O instante em que um sensor s transmite seu quadro varia a cada rodada. Seja $t(s, c)$ o instante onde a transmissão do sensor s na rodada c começa. Por exemplo, a transmissão do sensor s na rodada c pode começar logo após a transmissão de difusão na rodada c, $t(s, c) \approx t(c) + p$; e a transmissão do sensor s na rodada $c + 1$ pode começar próxima ao início da rodada $c + 2$, $t(s, c + 1) \approx t(c + 2) - p$. A figura 2 ilustra como o instante da transmissão do quadro de um sensor s varia entre rodadas.

Cada sensor s calcula o instante de sua transmissão na rodada c, $t(c, s)$ usando uma função pseudo-aleatória f compartilhada por todos os sensores. A função f recebe como parâmetro o identificador do sensor s e o marcador da rodada atual c e retorna um valor aleatório entre $[0, 1)$ (nossa implementação de f usa SHA256). O valor $f(s, c)$ é

usado como o deslocamento da transmissão de s na rodada c (vide detalhes na seção 3.3). Por fim, cada sensor s mantém um conjunto $\mathcal{V}_s$ de identificadores vizinhos, ou seja, dos sensores que estão no seu raio de transmissão. Note-se que um sensor s tem todos os dados necessários para calcular os instantes de transmissão $t(s', c')$ de todos os seus vizinhos $s' \in \mathcal{V}_s$ em qualquer rodada c'. Isso permite a um sensor s saber quando "acordar" para receber quadros de um vizinho s'.

3.2. Transmissões de difusão

O ID-MAC reserva os primeiros q milissegundos de cada rodada para a transmissão de um quadro de difusão. Cada sensor s decide transmitir um quadro de difusão na rodada c verificando se

$$f(s, c) < f(v, c) \text{ para todo } v \in \mathcal{V}_s \text{ e} \tag{1}$$

$$f(s, c) < \frac{1}{|\mathcal{V}_s|}. \tag{2}$$

A eq. (1) distribui a banda disponível para difusão de forma justa entre os sensores. Cada sensor recebe uma fração da banda de difusão inversamente proporcional à quantidade de vizinhos que possui. Todos os sensores eventualmente conseguem enviar quadros de difusão e não existe inanição. A eq. (2) limita a utilização da banda de difusão (em torno de 70 a 75% para vizinhanças de 3 a 10 sensores) para que novos sensores entrando na rede possam informar sua presença aos sensores que lá já estão.

3.3. Transmissões ponto-a-ponto

Cada sensor pode transmitir um quadro por rodada de transmissão. Cada sensor s calcula o instante $t(s, c)$ durante a rodada c quando ele começa sua transmissão como

$$t(s, c) = t(c) + q + (d - 2q)f(s, c). \tag{3}$$

O termo $t(c) + q$ é o instante quando acaba a transmissão de difusão na rodada c e podem começar as transmissões ponto-a-ponto. O termo $d - 2q$ é a duração do intervalo onde podem começar transmissões durante a rodada c. Subtraímos $2q$ por que a transmissão de difusão ocupa q milissegundos no início da rodada e por que precisamos começar a transmissão de um quadro antes de $t(c) + d - q$ para terminar a transmissão do quadro antes do fim da rodada.

Todos os vizinhos de um sensor s conseguem calcular $t(s, c)$. Vizinhos interessados em receber dados de s "acordam" no instante $t(s, c)$ para verificar se s tem dados para transmitir. Em particular, em redes de sensores onde os dados são transmitidos para um sorvedouro, o próximo sensor na árvore de roteamento de s até o sorvedouro precisa "acordar" nos instantes $t(s, \cdot)$ para receber e reencaminhar o quadro de s.

3.4. Detalhes de implementação e operação

Construção da rede. Sensores precisam saber o contador da rodada atual, c, e quando a rodada de transmissão se inicia, $t(c)$.[4] Vale lembrar que a escolha de c e $t(c)$ para a junção de múltiplos agrupamentos (*clusters*) em uma única rede pode ser feita com algoritmos

clássicos para eleição de líder [Lynch 1996] e, portanto, tal questão não foi alvo deste trabalho.

Sincronização. Sensores precisam sincronizar o início das rodadas de transmissão $t(c)$. Os instantes de início das rodadas podem ser sincronizados usando mensagens de difusão de referência [Elson and Estrin 2002].

Modo de operação. Descrevemos o ID-MAC assumindo que um sensor s sabe de qual vizinho s' ele deve receber dados e, então, "acorda" no instante $t(s', c)$. Este modo de operação é adequado para redes em que os sensores encaminham dados para um sorvedouro e reduz colisões pois as transmissões começam em instantes aleatórios.

O ID-MAC ainda é eficiente em cenários em que os sensores não sabem de quais outros sensores devem receber dados. Para tal, basta utilizar a eq. (3) para determinar o instante no qual o sensor s "acorda" para receber dados (em vez de enviar). Este modo de operação está mais sujeito a colisões, pois dois nós que querem transmitir para s podem transmitir no mesmo instante $t(s, c)$.

Confirmação e retransmissão. Destinatários executando o ID-MAC confirmam o recebimento de um quadro para o remetente com um pequeno quadro de confirmação (ACK). No caso de colisão ou falha na decodificação de um pacote recebido, o remetente não recebe o quadro de confirmação e tenta retransmitir o quadro.

Escalabilidade. Quando a densidade da rede e o número de vizinhos dos sensores aumenta, a duração da rodada de transmissão também deve aumentar. O aumento da duração da rodada de transmissão evita que a probabilidade de colisão em transmissões ponto-a-ponto aumente e degrade o desempenho da rede.

4. Resultados

Nessa seção descrevemos o ambiente de simulação no ns-2 [ns2 2002] (seção 4.1) utilizado na avaliação do ID-MAC e discutimos os resultados (seção 4.2).

4.1. Ambiente de Simulação

A proposta das simulações é realizar uma coleta de dados sobre eventos que acontecem numa área de interesse. Utilizamos o modelo de Mini *et al.* [Mini et al. 2004], no qual eventos ocorrem em locais uniformemente distribuídos na área de interesse e o tempo entre eventos é dado por um processo de Poisson [Ross 1996]. Sensores que detectam um evento coletam e transmitem informações sobre o evento para o sorvedouro.

Consideramos uma rede com 200 sensores estáticos dispostos aleatoriamente numa área de interesse de $100 \times 100\, m^2$. O sorvedouro é posicionado no canto inferior esquerdo, isto é, na posição $(0, 0)$. Executamos simulações variando a taxa de ocorrência de eventos em $0,03$, $0,09$ e $0,12$ eventos por segundo. O raio de influência r de cada evento é uniformemente distribuído entre 2 e $10\, m^2$. A duração de um evento é uniformemente distribuída entre 5 e 50 segundos. O tempo entre os eventos é modelado pelo processo de Poisson e definido pela equação $f(x) = \lambda e^{-\lambda x}$ [Mini et al. 2004].

[4]Note-se que a manutenção de um contador de rodada global c não é obrigatória. Alternativamente, cada sensor s pode ter seu próprio contador de rodada c_s; basta aos vizinhos de s armazenar c_s para calcular as eqs. (1), (2) e (3).

A rede é homogênea, i.e., todos os sensores têm a mesma configuração. A energia inicial atribuída a cada sensor é de 25 J. Tal valor foi escolhido de modo a não permitir a morte prematura de um sensor devido à escassez de energia. O raio de comunicação e sensoriamento de cada sensor é 10 m. O modelo de dissipação de energia utilizado segue o modelo do ns-2 [ns2 2002], o qual decrementa o consumo de energia dos sensores de acordo com o modo de operação do sensor. O consumo de cada modo de operação é definido com parâmetros reais de consumo do sensor Mica2 [Mica2 2004]. O ciclo de trabalho dos sensores é fixo em 20 % e a duração da rodada de transmissão resultante é de aproximadamente 140 ms. Este valor de 20 % foi escolhido para que todos os sensores possam enviar dados sem muita sobreposição nos seus intervalos de envio. Por fim, na camada de rede foi utilizado um protocolo que constrói a árvore de roteamento a partir da distância Euclidiana entre os sensores. A escolha por um protocolo simples como esse se deve ao fato de que o foco deste trabalho é na camada MAC. A tabela 2 sumariza a configuração dos parâmetros utilizados nas simulações.

Tabela 2. Configuração dos cenários de simulação.

Parâmetros Gerais	
Topologia da rede	plana
Sensores	estáticos & homogêneos
Número de sensores	200
Área	$100 \times 100\, m^2$
Posição do sorvedouro	$(0, 0)$
Energia inicial	25 J
Potência de transmissão	+5 dBm
Raio de comunicação e sensoriamento	10 m
Ciclo de trabalho	20 %
Período de operação	1200 s
Parâmetros de Consumo de Energia do Rádio	
Envio	27 mA
Recepção	10 mA
Dormente	1 μA
Parâmetros do Modelo de Eventos	
Raio do evento	Uniforme(2, 10) m
Duração do evento	Uniforme(5, 50) s
λ	$\{0.03, 0.09, 0.12\}$

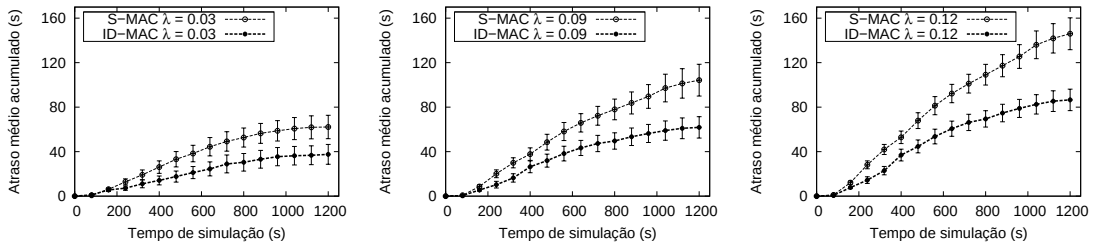
Para termos uma base de comparação na nossa avaliação, simulamos os mesmos cenários para o ID-MAC e para S-MAC. Conforme mencionamos anteriormente (seção 2), em RSSFs o S-MAC é comumente utilizado como base de comparação para novas propostas de protocolos MAC. Como o S-MAC retransmite pacotes apenas uma vez, configuramos o ID-MAC para retransmitir pacotes apenas uma vez e remover o impacto do número de retransmissões nos resultados. As métricas utilizadas para avaliar o desempenho dos dois protocolos foram as seguintes:

1. *atraso fim-a-fim*: o intervalo de tempo entre envio de um quadro pelo sensor e sua recepção pelo sorvedouro;
2. *consumo de energia*: a energia consumida por todas as operações realizadas por um sensor na rede, durante o funcionamento da rede;
3. *entrega fim-a-fim:* o número de quadros que foram entregues com sucesso para o sorvedouro.

Todos os resultados abaixo descartam os 60 primeiros segundos da simulação, onde os nós executam descoberta de vizinhos e construção da árvore de roteamento, para focar no desempenho dos protocolos MAC para transmissão de dados ao sorvedouro. Simulamos cada cenário 33 vezes com sementes para geração de números aleatórios distintas. Quando não mostramos a variação dos dados, o coeficiente de variação foi menor que $0,1$.

4.2. Resultados

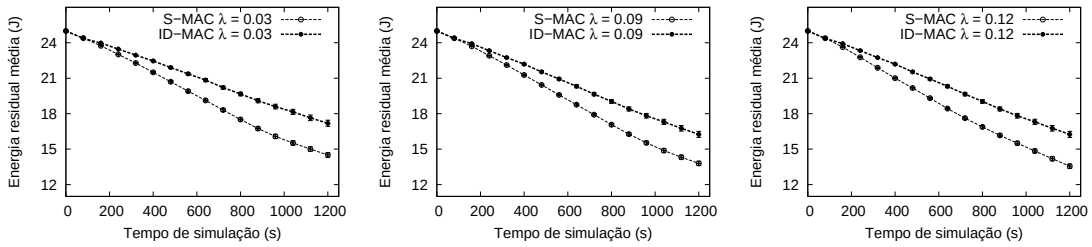
A figura 3 apresenta o atraso médio acumulado ao longo da operação da rede. Para certa taxa de ocorrência de eventos (por exemplo, na figura 3(a)), a latência média aumenta ao longo do tempo. Como apresenta-se a média, com o decorrer do tempo de simulação, quadros de eventos mais afastados do sorvedouro, com atraso maior, chegam e elevam o atraso médio. O atraso também cresce com a taxa de ocorrência de eventos λ, pois quão maior a frequência dos eventos, maior também a quantidade de dados trafegada na rede. Em relação aos protocolos, é possível verificar que o protocolo ID-MAC apresentou melhor desempenho. O atraso do ID-MAC e a variância do seu atraso são menores que os do S-MAC. Duas propriedades do ID-MAC contribuem para a redução do atraso: (i) as transmissões são feitas em instantes aleatórios, o que diminui a probabilidade de colisão; e (ii) mais de uma transmissão pode acontecer numa rodada de transmissão.



(a) $\lambda = 0.03$ (b) $\lambda = 0.09$ (c) $\lambda = 0.12$

Figura 3. Atraso médio em função de λ.

A avaliação do consumo de energia dos protocolos pode ser observada através da energia residual média nos sensores da rede. A figura 4 mostra o decaimento da energia média na bateria dos nós da rede ao longo do tempo para ambos os protocolos. Nota-se um pequeno acréscimo do consumo de energia quando a frequência de eventos aumenta. Isto ocorre porque uma quantidade de dados maior será trafegada, o que necessita de mais transmissões. Além disso, o protocolo ID-MAC apresentou um consumo de energia menor que o S-MAC, cerca de 10 %, nos três cenários. Duas características do ID-MAC contribuem para essa economia de energia: (i) o ID-MAC não possui quadros RTS-CTS para reserva do canal; e (ii) redução de colisões e, consequentemente, de retransmissões pela utilização da função pseudo-aleatória definindo o instante de transmissão.



(a) $\lambda = 0.03$ (b) $\lambda = 0.09$ (c) $\lambda = 0.12$

Figura 4. Energia residual média em função de λ.

A figura 5 apresenta a taxa de entrega de pacotes na rede. Pode-se notar que o aumento na quantidade de eventos também aumentou a entrega fim-a-fim de dados para os protocolos comparados. Entretanto, percebe-se que o aumento na entrega fim-a-fim não foi expressivo, algo em torno de 15 %. Isso ocorre pois o tamanho do período de escuta e transmissão permanece constante nos cenários avaliados, o que limita a quantidade de transmissões. Nota-se que o tamanho desse período é um fator limitante para a entrega fim-a-fim. Também é possível verificar que o protocolo proposto apresentou uma entrega fim-a-fim 20 % maior. Esse aumento médio de 20 % é alcançado devido ao modo de operação do ID-MAC, permitindo mais de uma transmissão por rodada, quando necessário, enquanto o S-MAC realiza no máximo uma transmissão por rodada.

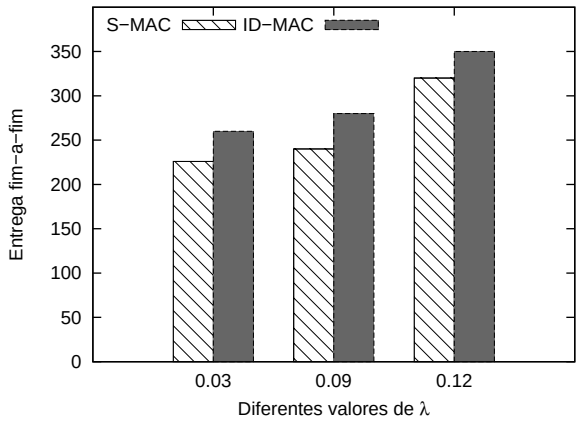


Figura 5. Entrega fim-a-fim em função de λ.

5. Trabalhos Relacionados

Protocolos MAC desempenham um papel chave em RSSFs e diversas propostas foram desenvolvidas exclusivamente para este tipo de rede [Ye et al. 2004, Hill and Culler 2002, Van Dam and Langendoen 2003, Polastre et al. 2004, Lu et al. 2004, El-Hoiydi and Decotignie 2004, Zheng et al. 2005, Rhee et al. 2008, Ye et al. 2006, Halkes and Langendoen 2007, Kim et al. 2008, Liu et al. 2009, Hsu et al. 2009, Parker et al. 2010, de Andrade et al. 2012].

De acordo com a linha do tempo apresentada na figura 6, Ye *et al.* propuseram o protocolo S-MAC [Ye et al. 2004], um dos primeiros para RSSFs. O protocolo segue a abordagem síncrona, utiliza ciclos de trabalho e foi discutido na seção 2 deste trabalho.

Em seguida, outros protocolos foram propostos com o objetivo de aprimorá-lo. Dentre eles, o protocolo T-MAC [Van Dam and Langendoen 2003], proposto por Dam *et al.* é um protocolo cujo objetivo é reduzir a escuta-ociosa e a latência na comunicação. O T-MAC ajusta o ciclo de trabalho dos sensores de acordo com as flutuações de tráfego da rede. Para tal, antes de iniciar o modo dormente, um sensor verifica se não há nenhuma comunicação corrente ou iminente. Caso haja, o sensor permanece ativo e apto para realizar a transferência. A desvantagem desta abordagem é o sensor ficar ativo sem que de fato ocorra uma nova transmissão, gerando um gasto desnecessário de energia.

Seguindo a evolução dos protocolos, Polastre *et al.* e Lu *et al.* propuseram, respectivamente, o B-MAC [Polastre et al. 2004] e D-MAC [Lu et al. 2004]. Ambos empregaram períodos dormentes para economizar energia. O protocolo B-MAC reduz o tempo do ciclo de trabalho e minimiza a escuta-ociosa ao empregar um esquema de preâmbulo adaptativo configurado pelas camadas superiores. O protocolo D-MAC, por sua vez, concentra-se em melhorar a comunicação convergente. Seu funcionamento evita a quebra nas transmissões em direção ao sorvedouro. Para isso, cada sensor escolhe o instante de "dormir" de acordo com sua posição na árvore de roteamento. Além disso, o D-MAC ajusta a duração do seu ciclo de trabalho em função do tráfego de rede.

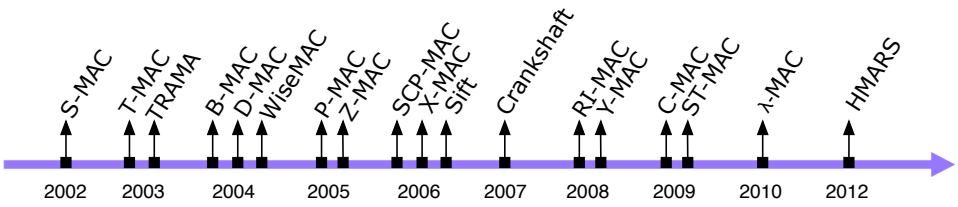


Figura 6. Linha do tempo de protocolos MAC para RSSFs.

O protocolo SCP-MAC [Ye et al. 2006] de Ye *et al.*, por sua vez, tem como principal caraterística a redução expressiva do ciclo de trabalho. Ele pode operar com valores de 0.1% ou até menores. Para a utilização desta abordagem os autores propõem um esquema de otimização do agendamento e mecanismos de sondagem do canal de comunicação, utilizando melhor o mesmo. Neste cenário, os sensores vão acordar em curtos espaços de tempo, apenas para checar a atividade do canal. Tal abordagem, no entanto, apresenta melhor desempenho em situações de pouco tráfego.

Por fim, há ainda protocolos que foram concebidos para contextos específicos. Por exemplo, Andrade *et al.* [de Andrade et al. 2012] propuseram o HMARS, um protocolo MAC específico para a integração de RSSFS com rádio sobre fibra (*radio over fiber – RoF*); e Hsu *et al.* propuseram o ST-MAC, um protocolo para RSSFs Aquáticas [Hsu et al. 2009].

6. Conclusão

A substituição da fonte energética é comumente inviável em RSSFs. Assim, a energia se torna um recurso crítico que deve ser considerado por toda a pilha de protocolos. Neste contexto, o protocolo da camada MAC desempenha um papel especialmente importante. Objetivando poupar energia, os protocolos MAC exclusivamente voltados para RSSFs implementam técnicas de ligar e desligar o rádio dos sensores. Nos protocolos síncronos,

tal operação de liga/desliga segue uma agenda e sensores precisam divulgá-la/atualizá-la, o que acarreta sobrecarga de comunicação.

Neste trabalho propusemos o ID-MAC, um protocolo em que os sensores participantes da comunicação são capazes de derivar, de forma não interativa, quando cada um deles estará pronto para enviar/receber. Para tal, os sensores utilizam seus identificadores e uma função comum a todos. As principais contribuições do trabalho foram as seguintes: (i) a concepção de uma abordagem inovadora de protocolos MAC para RSSFs que se baseia na identidade dos sensores; (ii) a apresentação do ID-MAC, um protocolo MAC concreto baseado nesta nova abordagem; (iii) e a avaliação de desempenho do ID-MAC frente ao protocolo S-MAC. Os resultados mostraram que essa estratégia é capaz de prolongar o tempo de vida útil da rede sem, no entanto, degradar seu desempenho.

Referências

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

de Andrade, T. P. C., da Fonseca, N. L. S., Oliveira, L. B., and Branquinho, O. C. (2012). Protocolos MAC para integração de redes de sensores sem fio baseado em rádio-sobre-fibra. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'12)*.

Demirkol, I., Ersoy, C., and Alagoz, F. (2006). MAC protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115–121.

El-Hoiydi, A. and Decotignie, J.-D. (2004). WiseMAC: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In *International Symposium on Computers and Communications 2004 Volume 2 (ISCC'04)*, ISCC '04, pages 244–251, Washington, DC, USA. IEEE Computer Society.

Elson, J. and Estrin, D. (2002). Fine-Grained Network Time Synchronization using Reference Broadcast. In *OSDI*.

Halkes, G. P. and Langendoen, K. G. (2007). Crankshaft: An energy-efficient mac-protocol for dense wireless sensor networks. In *4th European Conference on Wireless Sensor Networks (EWSN'07)*. Blackwell.

Hill, J. and Culler, D. (2002). Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24.

Hsu, C.-C., Lai, K.-F., Chou, C.-F., and Lin, K. C.-J. (2009). ST-MAC: Spatial-temporal mac scheduling for underwater sensor networks. In *INFOCOM'09*, pages 1827–1835. IEEE.

Kim, Y., Shin, H., and Cha, H. (2008). Y-MAC: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In *7th International Conference on Information Processing in Sensor Networks IPSN'08*, pages 53–63, Washington, DC, USA. IEEE Computer Society.

Langendoen, K. and Meier, A. (2010). Analyzing mac protocols for low data-rate applications. *ACM Trans. Sen. Netw.*, 7(2):19:1–19:40.

Liu, S., Fan, K.-W., and Sinha, P. (2009). CMAC: An energy-efficient mac layer protocol using convergent packet forwarding for wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(4):29:1–29:34.

Loureiro, A. A. F., Nogueira, J. M. S., Ruiz, L. B., Mini, R. A., Nakamura, E. F., and Figueiredo, C. M. S. (2003). Redes de sensores sem fio. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'03)*, pages 179–226, Natal, RN, Brazil. Tutorial.

Lu, G., Krishnamachari, B., and Raghavendra, C. (2004). An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *18th International Parallel and Distributed Processing Symposium (IPDP'04)*, pages 224–231.

Lynch, N. A. (1996). *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Mica2 (2004). Mts/mda sensor and data acquisition boards user's manual. www.xbow.com.

Mini, R. A. F., Loureiro, A. A. F., and Nath, B. (2004). A more realistic energy dissipation model for sensor nodes. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'04)*, pages 365–378, Gramado, RS, Brazil. ISBN 85-88442-80-9.

ns2 (2002). The network simulator. www.isi.edu/nsnam/ns.

Parker, T., Halkes, G., Bezemer, M., and Langendoen, K. (2010). The lambdamac framework: redefining MAC protocols for wireless sensor networks. *Wireless Networks*, 16(7):2013–2029.

Polastre, J., Hill, J., and Culler, D. (2004). Versatile low power media access for wireless sensor networks. In *2nd international conference on Embedded networked sensor systems (SenSys'04)*, pages 95–107, New York, NY, USA. ACM.

Rhee, I., Warrier, A., Aia, M., Min, J., and Sichitiu, M. L. (2008). Z-MAC: a hybrid mac for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 16(3):511–524. SenSys'05.

Ross, S. M. (1996). *Simulation (Statistical Modeling and Decision Science)*. Academic Press, second edition.

Tanenbaum, A. S. (2002). *Computer networks (4. ed.)*. Prentice Hall.

Van Dam, T. and Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. In *International Conference on Embedded networked Sensor Systems SenSys'03*, pages 171–180, New York, NY, USA. ACM.

Ye, W., Heidemann, J., and Estrin, D. (2004). Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506. INFOCOM'02.

Ye, W., Silva, F., and Heidemann, J. (2006). Ultra-low duty cycle mac with scheduled channel polling. In *4th ACM Conference on Embedded Network Sensor Systems SenSys'06*, pages 321–334, New York, NY, USA. ACM.

Zheng, T., Radhakrishnan, S., and Sarangan, V. (2005). PMAC: an adaptive energy-efficient MAC protocol for wireless sensor networks. In *19th International Parallel and Distributed Processing Symposium (IPDP'05)*.

NodePM: Um Sistema de Monitoramento Remoto do Consumo de Energia Elétrica via Redes de Sensores sem Fio

Geraldo P. R. Filho[1], Jó Ueyama[1], Leandro A. Villas[2], A. R. Pinto[3], Sibelius Seraphini[1]

[1]Instituto de Ciência Matemáticas e de Computação – ICMC
Universidade de São Paulo – USP
13566 – 590 – São Carlos – SP – Brasil

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

[3]Universidade Estadual Paulista – UNESP
São José do Rio Preto – SP – Brasil

{geralop, joueyama}@icmc.usp.br, leandro@ic.unicamp.br,
arpinto@ibilce.unesp.br, sibelius@grad.icmc.usp.br

Abstract. *This article proposes NodePM, a autonomous method for novelty detection in electronic equipments monitored by a smart grid. Considering the entropy of each equipment, which is calculated based on a markov chain model, the method uses a machine learning algorithm for the detection of novelties. NodePM is integrated to a energy consumption monitoring platform composed of a WSN and a cloud-based application. The results of the experiments showed the efficiency of the method in detecting novelties in electronic equipment which has made their use viable in our platform.*

Resumo. *Neste artigo, é proposto um método inteligente, nomeado de NodePM, para detectar novidades em equipamentos eletrônicos monitorados por uma smart grid. Considerando a entropia de cada equipamento monitorado, a qual é calculada com base em um modelo de cadeia de markov, o método proposto detecta as novidades por meio de um algoritmo de aprendizado de máquina. Além disso, o NodePM é integrado a uma plataforma de monitoramento remoto de consumo de energia, que consiste de uma RSSF associada a uma aplicação em nuvem. Os resultados obtidos evidenciaram a eficiência do método em detectar novidades nos equipamentos, demonstrando a viabilidade do uso deste na plataforma de monitoramento.*

1. Introdução

Nos últimos anos, percebeu-se uma crescente demanda de energia elétrica por parte das indústrias, comércios e residências. Essa situação está presente tanto no cenário brasileiro quanto mundial. No período entre 1999 e 2009, o consumo *per capta* de energia elétrica no Brasil e no mundo aumentou 20% e 22% respectivamente [DATA 2012]. Tal cenário, portanto, exige um sistema elétrico mais inteligente que permita reduzir o consumo de energia elétrica em cada equipamento eletrônico, encorajando os consumidores a implementar estratégias eficientes para a redução do consumo de energia.

A tecnologia da informação para o sistema elétrico de potência, integrada aos sistemas de comunicação e infraestrutura de rede elétrica, conhecida como *smart grid*

[ENERGY 2012], permite monitorar e gerenciar o sistema de energia elétrica, em qualquer lugar a qualquer momento. Nesse sentido, a utilização das *smart grids* tem se tornado cada vez mais importante no cenário urbano, pois oferecem integração em diversas fontes de energia, tais como hidrelétrica, eólica, solar e atômica.

Espera-se que a utilização da *smart grid* venha a ser uma realidade nos próximos anos, uma vez que as indústrias, universidades e governos do mundo inteiro têm dedicado recursos expressivos para o desenvolvimento dessa tecnologia. Essa tendência pode ser confirmada por meio de diferentes projetos e iniciativas nacionais e internacionais do governo, indústria e academia, dedicados às *smart grids* [AlertMe 2012], [ENERGY 2012], [MAGGI 2012], [Erol-Kantarci and Mouftah 2010], [Will et al. 2009].

Apesar de todos os avanços conquistados nessa área, as concessionárias de energia elétrica não oferecem suporte para as instalações da rede elétrica nas residências (para, por exemplo, gerenciar o consumo de energia remotamente). Isso significaria mais custos para a empresa. Além disso, essas concessionárias fornecem somente o consumo total de energia gasto em uma casa. Determinar quais são as fontes individuais (equipamentos eletrônicos) que possuem maior influência na conta de luz não é uma tarefa trivial.

Assim, um dos possíveis caminhos propostos [Power-Meter 2012], [AlertMe 2012], [Campus-Metabolism 2012], [Erol-Kantarci and Mouftah 2011], [Duarte et al. 2011], [Jota et al. 2006], [Botte et al. 2005], para resolver esse problema é integrar uma Rede de Sensores Sem Fio (RSSF) nos equipamentos eletrônicos da residência. A partir de uma RSSF é possível estabelecer um sistema que monitora em tempo real o uso de cada tomada da casa, permitindo ao usuário conhecer o seu perfil de consumo de energia. A partir disso, o usuário pode descobrir se existe algum tipo de desperdício de forma pontual e efetuar as devidas correções.

Estudos mostram que fornecer informações sobre como, quando, onde e o que os usuários estão usando os ajuda a tomar decisões corretas [Mcmakin et al. 2002], [Stern 1999], [Ester 1985]. É fundamental, portanto, usar nas *smart grids* métodos inteligentes para detectar e enviar novidades, de forma individual e autônoma, para os usuários quando algo anômalo surge nos seus equipamentos eletrônicos. Tais anomalias podem surgir, por exemplo, quando um equipamento consome mais energia do que o esperado e/ou quando o equipamento começa a ter um comportamento fora do padrão.

Nesse contexto, este artigo propõe um método inteligente, nomeado de *Novelty Detection Power Meter* (NodePM), para detectar novidades em equipamento eletrônicos monitorados por uma *smart grid*. Considerando a entropia de cada equipamento monitorado, a qual é calculada com base em um modelo de cadeia de markov, o método proposto detecta as novidades por meio de um algoritmo de Aprendizado de Máquina (AM). O NodePM é integrado a uma plataforma de monitoramento remoto de consumo de energia elétrica, que consiste de uma RSSF associada a uma aplicação em nuvem. Dessa forma, é possível enviar alertas autônomos de forma inteligente aos usuários (por exemplo, em um *smartphone*) quando algo anômalo surge nos equipamentos eletrônicos. Para avaliar a qualidade do método proposto, foi realizada uma análise de desempenho tendo como parâmetro de comparação o *Self-Organizing Novelty Detection* (SONDE).

Os resultados obtidos mediante a análise estatística evidenciaram a viabilidade do método na plataforma desenvolvida, obtendo desempenho satisfatório na detecção de

novidades no ambiente monitorado (veja a Seção 6).

Nesse sentido, este trabalho diferencia-se das soluções existentes em pelo menos três aspectos: (i) faz uso de um método inteligente para monitorar o consumo de energia elétrica; (ii) usa técnicas de AM para analisar o comportamento dos equipamentos eletrônicos por meio das RSSFs; (iii) e envia alertas de forma inteligentes ao *smartphone* quando algo anômalo surge.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a estratégia utilizada para o desenvolvimento do trabalho e o método proposto. A Seção 4 apresenta o *baseline* para o NodePM. A Seção 5 apresenta a base de dados utilizada. A Seção 6 apresenta os resultados dos experimentos. Finalmente, a Seção 7 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Diversos trabalhos têm sido publicados na área de *smart grid* ao longo dos anos e esta seção apresenta trabalhos científicos [AlertMe 2012, Campus-Metabolism 2012, Power-Meter 2012, Duarte et al. 2011, Erol-Kantarci and Mouftah 2010, Jota et al. 2006, Botte et al. 2005] cujo foco corresponde, principalmente, ao monitoramento do consumo de energia elétrica. No entanto, apesar de todos os recentes avanços conquistados, ainda existem vários desafios e problemas em aberto nessa área. Por exemplo, a inexistência de uma metodologia para detectar anomalias/novidades no ambiente monitorado.

Um dos modelos mais antigo de uma *smart grid* é o projeto Telegestore. O Telegestore [Botte et al. 2005] é um sistema que gerencia os medidores residenciais e comerciais remotamente, visando explorar a rede de distribuição de baixa tensão entre os transformadores e os medidores. As principais desvantagens do Telegestore são: (i) a desconsideração de um método para detectar anomalias (por exemplo, *black-out*) na rede de baixa tensão; (ii) e, o monitoramento por setores não é explorado.

[Erol-Kantarci and Mouftah 2010] propõem o uso da RSSF para gerenciar a energia elétrica de uma residência por meio da *smart grid*. Para tanto, os pesquisadores propuseram um *Appliance Coordination* (ACCORD) para reduzir o custo de energia na hora de pico. As desvantagens do ACCORD são: (i) não aproveitar os benefícios que as RSSFs oferecem (por exemplo, monitoramento por setores); (ii) a falta de um método para detectar novidades no ambiente monitorado; (iii) e, por fim, os experimentos são simulados.

No Brasil, também há iniciativas em relação às *smart grids*, por exemplo, o Centro de Monitoramento de Usos Finais (CMUF) [Jota et al. 2006]. O projeto CMUF objetiva ajudar as pessoas a gerenciar o consumo de energia elétrica remotamente, consistindo em um sistema de baixo custo. Porém, as principais desvantagens do CMUF são: (i) não integrar na plataforma um método para detectar novidades no ambiente monitorado; (ii) e, devido à sua natureza centralizada, o sistema pode ser vulnerável a sobrecarga de dados.

O trabalho que mais se assemelha a este artigo é do [Duarte et al. 2011]. Os pesquisadores propõem um *smart meter* para fazer a medição da energia elétrica em uma residência. O medidor consiste em monitorar o consumo de energia elétrica indicando os equipamentos de maior consumo dentro do domicílio. Apesar da semelhança, as desvantagens do *smart meter* são: (i) a plataforma não utiliza um método para detectar novidades nos equipamentos eletrônicos; (ii) e, não usa um servidor em nuvem para facilitar o geren-

ciamento das informações.

3. Sistema de Monitoramento Remoto do Consumo de Energia em uma Residência

Algumas etapas foram necessárias para a realização deste trabalho, sendo elas: (i) o desenvolvimento de uma plataforma na qual utiliza uma RSSF e computação em nuvem; (ii) o uso do NodePM para detectar novidades no ambiente a ser monitorado; (iii) e, por fim, o desenvolvimento de uma aplicação para o usuário visualizar as informações de consumo de energia em sua residência.

Deve-se ressaltar, portanto, que a plataforma desenvolvida e apresentada a seguir é de simples instalação e fácil utilização, assim não é necessário ter um especialista na área. O NodePM é integrado na plataforma a fim de detectar novidades no ambiente monitorado. Além disso, o sistema desenvolvido permite monitorar o consumo de energia elétrica em qualquer lugar a qualquer momento.

3.1. Plataforma construída usando RSSF e computação em nuvem

A plataforma proposta é composta por três etapas:

- A Etapa 1 é responsável pela aquisição dos dados de consumo dos equipamentos eletrônicos por meio de uma RSSF.
- A Etapa 2 está relacionada com o processamento dos dados coletados na Etapa 1. Para tanto, foi desenvolvida uma aplicação em um servidor em nuvem (*app engine*[1]), no qual gerencia os dados recebidos da RSSF.
- Por fim, a Etapa 3 é responsável pela disponibilização das informações sobre o consumo de energia elétrica para os interessados. Por isso, uma aplicação para o *smartphone* foi desenvolvida.

Na Figura 1, é apresentado o funcionamento da plataforma de monitoramento remoto do consumo de energia elétrica. Para isso, foram construídos protótipos de RSSF (Rótulo 1 Figura 1) a partir da montagem de *wattímetros* (equipamentos que mede o consumo de energia elétrica) *Kill-a-Watt* da empresa P3[2]. Os *wattímetros* são ligados diretamente em uma tomada e junto a ela é conectado o equipamento que deseja monitorar. Como os *wattímetros* não possuem acesso a nenhum meio de comunicação, foi adicionado a eles um módulo XBee[3] a fim de transmitir as informações de consumo de energia para um servidor. Dessa forma, construiu-se uma infraestrutura, *gateway* Arduino[4], (Rótulo 2 da Figura 1) para enviar as informações via RSSFs e transmitir os dados lidos para o servidor em nuvem (Rótulo 3 da Figura 1, onde é implementado o método proposto). O servidor recebe os dados da RSSF, processa tais dados (detectando novidades) e gerencia (dados de consumo) o envio das informações, sendo possível fazer o monitoramento remoto por meio de um *smartphone* (Rótulos 4 e 5 da Figura 1).

Nesse sentido, o usuário recebe os alertas e visualiza as informações mais importantes do consumo de energia no Rótulo 4 da Figura 1. Já no Rótulo 5 da Figura 1, é feito o monitoramento por setores, tais como: o equipamento, cômodo da casa e/ou toda região de atuação, somente possível por causa da RSSF.

[1] App Engine - Developers, https://developers.google.com/appengine/

[2] P3 International innovative electronic solutions, http://www.p3international.com

[3] Xbee-pro module datasheet, http://ftp1.digi.com/support/documentation/90000976G.pdf

[4] Arduino, http://www.arduino.cc

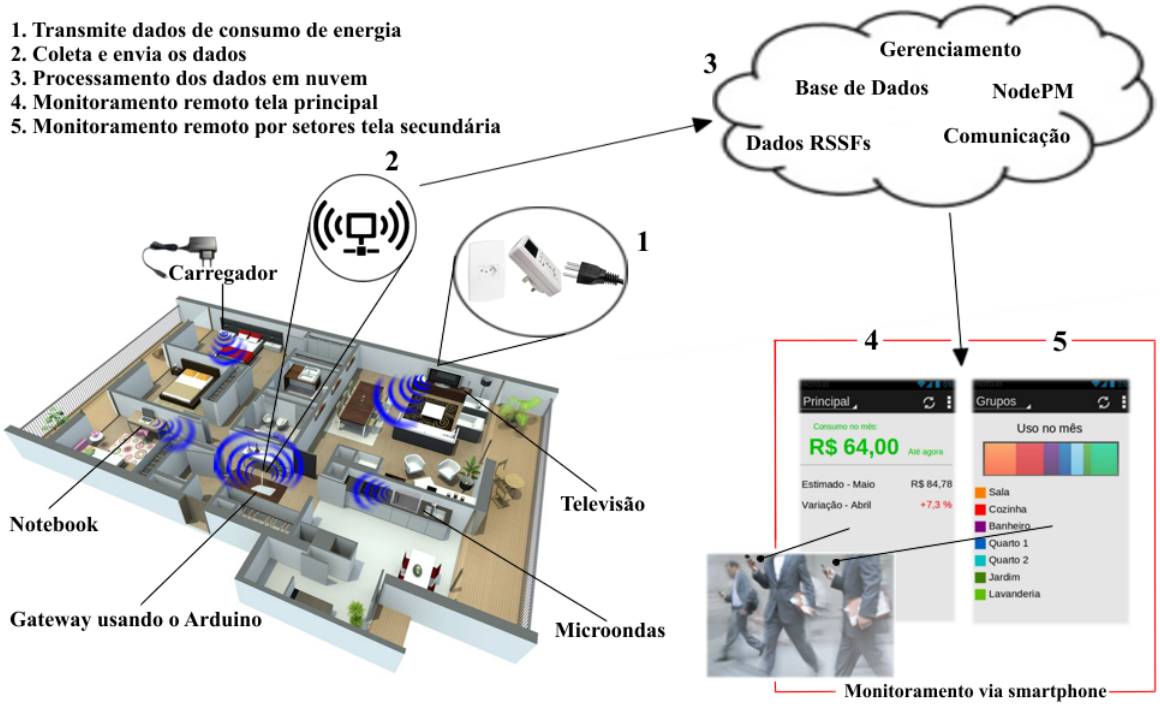


Figura 1. Cenário de funcionamento da plataforma.

3.2. Visão Geral do *Novelty Detection Power Meter* (NodePM)

O método proposto, Algoritmo 1, é nomeado como *Novelty Detection Power Meter* (NodePM) justamente por ser integrada na plataforma, servindo como um ponto de interconexão de envios de alertas entre o medidor e o usuário.

Algoritmo 1: *Novelty Detection Power Meter*

```
Entrada: X=(X1...Xn);                          /* conjunto de dados */
Saída: mensagens de alertas para o dispositivo móvel
Dados: Threshold, Markov;
1 inicio
2     Estado ⇐ KNN(X);                  /* classifica a instância desconhecida */
3     Markov ⇐ AtualizarCadeiaMarkov(Markov, Estado);   /* incorporar a
      instância classificada em um estado da cadeia de Markov */
4     Probabilidade ⇐ GetProbabilidade(Markov);   /* vetor de probabilidade */
5     DeltaEntropia ⇐ Shannon(Probabilidade);     /* variação da entropia */
      /* detectando novidade no equipamento eletrônico                    */
6     se DeltaEntropia ≥ Threshold então
7         Enviar(Dispositivo);        /* envia mensagem para o dispositivo móvel */
8     fim
9 fim
```

O NodePM é modelado usando os conceitos de AM, cadeias de Markov e entropia. Foram identificadas três etapas distintas para o funcionamento do Algoritmo 1.

- Primeira etapa, processar os dados por meio de um classificador, *K-Nearest Neighbors* (KNN) (Linha 2 do Algoritmo 1), para catalogar o estado comportamental

dos equipamentos eletrônicos.

- Segunda etapa, capturar o elemento catalogado na primeira etapa e adicionar no estado comportamental da cadeia de Markov (Linha 3 do Algoritmo 1).
- Terceira etapa, obter a matriz de probabilidade da cadeia de Markov e calcular o grau de incerteza dos equipamentos eletrônicos usando a variação da entropia (Linhas 4 e 5 do Algoritmo 1).

As etapas descritas anteriormente são reorganizadas, nas próximas subseções, para um melhor entendimento do método.

3.2.1. Classificação de Padrões Comportamentais

Nesta subseção é apresentada a técnica escolhida, KNN, para classificar os dados recebidos da RSSF, com o intuito de incorporá-los na cadeia de Markov (Subseção 3.2.2).

A escolha do KNN é justificada por dois motivos: (i) apesar de simples, tem mostrado ser uma das técnicas mais eficazes já proposta na literatura; (ii) e, é uma técnica, tradicional e efetiva aplicada em vários problemas de classificação [Yang and Liu 1999].

Para facilitar o entendimento da técnica e o seu funcionamento na plataforma, a Figura 2 ilustra como os dados de entrada são classificados. Os gráficos da Figura 2 representam a potência de um equipamento em função do tempo (minuto), sendo possível perceber quatro comportamentos distintos (*standby*, desligado, normal[5] e não esperado[6]) de consumo de energia.

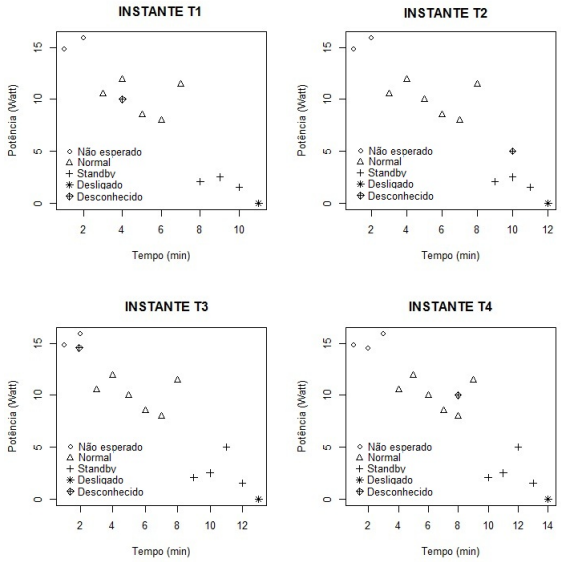


Figura 2. Exemplo de classificação do KNN.

Deve-se ressaltar, portanto, que o KNN classifica as instâncias desconhecidas analisando os K vizinhos mais próximos. Neste caso, a instância "desconhecida" no INSTANTE T1 (Figura 2) é classificada como uma instância "normal". Após a classificação, é fundamental destacar, que o elemento classificado é adicionado incrementalmente na base de treinamento, a fim do método se adaptar de forma autônoma em ambientes dinâmicos.

[5]Representa o equipamento dentro de um padrão, a geladeira por exemplo fica ligada constantemente.
[6]Representa o equipamento fora do padrão e/ou consumindo energia acima do esperado.

A adaptação do ambiente, neste trabalho, ocorre a cada 5 segundos, tempo estipulado pela plataforma para enviar os dados. Dessa forma, o INSTANTE T2, T3 e T4 seguem o mesmo processo que o INSTANTE T1.

Assim, a próxima etapa do método consiste em adicionar a instância classificada, a cada instante de tempo, em um estado da cadeia de Markov.

3.2.2. Obtenção de Padrões Comportamentais

Para obtenção de padrões comportamentais a cadeia de Markov [Markov 1971] é utilizada. Assim, conforme o KNN classifica as instâncias desconhecidas, uma nova cadeia de Markov é gerada. Na Figura 3 é exibida a matriz de probabilidade e a cadeia de Markov em cada um dos instantes de tempo, como apresentado anteriormente na Figura 2, a fim de representar o comportamento do equipamento naquele instante de tempo.

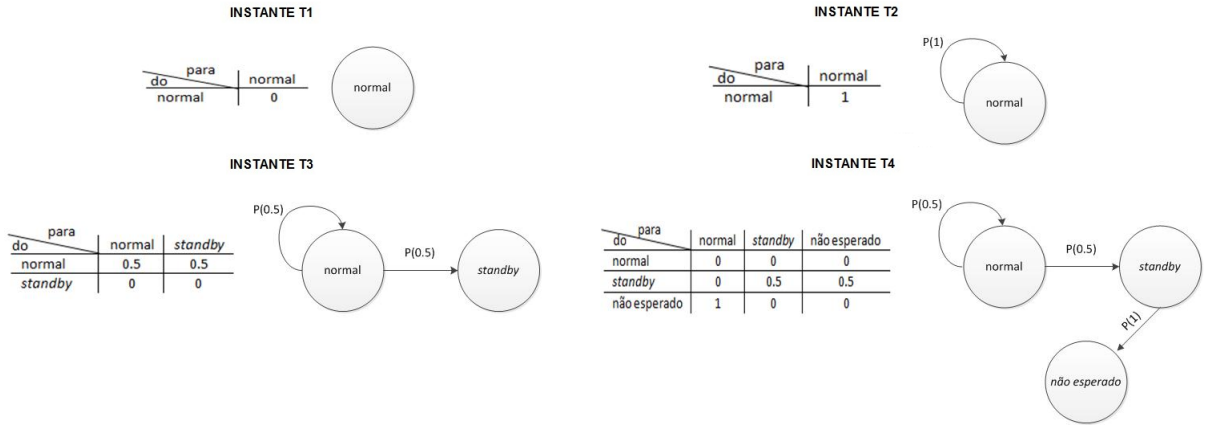


Figura 3. Matriz de probabilidade e cadeia de Markov a cada instante de tempo de acordo com a Figura 2.

Nesse sentido, a matriz de probabilidade e a cadeia de Markov são atualizadas de acordo com cada nova classificação do KNN (veja o paralelo da Figura 2 com a Figura 3). Neste caso, no INSTANTE T1 (Figura 3) classificou-se a primeira instância como "normal", porém sem transição na cadeia de Markov. No INSTANTE T2 (Figura 3), a segunda instância também foi classificada como "normal", no entanto há uma transição entre a primeira e segunda instância classificada, com isso, a matriz de probabilidade é atualizada, apresentando uma transição **do** estado "normal" **para** o estado "normal". O INSTANTE T3 e T4 da Figura 3 seguem o mesmo processo. Assim, os estados da cadeia de Markov são criados dinamicamente a medida que o KNN classifica o comportamento do equipamento para aquele instante de tempo.

Após classificar, representar o comportamento dos equipamentos mediante a cadeia de Markov e obter as probabilidades a cada instante de tempo é necessário calcular o grau de incerteza dos equipamentos por meio da entropia da cadeia de Markov.

3.2.3. Detecção de Novidades com a Variação da Entropia da Cadeia de Markov

Com o conjunto das probabilidade das cadeias de Markov gerado torna-se possível calcular a variação da entropia a cada instante de tempo, por meio da seguinte fórmula:

$$H(X) = -\sum_{i=1} p(x_i) \log_b p(x_i)$$

Onde $\log_b$ denota o logaritmo de *x* na base 2 e o $p(x_i)$ é a probabilidade de um evento ir para um outro estado (matriz de probabilidade da Figura 3). Dessa forma, cada interação do usuário com o equipamento gera uma curva de energia, a qual representa as alterações comportamentais dos equipamentos. Na Seção 6.1 são realizados experimentos dos tipos de novidades detectadas nos equipamentos.

4. O *Baseline* para o Método Proposto

O *Self-Organizing Novelty Detection* (SONDE) é uma rede neural que adapta incrementalmente sua estrutura de conhecimento (neurônio), a fim de detectar novidades em ambientes dinâmicos. Para isso, a SONDE classifica em um mesmo neurônio padrões de entradas similares. Quando nenhum neurônio é capaz de classificar um padrão de entrada, um novo neurônio é criado indicando uma novidade no ambiente [Albertini and Mello 2007].

A escolha da SONDE pode ser justificada por quatro motivos: primeiro, a SONDE pode ser utilizada em qualquer sistema e/ou aplicação independente da base de dados; segundo, o método detecta eventos inesperados em ambientes dinâmicos; terceiro, o treinamento e adaptação são realizados sem a intervenção de um especialista; e, por fim, quarto, a SONDE obteve melhor desempenho quando comparado com *Grow When Required* [Albertini and Mello 2007] e [Marsland et al. 2002].

5. Base de Dados

O método proposto utilizou um conjunto de dados (*dataset*), contendo informações reais sobre a interação do usuário com o equipamento eletrônico. Essa base de dados possui quatro atributos, sendo eles: (i) identificador do equipamento atribuído pelo sensor; (ii) potência em *Watt*; (iii) a data de utilização do equipamento; (iv) e o tempo de uso do equipamento naquele momento.

Deve-se ressaltar, no entanto, que o único atributo normalizado para o tipo inteiro é a data, a qual representa o número de dias decorridos a partir de uma data fixa. Esse atributo é normalizado pela própria plataforma. Esta utiliza o JSON[7] para transmitir os dados em um formato padronizado. Assim, o método não necessita fazer qualquer normalização dos atributos recebidos.

Após o envio dos dados, foram realizadas oito partições das amostras do conjunto de dados coletados de três meses (veja Seção 6). Cada partição contém um conjunto de dados de 1 e 2 semanas. Cada elemento do conjunto (1 semana e 2 semanas) foi dividida em 25% de instâncias de treinamento e 75% em instâncias de testes. Essa técnica é conhecida como *hold-out*, no qual divide a base de dados em dois conjuntos (conjunto de treino e teste) [Mitchell 1997]. Para não existir grau de dependência nos experimentos, da Subseção 6.2, a divisão do conjunto de treino e teste foi replicada aleatoriamente 11 vezes para cada conjunto de parâmetros da Tabela 1. As replicações em um mesmo subconjunto têm o objetivo de contemplar os diferentes comportamentos que os usuários possuem (por exemplo, comportamento diferente no final de semana em relação aos demais dias).

[7]JSON, http://www.json.org/

6. Experimentos e Análise dos Dados Obtidos

A validação do método proposto, foi dividida em duas etapas: a primeira, Seção 6.1, apresenta e discute as novidades/anomalias encontradas nos equipamentos eletrônicos; a segunda, Seção 6.2, é realizado uma avaliação de desempenho do NodePM com a SONDE.

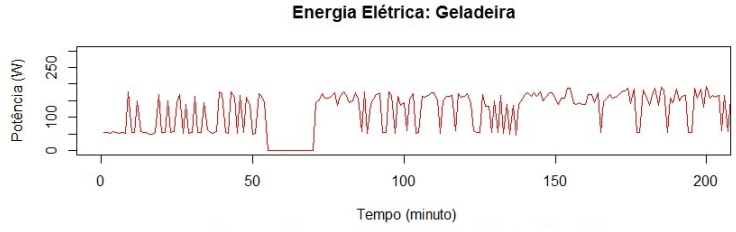
Para produzir resultados com boa precisão, foi montando um ambiente real para monitorar o consumo de energia dos equipamentos eletrônicos em uma residência. Nesse cenário, os equipamentos se localizavam em cômodos diferentes e, por conveniência, o *gateway* Arduino estava junto do roteador para comunicar com a Internet. A coleta de dados foi feita durante três meses e nesse período foi observado o comportamento do sistema por meio do perfil de consumo do usuário.

6.1. Experimentos de Detecção de Novidades com o NodePM

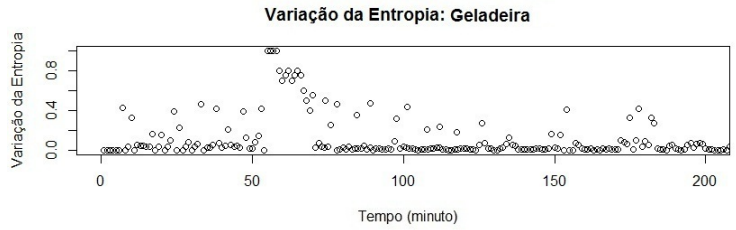
Com os dados coletados foi possível realizar experimentos com o método, o qual apresentou dois tipos de novidades (qualitativa e quantitativa):

- A 1ª ocorre quando o equipamento eletrônico tem uma mudança abrupta no seu comportamento padrão, ou seja, ocorreu uma troca não habitual no seu estado, passando do estado x para um estado y, no qual não era esperado. (qualitativa)
- A 2ª acontece quando o equipamento eletrônico começa a consumir energia mais que o esperado durante um determinado período. (quantitativa)

A Figura 4 apresenta o primeiro tipo de novidade, como descrito anteriormente, para um equipamento eletrônico (geladeira). O Gráfico 4b, no qual apresenta a variação da entropia em função do tempo (minuto) é possível notar tanto o comportamento padrão (sem novidade) entre o período de 0 a 54 e 71 a 200 minutos quanto o comportamento não esperado (com novidade) entre o período de 55 a 70 minutos.



(a) Consumo de energia elétrica da geladeira durante um período acontecendo algo inesperado entre o período de 55 a 70 minutos.

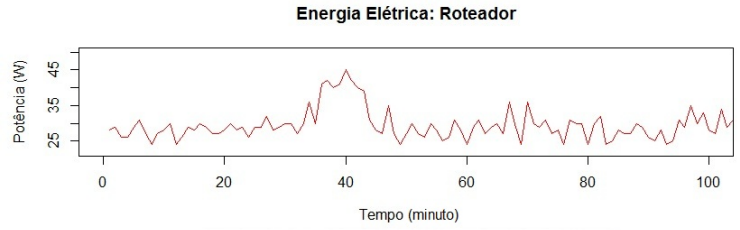


(b) Detectando novidade com uma mudança inesperada no comportamento da geladeira de acordo com o Gráfico 4a.

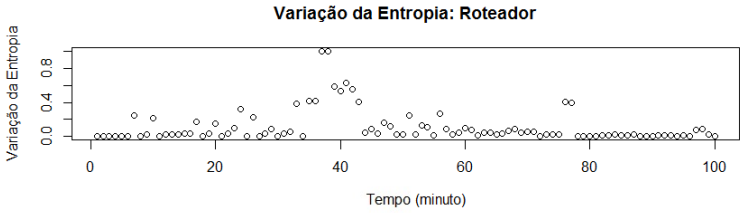
Figura 4. Detecção de novidades causada pela mudança no comportamento do equipamento.

Como o Gráfico 4a apresenta a potência consumida em função do tempo (minutos) é essencial observar o comportamento da geladeira a cada período. Nesse sentido, no intervalo de tempo entre 55 e 70 (Gráfico 4b) nota-se uma alteração repentina na entropia, causada pela mudança no padrão de comportamento da geladeira (Gráfico 4a). Esta mudança abrupta surgiu devido ao desligamento da geladeira durante 15 minutos (veja o Gráfico 4a no intervalo de 55 a 70), ou seja, o equipamento saiu do seu estado habitual o que não era esperado. Apesar disso, a variação da entropia começa a estabilizar depois dos 66 minutos. Essa estabilização pode ocorrer de duas maneiras: a primeira é devido à mudança de hábito do usuário (causada por *feedback* por intermediação do *smartphone*), e a segunda é que no decorrer do tempo o método considera que aquela novidade já não é algo inesperado.

O segundo tipo de novidade (Figura 5) está relacionado com o consumo elevado de energia elétrica, do roteador, durante um determinado período (quantitativo). O Gráfico 5a representa bem esse comportamento, expondo a potência consumida em função do tempo (minuto). Nesse gráfico é perceptivo um elevado pico de energia no período de 36 a 43 minutos. Sendo assim, o Gráfico 5b que representa a variação da entropia em função do tempo (minuto) constata a ocorrência de uma novidade no mesmo instante de tempo, que o Gráfico 5a, devido a uma mudança abrupta na variação da entropia. Vale destacar, que a estabilização da variação da entropia ocorre similar ao primeiro tipo de novidade, ou seja, a estabilização pode ocorre por dois motivos, como descrita anteriormente.



(a) Consumo de energia elétrica do roteador durante um determinado período.



(b) Detectando novidade no mesmo instante de tempo que o Gráfico 5a.

Figura 5. Detecção de novidade com o aumento do consumo inesperado de energia elétrica do roteador.

Apesar de existir dois tipos de novidades, ambas podem estabilizar no erro como ocorrido no Gráfico 4b depois de 66 minutos. Este tipo de situação é considerado uma vantagem do ponto de vista do usuário. Visto que, caso o usuário não queira mudar o seu comportamento diante do equipamento defeituoso e/ou do consumo exagerado de energia elétrica, a metodologia utilizada não irá incomodar o usuário com os alertas enviados para o seu *smartphone*.

6.2. Avaliação de Desempenho com o NodePM

Esta subseção avalia o desempenho do NodePM, considerando as seguintes medidas de desempenho: sensibilidade (conhecida por alguns autores como taxa de detecção), precisão, especificidade e acurácia. Essas são calculadas a partir de uma matriz de confusão apresentada na Figura 6, no qual avalia os resultados com base nas perdas causadas.

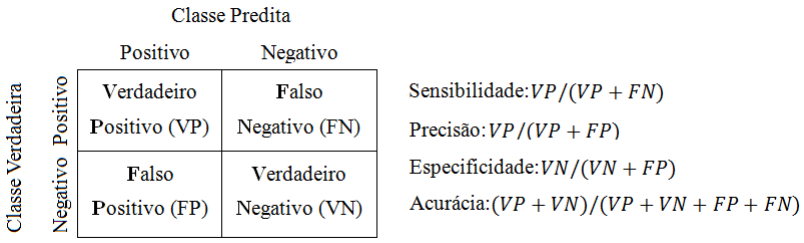


Figura 6. Medidas de desempenho calculadas a partir da matriz de confusão.

Segundo [Fawcett 2006], tais medidas possuem características inerentes, sendo a sensibilidade o total de amostras cuja classe resultante é realmente positiva (verdadeiros positivos); a precisão é o total de exemplos classificados como positivo, mas que nem sempre são, ou seja, os verdadeiros negativos; já a especificidade é o oposto da sensibilidade, importando somente em classificar como negativo os exemplos que de fato são negativos (falsos positivos) e a acurácia permitirá analisar o quanto preciso o método classifica adequadamente o comportamento de um equipamento.

Os conjuntos de parâmetros estabelecidos para realizar a avaliação de desempenho são apresentados pela Tabela 1. O sistema é avaliado com a alteração do *dataset* (1 semana e 2 semanas), com a mudança do equipamento (geladeira e roteador), e, com a troca dos métodos (NodePM e SONDE). Tais métodos utilizam o *dataset* no qual contém informações reais da interação do usuário com o equipamento. Cada conjunto de parâmetros foi executado 11 vezes para cada partição do conjunto de dados (apresentado anteriormente, Seção 5), a fim de alcançar uma estabilidade dos dados. Os resultados são apresentados na Figura 7, com um intervalo de confiança de 95%.

Tabela 1. Conjunto de parâmetros escolhidos para serem avaliados.

Experimento	Método	*Dataset*	Equipamento
E1	NodePM	1 Semana	Geladeira
E2	NodePM	1 Semana	Roteador
E3	NodePM	2 Semanas	Geladeira
E4	NodePM	2 Semanas	Roteador
E5	SONDE	1 Semana	Geladeira
E6	SONDE	1 Semana	Roteador
E7	SONDE	2 Semanas	Geladeira
E8	SONDE	2 Semanas	Roteador

O Gráfico 7 apresenta a porcentagem dos resultados obtidos dos experimentos de E1 até E8 (Tabela 1) em função das medidas sensibilidade, precisão, especificidade e acurácia para o *dataset* de 1 e 2 semanas. O resultado destaca que o método NodePM possui um desempenho superior em relação a SONDE quando considerado o *dataset* de 1 semana, independente do equipamento utilizado (experimentos {(E1,E5),(E2,E6)}). Isso ocorre devido ao NodePM usar uma abordagem supervisionada, ou seja, a base de dados possuem exemplos que estão rotulados com uma classe predita. Logo, o método não

necessita de muitas instâncias de treinamento para atingir bons resultados, pois com o *dataset* de 2 semanas o seu desempenho foi significativo (aproximadamente 3% de aumento, experimentos de E1 a E4). Porém, ao levar em consideração o *dataset* de 2 semanas os papéis se invertem, ou seja, a SONDE tem melhor desempenho que o NodePM, independente do equipamento utilizado (experimentos {(E3,E7),(E4,E8)}). Isto faz sentido, pois a SONDE é uma rede neural artificial não-supervisionada e auto-organizável. Assim, é necessário ter um *dataset* maior para que os neurônios da SONDE se adapte, de forma incremental, de acordo com novos padrões de entrada. Por isso, o seu elevado desempenho (aproximadamente 13% de aumento, experimentos de E5 a E8) está relacionado com o aumento das instâncias de treinamento.

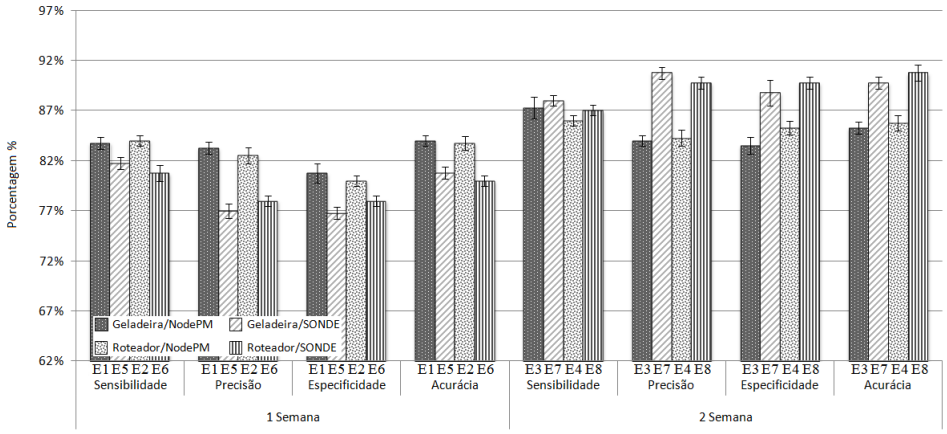


Figura 7. Análise de desempenho do NodePM de acordo com a Tabela 1.

6.2.1. Análise das Influências dos parâmetros

Esta subseção apresenta as análises das influências do conjunto de parâmetro consideradas na Tabela 1 em função das medidas de desempenho. As análises foram feitas utilizando um modelo de regressão multivariado, no qual inclui uma relação causal com mais de duas variáveis [Jain 1991]. Neste caso, o comportamento de uma medida de desempenho é explicado por mais de um parâmetro. Dessa forma, é possível analisar quais parâmetros e/ou suas combinações que mais influenciou nos resultados.

Tabela 2. Influência dos parâmetros em função das medidas de desempenho.

Medidas de Desempenho	A (%)	B (%)	C (%)	AB (%)	BC (%)	AC (%)	ABC (%)
Sensibilidade	3,06	81,05	2,25	12,25	0,57	0,25	0,57
Precisão	0,17	59,14	0,02	39,9	0,07	0,02	0,68
Especificidade	1,09	77,85	0,82	19,16	0,38	0,12	0,58
Acurácia	0,64	64,97	0,07	33,16	0,95	0,01	0,02

Na Tabela 2, são apresentados os parâmetros métodos, *dataset* e equipamento com as respectivas letras A, B e C. As combinações de duas ou mais letras são os percentuais das interações entre os parâmetros. Nota-se que o parâmetro com mais influência para todas as medidas de desempenho é justamente o *dataset* (B), no qual também pode ser observado nos resultados pelos pares dos experimentos (E1,E3), (E2,E4), (E5,E7), (E6, E8). O segundo parâmetro com mais influência é a combinação do *dataset* com o método

(AB). Tal influência faz sentido, pois é relacionada com a forte interação que o parâmetro A possui com o B. Deve-se ressaltar, que o parâmetro com menos influência é o equipamento independente do conjunto de parâmetro considerado. Essas afirmações estatísticas evidenciam a eficácia do NodePM na plataforma desenvolvida.

7. Conclusões e Trabalhos Futuros

Com base nos resultados, constatou-se que o NodePM foi viável na plataforma. Porém, cada método teve suas peculiaridades. Assim sendo, a escolha do método ficou intrínseco ao *dataset*, pois os dados coletados em tempo real da interação do usuário com o equipamento possuem características distintas. Dessa forma, se usuário utilizar equipamentos que são ligados constantemente, tais como geladeira, roteador e/ou *freezer*, no qual possuem mais dados com interação, o mais adequado é utilizar a SONDE (veja Figura 7, experimentos E7 e E8). Porém, caso o usuário interaja com equipamentos, tais como microondas, máquina de lavar e/ou cafeteira, no qual os aparelhos não são ligados constantemente tendo poucos dados de interação, o mais adequado é o NodePM (veja Figura 7, experimentos E1 e E2).

Este artigo apresentou as seguintes contribuições: (i) o uso de uma RSSF para a construção de um protótipo, no qual monitora o consumo de energia dos equipamentos eletrônicos individualmente; (ii) um método inteligente, NodePM, baseado nos conceitos de AM para detectar novidades em um ambiente monitorado; (iii) uma aplicação para *smartphone* no qual recebe as informações do consumo de energia pelo servidor em nuvem, apresentando as informações para o usuário; (iv) a utilização de um servidor em nuvem como solução para gerenciar tanto as informações recebidas das RSSFs quanto da utilização do método; (v) e finalmente, uma avaliação de desempenho do NodePM, apresentando o seu desempenho na detecção de novidades nos equipamentos eletrônicos monitorados por uma *smart grid*.

Como trabalho futuro, pretende-se desenvolver um *Middleware* adaptativo com base no OpenCom [Coulson et al. 2008], para mudar em tempo de interação qualquer método de acordo com o ambiente monitorado.

8. Agradecimentos

Os autores agradecem o apoio financeiro da CAPES, processo DS-7901025/M; FAPESP, processos 2012/12061-1, 2008/05346-4 e 2008/57870-9; e CNPq, 573963/2008-8.

Referências

Albertini, M. K. and Mello, R. F. (2007). A self-organizing neural network for detecting novelties. In *Proceedings of the 2007 ACM symposium on Applied computing*.

AlertMe (2012). Energy monitor save energy & lower electricity costs alerme. http://www.alertme.com. visitado em 13 de Março de 2012.

Botte, B., Cannatelli, V., and Rogai, S. (2005). The telegestore project in enel´s metering system. *International Conference on Electricity Distribution*, 18th.

Campus-Metabolism (2012). Arizona state university, (asu). http://cm.asu.edu/#app=4c23&6741-selectedIndex=1. visitado em 05 de julho de 2012.

Coulson, G., Blair, G., Grace, P., Taiani, F., Joolia, A., Lee, K., Ueyama, J., and Sivaharan, T. (2008). A generic component model for building systems software.

DATA, P. (2012). Indicadores do desenvolvimento mundial. http://www.google.com.br/publicdata/explore?ds=d5bncppjof8f9_. visitado em 02 de julho de 2012.

Duarte, L. F. C., Zambiaco, J. D., Airoldi, D., Ferreira, E. C., and Dias, J. A. S. (2011). Characterization and breakdown of the electricity bill using custom smart meters: a tool for energy-efficiency programs. *International journal of circuits, system and signal processing*.

ENERGY (2012). Smart grid — department of energy. http://energy.gov/oe/technology-development/smart-grid. visitado em 05 de março de 2012.

Erol-Kantarci, M. and Mouftah, H. (2011). Wireless sensor networks for smart grid applications. In *Electronics, Communications and Photonics Conference (SIECPC)*.

Erol-Kantarci, M. and Mouftah, H. T. (2010). Wireless sensor networks for domestic energy management in smart grids. *Biennial Symposium on Communications*, 25th.

Ester, P. (1985). Consumer behavior and energy conservation : a policy-oriented experimental field study on the effectiveness of behavioral interventions promoting residential energy conservation / by peter ester.

Fawcett, T. (2006). An introduction to roc analysis. *Elsevier*, 27:861 – 874.

Jain, R. (1991). The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling. *SIGMETRICS Perform.*

Jota, F. G., Jota, P. R. S., and Nobre, E. C. (2006). Gerenciamento efetivo de energia por uso final: Um sistema de monitoramento de baixo custo via internet. *Seminário Nacional de Distribuição de Energia Elétrica*, XVII.

MAGGI, B. (2012). Projeto de lei nº 84/2012. http://www.senado.gov.br/atividade/materia/-getPDF.asp?t=105231&tp=1.

Markov, A. A. (1971). *Extension of the Limit Theorems of Probability Theorty to a Sum of Variables Connected in a Chain*. In Dynamic Probabilistic System.

Marsland, S., Shapiro, J., and Nehmzow, U. (2002). A self-organising network that grows when required. *Neural Netw.*, 15:1041–1058.

Mcmakin, A. H., Malone, E. L., and Lundgren, R. E. (2002). Motivating residents to conserve energy without financial incentives abstract:.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science.

Power-Meter (2012). Google powermeter: A google.org project. http://www.google.com/powermeter. visitado em 14 de Março de 2012.

Stern, P. C. (1999). Information, incentives, and proenvironmental consumer behavior. *Journal of Consumer Policy*, 22:461–478.

Will, J., O´Connell, T., and Lange, C. (2009). Smart grid is a global priority. *Smart Grid*. https://courses.cit.cornell.edu/crp384/2009reports/LangeOt

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*.

QoE-aware Multiple Path Video Transmission for Wireless Multimedia Sensor Networks

Denis Rosário[1,2], Rodrigo Costa[1], Aldri Santos[3], Torsten Braun[2], Eduardo Cerqueira[1]

[1]Federal University of Pará (UFPA), Belém – PA – Brazil

[2]Institute of Computer Science and Applied Mathematics, University of Bern, Bern – Switzerland

[3]Federal University of Paraná (UFPR), Curitiba – PR – Brazil

denis@ufpa.br, rodrigomc@ufpa.br, aldri@inf.ufpr.br,

braun@iam.unibe.ch, cerqueira@ufpa.br

Abstract. *Wireless Multimedia Sensor Networks (WMSNs) promise a wide scope of emerging potential applications in both civilian and military areas, which require visual and audio information to enhance the level of collected information. The transmission of multimedia content requires a minimal video quality level from the user's perspective. However, links in WMSN communications are typically unreliable, as they often experience fluctuations in quality and weak connectivity, and thus, the routing protocol must evaluate the routes by using end-to-end link quality information to increase the packet delivery ratio. Moreover, the use multiple paths together with key video metrics can enhance the video quality level. In this paper, we propose a video-aware multiple path hierarchical routing protocol for efficient multimedia transmission over WMSN, called video-aware MMtransmission. This protocol finds node-disjoint multiple paths, and implements an end-to-end link quality estimation with minimal overhead to score the paths. Thus, our protocol assures multimedia transmission with Quality of Experience (QoE) and energy-efficiency support. The simulation results show the benefits of video-aware MMtransmission for disseminating video content by means of energy-efficiency and QoE analysis.*

1. Introduction

The proliferation of multimedia content and the demand for new audio/video services in Internet of Things (IoT) applications [Zhou and Chao 2011] have fostered the development of a new era based on multimedia information. Those applications allowed the evolution of Wireless Multimedia Sensor Networks (WMSNs) [Almalkawi et al. 2010], which enable a large class of scenarios ranging across diverse areas, e.g., environmental monitoring, video surveillance, traffic control, smart cities, and other IoT applications. The multimedia content in such applications gives support to the end-users (or systems) to visually verify the real impact of events, and be aware of what is happening in the environment with rich visual information.

Multimedia delivery demands high bandwidth, real-time transmission, lower frame loss, tolerable end-to-end delay and jitter. Additionally, applications involving multimedia dissemination should support Quality of Service (QoS) and Quality of Experience

(QoE) to deliver video content with, at least, a minimal video quality level from the user's perspective together with energy-efficiency and scalability [Jaime et al. 2010]. Those issues impose more constraints on the design of a routing protocol for WMSNs. Further, frames with different priorities compose a compressed video, and from a human's experience, the loss of high priority frames causes severe video distortion. Thus, a key principle in a QoE-aware routing protocol for WMSNs in IoT applications is the transmission of high priority frames (protect) with minimum packet loss [Greengrass et al. 2009].

Nevertheless, low-power radios employed by WMSNs are very sensitive to noise, interference, and multipath distortion, which cause link unreliability. These issues also impose serious effects on wireless communication quality, due to significant link quality fluctuations and weak connectivity [Baccour et al. 2011]. In this context, link quality estimation is an essential metric to enable the nodes to dynamically plan, adapt and take appropriate actions when making a routing decision. However, a reliable mechanism to support route selection based on both cross-layer information and end-to-end link quality estimation with minimum overhead is still needed [Gomez et al. 2010].

Multi-path routing aims to provide load balance, bandwidth aggregation and reduced delay, compared to single-path transmission [Radi et al. 2012]. Though, the multiple paths must be node-disjoint to avoid a common node among different paths. Node-disjoint multi-path routing mitigates packet losses at relay nodes with restricted buffer, and thus, enhances the packet delivery ratio. Additionally, WMSN route selection should consider both cross-layer information and end-to-end link quality estimation to find node-disjoint multi-path routes. Moreover, priority frames must be protected in congestion/link error periods. However, current WMSN routing protocols do not consider all of these key characteristics to support QoE-aware multimedia transmission.

This paper proposes a video-aware multiple path hierarchical routing protocol for efficient multimedia transmission over WMSN, called *video-aware MMtransmission*. The proposed protocol assures multimedia transmission with both QoE and energy-efficiency support. In specific terms, video-aware MMtransmission finds reliable node-disjoint multiple paths, and evaluates in an end-to-end fashion by using cross-layer information. The route discovery includes a minimum signaling overhead, as expected for many IoT applications. Moreover, the protocol uses key video information to disseminate high priority video frames via more reliable paths with the aims of increase the video quality level.

We performed simulations to evaluate the energy-efficiency and video quality level of video-aware MMtransmission. We analyzed the video quality by means of two well-known QoE objective metrics, namely *Structural Similarity (SSIM) and Video Quality Metric (VQM)*. The results showed that the proposed protocol provides multimedia distribution with a higher quality level compared to other approaches. Our protocol has a SSIM gain of approximately 10% and a VQM gain of 10% to 40%.

The remainder of the paper is structured as follows. Section 2 outlines existing routing protocols for WMSNs. Section 3 describes the video-aware MMtransmission protocol for WMSNs, which was evaluated by means of simulation experiments as presented in Section 4. Section 5 presents main contributions and results of this paper.

2. Related Work

Several attempts have been made to achieve promising results in routing protocols for WMSN applications, and in general such works focus on minimizing energy consumption and meeting certain QoS requirements for multimedia distribution. Further, they establish a hierarchical architecture to achieve lower energy consumption, higher functionality, better scalability and greater reliability [Ehsan and Hamdaoui 2011].

[Kandris et al. 2011] proposed Power Efficient Multimedia Routing (PEMuR) based on a combination of hierarchical routing and video packet scheduling models to support efficient video communication in WMSNs. PEMuR considers only the remaining energy to find routes (not link quality), and this issue limits the transmission of multimedia content with QoS/QoE support due to unreliability of the low-power wireless links. As PEMuR also relies on a centralized scheme to create clusters, it does not consider multiple paths to balance load, aggregate bandwidth, and reduce delay. [Politis et al. 2008] introduce a multi-path routing over a hierarchical architecture, and employs two packet scheduling algorithms to transmit video packets over multiple paths dependent on their priority. Further, a scheduling mechanism drops packets according to their effect on the overall video distortion. This work, however, does not take into account the end-to-end link quality estimation and node-disjoint multiple paths.

[Lin et al. 2010] developed the Adaptive Reliable Routing Based on a Cluster Hierarchy for Wireless Multimedia Sensor Networks (ARCH) to balance energy consumption and meet required reliability, adjusting the transmission power, together with an energy prediction mechanism. In [Lin et al. 2011], Energy Efficiency QoS Assurance Routing in Wireless Multimedia Sensor Networks (EEQAR) employs a social network analysis to optimize network performance. Nevertheless, both proposals implement multi-hop communication inside a cluster, adding extra overhead for route discovery. Further, they do not consider link quality as metric for route selection, neither the usage of multiple paths. These solutions also lack of QoE-based evaluation.

[Lari and Akbari 2010] introduced a node-disjoint multiple routing, and consider free buffer size, residual energy, hop-count, and packet loss rate to score and classify paths. Paths with better conditions achieve higher scores and are used to transmit higher priority video packets. On the other hand, [Jayashree et al. 2012] propose to analyze the image to find some common regions (overlapping) and some not common regions (non-overlapping). The path with the highest score has the best condition for sending packets with overlapped area. However, these two works consider a flat architecture, which reduces network lifetime [Ehsan and Hamdaoui 2011], and the route selection does not take into account end-to-end link quality, which leads to unreliability. The proposal of [Jayashree et al. 2012] carries out image processing, which consumes time and energy.

From the related work analysis, we conclude that node-disjoint multiple path routing protocols for WMSNs, together with a route selection scheme that considers both cross-layer information and end-to-end link quality estimation with a minimal signaling overhead increases reliability. Moreover, a video-aware mechanism to protect priority frames in congestion/link error periods enhances the video quality from the human's experience. However, existing routing protocols for WMSNs do not take into account all of these relevant characteristics into a single hierarchical routing proposal to support QoE-aware multimedia transmission, while achieving energy-efficiency.

3. The Video-aware MMtransmission Routing Protocol

This section presents a video-aware multiple path hierarchical routing protocol for efficient multimedia transmission over WMSN, called video-aware MMtransmission. The protocol finds node-disjoint multiple paths and evaluates paths by introducing an end-to-end link quality estimation, which includes a minimal signaling overhead. The proposed protocol takes into account the frame relevance to provide load balance, and increase the video quality from user's perspective.

3.1. WMSN Scenario

We assume a WMSNs composed of heterogeneous nodes, categorized into sensor and camera nodes, as illustrated in Figure 1. The sensor nodes (SN_i, i = 1, 2, ..., n) are restricted in terms of energy supply, processing and memory space, while camera nodes (CN_j, J = 1, 2, ..., m) are equipped with an alternative energy source, complementary metal-oxide-semiconductor (CMOS) camera, and very low bit rate image encoder. In this paper, we denoted multiple SN_i or CN_j, as $SN_{i(s)}$ or $CN_{j(s)}$ respectively.

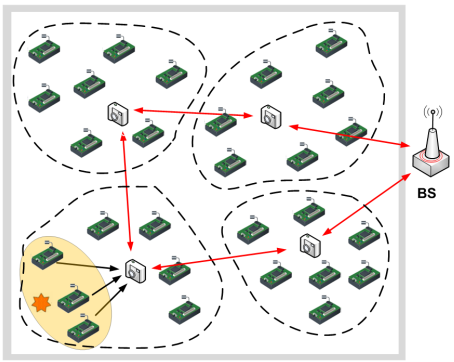


Figure 1. Network architecture

To support its operation, video-aware MMtransmission demands that $CN_{j(s)}$ act as Cluster-Heads (CHs), and $SN_{i(s)}$ act as non-CHs. $CN_{j(s)}$ carry out more complex tasks, such as time-slot allocation, synchronizing non-CH transmissions, data aggregation, real-time video retrieval, and routing packets. On the other hand, $SN_{i(s)}$ perform simple tasks, such as periodically sending physical scalar measurements to a CN_j. In this way, $CN_{j(s)}$ can predict event occurrence, such as flooding, fire, intruders and others, by using existing models or methods and physical scalar measurements. As soon as $CN_{j(s)}$ detect an event, they should start the multimedia retrieval and transmission to provide users more precise information than simple scalar data.

3.2. Operation Model of Video-aware MMtransmission

Video-aware MMtransmission relies on hierarchical network architecture with heterogeneous nodes to reduce the overall communication overhead, maximize the network lifetime, and improve scalability. The operation of MMtransmission has as basis the Multi-hop hierarchical routing protocol for Efficient VIdeo communication over WMSN (MEVI), proposed in [Rosário et al. 2012], since MEVI's properties enable multi-hop communication taking into account scalability and energy-efficiency.

The data transmission consists of intra-cluster and inter-cluster communications. For intra-cluster communication, nodes create clusters with low signaling overhead,

where non-CHs send sensed values in a specific time-slot to their CH. Further, each CH receives the data packets, aggregates them into a single packet and assigns the time-slots according to the TDMA (Time Division Multiple Access) schedule.

On the other hand, during the inter-cluster communication, there is a multi-hop communication between CHs and Base Station (BS) to enable the CHs forwarding the aggregated and multimedia data packets to the BS, and the BS can also request multimedia content to a CH if necessary. Thus, MEVI contains a route discovery period by exploiting a reactive scheme to find routes on demand to decrease the overhead and improve the scalability of the system. The route discovery process involves the CHs broadcasting route request (RREQ) and reply (RREP) messages. In this way, each received RREP means an available route to the destination node.

These messages search available routes and assist the route selection process by collecting information of *Remaining Energy (*RE_t*)* and *Hop Count (HC)*. Hence, each path has associated a Link Quality (LQ) value, which is applied to score and classify the links. The LQ value is computed according to Eq. 1 by considering RE_t, and *LQI (Link quality Indicator)* for the next hop, *HC*, and weights to give a degree of relevance to each metric.

$$LQ = \alpha \times \frac{RE_t}{E_0} + \beta \times \frac{LQI}{LQI_{max}} + \gamma \times \frac{HC_{max} - HC}{HC_{max}} \quad (1)$$

LQ returns values from 0 to 1, the sum of weights (α, β, γ) is equal to 1, E_0 represents the initial energy, LQI_{max} represents the maximum value for LQI, and the maximum Hop Count (HC_{max}) depends on the network diameter.

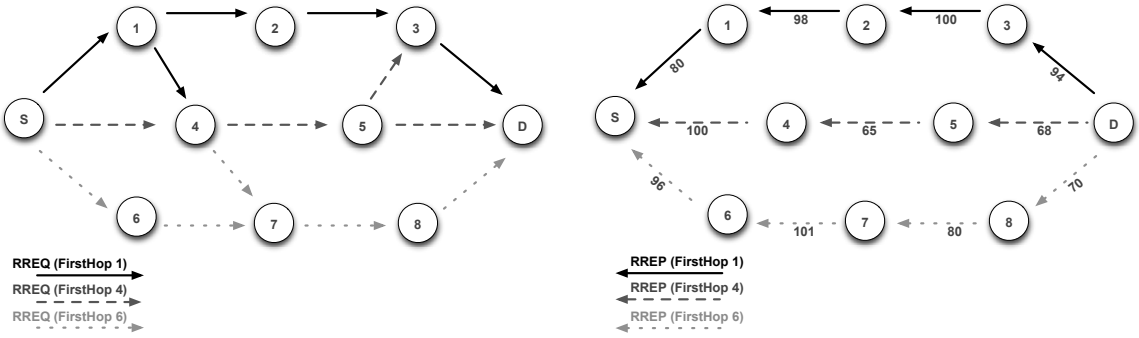
3.3. Node-disjoint Multiple Route Discovery

The video-aware MMtransmission protocol finds multiple paths between CHs and BS by transmitting RREQ and RREP messages, as shown in Figures 2(a) and 2(b). The proposed protocol gives a number for each possible path by including the first-hop into the RREQ message, which is the first node to broadcast RREQ, after the source node (S) initiates route discovery. Figure 2(a) illustrates this operation, where the S broadcasts a RREQ message to find possible multiple paths to the destination node (D). The RREQ message traverses by three possible paths with different first-hop, i.e. 1, 4 and 6. Additionally, a node never rebroadcasts duplicated RREQs or with different first-hop compared to prior received RREQs. For example, node 7 received RREQs with first-hop 4 and 6, and then it knows that those paths are not node-disjoint. In this case, node 7 does not rebroadcast and adds both paths into the routing table.

As soon as the RREQ messages reach the destination node, it creates new path entries to the source node for each incoming request, even when RREQs have a different first-hop. The destination node transmits a corresponding RREP message for every incoming RREQ message, with the aim to establishing a full bidirectional multiple path, as shown Figure 2(b). The RREP message travels back to reach the source node, which creates new path entries for every incoming RREP.

Existing works classify the paths according to hop-count or single-hop metrics. In contrast to that, the video-aware MMtransmission protocol evaluates the end-to-end link quality communication by using regions of connectivity presented by

[Baccour et al. 2011]. Such work classifies the links by means of the PRR (Packet Reception Ratio) value into three regions of connectivity, namely connected (PRR higher than 90%), transitional (PRR between 10% and 90%), and disconnected (PRR lower than 10%). Based on these information, video-aware MMtransmission must find routes composed of links with higher PRR to support multimedia transmission with higher transmission reliability. Moreover, the proposed protocol supports multimedia dissemination with higher quality level, by using information about the frame importance from the user's point-of-view to protect key frames of congestion/link error periods, as explained latter.



(a) Route request (b) Route reply

Figure 2. Video-aware MMtransmission route discovery

In specific terms, to estimate the end-to-end link quality, we count the number of disconnected and connected links by including two counters into RREQ and RREP messages. We define the bounds of disconnected and connected regions by means of two LQI thresholds, i.e. LQI_{bad} and LQI_{good}, which should be defined according to experiments, as shown Figure 3. LQI lower than LQI_{bad} implies a disconnected link, and with LQI higher than LQI_{good} provides a connected link. Thus, as soon as a node receives a RREQ/RREP, it derives the LQI value to classify the links into disconnected and connected, and then updates the counters of RREQ/RREP messages.

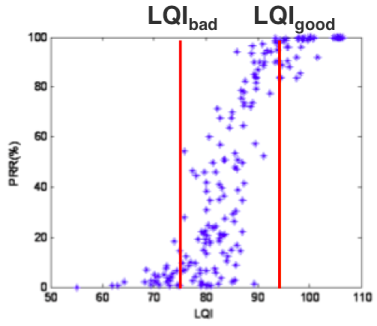


Figure 3. Regions of connectivity

We introduce **Path Quality (PQ)** to score the paths based on cross-layer and end-to-end link quality information. PQ is computed according to Eq. 2 by considering single hop metrics (RE_t and *LQI*), and end-to-end metrics (number of *Disconnected Links (DL)* and *Connected Links (CL)*). Moreover, PQ includes weights to give priorities for each

metric, which are defined based on the scenario characteristics and application requirements. As result, video-aware MMtransmission selects routes with high PQ to provide reliability for data delivery and increase the system performance.

$$PQ = \alpha \times \frac{CL}{HC} + \beta \times \frac{HC - DL}{HC} + \gamma \times \frac{RE_t}{E_0} + \sigma \times \frac{LQI}{LQI_{max}} \quad (2)$$

PQ gives values from 0 to 1 for each possible path, and the sum of weights $(\alpha, \beta, \gamma, \sigma)$ is equal to 1.

Figure 2(b) exemplifies the evaluation of the paths based on end-to-end information by assuming LQI_{bad} = 75 and LQI_{good} = 95, as defined in accordance with Figure 3. Table 1 shows the LQ (Eq. 1) and PQ (Eq. 2) values for each possible path between S and D. Considering the lowest number of hops, we select the route 2. However, this route is composed of two bad links, which will cause higher packet loss. Taking into account LQ, we select route 2 as the best and route 3 as alternative. However, these routes contain two bad links that also cause higher packet loss. Considering PQ, we select the route 1 as the best route, which is composed of more reliable links, and thus able to deliver video content with higher quality level.

Table 1. Routing table example

Route	Possible Paths	PQ	LQ
1	S-1-2-3-D	0.74	0.36
2	S-4-5-D	0.63	0.58
3	S-6-7-8-D	0.73	0.44

It is important to highlight that [Baccour et al. 2011] define connected links by means of PRR values higher than 90%. However, we decreased the threshold to 80%, because in certain cases it is difficult to find a higher number of links belonging to the connected region. Moreover, PRR higher than 80% still provides multimedia transmission with good quality from a user's experience. Additionally, we increase the threshold for disconnected region from 10 to 20%, because links with PRR lower than 20% deliver videos with very bad quality from a user's perspective.

3.4. Video-aware Multimedia Transmission

Three types of frames compose a compressed video. Intra frames (I-frames) provide a reference point for decoding a received video stream, i.e., reference for all the other frames. Predictive-coded frames (P-frames) give an increased level of compression compared to I-frames. Bi-directionally predictive-coded (B-frames) use the previous and next I-frames or P-frames as their reference points for motion compensation [Greengrass et al. 2009]. The frame sequence that depends on an I-frame composes a Group of Pictures (GoP). For example, a GoP length of 10 frames means that the GoP starts with an I-frame followed by a sequence of 9 P or B frames.

The frame loss has different influence on the user's perception, and causes distortion and error propagation. In this way, the loss of an I-frame affects the other B- or

P-frames of the same GoP by propagating the error within the whole GoP. For loss of P-frames, the error propagates by the remaining P- and B-frames within the GoP. Additionally, the loss of P-frames at the beginning of a GoP causes more video distortion than P-frames at the end of a GoP. Finally, for loss of B-frames, the error does not propagate, since B-frames are not used as a reference for other frames.

To support QoE-aware multimedia transmission and load balance, the proposed protocol protects priority frames of congestion/link error periods by selecting reliable paths to send the video packets based on frame importance. The path with highest PQ is more reliable for transmitting priority frames (I-frame and the firsts P-frames), and thus, ensures a minimum packet loss. On the other hand, less priority frames (B-frames and the last P-frames) are transmitted via less reliable paths (alternative paths).

However, there are cases that the alternative paths provide a low reliability, because it suffers a higher packet loss rate. Then, as soon as the video-aware MMtransmission protocol detects an alternative route with a PQ lower than a minimal PQ (PQ_{min}), the protocol transmits the video packets, by using single path communication. To define PQ_{min}, we performed experiments and present the results in the next section.

4. Performance Evaluation

This section describes the simulations conducted to evaluate the video-aware MMtransmission protocol, and also shows the impacts and benefits of our protocol for multimedia distribution over WMSNs. First, we present the methodology used to evaluate video-aware MMtransmission. Then, we present and analyze the simulation results.

4.1. Simulation Parameters and Evaluation Metrics

We evaluated the performance of video-aware MMtransmission routing protocol by means of simulations. For that, we used the Mobile MultiMedia Wireless Sensor Network (M3WSN) OMNeT++ framework [Rosario et al. 2013]. This is because M3WSN framework includes a model to generate sensor data, which corresponds to a real process with spatial data correlation and also variability over time. This data was used to predict event occurrence, such as flooding, fire, intruders and others. Hence, as soon as $CN_{j(s)}$ detect an event, they should start the multimedia retrieval and transmission. In our simulations, we considered the features and energy consumption of $SN_{i(s)}$ equipped with a CC2420 radio, and $CN_{j(s)}$ equipped with CC2420 radio and CMOS camera (CMUcam).

We conducted simulations with different solutions of multi-path routing under MEVI to compare the results with video-aware MMtransmission. These solutions use similar approaches compared to existing routing protocols. The first solution classifies the routes according to the number of hops (*MMEVI-HC*), which is similar to Ad hoc On-demand Multipath Distance Vector (AOMDV), and [Hurni and Braun 2008] for route selection. The second solution, called *MMEVI-LQ*, evaluates the routes according to a single-hop metric, i.e., LQ (Eq. 1), which is the metric used by MEVI. Additionally, LQ uses metrics similar to [Lari and Akbari 2010] and [Jayashree et al. 2012] (see Section 2). For the last solution, called *MMEVI-PQ*, the paths are evaluated based on end-to-end link quality estimation, i.e., PQ (Eq. 2). This solution is used to evaluate the mechanism of video-aware MMtransmission to exclude alternative paths with PQ lower than PQ_{min}. Simulations were carried out and repeated 20 times to provide a confidence interval of 95% (vertical bars in graphics). Table 2 summarizes the basic simulation parameters.

Table 2. Simulation parameters

Parameter	Value
Field size	80x80
Base station location	40, 0
Total number of Nodes	100
Number of CHs	36
CHs topology	Grid
Non-CHs topology	Uniform
Initial Energy for non-CHs	14 J
Transmission Power	-10 dbm
Path loss model	Lognormal shadowing model
Video sequence	Container
Video Encoding	H.264
Video Format	QCIF (176 x 144)
Frame Rate	26 fps

It is important to highlight that $SN_{i(s)}$ are used to periodically send packets to a CN_j. Thus, different $SN_{i(s)}$ density does not reduce the video quality. Regarding to overhead of video-aware MMtransmission, our protocol adds low byte overhead to include the additional fields in RREQ/RREP messages compared to existing solutions. On the other hand, the packet overhead depends on the number of alternative paths.

Existing works classify the videos according to their motion into three categories, namely *low, median and high*. For example, [Aguiar et al. 2012] classify the Container video sequence (taken from Video Trace Library [Library 2012]) with low movement, which means that there is a small moving region of interest on a static background, i.e., a ship crossing a lake. This characteristic is required for many WMSN/IoT applications, such as environmental monitoring and smart parking. Thus, these factors explain our option for transmitting the Container video sequence.

Traditionally, routing protocols are evaluated from a network/packet level point-of-view by using QoS metrics, e.g., delay, jitter, or loss. However, QoS metrics do not reflect user's perception and, thus, fail in capturing subjective aspects associated with human's experience. In this context, QoE metrics/approaches overcome the limitations of current QoS schemes regarding human's perception and subjective aspects. Therefore, to highlight the protocol reliability from the user's point-of-view, we evaluate the video-aware MMtransmission by using two well-known objective QoE metrics: Structural Similarity (SSIM) and Video Quality Metric (VQM).

SSIM measures the structural distortion of the video to obtain a better correlation with the user's subjective impression. SSIM has values ranging from 0 to 1, and higher value means better video quality. On the other hand, VQM measures the "perception damage" of video experienced, based on features of the human visual system, including distinct metric factors such as blurring, noise, color distortion and distortion blocks. For VQM, a value closer to 0 means a video with a better quality.

We measure the energy-efficiency of our protocol by devising an ad hoc metric called of Energy-Quality Index (EQI). EQI represents the trade-off between the video

quality achieved (i.e., by means of SSIM) to the burden energy (E) at the intermediate nodes along a path, and is computed according to Eq. 3. The values of SSIM and E are normalized with number of hops [Boluk et al. 2011].

$$EQI = \frac{\Delta SSIM}{\Delta E} \tag{3}$$

4.2. Simulation Results

This section presents the experiments to find PQ_{min}, which impacts on the performance of video-aware MMtransmission. We conducted simulations to measure the video quality level and energy-efficiency. PQ_{min} is used to avoid packet loss on unreliable paths, and thus, ensures multimedia transmission with a minimal video quality level. Alternative paths with PQ lower than PQ_{min} are not used. Based on the results shown in Figure 4, a video transmitted through a path with PQ below 0.4 obtains poor video quality level from the user's perspective, and for this reason we selected PQ_{min} = 0.4.

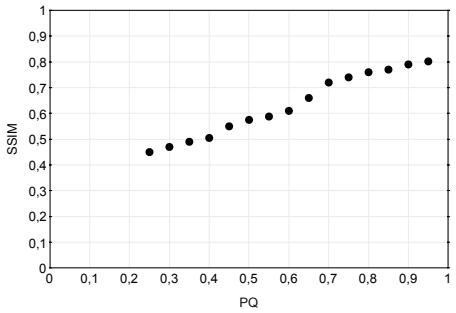


Figure 4. SSIM according to path quality value

We measure the average of SSIM and VQM for all transmitted videos with respect to the length of route (number of hops), as shown in Figures 5 and 6. By analyzing Figure 5, we see a higher and similar video quality for a route up to 2 hops, regardless of protocols. This happens due to the source node is closer to BS, which decreases interference and packet drops at intermediate nodes. Nevertheless, in the best case video-aware MMtransmission improves the SSIM by 5% compared to the other solutions, and in the worst case the protocols have the same performance.

On the other hand, a path can experience links with bad communication quality when the routing protocol does not estimate the end-to-end link quality for routes with more than 3 hops. For example, MMEVI-HC selects routes based on the number of hops, which makes it unreliable. This can be explained by the fact that short routes in terms of hops are more susceptible to packet loss due to both noise and interference. By evaluating single-hop links, MMEVI-LQ increases the video quality compared to MMEVI-HC. However, this is still lower than solutions that evaluate the end-to-end link quality. Our protocol provides a gain in around 12% for SSIM metric compared to MMEVI-HC and MMEVI-LQ, for routes with more than 3 hops. This is because our protocol evaluates the end-to-end link quality communication with a minimal signaling overhead to increase the packet delivery ratio. Moreover, video-aware MMtransmission provides higher reliability than MMEVI-PQ, because when the alternative routes are not reliable enough, the proposed protocol uses the single-path communication.

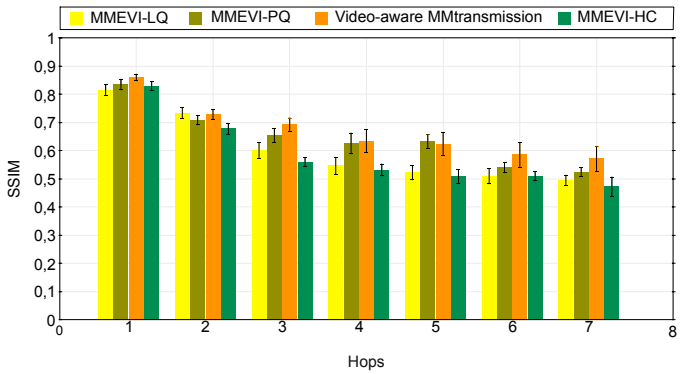


Figure 5. SSIM with respect to number of hops

Figure 6 shows the video quality by using the VQM metric. It is important to highlight that low VQM values means higher video quality. By analyzing VQM results, we find similar observations compared to SSIM results. By transmitting video packets via video-aware MMtransmission, the VQM improves by around 10%, and in the worst case our protocol has a similar video quality as the other solutions for routes up to 2 hops. On the other hand, for routes with more than 3 hops, the video-aware MMtransmission routing increases the VQM by 40%, 30% and 10% compared to MMEVI-HC, MMEVI-LQ and MMEVI-PQ, respectively. Thus, based on SSIM and VQM results, we confirm the benefits of video-aware MMtransmission to assure higher video quality from user's perception for disseminating video flows.

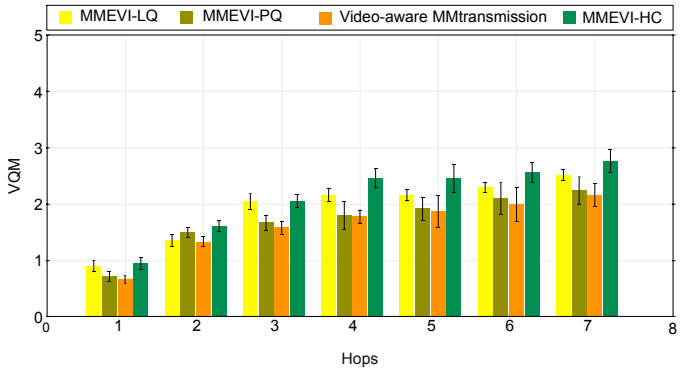


Figure 6. VQM with respect to number of hops

Figure 7 illustrates the energy costs to provide a certain video quality level. Upon analyzing the results, we can see that video-aware MMtransmission increases the EQI by around 10%, which means that the proposed protocol provides the best trade-off between the video quality per unit of spent energy. This is because our protocol evaluates the end-to-end link quality communication, which improves the packet delivery. Additionally, it avoids alternative paths with low reliability by considering a minimal PQ value.

To show the benefits of transmitting video streams using video-aware MMtransmission from the standpoint of the end-user, a random frame was selected (frame 257) from the transmitted video, as displayed in Figure 8. Frame 257 is the moment when two birds fly across the scene. For some applications, the moment/frame with a moving region of interest, e.g., intruder/event and, in this case a bird, crossing a static background

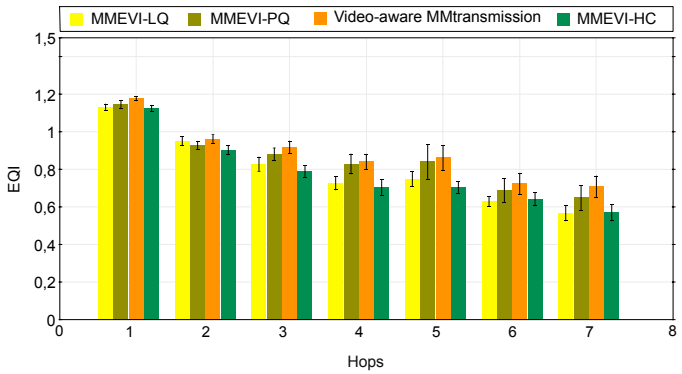


Figure 7. EQI with respect to number of hops

is useful to detect an intruder or to analyze the impact of an event. Thus, the video stream provides more precise information, gives support to end-users (or systems) to visually verify the real impact of event, and be aware of what is happening in the environment based on visual information.

(a) Original (b) Video-aware MMtransmission (c) MMEVI-PQ

(d) MMEVI-HC (e) MMEVI-LQ

Figure 8. Frame 257 of container video sequence

The benefits of video-aware MMtransmission are evident by comparing its received frame (Figure 8(b)) with the other solutions (Figures 8(d), 8(e) and 8(c)) and also with the original frame (Figure 8(a)). For example, the transmitted frames by using MMEVI-HC, MMEVI-LQ and MMEVI-PQ have a higher distortion in the ship and on the lake compared to the original frame (see Figure 8(a)) compared to the frame transmitted via our protocol (see Figure 8(b)). Moreover, the bird does not appear in the same position because this frame was lost, and thus it was reconstructed based of the previously received frames. In contrast to that, the frame transmitted via video-aware MMtransmission presents only few distortions, and the bird appears in the same position. This is because our protocol proposes a mechanism to protect priority frames in congestion/link error periods by transmitting via reliable paths, and thus, enhances the video quality level

from the human's experience. Additionally, video-aware MMtransmission evaluates the routes in an end-to-end fashion by a cross-layer approach with a minimum overhead.

5. Conclusion

This paper introduced the video-aware MMtransmission routing protocol to provide load balance and QoE-aware multimedia transmission, while achieving energy-efficiency. In specific terms, it searches for node-disjoint multiple paths, evaluates the paths according to cross-layer information and end-to-end link quality estimation. Thus, it gets to select the most reliable route to send priority frames (i.e., based on user's perspective), and alternative routes to distribute lower priority frames.

Simulations were carried out to show the benefits of video-aware MMtransmission for multimedia dissemination. Analyzing the simulation results, it was found that video-aware MMtransmission enables video distribution with a minimal quality level from a user's perspective. This was found by evaluating the proposed protocol through well-known objective metrics (SSIM and VQM) as well as showing video frames. Transmitting video packets via video-aware MMtransmission, the SSIM has a gain of 10% and the VQM improves by 10% and 40% compared to other solutions. Thus, we can conclude that the proposed video-aware MMtransmission protocol delivers video with QoE assurance, while also achieving energy-efficiency.

6. Acknowledgment

The National Council for Scientific and Technological Development (CNPq) and also the Foundation for Research of the State of Para (FAPESPA) supported this work.

References

Aguiar, E., Riker, A., Abelém, A., Cerqueira, E., and Mu, M. (2012). Video quality estimator for wireless mesh networks. In *20th International Workshop on Quality of Service (IWQoS)*, pages 1–9. IEEE.

Almalkawi, I., Guerrero Zapata, M., Al-Karaki, J., and Morillo-Pozo, J. (2010). Wireless Multimedia Sensor Networks: Current Trends and Future Directions. *Sensors*, 10(7):6662–6717.

Baccour, N., Koubâa, A., Jamâa, M., Rosário, D., Youssef, H., Alves, M., and Becker, L. (2011). Radiale: a framework for designing and assessing link quality estimators. *Ad Hoc Networks*, 9(7):1165–1185.

Boluk, P., Baydere, S., and Harmanci, A. (2011). Robust image transmission over wireless sensor networks. *Mobile Networks and Applications*, 16(2):149–170.

Ehsan, S. and Hamdaoui, B. (2011). A survey on energy-efficient routing techniques with qos assurances for wireless multimedia sensor networks. *IEEE Communications Surveys Tutorials*, PP(99):1 –14.

Gomez, C., Boix, A., and Paradells, J. (2010). Impact of lqi-based routing metrics on the performance of a one-to-one routing protocol for ieee 802.15. 4 multihop networks. *EURASIP Journal on Wireless Communications and Networking*, 2010(1).

Greengrass, J., Evans, J., and Begen, A. (2009). Not all packets are equal, part i: Streaming video coding and sla requirements. *IEEE Internet Computing*, 13(1):70–75.

Hurni, P. and Braun, T. (2008). Energy-efficient multi-path routing in wireless sensor networks. *Ad-hoc, Mobile and Wireless Networks*, pages 72–85.

Jaime, G., e Silva, E., Leão, R., and De Marca, J. (2010). On the reduction of scalable video base-layer packet loss rate on fifo/drop-tail queues. In *XXVIII Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, pages 659–672. SBC.

Jayashree, A., Biradar, G., and Mytri, V. (2012). Energy efficient prioritized multipath qos routing over wmsn. *International Journal of Computer Applications*, 46(17):33 – 39.

Kandris, D., Tsagkaropoulos, M., Politis, I., Tzes, A., and Kotsopoulos, S. (2011). Energy efficient and perceived qos aware video routing over wireless multimedia sensor networks. *Ad Hoc Networks*, 9(4):591 – 607. Multimedia Ad Hoc and Sensor Networks.

Lari, A. and Akbari, B. (2010). Network-adaptive multipath video delivery over wireless multimedia sensor networks based on packet and path priority scheduling. In *International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA)*, pages 351–356. IEEE.

Library, V. T. (2012). Yuv video sequences. Available at: `http://trace.eas.asu.edu/yuv/`. Accessed at Dec. 2012.

Lin, K., Chen, M., and Ge, X. (2010). Adaptive reliable routing based on cluster hierarchy for wireless multimedia sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2010(1).

Lin, K., Rodrigues, J., Ge, H., Xiong, N., and Liang, X. (2011). Energy efficiency qos assurance routing in wireless multimedia sensor networks. *IEEE Systems Journal*, 5(4):495 –505.

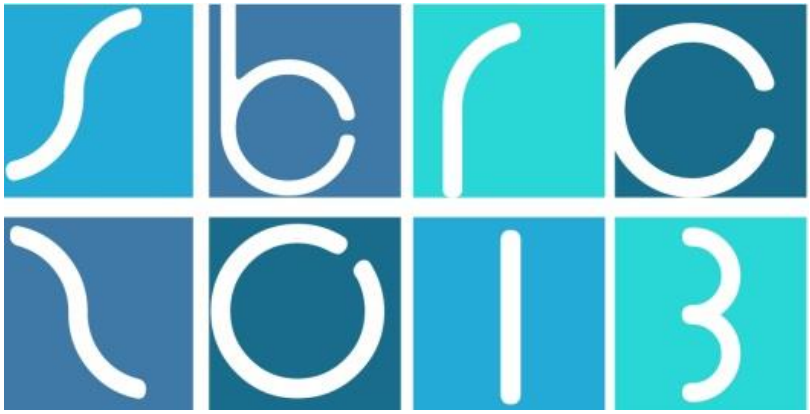
Politis, I., Tsagkaropoulos, M., Dagiuklas, T., and Kotsopoulos, S. (2008). Power efficient video multipath transmission over wireless multimedia sensor networks. *Mobile Networks and Applications*, 13(3):274–284.

Radi, M., Dezfouli, B., Bakar, K., and Lee, M. (2012). Multipath routing in wireless sensor networks: Survey and research challenges. *Sensors*, 12(1):650–685.

Rosário, D., Costa, R., Paraense, H., Machado, K., Cerqueira, E., Braun, T., and Zhao, Z. (2012). A Hierarchical Multi-hop Multimedia Routing Protocol for Wireless Multimedia Sensor Networks. *Network Protocols and Algorithms*, 4(2).

Rosario, D., Zhao, Z., Silva, C., Cerqueira, E., and Braun, T. (2013). An omnet++ framework to evaluate video transmission in mobile wireless multimedia sensor networks. In *Proceedings of the 6th International Workshop on OMNeT++*, Cannes, France.

Zhou, L. and Chao, H.-C. (2011). Multimedia traffic security architecture for the internet of things. *IEEE Network*, 25(3):35–40.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 2

Monitoração e Caracterização de Redes

Explorando o processamento paralelo na classificação de tráfego em redes de alta velocidade

Alysson F. Santos, Petrônio G. L. Júnior, Stenio F. L. Fernandes, Djamel F. H. Sadok

Centro de Informática – Universidade de Federal de Pernambuco (UFPE)
Recife – PE – Brasil

{afs,pglj,sflf,jamel}@cin.ufpe.br

Abstract. *Traffic Identification is a crucial function performed by ISPs administrators to evaluate and improve its network service. Deep Packet Inspection (DPI) is a well-known technique used to classify traffic and relies mostly in Regular Expressions (REs) evaluated by Finite Automata. No previous studies have discussed the impact of DPI on the overall system throughput. This work presents a novel technique to perform DPI on GPU called Flow-Based Traffic Identification (FBTI). Basically we want to increase DPI systems performance on commodity platforms to classify networked traffic at high speed links. We achieve a raw throughput classification of approximately 114 Gbps on GPUs. Our prototype solution reach a real total throughput of approximately 1 Gbps.*

Resumo. *Identificação de tráfego é uma tarefa importante para provedores de serviço de internet (ISP), com o objetivo de otimizar a qualidade de serviço (QoS) para os usuários. Inspeção Profunda de Pacotes (DPI) é a técnica mais utilizada para realizar a identificação e se baseia em expressões regulares (ER) para representar padrões de tráfego. Dentre os vários problemas, o principal gargalo ao identificar tráfego em tempo real de alta velocidade é o casamento das ER, que costuma ser computacionalmente custoso. A fim de aumentar o desempenho de um DPI, este trabalho utiliza uma unidade de processamento gráfico (GPU). Nesse contexto, a solução proposta alcançou uma vazão de processamento de 114 Gb/s na GPU, além de cerca de 1 Gb/s de vazão total.*

1. Introdução

A identificação de tráfego exerce um papel importante no gerenciamento das redes dos Provedores de Serviço da Internet (ISPs) visando, principalmente, melhorar a qualidade de serviço (QoS) de alguns serviços ou aplicações. Nesse contexto, a inspeção de pacotes (DPI) é uma técnica que fornece informações precisas sobre o tráfego de uma rede monitorada. Embora outras técnicas sejam utilizadas (como análise de fluxos, por exemplo), sistemas DPI apresentam maior flexibilidade e precisão. Um dos maiores problemas enfrentados por esses sistemas é, entretanto, a necessidade de analisar todos os pacotes que trafegam por um ponto da rede, o que demanda um grande esforço computacional [Dainotti et al. 2012]. A fim de melhorar o desempenho de um sistema DPI, várias soluções baseadas em *hardware* foram desenvolvidas. Por exemplo, circuitos FPGA ou ASIC [Becchi et al. 2007] [Yu et al. 2006] mapeiam expressões regulares para representar assinaturas de aplicações bem conhecidas em seus

hardwares, processando rapidamente. Essas soluções são custosas e pouco flexíveis. Por esse motivo, alguns trabalhos utilizam CPUs tradicionais para analisar pacotes [Antonello et al. 2012]. Outra plataforma que tem sido utilizada é a GPU (*Graphics Processing Unit*) em conjunto, principalmente, com a arquitetura CUDA (*Compute Unified Device Architecture*) [Sanders et al. 2010]. As GPUs apresentam baixo custo, grande poder computacional e flexibilidade (quando comparados com analisadores feitos em *hardware*). Trabalhos recentes trazem soluções eficientes para melhorar sistemas DPI [Cascarano et al. 2010], além de técnicas de aprendizagem de máquina [Sharp 2008] e roteamento IP [Mu et al. 2010] aplicados nessa área.

Sistemas DPI baseados em GPU frequentemente utilizam técnicas de casamento de cadeias de caracteres. Dentre eles, destacam-se alguns trabalhos que utilizam algoritmos baseados em casamento exato de cadeias de caracteres [Szabo et al. 2010]. Para esse fim, pesquisadores criam estruturas de dados capazes de representar as expressões regulares (ERs) e armazená-las em memória. As ERs oferecem maior expressividade quando comparadas a cadeias fixas de caracteres, além de serem mais flexíveis. Para representar ERs, são criados autômatos finitos determinísticos (DFA) [Wang et al. 2011] ou autômatos finitos não-determinísticos (NFA) [Cascarano et al. 2010]. De maneira geral, o DFA apresenta um melhor desempenho (em termos de velocidade de processamento dos pacotes), sendo bastante utilizado em sistemas DPI. Diante desse cenário, o escopo deste trabalho será restrito à construção de um sistema DPI baseado em GPU para identificação de tráfego. Nesse caso, é importante notar que o sistema DPI será projetado para fazer uso do grande poder computacional de uma GPU, se diferenciando, nesse ponto, de trabalhos anteriores, que utilizaram apenas CPUs. Dessa forma, este trabalho propõe novas arquiteturas baseadas em GPU que buscam a otimização do processo de classificação de tráfego realizado pelo DPI. A solução proposta alcançou resultados expressivos, como uma vazão de processamento (volume total de dados na GPU) de 114 Gb/s na GPU, além de cerca de 1 Gb/s de vazão total (volume total de dados processado), e uma vazão de fluxos classificados correspondente a um volume de mais de 20 Gb/s.

Este trabalho apresenta, na Seção 2, a teoria básica sobre GPU e CUDA. Em seguida, na Seção 3, alguns trabalhos relacionados são apresentados e, na Seção 4, a arquitetura proposta é descrita. A metodologia de avaliação utilizada no desenvolvimento do trabalho é exposta na Seção 5. Os resultados da avaliação estão na Seção 6. Por fim, as conclusões e trabalhos futuros são apresentados na Seção 7.

2. GPU e a arquitetura CUDA

Originalmente concebidas para auxiliar na renderização de componentes gráficos, reduzindo o uso de recursos da CPU, as GPUs evoluíram, aumentando o seu poder computacional, e se tornaram úteis na área da computação científica. Além da programação direta e intuitiva, essas unidades de processamento gráfico podem ser utilizadas como unidades complementares às CPUs, ampliando suas capacidades [Sanders et al. 2010]. É possível utilizar uma GPU para diminuir a carga na CPU, aumentando o paralelismo e o desempenho final do sistema. Tais benefícios são possíveis devido à grande quantidade de núcleos de processamento em uma única placa, na ordem de centenas. Elas evoluíram com o passar do tempo, fazendo com que as novas gerações de placas gráficas se tornassem mais facilmente programáveis pelos desenvolvedores, mantendo o alto poder de processamento [Ryoo et al. 2008]. Nesse

contexto, a NVIDIA desenvolveu a arquitetura de programação (CUDA), voltada para a execução de tarefas que precisam ser realizadas em paralelo sem constantes interrupções. CUDA se caracteriza por ser uma extensão da linguagem C/C++, possuindo funcionalidades e diretivas para processar tarefas diretamente na GPU. Tais funcionalidades e diretivas são inicializadas através de um programa principal que executa na CPU, chamado de *host* [Sanders et al. 2010], capaz de invocar o conjunto de instruções que será executado na GPU denominado *kernel.*.

Para que a GPU seja eficiente, é necessário entender o paradigma SIMD (*Single Instruction Multiple Data*), para o qual ela foi projetada. Esse paradigma consiste na execução de um mesmo conjunto de instruções sobre múltiplos dados de entrada. Uma soma de dois vetores poderia ser feita em um único passo, obtendo o vetor resultado, por exemplo. Sendo composta por várias *threads* (ramificações paralelas da execução do programa), a colaboração entre elas deve ser a menor possível. O objetivo desse modelo é ter várias *threads* executando o mesmo conjunto de instruções simultaneamente, aumentando o paralelismo. Em CUDA, as *threads* são agrupadas em conjuntos de blocos, que são reunidos em conjuntos de *grids*. *Grids* e blocos são apenas *containers* lógicos criados para dividir os componentes de CUDA e facilitar o escalonamento das *threads*. Além disso, a arquitetura é composta por diversos processadores simétricos, chamados *streaming multiprocessors* (SM), capazes de executar diversos *grids* de blocos simultaneamente. *Threads* de um mesmo bloco compartilham o mesmo espaço de endereçamento, a região de memória compartilhada e os registradores do SM em que o bloco executa. Elas se comunicam unicamente através da memória compartilhada de cada bloco, não sendo possível *threads* de blocos distintos trocarem informações. Um conjunto de 32 *threads* é chamado de *warp*. Em resumo, o paradigma SIMD prevê que as *threads* executem o mesmo conjunto de instruções sem divergência entre elas, garantindo um bom desempenho, comparado com um paradigma utilizado na CPU.

3. Trabalhos relacionados

Embora a maioria dos sistemas DPI utilizem DFAs para representar suas ERs, (devido ao desempenho de processamento), os NFAs consomem muito menos memória. Em [Cascarano et. al. 2010], foi criado o iNFAnt, um mecanismo para realizar casamento de ERs em GPU baseado em NFAs. Nesse caso, o iNFAnt foi criado com o intuito de permitir que várias *threads* pudessem executar simultaneamente, sendo cada uma delas responsável por realizar uma transição para um determinado estado ativo. No entanto, devido às características de CUDA, o algoritmo proposto se torna bastante lento, pois é preciso que as *threads* sempre estejam sincronizadas ao longo da execução, o que não é previsto pelos autores. No melhor caso, o algoritmo alcançou uma vazão de 1 Gb/s, ao realizar o casamento de pacotes TCP com apenas duas ERs simples, em contraste com sistemas DPI reais que possuem centenas de assinaturas (como L7-Filter, Snort, e Bro).

Um dos primeiros trabalhos que tratou do casamento de expressões regulares utilizando uma GPU foi apresentado em [Smith et. al. 2009]. Eles realizaram uma investigação aprofundada visando encontrar a melhor plataforma para realizar casamento de ERs de forma eficiente com foco em sistemas DPI. Foram analisadas as plataformas CUDA, FPGA e arquiteturas baseadas em vários núcleos (*multicore*). Alguns testes com diferentes fluxos de controle de execução e estratégias de paralelismo em nível de pacote foram realizados. No entanto, nos experimentos realizados, o

desempenho do sistema era degradado quando os tamanhos dos pacotes eram diferentes. Os autores concluíram que os pacotes deviam possuir o mesmo tamanho (sendo normalizados), para que não houvesse divergência durante a execução das *threads*. Além disso, em [Smith et. al. 2009], o impacto da utilização de diferentes autômatos finitos para representar as ERs na memória da GPU foi investigado. Os resultados finais apontaram que a GPU alcançou um desempenho 9 vezes maior que um processador Pentium 4. Porém, eles não apresentaram os valores para a vazão geral do sistema DPI.

Em [Vasiliadis et. al. 2009], os autores desenvolveram o Gnort, uma versão do Snort que utiliza uma GPU para realizar o casamento de expressões regulares. No Gnort, apenas o casamento de ERs é realizado em uma GPU. A solução proposta armazena um conjunto de pacotes de rede em filas temporárias. Quando esta fila encontra-se cheia, ela é transferida em lote para a GPU para que a análise seja realizada. O Gnort mantém a fila com posições de memória de tamanho fixo. Os autores não mostraram qual foi a organização dos blocos e *threads* utilizados no algoritmo. O desempenho de melhor caso de execução para o Gnort foi de aproximadamente 16 Gb/s, para a vazão de processamento dos pacotes e a vazão total foi de 800 Mbps.

O Gregex [Wang et. al. 2011] é capaz de atingir uma vazão de processamento de 122 Gb/s e um desempenho geral para identificar pacotes de 25.6 Gb/s. Os autores utilizaram um subconjunto de assinaturas do Snort, agrupando-as em um único DFA para representar todo o conjunto. Entretanto, não foi informada qual técnica de agrupamento foi utilizada, assim como outros detalhes de implementação não foram descritos. O Gregex utiliza uma fila de pacotes normalizada, onde cada pacote ocupa um espaço de 2048 Bytes (2 kB). Caso um pacote tenha carga útil menor que 2 kB, o restante da posição de memória da fila para onde o pacote está sendo copiado é preenchida com zeros. Isso gera imprecisões durante o cálculo da vazão, já que os zeros adicionados não fazem parte do tráfego original, e não deveriam ser contabilizados.

Considerando a solução apresentada neste trabalho, apesar dos benefícios associados ao uso da arquitetura CUDA, a criação de uma solução eficiente para o casamento de expressões com ER não é uma tarefa simples. Para implementar algoritmos paralelos, é necessário explorar diferentes *layouts* de memória de CUDA e diferentes modelagens do problema. A partir da melhor utilização da memória, diversas otimizações podem ser implementadas para maximizar o desempenho da GPU. Neste trabalho, também foram realizadas otimizações através da transposição de memória e da utilização de memória fixa sem paginação. Além disso, este trabalho prevê a utilização de fluxos, a fim de alcançar uma maior vazão total referente ao volume agregado.

4. Arquitetura proposta

Este trabalho tem como objetivo aumentar a vazão final de sistemas DPI que utilizam ERs como assinaturas através de uma GPU. Para isso, foram criadas 3 arquiteturas diferentes, chamadas 1CTB (1-CUDA *Thread per Block*), 1CTP (1-CUDA *Thread per Packet*) e ITBF (Identificação de tráfego baseada em fluxo), que utilizam diferentes disposições e combinações de blocos e *threads* em CUDA para aplicar o conjunto de expressões regulares a vários pacotes simultaneamente. Em resumo, queremos achar a melhor configuração para processar paralelamente os pacotes de uma determinada rede.

Como pode ser observado na Figura 1, a arquitetura geral é composta pela interação entre *host* (CPU) e GPU. Quando o *host* precisa classificar um conjunto de

pacotes utilizando uma determinada ER, eles são previamente copiados para o espaço de endereçamento da GPU, uma vez que CUDA acessa posições de memórias locais, estejam elas na memória global, constante, compartilhada ou de textura. Além de copiar os pacotes, também deve ser indicado qual DFA deverá ser utilizado para realizar o casamento. A tabela de transição dos DFAs utilizados é copiada e armazenada na memória global da GPU durante a fase de inicialização do DPI. O conjunto de pacotes que será inspecionado é armazenado na memória global da GPU, durante a fase de processamento. Durante a execução do código no *host*, o *kernel* é chamado e a GPU inicia o processamento, já tendo sido informada sobre a quantidade de blocos e *threads* que deverão ser instanciados. A disposição das *threads* e blocos varia de acordo com a solução utilizada (1CTB, 1CTP, ITBF). Após o término da execução do *kernel*, os resultados são copiados da GPU para o *host*, a fim de que os pacotes que casaram com alguma ER possam ser identificados. Essa arquitetura geral foi desenvolvida através de um protótipo funcional, sem paralelizar atividades desempenhadas no *host*. Todo o tráfego encontra-se armazenado em disco, de modo que o protótipo executa em modo *off-line* (lendo o disco e efetuando a classificação com a GPU).

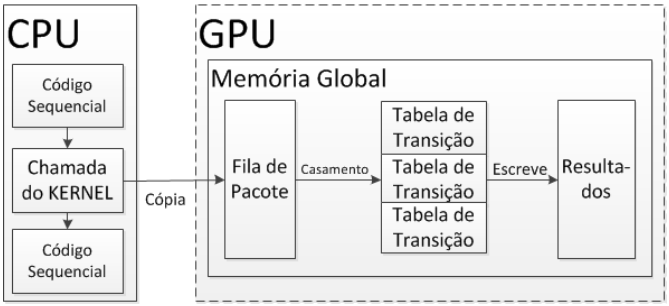


Figura 1. Arquiterura Geral de um sistema DPI utilizando uma GPU para realizar casamento de ER.

A Figura 2 ilustra a arquitetura 1CTB, onde é instanciado um único bloco CUDA por cada pacote da fila de entrada e cada um desses blocos possui apenas uma única *thread*. Ela é responsável por comparar o conteúdo de um único pacote com apenas um DFA, escrevendo o resultado final da classificação num vetor de resultados. Além da arquitetura 1CTB, também foi criada a arquitetura 1CTP, a qual cria várias *threads* por bloco, sendo cada uma das *threads* responsáveis por um único pacote. Dessa forma, várias *threads* são executadas em um mesmo bloco, porém cada uma trata uma posição diferente da memória global. Dessa forma, os acessos à memória podem ser disparados por *warps*, diminuindo a latência inerente ao processo.

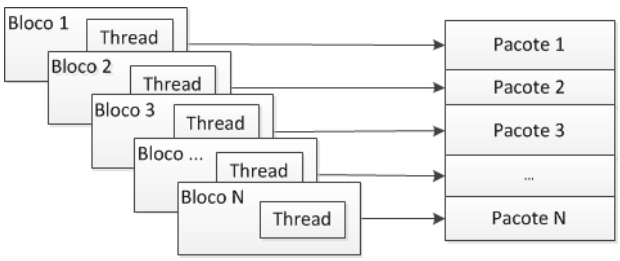


Figura 2. Solução 1 - Um *thread* por Bloco

A Figura 3 ilustra a solução 1CTP. A solução 1CTP baseia-se no correto posicionamento de cada uma das *threads* criadas através do conjunto de pacotes, de

modo a garantir que uma *thread* seja responsável apenas por classificar um único pacote. Percebe-se que a primeira *thread* do bloco 1 fica responsável por inspecionar o primeiro pacote, a segunda thread desse mesmo bloco é responsável pelo pacote 2 e consequentemente o *thread* N deste bloco é responsável pelo pacote N.

Por fim, a última versão criada denomina-se IBTF. Nessa solução, foi implementada a transposição de memória [Ruetsch et al. 2009], que é comumente utilizada na área de computação gráfica para aumentar o desempenho de algoritmos que executam em CUDA. A Figura 4 ilustra a solução IBTF. Vários estudos anteriores [Fernandes et al. 2009] [Bernaille et al. 2006] investigaram os benefícios obtidos ao realizar inspeção de pacotes agregando os dados em fluxos para identificar trafego, inspecionando apenas os primeiros bytes de cada fluxo. Mesmo analisando apenas parte dos dados trocados pelo canal de comunicação, os autores provaram que é possível obter uma boa classificação quanto às aplicações responsáveis pelos fluxos. No entanto, nenhum trabalho anterior analisou o impacto que utilizar essa técnica produz na vazão do sistema DPI.

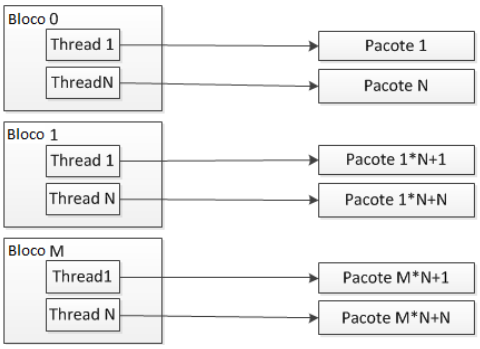


Figura 3. Solução 1CTP alocando um *thread* por pacote da fila de entrada.

Neste trabalho, essa técnica é aplicada através da arquitetura IBTF utilizando uma GPU para paralelizar o processamento. O IBTF aloca um conjunto de *threads* por bloco. Porém, ao invés de uma *thread* por pacote, uma *thread* para cada um dos fluxos que será classificado é definida.

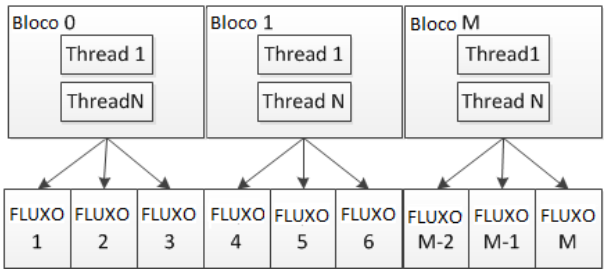


Figura 4. IBTF – Identificação de Tráfego Baseada em Fluxo

Como já reportado por estudos anteriores, é preciso que a fila de entrada seja formada por um vetor de dados uniforme, onde a área de memória ocupada por cada elemento é igual. Desse modo, o IBTF aloca um espaço fixo de dados por fluxo, onde os primeiros *K* bytes do fluxo são armazenados. A arquitetura IBTF também implementa a técnica de transposição de memória para diminuir o tempo de acesso a memória, distribuindo-os entre *warps*. Para que seja possível realizar a transposição, é necessário uma matriz quadrada. Desse modo, o IBTF trabalha com um vetor de fluxos *K* x *K*, contendo *K* fluxos, cada um com os *K* primeiros bytes de cada fluxo.

5. Metodologia de Avaliação

A fim de construir as soluções propostas, uma arquitetura Intel Quad-Core foi utilizada, onde cada núcleo é composto por um processador Core i7-2600. Além disso, foi utilizada a GPU NVIDIA GeForce GTX 480 com CUDA 2.0. A GPU GTX 480 é composta por 480 processadores e permite a criação de *kernels* com no máximo 65536 blocos, onde cada um dos blocos só é capaz de alocar 1024 *threads*. Para criar os cenários de avaliação, foi construído um gerador sintético de *traces* [Santos et al. 2011]. Esse gerador é capaz de criar arquivos de pacotes contendo carga útil segundo um conjunto de ERs, de modo que um sistema DPI possa encontrar ou não um casamento, de acordo com o *trace* gerado. Adicionalmente, este trabalho utilizou diferentes *traces* sintéticos de pacotes, assumindo cenários de melhor e de pior caso. Os cenários de melhor caso são aqueles em que um casamento irá ocorrer com alguma assinatura. Os cenários de pior caso são aqueles em que não ocorre casamento com nenhuma ER do conjunto. Nesse caso, foram gerados *traces* utilizando o gerador sintético de pacotes sobre um conjunto genérico de assinaturas, composto por 15 ER que especificam protocolos de aplicações comuns de rede, tais como HTTP, MSN Messenger, SSH, POP3, SMTP e RTSP. Além de realizar testes de melhor e de pior caso, também foi utilizado um *trace* real*,* identificado como caso médio, capturado em um enlace de alta velocidade de um provedor de serviços brasileiro, correspondente a uma coleta com 24 horas de duração, realizada entre novembro e dezembro de 2008. O *trace* completo possui 263 GB, porém apenas 75 GB foram utilizados nos testes, pois apenas pacotes IP, carregando segmentos TCP foram considerados. Mais informações sobre o *trace* utilizado podem ser encontradas em [Fernandes et al. 2009].

A principal métrica utilizada para medir o desempenho dos protótipos construídos neste trabalho é a vazão, que representa a quantidade de informações que a arquitetura proposta consegue processar por unidade de tempo. Nesse contexto, a vazão total (VT) trata do volume total de dados processado considerando um determinado período de tempo, contabilizando ainda as transferências de dados entre *host* e GPU. O presente trabalho trata não apenas da vazão de execução total da arquitetura proposta, mas também de outras medições a fim de obter diferentes tipos de vazão. Nesse caso, foram definidos outros dois tipos de vazão a serem avaliados.

A vazão de processamento (VP) caracteriza-se por ser a vazão para classificar todos os dados na GPU. A VP é dada pela razão entre a quantidade total de bytes copiados para a GPU e o tempo total necessário para processar e realizar o casamento de ERs. É importante salientar que esse tempo de processamento não inclui o tempo necessário para transferir os dados do *host* para a GPU e da GPU de volta para o *host*. Para medir a VP de maneira precisa, é necessário medir apenas o tempo gasto para processar os pacotes na GPU. Para isso, CUDA possibilita a utilização de eventos de medição. A Figura 5 retrata o ponto de medição utilizado para obter a duração do processamento dos pacotes na GPU e as diretivas de CUDA utilizadas.

A vazão de fluxo (VF) remete ao volume total de processamento do tráfego representado pelos fluxos. A VF será contabilizada apenas para a arquitetura IBTF, já que as demais não possuem agregação de tráfego por fluxos. De forma resumida, a VF é a razão entre o volume total de tráfego agregado dos fluxos que estão sendo classificados e o tempo necessário para processá-los na GPU. O mesmo intervalo de tempo utilizado no cálculo da VP é usado para a VF.

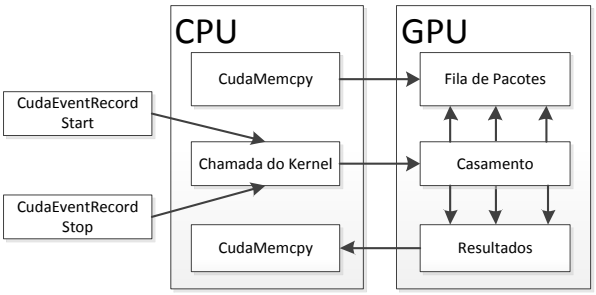


Figura 5. Ponto de medição para calcular a vazão de processamento

6. Resultados Experimentais

Conforme descrito, foram avaliadas as vazões de processamento e total para cada versão da arquitetura proposta, além da vazão de fluxo para a arquitetura IBTF. Assim, os resultados serão apresentados, analisados e uma breve discussão acerca dos resultados será exposta. Os testes foram realizados utilizando os *traces* de melhor e pior caso, além do *trace* do provedor de serviço, o qual representa o caso médio de execução. Em todos os casos, os valores obtidos para os diferentes tipos de vazão tratam-se de médias referentes a 10 execuções independentes.

O protótipo desenvolvido para testar a arquitetura 1CTB é bastante simples. Cada um dos pacotes é lido sequencialmente do *trace* e são, em seguida, armazenados em filas ainda na CPU. Quando uma fila possui uma quantidade pré-determinada de pacotes, ela é enviada para a GPU e o casamento de expressões regulares é executado. Na Figura 6-a, estão as médias da vazão de processamento (VP) referentes à execução da arquitetura 1CTB para diferentes tamanhos de fila de pacotes. A análise da Figura 6-a aponta que quanto maior a fila, maior é o valor da vazão de processamento para cada um dos *traces* utilizados. Como esperado, o *trace* do melhor caso de execução possui a melhor vazão de processamento e o *trace* de pior caso teve o menor desempenho.

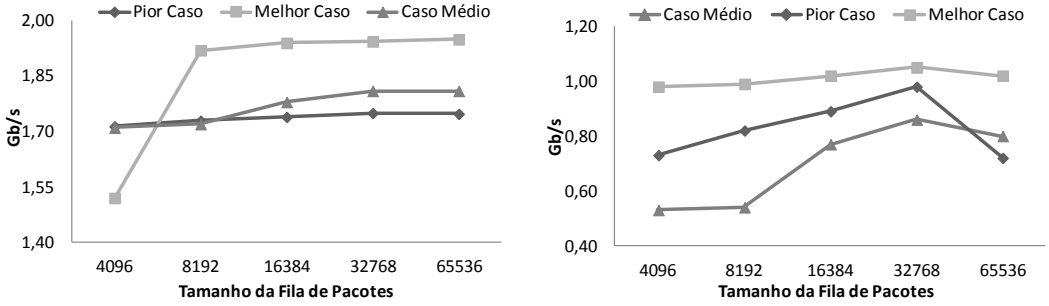


Figura 6. Vazão relacionada ao 1CTB. (a) Vazão de processamento da arquitetura 1CTB. (b) Vazão total para a arquitetura 1CTB.

No entanto, o aumento na vazão é muito pequeno quando a fila possui mais que 16384 pacotes. Isso se deve ao fato de que como a arquitetura utiliza apenas uma *thread* por bloco, não é vantajoso criar uma grande quantidade de blocos, pois CUDA escalona uma quantidade fixa de blocos por SMs. Nesse caso, a concorrência entre os blocos passa a ser alta, resultando em uma maior quantidade de mudanças de contexto e fazendo com que o ganho em desempenho não seja tão relevante, apesar do aumento. Assumindo os resultados obtidos para o *trace* de caso médio como sendo o comportamento esperado, a vazão total com a melhor configuração possível para a

arquitetura 1CTB foi de aproximadamente 1,8 Gb/s. Para a vazão total (VT), cujos resultados encontram-se ilustrados na Figura 6-b, o mesmo comportamento encontrado para a vazão de processamento ocorreu para valores de fila até 32768, ou seja, quanto maior a fila de pacotes, maior a vazão total. No entanto, para valores de fila maiores que 32768 pacotes, o desempenho começou a piorar. Isso se deve ao fato de que é preciso copiar a fila de pacotes do *host* para a GPU, acarretando em atrasos que reduzem o tempo total de processamento. A vazão total com a melhor configuração possível para a arquitetura 1CTB foi de 0,98 Gb/s com os resultados de caso médio.

A Figura 7-a ilustra os resultados obtidos para a vazão de processamento para a arquitetura 1CTP. É possível verificar que a variação da quantidade de blocos e *threads* alocados causa um forte impacto no desempenho, pois cada um dos blocos será responsável por um pacote diferente. De maneira geral, a vazão de processamento aumenta à medida que mais *threads* são alocados dentro de cada bloco, reduzindo a quantidade de blocos (a vazão de processamento aumenta quando a relação *threads*/bloco aumenta). Entretanto, o melhor resultado obtido foi com a configuração que utiliza 1024 blocos, cada um deles contendo 64 *threads*. Isso se deve ao fato de que CUDA escalona *threads* de um mesmo bloco em *warps* e um mesmo *warp* ocupa todos os processadores de um SM, provocando uma ocupação ótima de recursos. A melhor vazão de processamento alcançada, ao utilizar um *trace* de melhor caso, foi de aproximadamente 23 Gb/s. Como o desempenho esperado da arquitetura é aquele produzido pelo caso médio, a vazão de processamento para a arquitetura 1CTP é de aproximadamente 14 Gb/s.

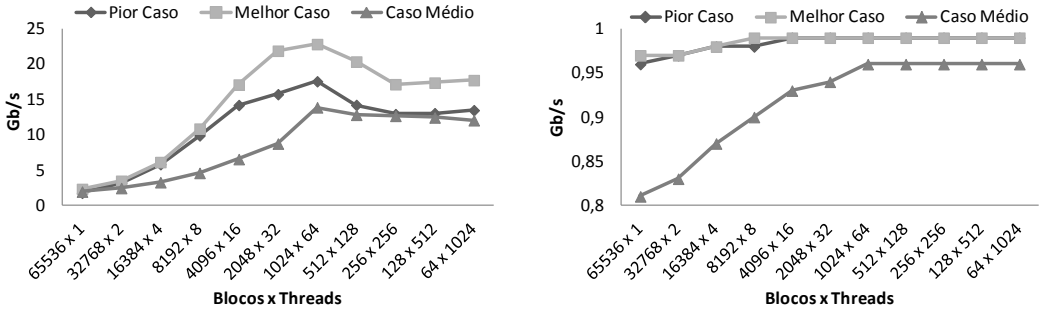


Figura 7. Vazão relacionada ao 1CTP. (a) Vazão de processamento para a arquitetura 1CTP. (b) Vazão total para a arquitetura 1CTP.

Os resultados obtidos para a vazão total na arquitetura 1CTP encontram-se na Figura 7-b. Conforme esperado, assim como nos resultados obtidos para a vazão de processamento, em todos os *traces* utilizados, quanto maior a quantidade de *threads* por bloco (a relação *thread*/bloco), maior a vazão total. Da mesma forma que na vazão de processamento, o melhor resultado obtido foi com a configuração que utiliza 1024 blocos, cada um com 64 *threads*. No entanto, para casos de teste contendo mais *threads* por bloco, a vazão total manteve-se a mesma, e não mais diminuiu como ocorreu para a vazão de processamento. Seguindo a mesma abordagem utilizada para os demais testes, o resultado considerado para a vazão total foi aquele encontrado na melhor configuração para o caso médio, que foi de aproximadamente 0,96 Gb/s.

A arquitetura IBTF utiliza a técnica transposição de memória para aumentar o desempenho no acesso à memória global, fazendo com que *threads* de um mesmo *warp* realizem acessos contíguos. O mesmo modelo é utilizado para avaliar as outras

arquiteturas, realizando testes de melhor caso, pior caso e caso médio. Nesse, caso a quantidade de blocos e *threads* foi modificada de modo que seja possível encontrar o conjunto ótimo para esses parâmetros. Devido ao uso da técnica de transposição, é preciso que a fila de fluxos seja uma matriz quadrada, ou seja, para fluxos de *K* bytes de dados, é preciso que existam *K* fluxos na fila. Para certo valor *K* de fluxos na fila, valores diferentes de *threads* e blocos foram aplicados. Percebe-se na Figura 8 que, para cada um dos valores de *K*, diferentes configurações produzem resultados distintos. Por exemplo, ao utilizar 8 kB de dados dos fluxos (Figura 8-d), a configuração que produziu a melhor vazão foi com 64 blocos, cada um contendo 128 *threads*. Já para testes onde foram inspecionados os 4 kB de dados (Figura 8-c), a melhor configuração foi de 64 blocos, com 64 *threads*. Os gráficos apontam que quanto maior for a quantidade de dados analisados de cada fluxo, maior a vazão de processamento. No entanto, podemos perceber que apenas na situação em que os primeiros 8 kB de dados foram analisados a vazão para o melhor caso é muito acima do pior caso e do caso médio.

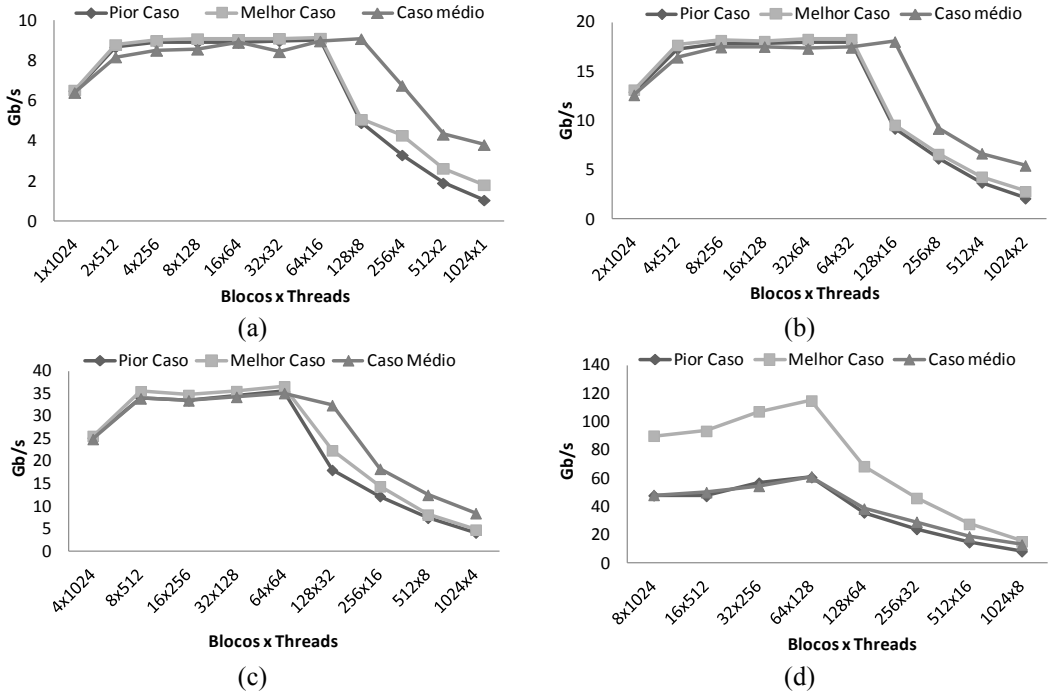


Figura 8. Vazão de processamento para a arquitetura IBTF. (a) Vazão para K = 1kB. (b) Vazão para K = 2kB. (c) Vazão para K = 4kB. (d) Vazão para K = 8kB.

Isso se deve ao fato de que o protótipo construído para executar os testes da arquitetura IBTF classifica 100% do tráfego do *trace* gerado sinteticamente para representar o melhor caso apenas quando os primeiros 8 kB estão presentes. Inspecionar menos que 8kB não é suficiente para conseguir identificar todo o *trace*, fazendo com que o pior caso, melhor caso e caso médio possuam processamento semelhantes. Porém, é importante frisar que mesmo no caso médio e no pior caso a vazão de processamento foi bastante próxima. Isso se deve ao fato de que grande parte do *trace* de caso médio não pode ser identificado, fazendo com que o cenário de pior caso tivesse desempenho semelhante.

Os gráficos da Figura 9 ilustram os resultados obtidos para a vazão de fluxos, realizando a mesma variação entre blocos e *threads* feita para avaliar a vazão de processamento, para cada um dos níveis e para cada um dos *traces*. Uma vez

armazenados em memória, os fluxos continuam possuindo apenas os seus valores de quantidade de pacotes e o seu volume de dados atualizados. Quando eles são classificados, todo o volume agregado é utilizado para os cálculos da vazão de fluxo. Esse é um limite teórico, pois depende que todos os fluxos estejam armazenados em memória e que haja um mecanismo eficiente. Esse mecanismo é capaz de lidar com todo o tráfego que chega até o ponto de medição, sem que haja perda ou descarte de pacotes, e de garantir que todos os fluxos não serão removidos da memória até que sejam identificados. Caso haja esse mecanismo e a GPU possam receber os fluxos na mesma taxa utilizada em nossos experimentos, podemos dizer que o sistema DPI é capaz de lidar com a mesma vazão de dados reportada pela vazão de fluxos. Assim, com o aumento no tamanho da quantidade de *bytes* analisados por fluxo, a vazão de fluxo manteve-se estável, onde apenas o *trace* de caso médio obteve uma vazão de fluxo maior para K = 8 kB. Isso acontece porque os *traces* gerados sinteticamente possuem fluxos com volume de dados fixos e menores. Para os testes utilizando 1 kB, 2 kB e 4 kB, a vazão de fluxo obtida para os testes de pior caso foi maior do que aquela encontrada para os testes de melhor caso. Isso se deve ao fato de que o *trace* que representa o pior caso é composto apenas por pacotes grandes de 1500 *bytes*, fazendo com que seus fluxos possuam um volume agregado maior. Já o *trace* de melhor caso possui fluxos contendo tanto pacotes de 1500 quanto 64 *bytes*, fazendo com que a vazão de fluxo seja menor. Já para os testes de melhor caso onde os primeiros 8 kB foram inspecionados (o *trace* é totalmente identificado), o resultado da vazão de fluxo foi acima dos 20 Gb/s.

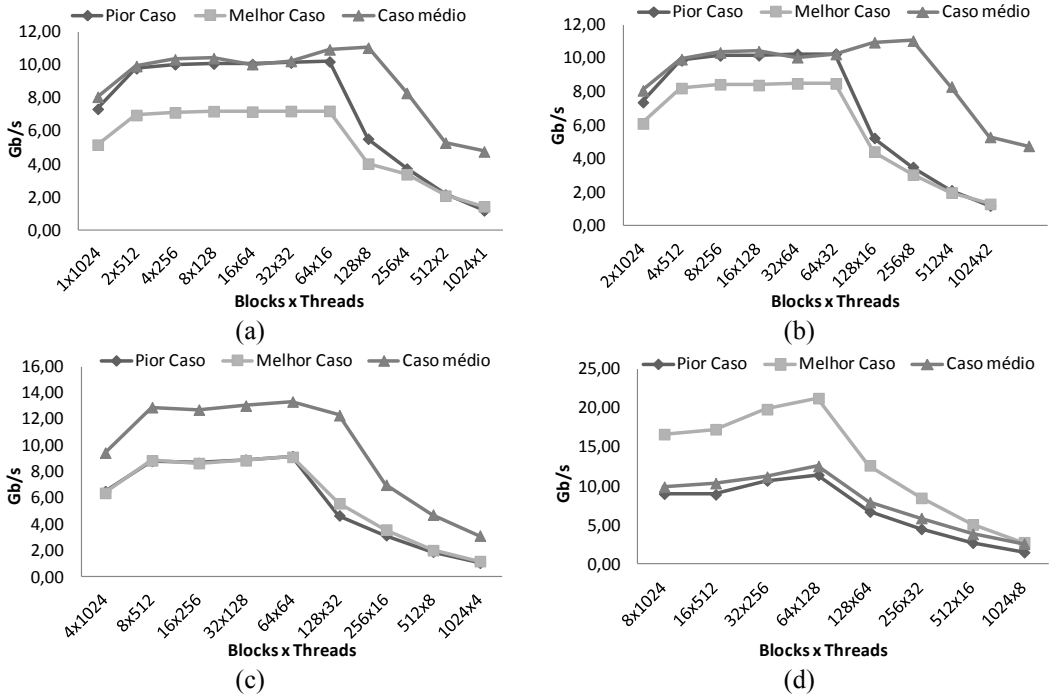


Figura 9. Vazão de Fluxo para a arquitetura IBTF. (a) Vazão para K = 1kB. (b) Vazão para K = 2kB. (c) Vazão para K = 4kB. (d) Vazão para K = 8kB.

Os gráficos da Figura 10 ilustram os valores da vazão total para cada um dos testes realizados, variando a quantidade K de fluxos e o *trace* utilizado. É importante frisar que o arquivo de *trace* do melhor caso de execução possui uma mistura de pacotes grandes e pequenos. Além disso, o *trace* que representa o melhor caso possui vários fluxos contendo pacotes de 1500 *bytes* e outros com apenas pacotes de 64 *bytes*, o que

acarreta um aumento na quantidade de pacotes que não serão enviados a GPU para processamento, causando um impacto negativo na vazão total de processamento. O *trace* do caso médio possui fluxos com diferentes tamanhos, o que também causará impacto negativo na vazão total. Já o *trace* do pior caso possui apenas fluxos grandes, contendo 1500 *bytes*, os quais sempre serão enviados para processamento. Analisando os gráficos somos capazes de perceber que, ao contrário do esperado, a vazão total foi maior, para todos os valores de *K*, no *trace* de pior caso. Apenas nos testes realizados utilizando os primeiros 8 kB de dados é que a vazão total do *trace* de caso médio e do *trace* de pior caso assumiram valores próximos. Os testes realizados com o *trace* de melhor caso apontaram o pior desempenho na vazão total de processamento devido à presença de vários fluxos pequenos, os quais não possuem ao menos *K* bytes de dados. O mesmo ocorre para os resultados obtidos com o *trace* de caso médio.

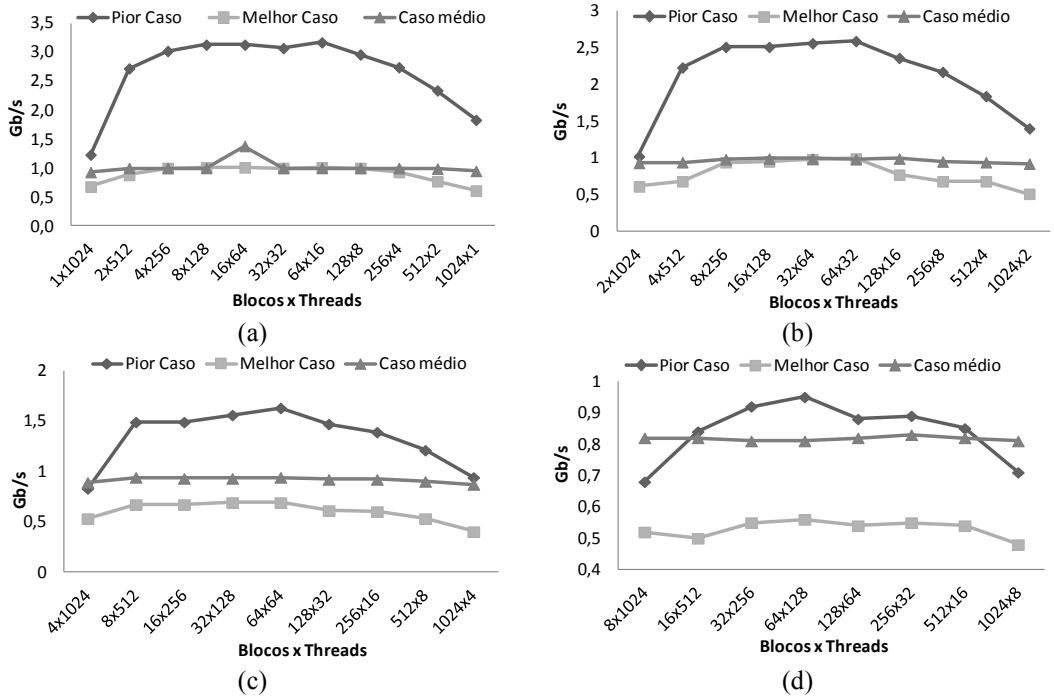


Figura 10. Vazão total para a arquitetura IBTF. (a) Vazão para K = 1kB. (b) Vazão para K = 2kB. (c) Vazão para K = 4kB. (d) Vazão para K = 8kB.

O motivo que explica os resultados obtidos para a vazão total terem sido muito abaixo dos encontrados para a vazão de processamento e para a vazão de fluxo é o fato de que há um custo computacional referente a tarefas de E/S ao realizar leitura dos pacotes do disco. Essa tarefa consiste em copiar o fluxo de dados para a GPU, em seguida aguardar o fim do processamento e copiar os resultados. Nesse caso, ainda existe todo processamento adicional para montar a estrutura de dados, adicionar e remover novos fluxos e para realizar as tarefas de manutenção dessa tabela em memória. Dessa forma, o custo adicional encontrado é um fator diretamente relacionado ao não paralelismo adotado na construção do protótipo. Segundo [Amdahl 1967], para que seja possível medir o ganho em desempenho obtido ao utilizar o conceito de paralelismo para aumentar o desempenho de uma determinada tarefa é preciso considerar o ganho em desempenho obtido ao realizar essa tarefa em paralelo, adicionada ao tempo que continua sendo realizado de maneira sequencial. É possível concluir que a arquitetura que ofereceu o melhor ganho em desempenho para a vazão de processamento foi a IBTF, produzindo um aumento significativo em relação ao 1CTB,

atingindo aproximadamente 114 Gb/s. Conforme apresentado na Tabela 1, percebe-se que a vazão de processamento alcançada é próxima ao máximo obtido na literatura.

Já para a vazão total, a solução que produziu o maior ganho de desempenho foi a 1CTP, com cerca de 1 Gb/s como pode ser observado em Tabela 2.

Tabela 1. Comparação da vazão de processamento de diferentes abordagens

Arquitetura	Vazão de Processamento
1CTB	1,8 Gb/s
1CTP	14 Gb/s
IBTF 8 kB (Caso Médio)	60 Gb/s
IBTF 8 kB (Melhor Caso)	114 Gb/s
Gnort	16 Gb/s
Gregex	122 Gb/s

Isso acontece porque, como apenas o processamento dos pacotes foi acelerado, mantendo as leituras dos pacotes de um disco, realizando cópias da fila de pacotes para a GPU e em seguida copiando os resultados de volta para o *host*, o ganho em desempenho na vazão total não acompanhou o aumento encontrado para a vazão de processamento. Por fim, é importante destacar que a vazão de fluxo, obtida pelo IBTF, de aproximadamente 20 Gb/s é uma abordagem inovadora, que permite que essa vazão seja correspondente a um grande volume real de dados. Em outras palavras, essa vazão deve ser comparada com a vazão total, representando um ganho superior a 20 vezes em comparação com os resultados apresentados na Tabela 2.

Tabela 2. Vazão total para diferentes abordagens

Arquitetura	Vazão total
Gnort	800 Mbps
1CTP	950 Mbps
IBTF 8 kB	930 Mbps

7. Conclusão e trabalhos futuros

O presente trabalho teve como objetivo descrever e desenvolver arquiteturas capazes de realizar identificação de tráfego utilizando expressões regulares em alta velocidade, utilizando o poder computacional proporcionado pelo uso de uma GPU. Além de utilizar uma GPU, as ERs foram transformadas em DFAs de modo que uma determinada cadeia de entrada pudesse ser avaliada através da execução de um único algoritmo de casamento, utilizando uma única tabela de transição que representa várias assinaturas simultaneamente.

O trabalho apresentou resultados que apontaram uma vazão de processamento de aproximadamente 114 Gb/s e uma vazão total acima de 960 Mb/s. É interessante destacar que a vazão obtida através da agregação de fluxos alcançou uma vazão real de mais de 20 Gb/s, mesmo considerando os custos computacionais relativos ao uso da GPU, superando de maneira satisfatória outras abordagens. Como trabalho futuro, consideramos explorar outras funcionalidades da arquitetura CUDA, que oferece um mecanismo para otimizar o agendamento de tarefas para serem processadas na GPU,

denominado *streams*. O uso de s*treams* permite que vários *kernels* sejam chamados em instantes diferentes, sendo uma ótima solução a ser adotada para arquiteturas *multithread*.

Referências

Amdahl, G., "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities", AFIPS Conference Proceedings (30): 483–485, 2007.

Antonello, R. et al., “Deterministic Finite Automaton for Scalable Traffic Identification: the Power of Compressing by Range”, Proc. of IEEE NOMS 2012, Hawai, USA.

Becchi, M. e Crowley, P., “An improved algorithm to accelerate regular expression evaluation”, In Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems 2007. NY, USA, 145-154.

Bernaille, L. et al."Traffic classification on the fly" SIGCOMM CCR 36,2(2006),23-26.

Cascarano, N. et al., “iNFAnt: Nfa Pattern Matching on GPGPU Devices”, ACM SIGCOMM Computer Communication Review 40, 5, 20-26, 2010.

Dainotti, A. et al., "Issues and future directions in traffic classification," IEEE Network, vol.26, no.1, pp.35-40, January-February 2012.

Fernandes, S. et al., "Slimming Down Deep Packet Inspection Systems," INFOCOM Workshops 2009. IEEE Press, Piscataway, NJ, USA, 61-66.

Mu, S. et al., “IP routing processing with graphic processors”, Design, Automation and Test in Europe, 2010, pp. 93-99.

Ruetsch, G. e Micikevicius, P., “Optimizing matrix transpose in cuda”, NVIDIA 2009.

Ryoo, S. et al. “Optimization principles and application performance evaluation of a multithreaded GPU using CUDA”, PPoPP '08. ACM, New York, NY, USA, 73-82.

Sanders, J. e Kandrot, E. “CUDA by example: an introduction to general-purpose GPU programming”, Addison Wesley (2010).

Santos, A. et al., “High-Performance Traffic Workload Architecture for Testing DPI Systems”, IEEE GLOBECOM 2011, vol., no., pp.1,5, 5-9 Dec. 2011.

Smith, R. et al. "Evaluating GPUs for network packet signature matching" ISPASS 2009, vol., no., pp.175,184, 26-28 April 2009.

Szabo , G. et al., “Traffic Classification over Gbit Speed with Commodity Hardware” IEEE J. Communications Software and Systems, vol. 5, 2010.

Sharp, T., “Implementing decision trees and forests on a GPU”, in Proc. ECCV, 2008, Vol. 5305 (2008), pp. 595-608.

Yu, F. et al. ,“Fast and memory-efficient regular expression matching for deep packet inspection”, ANCS '06, ACM, New York, NY, USA, 93-102, 2006.

Vasiliadis, G. et al., "Regular expression matching on graphics hardware for intrusion detection" Proceedings of the 12th RAID, Saint-Malo, France, 2009, pp. 265-283.

Wang, L. et al. "Gregex: GPU Based High Speed Regular Expression Matching Engine" IMIS 2011, vol., no., pp. 366, 370, June 30 2011-July 2.

Nemo: Procurando e Encontrando Anomalias em Aplicações Distribuídas

Brivaldo A. da Silva Junior[1], **Fabrício B. de Carvalho**[1], **Ronaldo A. Ferreira**[1]

[1]Faculdade de Computação (FACOM)
Universidade Federal de Mato Grosso do Sul (UFMS)
CEP – 79070-900 – Campo Grande – MS – Brasil

{brivaldo,fabricio,raf}@facom.ufms.br

Abstract. *Diagnosing anomalies in large enterprise networks consumes significant time of technical support teams, mainly because of the numerous complex interactions among the applications and network elements (servers, routers, links, etc.). The most complete and promising approach for solving this problem, called Sherlock, uses network traces to automatically build an Inference Graph (IG) that models the multiple interactions and dependencies present in a distributed environment. Despite the progress provided by Sherlock in the problem modeling, its execution time for inferring the probable causes and the precision of its anomaly detection results leave much room for improvements. This work proposes Nemo, a tool that explores domain-specific knowledge and a theoretical property of Bayesian Networks to significantly reduce the IG and consequently the execution time. Simulation results using real and synthetic data show that Nemo reduces Sherlock's execution time by over 90% and improves its precision in all simulated scenarios.*

Resumo. *Diagnosticar anomalias em grandes redes corporativas consome tempo considerável das equipes de suporte técnico, principalmente pela complexidade das inúmeras interações existentes entre as aplicações e os elementos de rede (servidores, roteadores, enlaces, etc.). A abordagem mais completa e promissora para a solução desse problema, denominada Sherlock, utiliza traços de rede para construir automaticamente um Grafo de Inferência (GI) que modela as múltiplas interações e dependências presentes em um ambiente distribuído. Apesar do progresso feito por Sherlock na modelagem do problema, o seu tempo de execução para se inferir as possíveis causas e a precisão dos resultados de detecção das anomalias ainda deixam a desejar. Este trabalho propõe Nemo, uma ferramenta que explora conhecimento específico do domínio do problema e uma propriedade teórica de Redes Bayesianas para reduzir significativamente um GI e, consequentemente, o tempo de execução. Resultados de simulação utilizando dados reais e sintéticos mostram que Nemo reduz o tempo de execução de Sherlock em mais de 90% e melhora sua precisão em todos os cenários simulados.*

1. Introdução

As grandes corporações dependem cada vez mais da Internet para desenvolver suas atividades cotidianas. Elas utilizam a grande rede para vender produtos, para troca de mensagens entre funcionários e parceiros, ou para divulgar suas marcas e serviços. A proliferação de atividades desenvolvidas na Internet torna necessária a construção de ambientes distribuídos mais sofisticados para dar vazão a essas novas demandas, além de levar a aplicações distribuídas

mais complexas e dependentes de vários componentes da infraestrutura de comunicação. Uma aplicação Web, por exemplo, pode ser executada em um cluster com servidores interligados por inúmeros roteadores em um *Data Center*, interagir com um servidor de banco de dados e fazer consultas a vários serviços distribuídos, como *Domain Name System* (DNS), *Active Directory* (AD), *Windows Internet Name Service* (WINS), *Radius*, etc. Consequentemente, anomalias, provenientes de falhas ou sobrecargas, em um dos componentes desse novo ambiente podem levar ao mal funcionamento de uma aplicação e uma percepção ruim dos clientes.

Nesse novo cenário, detectar anomalias em ambientes distribuídos tem se tornado uma tarefa extremamente desafiadora. Ferramentas tradicionais de gerenciamento, como Openview [HP 1990], Tivoli [IBM 1996], nmap [Lyon 1997], Nagios [Galstad 1999], ping e traceroute, deixam muito a desejar, pois fornecem informações de alta granularidade e dependem demasiadamente das habilidades dos administradores de rede para detecção e diagnóstico de problemas. Essas ferramentas, normalmente, fornecem apenas informações estatísticas, de conectividade ou de disponibilidade de um componente de software ou de hardware da rede, mas são incapazes de determinar os relacionamentos e as dependências entre eles, dificultando o rápido diagnóstico de um problema observado pelos usuários da rede.

As principais propostas para detecção e diagnóstico de anomalias em ambientes distribuídos existentes na literatura podem ser divididas em dois grandes grupos: intrusivas e não intrusivas. Nas propostas intrusivas, os principais exemplos são Magpie [Barham et al. 2004], Pinpoint [Chen et al. 2004], PIP [Reynolds et al. 2006] e vPath [Tak et al. 2009]. Nessas propostas, as aplicações são modificadas e instrumentadas para registrar os caminhos causais, ou seja, os caminhos percorridos por uma determinada requisição. Com os caminhos causais, é possível determinar os componentes de software e hardware envolvidos no processamento de uma requisição. Dessa forma, uma anomalia em uma aplicação pode ser mais facilmente diagnosticada analisando-se somente o conjunto de componentes utilizados no processamento da requisição. Algumas propostas, como o X-Trace [Fonseca et al. 2007], não se limitam a apenas instrumentar as aplicações, mas toda a pilha de protocolos, aumentando a visibilidade de execução das aplicações.

Muito embora as abordagens intrusivas produzam resultados mais confiáveis, elas dependem fortemente de programadores para instrumentar as aplicações, além de não serem úteis para monitorar aplicações legadas. Algumas abordagens sugerem a instrumentação de *frameworks* para essas situações [Barham et al. 2004], mas nesses casos a efetividade do diagnóstico cai bastante, além de não cobrir as aplicações legadas que não utilizam *frameworks*.

Os principais exemplos de abordagens não intrusivas Sherlock [Bahl et al. 2007], Constellation [Barham et al. 2008], Spotlight [John et al. 2010] e NSDMiner [Natarajan et al. 2012] não modificam as aplicações ou *frameworks* e utilizam somente informações de traços de rede para inferir as causas dos problemas. As principais diferenças de cada abordagem são determinadas, basicamente, pelas informações analisadas, pelas metodologias estatísticas de inferência e pelas soluções sistêmicas dos problemas inerentes à coleta de informações.

Embora as abordagens não intrusivas sejam menos precisas que as intrusivas, elas são consideradas mais viáveis na prática, pois não exigem conhecimento prévio das aplicações e nem modificações para incluir mecanismos de registro de eventos, além de serem úteis e apropriadas para ambientes legados e pouco documentados.

A abordagem não intrusiva mais completa e promissora para detecção e diagnóstico de anomalias em ambientes distribuídos, denominada Sherlock, utiliza traços de rede para construir automaticamente um Grafo de Inferência (GI) que modela as múltiplas interações e dependências presentes em um ambiente distribuído. As dependências são probabilísticas e representadas por pesos (probabilidades) nas arestas que interligam os nós (serviços, roteadores, enlaces, etc.). Apesar do progresso feito por Sherlock na modelagem do problema, o seu tempo de execução para se inferir as possíveis causas e a precisão dos resultados de detecção das anomalias ainda deixam a desejar. Sherlock utiliza um algoritmo de inferência Bayesiana (Ferret) bastante custoso cuja função de pontuação inclui todos os nós do grafo, tornando-o não escalável em cenários em que o GI possui milhares de nós.

Este trabalho propõe Nemo (*Network Monitor*), uma ferramenta não intrusiva que explora conhecimento específico do domínio do problema e uma propriedade teórica de Redes Bayesianas para reduzir significativamente um GI e, consequentemente, o tempo de execução. Resultados de simulação utilizando dados reais e sintéticos mostram que Nemo reduz o tempo de execução de Sherlock em mais de 90% e melhora sua precisão em todos os cenários simulados.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 descreve a ferramenta Nemo, seus algoritmos e suas propriedades teóricas. A Seção 3 descreve brevemente aspectos de implementação da ferramenta. A Seção 4 apresenta os resultados dos experimentos realizados e a Seção 5 conclui este trabalho.

2. Ferramenta Nemo

A ferramenta Nemo foi inspirada no Sherlock e utiliza boa parte de sua modelagem conceitual, bem como parte das estruturas de dados que foram inicialmente introduzidas em [Bahl et al. 2007]. Entretanto, Nemo introduz novos conceitos teóricos e algoritmos bem mais eficientes que os utilizados por Sherlock. Assim como Sherlock, a ferramenta Nemo é dividida em dois componentes principais: agente e gerente. A organização geral de Nemo está ilustrada na Figura 1.

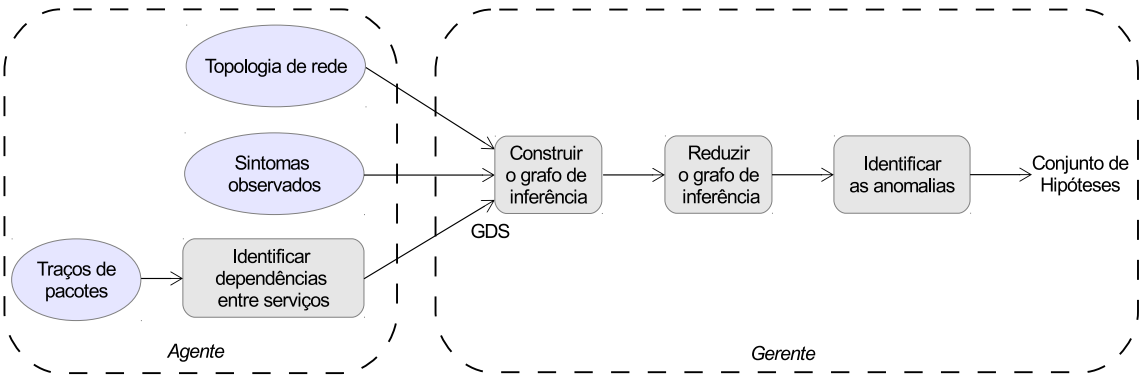


Figura 1. Visão geral da ferramenta Nemo.

Os agentes de Nemo são instalados em estações de trabalho e monitoram os pacotes que são enviados e recebidos pelas estações. A partir dos pacotes monitorados, um agente constrói

localmente um Grafo de Dependências de Serviços (GDS) para os serviços acessados pela estação e monitora os tempos de resposta de cada acesso para construir estatísticas confiáveis sobre o tempo médio de acesso de cada serviço. Quando o tempo de resposta é muito superior ao tempo médio, indicando possivelmente uma anomalia, os agentes solicitam ao gerente que seja executado o processo de diagnóstico de anomalias. Além disso, os agentes coletam informações de topologia da rede, por meio de um módulo de traceroute implementado internamente nos agentes, e enviam essas informações para o gerente.

O gerente é responsável por combinar todas as dependências entre serviços fornecidas pelos agentes e consolidar as visões locais em um GDS global para toda a rede. Além disso, o gerente faz a união do GDS com as informações de topologia para gerar um Grafo de Inferência (GI). O GI captura todas as dependências entre serviços e elementos de rede como enlaces, roteadores e servidores. O gerente também executa o processo de diagnóstico de anomalias quando solicitado pelos agentes. A Seção 2.1 descreve mais detalhadamente como os GDSs são construídos por Nemo.

As principais diferenças entre Nemo e Sherlock estão nos módulos de redução do GI (inexistente em Sherlock) e no módulo de identificação de anomalias. Muito embora a ferramenta tenha exigido um grande esforço de implementação e implantação em um ambiente de produção, o foco deste trabalho, devido à limitação de espaço, será nas melhorias teóricas e algorítmicas introduzidas nos módulos de redução do GI e no de detecção de anomalias. Esses módulos serão discutidos nas Seções 2.4 e 2.5, respectivamente. Os aspectos mais relevantes da implementação da ferramenta serão discutidos na Seção 3.

As próximas seções introduzem conceitos teóricos e descrevem as principais estruturas de dados utilizadas neste trabalho.

2.1. Grafo de Dependências de Serviços

O Grafo de Dependências de Serviços (GDS) descreve as dependências existentes entre serviços. Cada serviço acessado por um cliente, e capturado pelos agentes de Nemo, é mapeado para um nó do GDS. Um serviço S_1 é dependente de um serviço S_2 se o acesso a S_1 é normalmente precedido por um acesso a S_2. O GDS representa essa dependência por uma aresta direcionada de S_2 para S_1 e atribui um peso (probabilidade) a essa aresta de acordo com a força da dependência. A probabilidade de dependência (peso da aresta) é calculada dividindo-se o número de vezes que acessos ao serviço S_2 precedem acessos a S_1 dentro de um intervalo limite de tempo (intervalo de dependência) pelo número de vezes que S_1 é acessado em todos os traços coletados pelos agentes. O intervalo de dependência é um parâmetro do sistema e pode ser alterado de acordo com características específicas do sistema sendo monitorado. O valor sugerido por Sherlock e também utilizado neste trabalho é fixo e igual a 10ms. Esse valor se mostrou bastante eficaz na captura das dependências.

A Figura 2 ilustra o mapeamento da dependência entre os serviços HTTP e DNS. Nesse caso, como o acesso ao serviço HTTP ocorre dentro do intervalo de dependência do acesso ao serviço DNS, infere-se que o serviço HTTP é dependente do serviço DNS. A probabilidade da dependência é calculada de acordo com o número de vezes que esses dois acessos ocorrem dentro do intervalo de dependência. Observe que essa maneira de se calcular as probabilidades de dependência captura adequadamente situações em que as respostas de DNS são armazenadas em cache e que acessos seguidos ao mesmo endereço não precisam de novos acessos ao servidor de nomes. Nesse caso, a probabilidade de dependência será menor que 1,0.

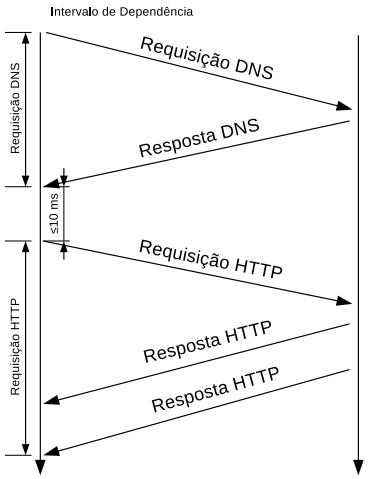


Figura 2. Mapeamento de dependências entre serviços.

2.2. Grafo de Inferência de Rede

Um GDS captura adequadamente as dependências entre serviços, mas não inclui informações sobre a infraestrutura de comunicação. Um Grafo de Inferência (GI) estende um GDS e é definido como um Grafo Direcionado Acíclico (GDA), com pesos nas arestas, que representa as dependências entre componentes de um ambiente distribuído, sendo que componentes incluem serviços e elementos de hardware. Em um GI, há três tipos de nós:

- **nós de observação:** modelam a experiência que o usuário está observando ao acessar um serviço. Essa experiência representa o tempo de acesso ao serviço e pode indicar que o serviço está *ativo* (operando normalmente e com tempos de resposta baixos), *problemático* (operando, mas com tempos de resposta muito altos), ou *inativo*.
- **nós raízes de problema:** representam servidores, equipamentos de rede, enlaces ou serviços. Quando um nó desse tipo apresenta um problema, alguns nós de observação acusarão o problema, fazendo com que o usuário perceba a degradação de um serviço.
- **meta nós ou nós de interligação:** servem para interligar os nós de observação a outros nós do GI que podem ser causadores de problemas. Eles são de três tipos: ruído máximo (ligações um-para-um), seleção (modelam balanceadores de carga) e *failover* (modelam redundância de servidores do tipo mestre/escravo).

O diferencial da modelagem realizada por Sherlock [Bahl et al. 2007] em relação a outras abordagens, como Shrink [Kandula et al. 2005] e SCORE [Kompella et al. 2005], foi a adição do estado *problemático* em cada nó. Cada nó do GI possui um estado que é representado pela probabilidade do nó estar *ativo*, *problemático* e *inativo*. Dessa forma, um nó com estado $(0,3; 0,5; 0,2)$ indica que o nó possui 30% de chance de estar *ativo*, 50% de chance de estar *problemático* e 20% de chance de estar *inativo*. A soma das probabilidades $P_{\text{ativo}} + P_{\text{problemático}} + P_{\text{inativo}}$ deve ser 1,0.

Assim como no GDS, as arestas no GI são direcionadas e possuem pesos (probabilidades) que representam a força da dependência entre dois nós. Para cada tipo de meta nó, há uma tabela-verdade que indica como as probabilidades de um nó são calculadas a partir das probabilidades de seus antecessores e dos pesos das arestas. As tabelas-verdade para cada tipo de nó estão definidas em [Bahl et al. 2007].

Finalmente, o GI possui dois nós especiais utilizados para possíveis erros no modelo, o nó *sempre problemático* (AT - *always troubled*) e o nó *sempre inativo* (AD - *always down*), que possuem seus estados definidos como $(0; 1; 0)$ e $(0; 0; 1)$, respectivamente. Esses nós são vinculados a cada nó de observação para representar possíveis informações que não estão mapeadas no modelo.

A Figura 3 mostra uma parte de um GI que foi gerado automaticamente por Nemo. Esse grafo foi construído com informações coletadas por 20 agentes instalados na rede da UFMS. O grafo mostra, entre outras coisas, as dependências dos serviços NetBios (137) e FTP (21) do serviço DNS (53). Nesse caso, os nós superiores são os nós raízes de problema, os do meio são os meta nós e os inferiores são os nós de observação.

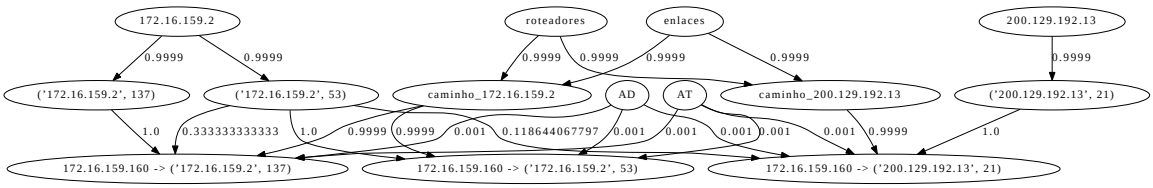


Figura 3. Parte de um GI de uma rede em produção gerado automaticamente por Nemo.

2.3. Redes Bayesianas

Redes Bayesianas são comumente utilizadas para realizar inferências probabilísticas em áreas como Estatística, Inteligência Artificial, Mineração de Dados, etc. Uma Rede Bayesiana (RB) nada mais é que um modelo gráfico que codifica relações probabilísticas entre variáveis de interesse. Sua utilidade advém de sua capacidade de codificar dependências entre todas as variáveis de interesse, de lidar com situações em que nem todas as informações são conhecidas, e de ser útil para se inferir relacionamentos causais. Segue, abaixo, a definição formal de uma Rede Bayesiana.

Definição 1 *Uma Rede Bayesiana $\mathfrak{B}$ é definida como um par $\mathfrak{B} = (G, P)$, em que $G = (V(G), A(G))$ é um grafo direcionado acíclico com conjunto de vértices $V(G) = \{X_1, \ldots, X_n\}$ e conjunto de arestas $A(G) \subseteq V(G) \times V(G)$, e P é uma distribuição de probabilidade conjunta definida nas variáveis correspondentes aos vértices do conjunto $V(G)$ como segue:*

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \pi(X_i))$$

em que $\pi(X_i)$ representa o conjunto de pais (ancestrais diretos) de X_i.

Uma grande contribuição de Sherlock foi a modelagem do problema de diagnóstico de anomalias como um GI. O GI pode ser visto como uma RB, pois é um grafo direcionado acíclico em que os nós representam variáveis aleatórias cujos valores são determinados pelos seus ancestrais diretos de acordo com uma tabela-verdade. Sendo assim, o problema de detecção de anomalias em Sherlock ficou reduzido ao de se explicar inteiramente as variáveis desconhecidas na RB, ou seja, descobrir as probabilidades dos nós raízes de problema que melhor explicam as probabilidades dos nós de observação.

2.3.1. *d*-separação e *d*-conexão

Por tentar explicar todas as variáveis desconhecidas da RB, Sherlock é bastante ineficiente, tanto do ponto de vista de armazenamento quanto do ponto de vista de tempo de execução. Nemo tira vantagem do domínio do problema e tenta explicar somente as variáveis que interferem diretamente em sintomas reportados pelos usuários. Para isso, Nemo explora a propriedade de independência condicional em RBs para descartar variáveis que sejam de fato independentes das variáveis de interesse e que, portanto, não interferem em seus valores. A propriedade de independência condicional em RBs foi bastante estudada no passado, resultando no conceito de *d*-separação. Geiger *et al.* [Geiger et al. 1989] propuseram um algoritmo bastante eficiente, de complexidade linear, para o cálculo da *d*-separação em uma RB.

O conceito de *d*-separação é melhor entendido quando associado ao conceito de *d*-conexão. A definição desses conceitos utiliza os termos nós colisores e nós não colisores. A função desses nós pode ser melhor compreendida pelas representações gráficas mostradas na Figura 4. Nós colisores e não colisores são nós cujos graus de saída são zero e diferente de zero, respectivamente.

Figura 4. Nó v_1 como (a) colisor e (b) não colisor.

Definição 2 *d-**conexão**: Seja G um grafo direcionado em que X, Y e Z são conjuntos disjuntos de vértices de G, então X e Y são d-conectados por Z em G, se, e somente se, existe um caminho não direcionado U entre vértices de X e de Y, tal que para cada nó colisor c em U, c ou um descendente de c está em Z e nenhum nó não colisor pertence a Z.*

Definição 3 *d-**separação**: X e Y são d-separados por Z em G, se, e somente se, eles não são d-conectados por Z.*

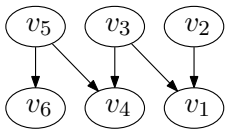


Figura 5. Considere os conjuntos de vértices $X = \{v_1\}$, $Y' = \{v_4\}$, $Y'' = \{v_5, v_6\}$ e $Z = \{v_2, v_3\}$. X e Y' são d-conectados por Z e, X e Y'' são d-separados por Z.

As Definições 2 e 3 formalizam os conceitos de *d*-conexão e *d*-separação, respectivamente, e a Figura 5 mostra um exemplo da utilização desses conceitos em um grafo. Na Seção 2.4, esses conceitos serão utilizados no desenvolvimento de um novo algoritmo que reduz significativamente os GIs gerados por Nemo.

2.4. Redução do Grafo de Inferência

O GI, como proposto por Sherlock, inclui todos os serviços, elementos de hardware e clientes monitorados em uma rede corporativa. Em situações de anomalias, é bem sabido que

apenas parte desses componentes realmente apresenta problemas. Além disso, dependendo da anomalia, poucos clientes são de fato afetados. Como exemplo, pode-se citar um defeito em um roteador que afeta apenas os clientes das sub-redes que são roteadas pelo equipamento defeituoso.

Com base nas observações acima, Nemo procura identificar apenas os nós do GI que influenciam os sintomas observados e descartar os demais nós que são comprovadamente independentes dos sintomas. Para se descartar os nós com a certeza que eles não são os causadores dos sintomas, Nemo utiliza o conceito de *d*-separação.

A questão chave para utilização da *d*-separação para redução do GI é a definição de pelo menos dois dos conjuntos X, Y e Z, pois o terceiro é obtido de modo eficiente (em tempo linear) pelo algoritmo de *d*-separação. Nemo constrói o conjunto Z como sendo os nós de observação que estão reportando problemas, ou seja, os nós sintomas. O conjunto X é definido como sendo os nós raízes de problema que são antecessores dos nós de Z, pois esses são nós que podem afetar os nós sintomas. Os nós de X podem ser obtidos utilizando um algoritmo de busca em largura a partir dos nós sintomas invertendo-se as orientações das arestas. O Algoritmo 1 sumariza essa ideia. Observe que o conjunto Y, obtido pela *d*-separação (Linha 4), contém os nós do GI que são independentes condicionalmente dos sintomas. Para se obter um grafo reduzido R, os nós de Y e suas arestas são removidas do grafo original. R conterá os nós de X, os nós de Z, os nós de observação que são dependentes condicionalmente de Z que podem explicar os sintomas observados, e os meta nós que unem os demais nós do grafo.

Algoritmo 1: Redução (G)

Entrada: Grafo de Inferência G.
Saída: Subgrafo R de G com apenas os nós e arestas que são relevantes para explicação dos sintomas.

```
1 início
2   Z ← Conjunto de nós de observação de G que estão observando degradações
    nos acessos (sintomas);
3   X ← Conjunto de nós raízes de problema de G que são ancestrais dos nós de Z;
4   Y ← d-separação(G, X, Z);  /* Y contém os nós de G tal que,
    X e Y são d-separados por Z em G.  */
5   R ← G − Y;
6   retorna R;
7 fim
```

2.5. Função de Pontuação

O processo de detecção e diagnóstico de anomalias a partir de um GI consiste em encontrar um vetor de atribuição de probabilidades aos estados *ativo, problemático* e *inativo* dos nós raízes de problema que melhor explique os estados observados pelos nós de observação. A ideia básica é propagar os estados (probabilidades) dos nós raízes de problema até os nós de observação e comparar os valores propagados com os estados dos nós de observação obtidos por meio das evidências. As evidências nada mais são que os tempos de acesso aos serviços. Para se determinar o melhor vetor, o algoritmo de inferência utiliza uma função de pontuação que leva em consideração os tempos de resposta dos serviços monitorados pelos agentes.

Em [Bahl et al. 2007], foi observado que os tempos de resposta dos serviços monitorados seguem uma distribuição bimodal em que uma moda caracteriza o tempo médio quando o serviço está normal (*ativo*) e uma segunda moda que caracteriza o tempo médio quando o serviço está enfrentando problemas (*problemático*). Nemo monitorou alguns serviços da rede da UFMS e constatou que a hipótese de distribuição bimodal é de fato verdadeira. A Figura 6 mostra a distribuição de tempos de um dos serviços monitorados. As amostras próximas às modas, quando consideradas separadamente, formam distribuições normais com médias bem próximas às modas. Nemo isola as duas distribuições utilizando o método de clusterização "*k-means*" para encontrar as duas modas e utiliza o ponto médio entre as duas modas para separar as distribuições (ativa e problemática).

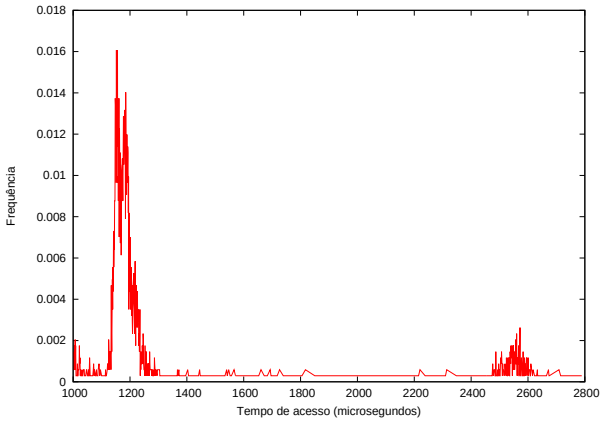


Figura 6. Distribuição de tempos de resposta de um serviço monitorado por Nemo. A primeira moda indica que o serviço está *ativo* e a segunda *problemático*.

Nemo e Sherlock diferem em suas funções de pontuação, pois Sherlock assume e utiliza diretamente a Função de Distribuição de Probabilidade Normal para pontuar um tempo de acesso observado. Isso faz com que tempos de acesso menores que a média recebam pontuações inferiores ao tempo médio ou até mesmo do que tempos ligeiramente superiores à média. Nemo, por outro lado, explora novamente conhecimento específico do domínio do problema para propor uma nova função de pontuação. A função de pontuação de Nemo é discreta e explora a ideia de que tempos de acesso inferiores a média ou ligeiramente superiores devem ter pontuações iguais, pois um tempo de acesso inferior a média indica que o serviço está operando normalmente. Nemo considera apenas os tempos de acesso válidos, tempos muito pequenos, como os decorrentes de *TCP reset*, são descartados. A função de pontuação de Nemo é mostrada no Algoritmo 2. Os resultados de simulação apresentados na Seção 4 mostram que essa função de pontuação aumenta a precisão do algoritmo de inferência nos casos avaliados.

3. Aspectos de Implementação

A ferramenta Nemo, composta de agentes e um gerente, foi implementada e testada na rede de produção da UFMS. Os agentes de Nemo foram desenvolvidos utilizando a linguagem multiplataforma Python com módulos funcionais em ambientes Windows e Linux (netifaces, socket, pcapy, etc.). Além disso, em cada sistema operacional, foi necessário instalar uma biblioteca de captura de pacotes de rede, em Windows a WinPcap e em Linux a LibPcap.

Os agentes de Nemo foram instalados em 20 estações de trabalho Windows. A implantação desses agentes foi feita utilizando a tecnologia de Políticas de Grupo (*Group*

Algoritmo 2: Pontuação (n)

Entrada: Nó de observação n.
Saída: Pontuação em n.

início
  P_{ativo}, $P_{\text{problemático}}$, $P_{\text{inativo}} \leftarrow$ Estado de n;
  μ_{ativo}, $\sigma_{\text{ativo}} \leftarrow$ Média e desvio padrão da distribuição *A*tivo de n;
  se *evidência*$(n) < \mu_{ativo} + \sigma_{ativo}$ **então**
    retorna $P_{\text{ativo}} * 1,0$;
  se *evidência*$(n) < \mu_{ativo} + 2 * \sigma_{ativo}$ **então**
    retorna $P_{\text{ativo}} * 0,7 + P_{\text{problemático}} * 0,25 + P_{\text{inativo}} * 0,05$;
  se *evidência*$(n) < \mu_{ativo} + 3 * \sigma_{ativo}$ **então**
    retorna $P_{\text{ativo}} * 0,4 + P_{\text{problemático}} * 0,5 + P_{\text{inativo}} * 0,1$;
  se *evidência*$(n) < \mu_{ativo} + 4 * \sigma_{ativo}$ **então**
    retorna $P_{\text{ativo}} * 0,05 + P_{\text{problemático}} * 0,75 + P_{\text{inativo}} * 0,2$;
  retorna $P_{\text{problemático}} * 0,8 + P_{\text{inativo}} * 0,2$;
fim

Policy) do AD (*Active Directory*) em execução no servidor de domínio com Windows Server 2008. Para a realização da instalação distribuída via AD, foi criado um pacote executável MSI (*Microsoft Software Installer*), exigência do ambiente, utilizando o *framework* py2exe do Python.

O gerente foi desenvolvido em Python e C++, de tal forma que a parte de comunicação e a interface foram desenvolvidas em Python e a parte de processamento em C++. Essa separação foi necessária devido ao enorme *overhead* existente em Python e que tornava o processamento extremamente lento. Para a construção do GI foi utilizada a biblioteca de estruturas de dados Lemon que é compatível com ambas as linguagens. Além disso, as trocas de mensagens entre os agentes e o gerente utilizam um protocolo simples de aplicação, criado para enviar dados de atualização do GDS e requisições de execução do processo de diagnóstico de anomalias. Os grafos de inferência, como o da Figura 3, foram gerados utilizando a biblioteca Graphviz.

Um ponto interessante da ferramenta, e que foi observado durante os testes do sistema, é que ela detectou um serviço desconhecido pela equipe de administração da rede. Um usuário havia instalado um Gerente SNMP em um Câmpus da instituição e não havia comunicado à equipe de suporte. Muito embora esse não seja o objetivo da ferramenta, ela pode ser utilizada para detectar serviços que ferem a política de uso da rede como, por exemplo, comunidades P2P e serviços não homologados.

4. Avaliação Experimental

Nesta seção, são apresentados resultados de simulação para avaliar vários aspectos de desempenho de Nemo. Muito embora Nemo tenha sido implementado e testado em um ambiente de produção, a avaliação por simulação permite que cenários maiores e mais complexos sejam explorados, além de se tratar de um ambiente controlado e passível de reprodução dos resultados. Inicialmente, é avaliado o algoritmo de redução do grafo de inferência. Este resultado é importante, pois o algoritmo de redução pode ser utilizado por outras abordagens que utilizam grafos de inferência para detecção de anomalias. Em seguida, Nemo é comparado com Sherlock e Spotlight. Spotlight [John et al. 2010] é uma proposta

mais recente e bem mais rápida que Sherlock. Entretanto, Spotlight utiliza uma heurística para reduzir o GI em um grafo bipartido que não preserva as propriedades teóricas do modelo inicial. Além disso, os resultados de precisão de Spotlight são bem inferiores aos de Sherlock. As métricas utilizadas para comparação são: precisão e tempo de execução. As próximas seções descrevem a metodologia de avaliação e apresentam os resultados numéricos.

4.1. Metodologia de Avaliação

Para a avaliação, foram gerados aleatoriamente diversos grafos de inferência correspondentes a topologias de rede, dependências de serviços e distribuições de tempo de acesso. A metodologia utilizada para geração do grafo é a mesma utilizada em [Bahl et al. 2007, John et al. 2010]. Os tempos de acesso são gerados de forma aleatória utilizando distribuições Normais com médias (ativo e problemático) e desvios padrão obtidos a partir de dados reais coletados pelos agentes de Nemo na rede da UFMS. As probabilidades de dependência são geradas aleatoriamente com distribuição uniforme.

As simulações foram realizadas em uma única máquina com dois processadores Intel Xeon E5530, 32GB de RAM e sistema operacional Linux com kernel 3.2.0. Todos os resultados foram obtidos a partir da média de pelo menos 50 execuções independentes. Além disso, um intervalo de confiança de 95% foi calculado para cada um dos resultados.

Para se avaliar a precisão, anomalias devem ser injetadas artificialmente nos grafos de inferência. Uma anomalia é introduzida selecionando-se aleatoriamente um nó raiz de problema e marcando-o como problemático. Após essa escolha, a anomalia deve ser propagada e refletida nos nós de observação. A propagação é feita por meio de um passeio probabilístico no grafo de inferência utilizando-se as probabilidades de dependência entre os nós. Observe que uma anomalia injetada em um nó raiz de problema pode refletir em vários nós de observação. Um algoritmo de inferência acerta quando indica corretamente os nós raízes de problema que foram marcados inicialmente como problemáticos.

Os grafos de inferência foram gerados com quantidades de nós raízes de problema que variaram de 100 a 20.000 para se avaliar o algoritmo de redução e de 150 a 250 para se avaliar a precisão e o tempo de execução. A quantidade de anomalias injetadas variaram de 1 a 4.

4.2. Redução do Grafo de Inferência

O Algoritmo 1, apresentado na Seção 2.4, é executado para reduzir a quantidade de nós do GI com o objetivo de se reduzir o tempo de execução do processo de diagnóstico de anomalias. Essa redução faz com que somente nós que estão relacionados às anomalias permaneçam no grafo e, consequentemente, que os nós que não auxiliam na detecção sejam descartados.

Os grafos utilizados na avaliação do algoritmo foram gerados variando-se as quantidades de anomalias injetadas e de nós raízes de problema. Para cada combinação, foram gerados 100 grafos diferentes utilizando-se a metodologia descrita na Seção 4.1. Pode-se observar na Figura 7 que a taxa de redução apresentada é diretamente proporcional a quantidade de nós raízes de problema e inversamente proporcional a quantidade de anomalias. A taxa de redução variou entre 50% até pouco mais de 99%. Essa redução elevada é explicada pela alta quantidade de nós raízes de problema que não explicam os sintomas provocados pelas anomalias. A figura também mostra os intervalos de confiança de 95% para a média.

Muito embora a redução seja significativa, ela será útil somente se o processo de diagnóstico não for afetado. A seção seguinte mostra que o algoritmo de redução não interfere

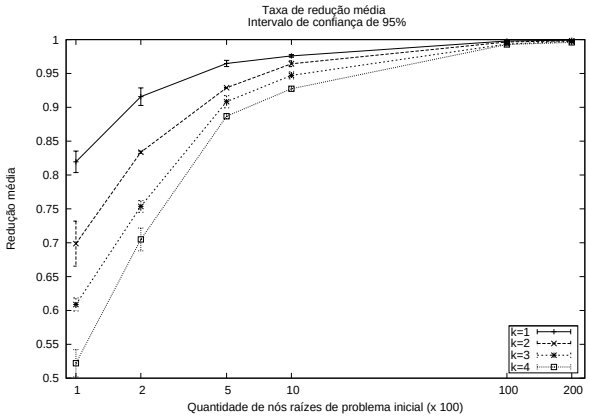


Figura 7. Taxa de redução média de nós raízes de problema, com um intervalo de confiança de 95%, de grafos com diferentes quantidades (100**,** 200**,** 500**,** $1k$**,** $10k$ **e** $20k$**) de nós raízes de problema e com** 1 **até** 4 **anomalias simultâneas.**

na precisão do processo de detecção de anomalias.

4.3. Precisão

A precisão de uma abordagem é definida como a razão entre a quantidade de execuções com as detecções corretas pela quantidade total de execuções. A Figura 8 mostra a precisão das abordagens com GIs de quantidade fixa de nós de observação, diferentes quantidades de nós raízes de problema e anomalias simultâneas. Nas comparações, Nemo se mostrou superior a Sherlock [Bahl et al. 2007] e ao Spotlight [John et al. 2010] em todos os cenários estudados.

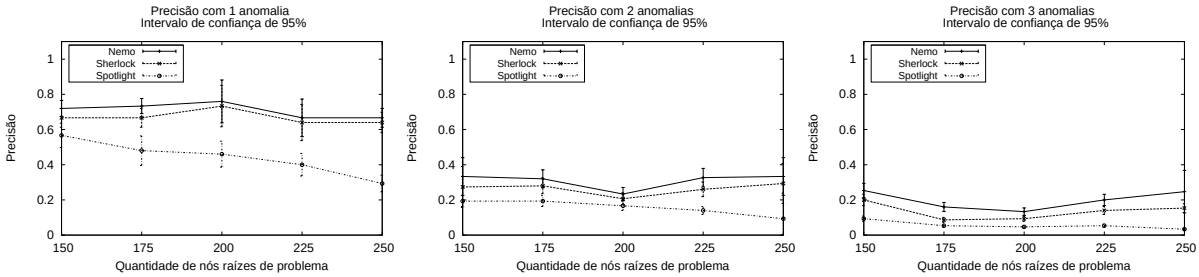


Figura 8. Comparação da precisão das abordagens com uma, duas e três anomalias simultâneas, variando a quantidade de nós raízes de problema.

Muito embora os valores de precisão para duas e três anomalias sejam baixos, os resultados apresentados consideram como acerto somente quando todas as anomalias estão no vetor de atribuição selecionado como melhor pelo algoritmo de inferência. Entretanto, Nemo identifica corretamente pelo menos uma das anomalias em mais de 82% dos casos, conforme mostra a Tabela 1. Quando se considera os cinco melhores vetores, Nemo acerta pelo menos uma das anomalias em mais de 97% dos casos.

A verificação de hipóteses com cinco diferentes pontuações é totalmente plausível por administradores de rede em um processo de diagnóstico de anomalias em redes corporativas. O processo pode ser desenvolvido de forma iterativa, corrigindo-se inicialmente as anomalias que foram indicadas corretamente e repetindo-se o processo até que todas sejam resolvidas.

Tabela 1. Percentual de detecção correta de pelo menos uma anomalia.

Quantidade de nós raízes de problema	Percentual de detecção correta					
	2 anomalias			3 anomalias		
	Nemo	Sher.	Spot.	Nemo	Sher.	Spot.
150	82,0	79,3	73,3	97,3	95,3	91,3
175	87,3	84,6	64,6	94,6	91,3	72,6
200	91,3	87,3	65,3	95,3	92,0	78,6
225	92,0	90,6	62,0	92,6	91,3	67,3
250	91,3	90,0	58,0	94,6	93,3	60,0

4.4. Tempo de execução

A Tabela 2 compara os tempos de execução das três abordagens quando executadas com os grafos de inferência utilizados na Seção 4.2. Como esperado, Nemo é bem mais rápido que Sherlock em todos os casos. Entretanto, Spotlight é mais rápido que Nemo quando duas e três anomalias são injetadas. Os baixos tempos de execução de Spotlight decorrem de uma abordagem bem diferente das demais e que é incapaz de produzir múltiplas hipóteses, além de ser bem menos precisa. O algoritmo de redução também foi testado com Spotlight, mas não resultou em redução significativa dos tempos de execução.

Tabela 2. Tempo médio de execução das três abordagens.

Quantidade de nós raízes de problema	Tempo médio de execução (s)								
	1 anomalia			2 anomalias			3 anomalias		
	Nemo	Sher.	Spot.	Nemo	Sher.	Spot.	Nemo	Sher.	Spot.
150	0,0028	0,0118	0,0064	0,0858	2,5260	0,0026	6,2356	653,4461	0,0228
175	0,0018	0,0180	0,0016	0,1162	3,9704	0,0034	6,7199	1548,27	0,0560
200	0,0032	0,0196	0,0004	0,1188	4,7354	0,0040	7,4836	3853,56	0,0584
225	0,0038	0,0198	0,0054	0,1104	6,1285	0,0046	24,1810	8289,36	0,0570
250	0,0056	0,0236	0,0050	0,1210	7,8026	0,0062	32,4695	14960,9	0,0672

5. Conclusão

Neste artigo é proposta e avaliada uma ferramenta para detecção e diagnóstico de anomalias. A ferramenta é não intrusiva, portanto não modifica aplicações, servidores ou sistemas legados. Ela utiliza somente traços de rede para inferir as possíveis causas dos problemas, sendo particularmente útil para ambientes legados. O código fonte da ferramenta está disponível em http://ndsg.facom.ufms.br/nemo. Uma funcionalidade ainda não implementada e testada por completo é uma interface gráfica para os usuários poderem alterar parâmetros e visualizar os resultados em tempo real.

As principais contribuições deste trabalho são: um algoritmo eficiente para redução de grafos de inferência e uma nova função de pontuação para inferência de anomalias. O algoritmo se mostrou bastante eficiente para redução não somente do grafo como também do tempo de execução da ferramenta e a função de pontuação melhorou a precisão do diagnóstico em todos os cenários avaliados.

Referências

Bahl, P., Chandra, R., Greenberg, A. G., Kandula, S., Maltz, D. A., and Zhang, M. (2007). Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Proceedings of The 2007 ACM SIGCOMM Conference on Applications,*

Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'07), pages 13–24, New York, NY, USA.

Barham, P., Black, R., Goldszmidt, M., Isaacs, R., MacCormick, J., Mortier, R., and Simma, A. (2008). Constellation: Automated Discovery of Service and Host Dependencies in Networked Systems. *Technical Report, MSR-TR-2008-67, Microsoft Research*.

Barham, P., Donnelly, A., Isaacs, R., and Mortier, R. (2004). Using Magpie for Request Extraction and Workload Modelling. In *Proceedings of The 6th Symposium on Operating Systems Design and Implementation (OSDI'04)*, pages 259–272, Berkeley, CA, USA.

Chen, M. Y., Accardi, A., Kiciman, E., Patterson, D. A., Fox, A., and Brewer, E. A. (2004). Path-Based Failure and Evolution Management. In *Proceedings of The 1st Symposium on Networked Systems Design and Implementation (NSDI'04)*, pages 309–322, Berkeley, CA, USA.

Fonseca, R., Porter, G., Katz, R. H., Shenker, S., and Stoica, I. (2007). X-Trace: A Pervasive Network Tracing Framework. In *Proceedings of The 4th Symposium on Networked Systems Design and Implementation (NSDI'07)*, Berkeley, CA, USA.

Galstad, E. (1999). Nagios. `http://www.nagios.org`.

Geiger, D., Verma, T., and Pearl, J. (1989). d-Separation: From Theorems to Algorithms. In *UAI*, pages 139–148.

HP (1990). HP Openview. `http://www.openview.hp.com`.

IBM (1996). IBM Tivoli. `http://www.ibm.com/tivoli`.

John, D., Prakash, P., Kompella, R. R., and Chandra, R. (2010). Shedding Light on Enterprise Network Failures Using Spotlight. In *Proceedings of The 29th IEEE Symposium on Reliable Distributed Systems (SRDS'10)*, pages 167–176.

Kandula, S., Katabi, D., and Vasseur, J. P. (2005). Shrink: A Tool For Failure Diagnosis in IP Networks. In *Proceedings of MineNet Workshop at 2006 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'06)*, pages 173–178.

Kompella, R. R., Yates, J., Greenberg, A. G., and Snoeren, A. C. (2005). IP Fault Localization Via Risk Modeling. In *Proceedings of The 2nd Symposium on Networked Systems Design and Implementation (NSDI'05)*.

Lyon, G. (1997). Nmap Security Scanner. `http://nmap.org`.

Natarajan, A., Ning, P., Liu, Y., Jajodia, S., and Hutchinson, S. E. (2012). NSDMiner: Automated discovery of Network Service Dependencies. In *Proceedings of The IEEE (INFOCOM'12)*, pages 2507–2515.

Reynolds, P., K., C. E., Wiener, J. L., Mogul, J. C., Shah, M. A., and Vahdat, A. (2006). Pip: Detecting the Unexpected in Distributed Systems. In *Proceedings of The 3rd Symposium on Networked Systems Design and Implementation (NSDI'06)*, Berkeley, CA, USA.

Tak, B. C., Tang, C., Zhang, C., Govindan, S., Urgaonkar, B., and Chang., R. N. (2009). vPath: Precise Discovery of Request Processing Paths from Black-Box Observations of Thread and Network Activities. In *Proceedings of The 34th USENIX Annual Technical Conference (USENIX'09)*, Berkeley, CA, USA.

Caracterizando o Impacto de Topologias no Mapeamento 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. *Network virtualization is a mechanism that allows the coexistence of multiple virtual networks on top of a single physical substrate. One of the research challenges addressed in the literature is the efficient mapping of virtual resources on physical infrastructures. Although this challenge has received considerable attention, state-of-the-art approaches present high rejection rate, i.e., the ratio between the number of denied virtual network requests and the total amount of requests is considerably high. In this work, we investigate the relationship between the quality of virtual network mappings and the topological structures of the underlying substrates. Exact solutions of an online embedding model are evaluated under different classes of network topologies. The obtained results demonstrate that the use of certain topologies on physical substrates significantly contributes to the reduction of rejection rates and, therefore, to improved resource usage.*

Resumo. *A virtualização de redes é um mecanismo que permite a coexistência de múltiplas redes virtuais sobre um mesmo substrato físico. Um dos desafios de pesquisa abordados na literatura é o mapeamento eficiente de recursos virtuais em infraestruturas físicas. Embora o referido desafio tenha recebido considerável atenção, as abordagens que constituem o estado-da-arte apresentam alta taxa de rejeição, i.e., a proporção de solicitações de redes virtuais negadas em relação ao total de solicitações efetuadas ao substrato é elevada. Neste trabalho, investiga-se a relação entre a qualidade dos mapeamentos de redes virtuais e as estruturas topológicas dos substratos subjacentes. Avalia-se as soluções exatas de um modelo de mapeamento online sob diferentes classes de topologias de rede. Os resultados obtidos evidenciam que o emprego de determinadas topologias em substratos físicos contribui significativamente para a redução das taxas de rejeição e, portanto, para um melhor aproveitamento dos recursos.*

1. Introdução

A virtualização de redes é um mecanismo que permite a coexistência de múltiplas redes virtuais (VNs – *Virtual Networks*) heterogêneas 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 que exploram o problema *online* de mapeamento de redes virtuais [Chowdhury et al. 2009, Houidi et al. 2011, Fajjari et al. 2011, Cheng et al. 2011, Cheng et al. 2012], constata-se que as taxas de rejeição para o conjunto de requisições entrantes são, normalmente, altas, podendo atingir até 80%. Supõe-se que um subconjunto dessas rejeições seja 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. Especula-se, contudo, que grande parte das rejeições ocorra 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.

Apesar dos esforços empreendidos para resolver o problema do mapeamento de redes virtuais, desconhecem-se trabalhos que investiguem a influência de topologias de rede no processo de mapeamento. Além disso, trabalhos existentes na área empregam topologias que nem sempre refletem as redes comerciais [Haddadi et al. 2008]. Acredita-se ser fundamental compreender a relação entre o uso de diferentes topologias e o processo de mapeamento, de forma a identificar como determinadas características topológicas influenciam nesse processo. Por exemplo, topologias organizadas de diferentes formas podem propiciar uma melhor utilização dos recursos físicos, podendo, assim, reduzir a taxa de rejeição de requisições, potencialmente elevando o lucro obtido por um provedor e, ao mesmo tempo, reduzindo custos para os solicitantes de redes virtuais.

Neste trabalho, objetiva-se caracterizar o impacto de diferentes classes de topologias tipicamente empregadas em infraestruturas comerciais na qualidade do mapeamento de redes virtuais. Mais especificamente, formaliza-se um modelo ótimo de mapeamento e emprega-se o mesmo na alocação *online* de redes virtuais sobre substratos com topologias estrela, *ladder* e *hub & spoke*. Os resultados obtidos são avaliados considerando diferentes métricas, como taxa de rejeição e consumo de recursos dos dispositivos da rede física. De forma resumida, destacam-se como principais contribuições deste artigo: *(i)* a formalização de um modelo de alocação *online* considerando restrições de localidade; *(ii)* a sistematização das características de redes tipicamente adotadas por provedores; e *(iii)* a avaliação e discussão do impacto de diferentes topologias no processo de mapeamento de redes virtuais.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta uma discussão dos trabalhos relacionados ao problema de mapeamento de redes virtuais, destacando as topologias consideradas e taxas de rejeição. Na Seção 3 são caracterizadas as topologias de rede consideradas neste trabalho. A Seção 4 formaliza o modelo de mapeamento *online* empregado. Na Seção 5 são apresentados e avaliados os resultados obtidos. Por fim, a Seção 6 conclui o artigo com considerações finais e indicações de trabalhos futuros.

2. Trabalhos Relacionados

Nesta seção apresenta-se os principais trabalhos relacionados ao problema de mapeamento de redes virtuais. Faz-se um breve resumo das soluções propostas, ressaltando o

tipo de topologia física e virtual utilizada, bem como as taxas de rejeição obtidas pelos métodos de mapeamento, quando disponíveis.

Yu et al. [Yu et al. 2008] apresentam um modelo de mapeamento *online* para redes virtuais com suporte a múltiplos caminhos e migração. A fragmentação de enlaces virtuais em múltiplos caminhos é usada para aprimorar a utilização de recursos físicos, ampliando as chances de acomodar um maior número de redes virtuais no substrato. Por sua vez, a migração de elementos de redes virtuais visa reotimizar a utilização dos recursos físicos. O procedimento de mapeamento de roteadores e enlaces é realizado em etapas distintas. Os experimentos foram realizados utilizando topologias geradas de forma aleatória tanto para o substrato quanto para as VNs, com conectividade fixa de 50%. Não são apresentadas taxas de rejeição.

Outro modelo *online*, formulado por Chowdhury et al. [Chowdhury et al. 2009], também realiza o mapeamento de roteadores e enlaces em fases distintas. Entretanto, restrições de localidade são usadas para pré-selecionar roteadores físicos onde roteadores virtuais serão hospedados, o que, segundo os autores, facilita o estágio subsequente de mapeamento de enlaces. Além disso, o modelo também possibilita a fragmentação de enlaces virtuais em múltiplos caminhos físicos. As topologias usadas na avaliação do modelo possuem as mesmas características das utilizadas por Yu et al. [Yu et al. 2008], sendo geradas de forma aleatória com conectividade fixa de 50%. As taxas de rejeição para o conjunto de requisições nos cenários estudados atingem entre 25% e 35%.

Alkmim et al. [Alkmim et al. 2013] propõem um modelo focado na necessidade de transferir imagens binárias de roteadores virtuais (armazenadas em repositórios conectados à rede) para os roteadores físicos que os hospedarão. Além de considerar requisitos relacionados às capacidades de roteadores e enlaces, bem como restrições de localidade, o modelo objetiva minimizar o tempo necessário para transferir tais imagens pela rede. As topologias empregadas nos cenários de avaliação são do tipo orgânicas, criadas utilizando o modelo BA-2 [Albert and Barabási 2000], e nos experimentos realizados, as taxas de rejeição variam entre 40% e 80%.

Cheng et al. [Cheng et al. 2011] realizam o mapeamento de elementos de redes virtuais baseados em um esquema de classificação de nós. Roteadores e enlaces, tanto virtuais quanto físicos, são classificados conforme sua capacidade e a capacidade de seus vizinhos. Elementos físicos e virtuais são ordenados conforme sua classificação, visando combinar elementos com classificação semelhante para produzir seus mapeamentos. Segundo os autores, tal estratégia visa evitar a criação de gargalos na rede física. Nos cenários propostos para a avaliação, as topologias do substrato e das VNs são geradas de forma aleatória, e as taxas de rejeição atingidas variam entre 15% e 25%.

Bays et al. [Bays et al. 2012], por sua vez, apresentam um modelo que considera a necessidade de prover, no processo de alocação, confidencialidade às redes virtuais. Redes virtuais podem requerer diferentes níveis de criptografia, e roteadores virtuais que necessitam realizar operações criptográficas devem ser mapeados a roteadores físicos capazes de dar suporte a tais operações. Requisitantes de redes virtuais podem, ainda, demandar que suas redes não compartilhem recursos físicos com outras redes virtuais específicas. A avaliação é realizada usando topologias orgânicas, geradas com o modelo BA-2, atingindo taxas de rejeição entre 48% e 58% nos cenários em que os requisitos de segurança são considerados.

Além dos trabalhos até então mencionados, cujo foco é o mapeamento de re-

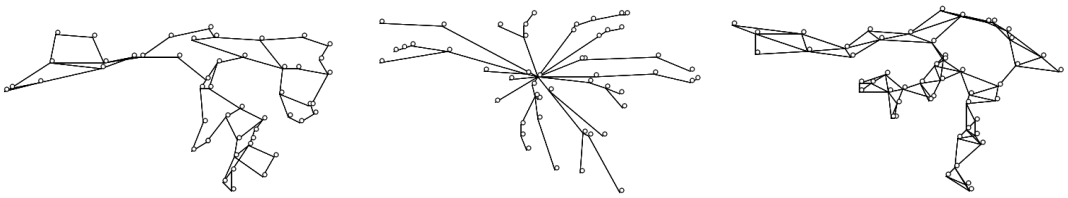
des virtuais, destacam-se também trabalhos que visam prover isolamento de recursos em ambientes de virtualização de redes. Tais abordagens visam garantir que as alocações de recursos definidas na etapa de mapeamento serão cumpridas corretamente. Carvalho et al. [Carvalho et al. 2011] apresentam uma abordagem dinâmica baseada em lógica nebulosa, nos perfis de utilização dos roteadores virtuais e nos contratos de nível de serviço. A ideia chave é punir redes virtuais que excedam a utilização dos recursos estabelecidos em seus contratos. Por sua vez, Fernandes e Duarte [Fernandes and Duarte 2011] propõem uma ferramenta para controle e gerenciamento de recursos que, por meio do monitoramento do tráfego de cada rede virtual, reajusta parâmetros das redes para garantir que os contratos de nível de serviço não serão violados. Como a avaliação de trabalhos dessa linha de pesquisa abstrai a etapa de mapeamento, não há dados relacionados à taxa de rejeição ou topologias empregadas. Os mesmos devem ser entendidos como complementares a este trabalho, tendo em vista que assumimos a oferta, pelo substrato, de mecanismos que forneçam isolamento de recursos e confidencialidade.

Como mencionado anteriormente, desconhecem-se trabalhos anteriores que avaliem de maneira justa o resultado de estratégias de mapeamento considerando diferentes topologias de rede. Os trabalhos desenvolvidos utilizam topologias genéricas ou orgânicas, como as geradas aleatoriamente ou pelo modelo BA-2, as quais não representam fielmente as propriedades topológicas presentes nas infraestruturas de redes de provedores. Portanto, o presente trabalho objetiva compreender como topologias tipicamente observadas em substratos físicos reais influenciam diferentes aspectos do mapeamento de redes virtuais, tais como a rejeição de requisições de redes virtuais e a utilização dos recursos físicos.

3. Topologias de Redes de Provedores

No contexto deste artigo, assumimos, sem perda de generalidade, que provedores de infraestrutura (InPs) apresentam topologias equivalentes àquelas empregadas por provedores de serviços de Internet (ISPs – *Internet Service Providers*). Nesse contexto, as topologias mais tradicionais são: *ladder*, estrela e *hub & spoke (H&S)*. Infraestruturas de rede organizadas em topologia *ladder* têm como principal característica a ausência de *hubs*, isto é, nós com alta conectividade e concentração de fluxo. Além disso, a infraestrutura é formada por um conjunto de *loops*. Esse tipo de topologia tende a possuir um custo baixo no que tange à implantação de enlaces (devido à baixa conectividade) e a apresentar distâncias longas (em termos de número de *hops*) entre nós. Redes *estrela* apresentam poucos *hubs* conectados a um número elevado de nós que, por sua vez, possuem conectividade baixa. Nesse tipo de rede, a distância entre nós tende a ser pequena, porém o tráfego tende a se concentrar nos *hubs*. A classe *H&S* apresenta um número maior de *hubs* em relação às outras topologias mencionadas, e há uma tendência de que esses *hubs* estejam interconectados. Além disso, existe uma grande quantidade de nós que se conecta a um ou mais *hubs*. A Figura 1 ilustra exemplos das três topologias recém mencionadas.

Kamiyama et al. [Kamiyama et al. 2010] realizaram um estudo em que sistematizam a classificação de infraestruturas de redes de provedores nas três classes de topologias previamente descritas. Para o estudo, foram analisadas 23 redes comerciais de *backbone* (disponíveis publicamente) com tamanhos variando entre 21 e 128 nós. A partir dessa análise, os autores definem um conjunto de métricas, as quais capturam as principais propriedades topológicas presentes nessas infraestruturas. Tais métricas incluem, por exemplo, o grau de conectividade da topologia e a presença de nós entendidos como *hubs*. Dessa forma, os autores mapeiam a relação entre tais métricas e o tipo de topologia da



(a) *Ladder* (b) Estrela (c) *Hub & Spoke*

Figura 1. Exemplos de classes de topologias de provedores de rede.

infraestrutura de rede, permitindo a classificação de topologias de redes de provedores em uma das classes descritas.

Tendo em vista a classificação bem aceita de topologias de redes de provedores recém apresentada e a abordagem sistemática proposta por Kamiyama et al. [Kamiyama et al. 2010] para caracterizar e, portanto, gerar essas topologias com elevado grau de fidedignidade (a redes reais), o presente artigo tomará as três classes como base para a investigação. Assim, pretende-se analisar como tais classes topológicas afetam o processo de mapeamento, identificando possíveis relações entre as taxas de rejeição e o consumo de recursos de dispositivos da infraestrutura.

4. Modelo Ótimo de Mapeamento de Redes Virtuais

Para analisar o impacto de diferentes tipos de topologias no processo de mapeamento de redes virtuais, formalizamos um modelo baseado em Programação Linear Inteira. 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).

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.

Cada roteador físico está associado a um identificador de localidade, armazenado no conjunto S^P. Tal permite que requisitantes de redes virtuais indiquem localidades específicas em que certos roteadores virtuais devem ser instanciados (por exemplo, para garantir conectividade entre duas ou mais localidades). Caso um roteador virtual possua requisito de localidade, o mesmo é armazenado no conjunto S^V.

Considera-se que roteadores físicos possuem capacidades limitadas de CPU e memória. Tais capacidades são representadas, respectivamente, por C_i^P e M_i^P. De forma análoga, os requisitos de CPU e de memória de cada roteador virtual de uma rede r são representados por $C_{r,i}^V$ e $M_{r,i}^V$. Enlaces físicos, por sua vez, possuem capacidade limitada de largura de banda, representada por $B_{i,j}^P$, enquanto que a largura de banda requisitada por cada enlace virtual é representada por $B_{r,i,j}^V$.

O modelo recebe requisições e aloca redes virtuais de forma *online*. Por conta disso, é necessário considerar os elementos virtuais já alocados no substrato. Roteadores virtuais previamente alocados são representados pelo conjunto $E_{i,r,j}^R$, e enlaces, pelo conjunto $E_{i,j,r,k,l}^L$.

As variáveis do modelo representam a solução ótima do problema de mapeamento para as entradas recebidas. Tais variáveis indicam onde estão alocados, no substrato físico, os roteadores e enlaces virtuais requisitados. Caso uma requisição seja aceita, cada um de seus roteadores é alocado a um roteador físico, enquanto que cada enlace virtual pode ser mapeado a um único enlace físico ou a um caminho composto por múltiplos enlaces.

- $A^R_{i,r,j} \in \{0,1\}$ – Alocação de roteadores, indica se o roteador físico i está hospedando o roteador virtual j da rede virtual r.
- $A^L_{i,j,r,k,l} \in \{0,1\}$ – Alocação de enlaces, indica se o enlace físico (i,j) está hospedando o enlace virtual (k,l) da rede virtual r.

Por fim, é apresentada a função objetivo e suas restrições. A função objetivo do modelo visa minimizar a largura de banda consumida pelas redes virtuais alocadas no substrato físico. O propósito de cada restrição será detalhado a seguir.

Objetivo:

$$min \sum_{(i,j)\in L^P} \sum_{r\in N^V,(k,l)\in L^V} A^L_{i,j,r,k,l} B^V_{r,k,l}$$

Sujeito a:

$$\sum_{r\in N^V, j\in R^V} C^V_{r,j} A^R_{i,r,j} \leq C^P_i \qquad \forall i \in R^P \quad \text{(R1)}$$

$$\sum_{r\in N^V, j\in R^V} M^V_{r,j} A^R_{i,r,j} \leq M^P_i \qquad \forall i \in R^P \quad \text{(R2)}$$

$$\sum_{r\in N^V, (k,l)\in L^V} B^V_{r,k,l} A^L_{i,j,r,k,l} \leq B^P_{i,j} \qquad \forall (i,j) \in L^P \quad \text{(R3)}$$

$$\sum_{i\in R^P} A^R_{i,r,j} = 1 \qquad \forall r \in N^V, j \in R^V \quad \text{(R4)}$$

$$\sum_{j\in R^V} A^R_{i,r,j} \leq 1 \qquad \forall i \in R^P, r \in N^V \quad \text{(R5)}$$

$$\sum_{j\in R^P} A^L_{i,j,r,k,l} - \sum_{j\in R^P} A^L_{j,i,r,k,l} = A^R_{i,r,k} - A^R_{i,r,l} \qquad \forall r \in N^V, (k,l) \in L^V, i \in R^P \quad \text{(R6)}$$

$$jA^R_{i,r,k} = lA^R_{i,r,k} \qquad \forall (i,j) \in S^P, r \in N^V, (k,l) \in S^V \quad \text{(R7)}$$

$$A^R_{i,r,j} = E^R_{i,r,j} \qquad \forall (i,r,j) \in E^R \quad \text{(R8)}$$

$$A^L_{i,j,r,k,l} = E^L_{i,j,r,k,l} \qquad \forall (i,j,r,k,l) \in E^L \quad \text{(R9)}$$

A restrição R1 garante que a capacidade de CPU de cada roteador físico não será excedida, garantindo, portanto, que os requisitos CPU de cada roteador virtual serão aten-

didos. A restrição R2 aplica o mesmo controle à capacidade de memória dos roteadores, e a restrição R3, à largura de banda dos enlaces. A restrição R4 garante que todo roteador virtual será alocado a um roteador físico. Por sua vez, a restrição R5 impede que múltiplos roteadores virtuais de uma mesma rede virtual sejam alocados a um mesmo roteador físico. A restrição R6 garante que todo enlace virtual será alocado a um caminho físico válido. Dessa forma, o caminho físico hospedando um enlace virtual (a, b) será garantidamente um caminho válido entre o roteador físico hospedando o roteador virtual a e o roteador físico hospedando o roteador virtual b. A restrição R7 garante que todo roteador virtual que possua um requisito de localidade será mapeado a um roteador físico na localidade solicitada. Por fim, as restrições R8 e R9 garantem que os elementos das redes virtuais previamente alocadas continuarão hospedados nos mesmos elementos físicos. A alocação dos roteadores é mantida pela restrição R8, enquanto que a alocação dos enlaces, pela restrição R9.

5. Avaliação do Impacto de Topologias no Mapeamento

Para avaliar o impacto de diferentes topologias de rede no processo de mapeamento de redes virtuais, o modelo formalizado na seção anterior foi implementado e executado no *CPLEX Optimization Studio*[1] versão 12.3. 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.

5.1. Carga de Trabalho

Para realizar os experimentos, desenvolveu-se um gerador de requisições de redes virtuais. Tal gerador é executado por um período de 500 janelas de tempo. Em cada janela, são geradas, em média, cinco requisições, seguindo uma distribuição de Poisson. Cada requisição gerada possui uma duração limitada, ou seja, após um determinado número de janelas de tempo, a mesma é removida. Tal duração é de, em média, cinco janelas de tempo, seguindo uma distribuição exponencial. Ressalta-se que essa forma de instanciação, baseada no emprego de janelas de tempo e com uso dos modelos de chegada de requisições e de duração de redes virtuais previamente citados, vem sendo empregada em trabalhos da área, como nos realizados por Yu et al. [Yu et al. 2008] e Houidi et al. [Houidi et al. 2011].

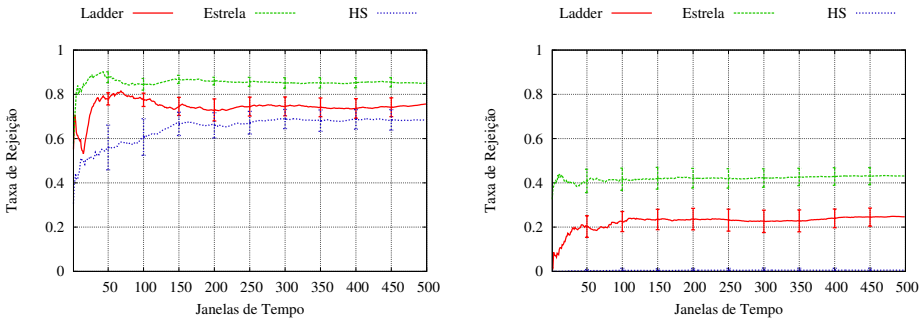
As topologias utilizadas como substrato físico são geradas por meio da ferramenta IGen[2]. Para gerar redes com as características topológicas das classes previamente apresentadas – *estrela*, *ladder* e *hub & spoke* – são usados, respectivamente, os métodos *Mentor*, *MultiTour* e *TwoTree*. Em linha com a caracterização de topologias apresentada anteriormente na Seção 3, na rede *ladder* os nós apresentam grau médio 3 e grau máximo normalizado igual a 4. A rede *estrela* possui uma proporção de nós altamente interconectados (*hubs*) menor que 0,25, enquanto que a rede *hub & spoke* apresenta uma proporção maior ou igual a 0,25. Tal proporção é definida como o número de nós com grau maior que o grau médio, dividido pelo total de nós da rede. Além dessas propriedades topológicas, as redes físicas possuem 50 roteadores, cada um com capacidade total de CPU definida como 100%, e 256 MB de memória. Os roteadores são distribuídos uniformemente entre 16 localidades e a largura de banda dos enlaces físicos é distribuída uniformemente entre 1 e 10 Gbps.

[1] http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/
[2] http://igen.sourceforge.net/

A topologia de cada rede virtual é gerada por meio da ferramenta BRITE[3], usando o modelo Barabási-Albert (BA-2) [Albert and Barabási 2000]. Acredita-se que tal modelo é capaz de representar adequadamente as características de requisições de redes virtuais. Além disso, ressalta-se que o uso das classes de topologias típicas de provedores, 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 entre 2 e 5 roteadores cada. Roteadores virtuais requerem entre 10% e 50% de CPU e entre 24 MB e 128 MB de memória, ambos parâmetros seguindo uma distribuição uniforme. A largura de banda dos enlaces virtuais é distribuída uniformemente entre 1 e 5 Gbps. Foram realizados dois cenários de avaliação sobre cada topologia física, os quais se diferenciam pela presença ou ausência de requisitos de localidade. No primeiro cenário, cada rede virtual possui dois roteadores (os *end points* dessa rede) com requisitos de localidade, gerados aleatoriamente entre as 16 localidades existentes. No outro, nenhuma rede virtual possui tais requisitos. Foram executadas 30 repetições de cada experimento, considerando como base diferentes instâncias para cada tipo de substrato.

5.2. Resultados

Primeiramente, analisa-se a taxa de rejeição de requisições de redes virtuais nos cenários previamente descritos. Ressalta-se que requisições de redes virtuais somente são rejeitadas caso não seja possível mapear todos os seus roteadores e enlaces virtuais no substrato físico. A Figura 2 ilustra a taxa média de rejeição obtida em cada cenário. Cada ponto no gráfico representa a taxa média de rejeição alcançada desde o início do experimento até a janela de tempo em questão. Observa-se claramente que, quando há restrições de localidade, as taxas de rejeição são substancialmente maiores (variando entre 65,38% e 83,71%), nas três topologias físicas, em relação aos cenários em que não há tais requisitos (nos quais variam entre 0,53% e 43,10%). O comportamento apresentado é influenciado pela redução no espaço de soluções factíveis causada pela presença de restrições de localidade.



(a) Cenários com requisitos de localidade. (b) Cenários sem requisitos de localidade.

Figura 2. Porcentagem média de requisições rejeitadas nos experimentos realizados.

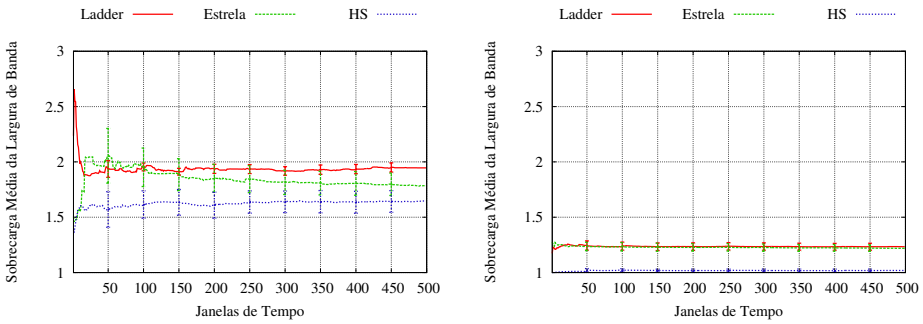
Ainda observando o gráfico ilustrado na Figura 2, percebe-se uma diferença nas taxas de rejeição ao se utilizar diferentes topologias físicas. A topologia do tipo *hub & spoke* apresenta uma taxa de rejeição inferior às demais em ambos cenários avaliados (68,44% no cenário com restrições de localidade e 0,53% no cenário sem tais restrições). Em contrapartida, a topologia *estrela* apresenta o pior desempenho (85,04% de rejeição

[3]http://www.cs.bu.edu/brite/

no cenário com localidade e 43,10% no cenário sem localidade). A topologia *ladder* apresenta taxas de rejeição de 75,63% e 24,66% para os cenários com localidade e sem localidade, respectivamente. Topologias do tipo *hub & spoke* tendem a causar a rejeição de um número menor de requisições, por possuírem, em média, um número maior de nós com alto grau de conectividade (*hubs*). A presença de múltiplos *hubs* leva a uma menor probabilidade de que o esgotamento dos recursos de um desses nós centrais cause um impacto significativo nos mapeamentos de requisições futuras. Ao contrário, topologias *estrela* apresentam poucos nós centrais, e por isso, há uma grande probabilidade de que tais nós se tornem um gargalo no processo de alocação de redes virtuais à medida em que ocorre o esgotamento de seus recursos. Já em redes da classe *ladder*, como não há nós centrais, o esgotamento de recursos de alguns enlaces físicos pode impossibilitar a criação de enlaces virtuais que utilizariam tais caminhos como "pontes" para interconectar determinados pontos da infraestrutura.

A Figura 3 ilustra a sobrecarga média dos mapeamentos de redes virtuais em cada experimento. Tal sobrecarga é medida como a razão entre a largura de banda efetivamente consumida por uma rede virtual hospedada no substrato físico e a largura de banda requisitada por tal rede. Em geral, o consumo real de largura de banda é maior do que a largura de banda total requisitada pelas redes virtuais, devido à frequente necessidade de alocar enlaces virtuais a caminhos compostos por múltiplos enlaces físicos. Somente não há sobrecarga quando cada enlace virtual é mapeado a apenas um enlace físico (razão 1,0). Ressalta-se que taxas menores de sobrecarga favorecem diretamente o provedor da infraestrutura, por economizar recursos que podem ser usados para acomodar requisições futuras. Além disso, tal economia pode levar a custos mais baixos para os solicitantes. Topologias do tipo *ladder* causam uma sobrecarga média maior em relação às demais (94,59% para cenários com localidade e 23,36% para cenários sem localidade), já que a estrutura topológica apresenta, em média, uma distância longa (em termos de número de *hops*) entre nós, bem como a ausência de (*hubs*) centrais. A topologia da classe *hub & spoke* apresenta a menor taxa de sobrecarga (64,67% para o cenário com localidade), por ser uma topologia com maior quantidade de interconexões e *hubs*. No cenário sem localidade (Figura 3(b)), nota-se que a sobrecarga média obtida nas infraestruturas *ladder* e *estrela* é similar – 23,36% e 21,92%, respectivamente – enquanto que a taxa obtida considerando-se a topologia *hub & spoke* é de apenas 1,89%. Isso evidencia que, quando consideradas restrições de localidade, enlaces virtuais tendem a ser mapeados a caminhos mais longos no substrato, devido a uma maior distância média entre os locais onde roteadores virtuais são hospedados. No entanto, topologias que contam com uma maior quantidade de *hubs* tendem a reduzir o impacto dessas restrições.

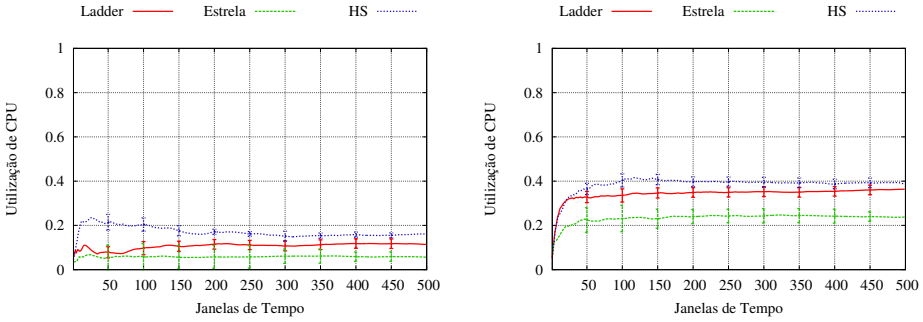
A seguir, avalia-se o consumo médio dos recursos físicos – CPU e memória dos roteadores e largura de banda dos enlaces – nos experimentos realizados. A Figura 4 apresenta o consumo médio de CPU dos roteadores físicos da infraestrutura. Considerando requisitos de localidade, nota-se que o consumo médio de CPU ao se empregar a topologia do tipo *ladder* (11,49%) é aproximadamente duas vezes maior que o consumo médio no experimento que utiliza a topologia do tipo *estrela* (5,36%). A topologia *hub & spoke* levou o consumo médio de CPU nos roteadores físicos a 16,82%. Nos experimentos em que não há requisitos de localidade (Figura 4(b)), nota-se um consumo de CPU maior, sendo consumido, em média, 23,80% dos recursos na topologia *estrela*, 35,22% na topologia *ladder* e 38,60% na topologia *hub & spoke*. O maior consumo dos recursos de CPU é diretamente influenciado pela quantidade de redes virtuais alocadas no substrato. Isso explica o consumo mais elevado nos cenários que não consideram localidade, visto



(a) Cenários com requisitos de localidade. (b) Cenários sem requisitos de localidade.

Figura 3. Sobrecarga média de largura de banda necessária para acomodar as requisições aceitas.

que nesses casos a taxa de rejeição é mais baixa. No entanto, o consumo relativamente baixo de CPU, cuja média geral é inferior a 40% em todos os experimentos, mostra que a rejeição de redes virtuais não é causada pela exaustão dos recursos de CPU dos roteadores da topologia de maneira global.

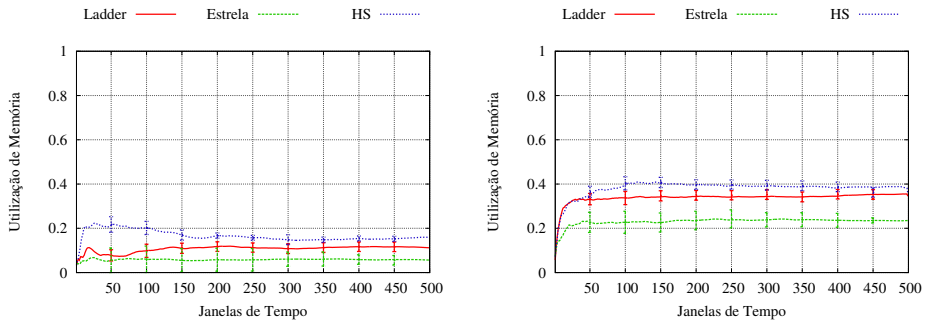


(a) Cenários com requisitos de localidade. (b) Cenários sem requisitos de localidade.

Figura 4. Utilização média de CPU dos roteadores físicos.

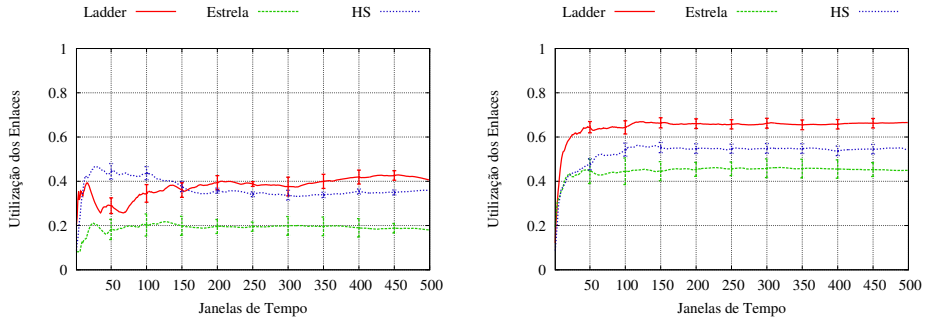
A Figura 5 apresenta o consumo médio de memória nos roteadores físicos. Percebe-se que o comportamento da utilização de memória dos roteadores é similar ao da utilização de CPU. Nos experimentos que consideram requisitos de localidade, a utilização média de memória é de 5,65% na topologia *estrela*, 11,21% na topologia *ladder* e 15,93% na topologia *hub & spoke*. Já os experimentos em que não há tais requisitos levam a uma utilização média de 23,49% na topologia *estrela*, 35,52% na topologia *ladder* e 38,03% na topologia *hub & spoke*. Nota-se que em nenhum dos experimentos ocorre o esgotamento desse recurso, podendo-se afirmar que tal fator, bem como o uso de CPU, não é a causa da rejeição de redes virtuais nesses cenários.

O consumo médio da largura de banda dos enlaces das infraestruturas é apresentado na Figura 6. Nos experimentos em que as redes virtuais possuem requisitos de localidade, há uma utilização de 18,01% dos recursos da topologia *estrela*, 40,63% da topologia *ladder* e 35,89% da topologia *hub & spoke*. Já nos experimentos sem restrições de localidade o consumo médio é de 44,93% na topologia *estrela*, 66,50% na topologia *ladder* e 54,08% na topologia *hub & spoke*. Tais resultados demonstram que, assim como o consumo de CPU e memória dos roteadores físicos, o consumo geral da largura de banda



(a) Cenários com requisitos de localidade. (b) Cenários sem requisitos de localidade.

Figura 5. Utilização média de memória dos roteadores físicos.



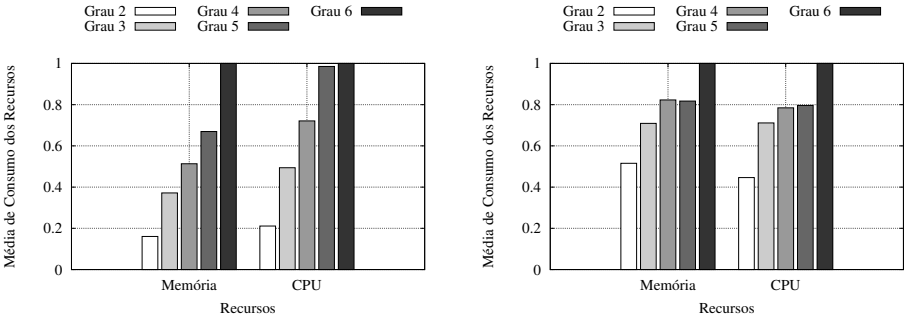
(a) Cenários com requisitos de localidade. (b) Cenários sem requisitos de localidade.

Figura 6. Utilização média de largura de banda dos enlaces físicos.

disponível na topologia não é a causa das taxas de rejeição observadas nos experimentos. A saturação de alguns pontos específicos das topologias faz com que não seja possível alocar um número maior de redes virtuais, ainda que uma visão global da rede revele que ainda há recursos disponíveis.

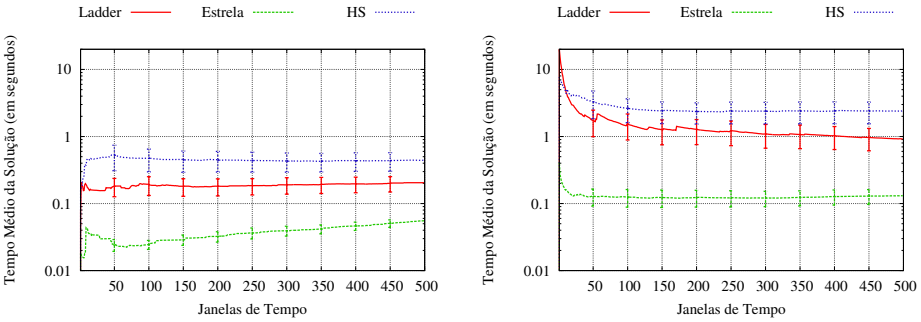
Para melhor compreender as taxas de rejeição observadas nos experimentos realizados, a Figura 7 ilustra a utilização média dos recursos de roteadores com diferentes graus de conectividade nos cenários que empregam a topologia *hub & spoke*. A média é calculada sobre as 100 últimas janelas de tempo (após a estabilização das taxas de rejeição), e normalizada em relação ao roteador com maior taxa de utilização. Nota-se uma relação clara entre o grau de conectividade e a utilização de CPU e memória dos roteadores físicos, evidenciando o fato de que os recursos dos roteadores com maiores graus de conectividade (*hubs*) são consumidos em uma proporção mais alta. Essa relação é ainda mais acentuada quando as redes virtuais possuem requisitos de localidade. Os resultados evidenciam a importância da presença de *hubs* na topologia física, e mostram como a exaustão de recursos em pontos específicos da rede de um provedor pode exacerbar a rejeição de requisições. Ressalta-se que fenômenos semelhantes de consumo são observados nas topologias estrela e *ladder* (em maior e menor escala, respectivamente), porém os resultados são omitidos por limitação de espaço.

Por fim, na Figura 8 apresenta-se o tempo médio necessário para encontrar a alocação ótima de cada requisição aceita. Ressalta-se que, nesse caso, o eixo vertical é apresentado em escala logarítmica, visto que há uma diferença significativa nos resul-



(a) Cenário com requisitos de localidade. (b) Cenário sem requisitos de localidade.

Figura 7. Consumo médio dos recursos de roteadores físicos com diferentes graus de conectividade nos cenários que empregam a topologia *hub & spoke*.



(a) Cenários com requisitos de localidade. (b) Cenários sem requisitos de localidade.

Figura 8. Tempo médio necessário para encontrar a alocação ótima.

tados de diferentes experimentos. Em todos os cenários com requisitos de localidade, o tempo necessário para a alocação nas topologias avaliadas permanece abaixo de 1 segundo. O emprego da topologia *estrela* leva a, em média, um tempo de 0,055 segundos, a topologia *ladder*, a um tempo médio de 0,20 segundos e a topologia *hub & spoke*, um tempo de 0,44 segundos. Nos demais cenários, em que não há restrições de localidade, as médias gerais permanecem abaixo de 3 segundos. A topologia *estrela* leva a, em média, um tempo de 0,13 segundos, a topologia *ladder* a um tempo médio de 0,92 segundos e a topologia *hub & spoke* a um tempo de 2,41 segundos.

As médias de tempo relativamente altas observadas nos cenários que consideram a topologia do tipo *hub & spoke* devem-se à maior quantidade de enlaces presentes nessa topologia. Tal característica leva a um conjunto maior de possíveis mapeamentos para as redes virtuais requisitadas, o que tende a aumentar o tempo necessário para encontrar a alocação ótima. De forma similar, a desconsideração de requisitos de localidade também leva a um maior espaço de soluções factíveis, o que explica as médias de tempo mais elevadas observadas nos cenários que possuem tal característica. Além disso, notam-se alguns picos no início dos experimentos, chegando a um máximo de 24,81 segundos no cenário em que é usada a topologia *ladder*, e 9,99 segundos no cenário em que emprega-se a topologia *hub & spoke*. Tal comportamento deve-se à maior quantidade de recursos disponíveis no início dos experimentos (quando as redes físicas encontram-se com uma grande quantidade de recursos disponíveis), o que também amplia as possibilidades de arranjo de redes virtuais nos substratos.

Os resultados apresentados mostram que há um impacto significativo na taxa de rejeição e no consumo dos recursos físicos das infraestruturas ao se utilizar diferentes classes de topologias de rede para a alocação de redes virtuais. A estrutura topológica exerce um impacto ainda maior ao se considerarem requisitos de localidade nas requisições de redes virtuais. Tais casos evidenciam que as rejeições de redes virtuais requisitadas não são causadas pelo esgotamento de recursos da infraestrutura, mas sim por fatores relacionados a determinadas características topológicas.

O principal fator que influencia a rejeição de redes virtuais é o esgotamento de recursos em regiões pontuais do substrato. Por exemplo, em topologias do tipo *estrela* e *hub & spoke*, em geral, o esgotamento dos recursos físicos dos *hubs*, além dos enlaces conectados aos mesmos, consiste na principal causa da impossibilidade de se mapear novas redes virtuais. Em topologias da classe *ladder*, em que não há *hubs*, a principal causa do aumento de requisições rejeitadas é o esgotamento dos recursos de enlaces específicos da topologia. Enlaces entendidos como "pontes" na infraestrutura podem se tornar gargalos e, se os recursos de um de tais enlaces se esgotam, a infraestrutura é particionada em dois grupos de roteadores sem conectividade entre si.

Em resumo, mapeamentos de redes virtuais em topologias *estrela* apresentam baixo tempo de resolução, porém alta taxa de rejeição e, por consequência, baixa utilização dos recursos físicos. Em topologias *ladder*, a taxa de rejeição e o tempo de resolução apresentam valores intermediários em relação às demais classes de topologias consideradas, porém o mapeamento de redes virtuais leva a um consumo maior de largura de banda. Por fim, topologias do tipo *hub & spoke* apresentam baixas taxas de rejeição e de sobrecarga média dos enlaces, porém levam a tempos de resolução mais altos em relação às demais topologias.

6. Conclusões

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. No entanto, desconhecem-se trabalhos anteriores que avaliem de maneira justa o resultado de estratégias de mapeamento considerando diferentes topologias de rede. Os trabalhos até então desenvolvidos, de maneira geral, utilizam topologias genéricas ou orgânicas, as quais não representam fielmente as propriedades topológicas presentes nas infraestruturas de redes de provedores.

Após formalizar um modelo ótimo de mapeamento *online* de redes virtuais e aplicá-lo sobre substratos com diferentes características topológicas tipicamente observadas em redes de provedores, caracterizou-se o impacto de diferentes classes de topologias no que diz respeito a taxas de rejeição e à utilização dos recursos da infraestrutura. Os resultados obtidos demonstram o impacto significativo causado pelo mapeamento de redes virtuais nas diferentes estruturas topológicas avaliadas. A alocação de redes virtuais é prejudicada pelo esgotamento de recursos em alguns pontos específicos da infraestrutura física, ainda que uma visão global da rede revele que ainda há recursos disponíveis no restante do substrato. Tal impacto é ainda maior quando consideradas restrições de localidade em requisições de redes virtuais.

Tendo avaliado o impacto do mapeamento de redes virtuais em substratos de redes com diferentes estruturas topológicas, pretende-se investigar estratégias para reduzir taxas de rejeição observadas em redes físicas de provedores. Mais especificamente, pretende-se, como trabalho futuro, investigar como um substrato físico pode ser pontualmente ex-

pandido ou redimensionado de forma a reduzir a rejeição de redes virtuais, considerando-se os custos necessários para realizar tais modificações na infraestrutura. A ideia chave é identificar os gargalos da rede com base em requisições negadas no passado e, por meio de ajustes pontuais na infraestrutura, ampliar consistentemente a aceitação de requisições de redes virtuais futuras.

Referências

Albert, R. and Barabási, A.-L. (2000). Topology of evolving networks: Local events and universality. *Physical Review Letters*, 85:5234 – 5237.

Alkmim, G., Batista, D., and da Fonseca, N. (2013). Mapping virtual networks onto substrate networks. *Journal of Internet Services and Applications*, 4(1):3.

Bays, L. R., Oliveira, R. R., Buriol, L. S., Barcellos, M. P., and Gaspary, L. P. (2012). Um modelo para mapeamento Ótimo de redes virtuais com requisitos de segurança. In *XII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 249 – 262.

Carvalho, H. E. T., Fernandes, N. C., and Duarte, O. C. M. B. (2011). Um controlador robusto de acordos de nível de serviço para redes virtuais baseado em lógica nebulosa. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 645 – 658.

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.

Cheng, X., Su, S., Zhang, Z., Wang, H., Yang, F., Luo, Y., and Wang, J. (2011). Virtual network embedding through topology-aware node ranking. *SIGCOMM Computer Communication Review*, 41(2):38 – 47.

Chowdhury, N., Rahman, M., and Boutaba, R. (2009). Virtual network embedding with coordinated node and link mapping. In *INFOCOM 2009, IEEE*, pages 783 – 791.

Fajjari, I., Aitsaadi, N., Pujolle, G., and Zimmermann, H. (2011). Vne-ac: Virtual network embedding algorithm based on ant colony metaheuristic. In *Communications, 2011 IEEE International Conference on*, pages 1 – 6.

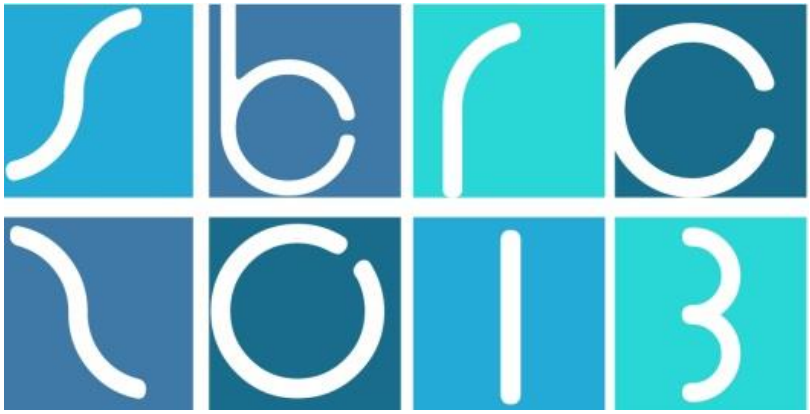
Fernandes, N. C. and Duarte, O. C. M. B. (2011). Provendo isolamento e qualidade de serviço em redes virtuais. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 295 – 308.

Haddadi, H., Rio, M., Iannaccone, G., Moore, A., and Mortier, R. (2008). Network topologies: inference, modeling, and generation. *Communications Surveys Tutorials, IEEE*, 10(2):48 – 69.

Houidi, I., Louati, W., Ameur, W. B., and Zeghlache, D. (2011). Virtual network provisioning across multiple substrate networks. *Computer Networks*, 55(4):1011 – 1023.

Kamiyama, N., Kawahara, R., Mori, T., Harada, S., and Hasegawa, H. (2010). Impact of topology on parallel video streaming. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 607 – 614.

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.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 3

Confiabilidade e Resiliência

Tolerância a Faltas Bizantinas usando Registradores Compartilhados Distribuídos

Marcelo Ribeiro Xavier Silva[1], Lau Cheuk Lung[1], Aldelir Fernando Luiz[2,3], Leandro Quibem Magnabosco[1]

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

[2]Câmpus Avançado de Blumenau - Instituto Federal Catarinense - Brasil

[3]Departamento de Automação e Sistemas - Universidade Federal de Santa Catarina - Brasil

marcelo.r.x.s@posgrad.ufsc.br, lau.lung@inf.ufsc.br, aldelir@das.ufsc.br

leandro.magnabosco@posgrad.ufsc.br

Abstract. *State Machine Replication (SMR) is a technique commonly used in the implementation of distributed services that tolerates Byzantine Faults. Originally the approaches based on this technique did require $3f+1$ servers to tolerate f faults. Recently, through the use of a tamper-proof component, some approaches reduced this number to $2f+1$. In general, the construction of this components enforce some complex hardware and software modification in the server machines. We present Registered Paxos (RegPaxos), an approach that explores virtualization and shared memory emulation to simplify the creation of a tamper-proof component. With this architecture we were also able to reduce the latency, in means of the number of communication steps, that is only comparable to speculative algorithms.*

Resumo. *Replicação de Máquina de Estados é uma técnica comumente utilizada na implementação de serviços distribuídos que toleram faltas bizantinas. Originalmente as abordagens baseadas nesta técnica necessitavam $3f+1$ servidores para tolerar f faltas. Recentemente, através do uso de componentes confiáveis, algumas abordagens conseguiram reduzir este número para $2f+1$. Para construir estes componentes confiáveis é necessário fazer algumas modificações complexas nos servidores, tanto do ponto de vista de software quanto de hardware. Nós apresentamos o Paxos Registrado (RegPaxos), uma abordagem que explora tecnologias de virtualização e compartilhamento distribuído de dados para simplificar a criação da componente inviolável de tolerância a faltas. Com esta arquitetura fomos capazes também de alcançar uma latência (em números de passos para comunicação) comparável apenas com algoritmos especulativos.*

1. Introdução

A aplicação de conceitos de dependabilidade para construir sistemas distribuídos seguros tem crescido consideravelmente sob a designação de tolerância a intrusão ou tolerância a faltas bizantinas [Veríssimo et al. 2003, Correia et al. 2004]. Esse tema de pesquisa tem sido muito explorada nas últimas décadas e, dentre as abordagens utilizadas, destacam-se as que fazem uso de técnicas de redundância. Estas técnicas são implementadas através da replicação de máquina de estados (do inglês, *State Machine Replication* - SMR) [Lamport 1978, Schneider 1982] que combina uma série de mecanismos que contribuem

para a manutenção da disponibilidade e integridade dos serviços e aplicações, bem como dos ambientes de execução.

A abordagem SMR tem sido utilizada para tolerar faltas Bizantinas (do inglês, *Byzantine Fault Tolerant* - BFT) [Reiter 1995, Castro and Liskov 2002], afim de manter o funcionamento correto do sistema a despeito da ocorrência de faltas. Os algoritmos tolerantes a faltas bizantinas [Veríssimo et al. 2003] têm como objetivo permitir que os sistemas continuem operando dentro de suas especificações de funcionamento, mesmo que alguns de seus componentes apresentem comportamentos arbitrários [Castro and Liskov 2002, Reiter 1995, Yin et al. 2003]. Alguns trabalhos comprovam que através do uso destes algoritmos é possível projetar serviços confiáveis como sistemas de arquivos em rede, *backup* cooperativo, serviços de coordenação de autoridades certificadoras, banco de dados, sistemas de gerenciamento de chaves, etc [Veronese et al. 2011, Bessani et al. 2007].

Os algoritmos BFT, em sua maioria, necessitam de um mínimo de $3f + 1$ réplicas [Castro and Liskov 2002, Reiter 1995, Kotla et al. 2008] para tolerar f faltosas. Na realidade, quando usada para tolerar faltas de *crash*, a redundância de máquinas de estado necessita apenas $2f + 1$ réplicas [Schneider 1990]. O número adicional de réplicas é necessário para tolerar comportamentos maliciosos e/ou arbitrários [Correia et al. 2012]. O custo é ainda mais alto se considerarmos que a heterogeneidade é uma premissa importante neste tipo de sistema [Obelheiro et al. 2005]. Trabalhos recentes conseguem melhorar a resiliência dos sistemas BFT tolerando faltas com apenas $2f + 1$ réplicas. Estes trabalhos baseiam-se em componentes invioláveis criadas através de modificações em *hardware* [Veronese et al. 2011, Chun et al. 2007], *software* da máquina servidora [Correia et al. 2004, Veronese et al. 2011] e/ou rede de comunicação [Correia et al. 2004]. Existem também trabalhos que propõe o uso de tecnologias de virtualização para implementar seus sistemas tolerantes [Reiser and Kapitza 2007, Stumm et al. 2010]. Esta abordagem com certeza diminui o custo de replicação, mas transforma a máquina física num ponto único de falha.

Além da resiliência, outro fator importante para avaliação do desempenho de sistemas BFT é o atraso no processamento de uma requisição, ou latência [Correia et al. 2012]. A quantidade de passos de comunicação necessários durante uma execução na ausência faltas é considerada como métrica de latência nestes sistemas [Veronese et al. 2011]. Os algoritmos especulativos, em que os clientes participam ativamente do progresso do sistema, são os que possuem a menor quantidade de passos [Kotla et al. 2008, Veronese et al. 2011].

Neste artigo, apresentamos o RegPaxos que também necessita de apenas $2f + 1$ réplicas para tolerar f faltosas. A arquitetura de sistema proposta neste trabalho é baseado em um modelo híbrido em que variam, de componente para componente, as suposições de sincronismo e presença/severidade de faltas e falhas [Correia et al. 2002, Veríssimo 2006, Correia et al. 2004]. Neste modelo, a rede é separada em duas. A primeira, chamada de rede de *payload*, é assíncrona e utilizada para que os clientes se comuniquem com os servidores. A segunda é uma rede inviolável e é utilizada pelos servidores para ordenar as requisições dos clientes. Nesta revisita ao modelo híbrido proposto em [Correia et al. 2002, Veríssimo 2006], contribuímos com a redução da complexidade na construção da rede inviolável através do uso de dois artifícios, virtualização, para isolar o serviço confiável do serviço disponível aos clientes, e a utilização de um componente, aqui denominado Registradores Compartilhados Distribuídos (DSR - acrônimo para *Distributed Shared Register*), para ordenar requisições dos clientes. Além disso, o RegPaxos é o pri-

meiro algoritmo não especulativo capaz de equiparar-se com algoritmos especulativos em relação à latência.

2. Trabalhos Relacionados

O uso de máquinas de estados foi introduzido por Lamport em sistemas onde não se considerava a ocorrência de faltas [Lamport 1978]. Schneider generalizou esta abordagem considerando sistemas sujeitos a faltas de *crash*, o que serviu como base para o Rampart [Reiter 1995], e para outros sistemas tolerantes a faltas bizantinas.

A considerar o uso de virtualização como suporte para sistemas BFT, o VM-FIT [Reiser and Kapitza 2007] foi uma das primeiras propostas a aplicar virtualização para tolerar faltas bizantinas. Os autores propõe o uso de várias máquinas virtuais sobre uma máquina física para atender requisições de clientes. A ordem adotada para execução das operações de clientes vem de um *virtual machine manager* (VMM) adaptado para atuar como ordenador. Esta abordagem é importante por ser uma das pioneiras e por reduzir o custo de replicação. Entretanto, esta redução transforma a máquina física em um ponto único de falhas, além disso, a necessidade de se alterar o VMM dificulta seu desenvolvimento.

O facilitador para sistemas distribuídos, *Attested Append-only Memory* (A2M) apresentado em [Chun et al. 2007] provê uma abstração de um sistema de *log* confiável, através da qual é possível criar protocolos imunes a faltas. Através do uso do A2M é possível criar variações do sistema BFT apresentado por Castro e Liskov [Castro and Liskov 1998]. Com o A2M o sistema mantém suas propriedades de segurança e progresso, mesmo que metade das réplicas estejam faltosas. É uma abordagem simples em termos de entendimento e atinge a melhor resiliência prática. Entretanto, sua implementação requer o gerenciamento de cinco arquivos de *log*, aumentando a necessidade de armazenamento e o torna mais complexo do que seus autores assumiram.

Os algoritmos assíncronos e tolerantes a faltas bizantinas por replicação de máquina estados MinBFT e MinZyzzyva apresentados em [Veronese et al. 2011] melhoram algoritmos anteriores requerendo apenas $2f + 1$ ao invés $3f + 1$ réplicas. Os autores apresentam um serviço confiável simples. O algoritmo é muito simples e usa o *chip* TPM (*Trusted Platform Module*) como componente inviolável para implementar o serviço USIG (*Unique Sequential Identifier Generator*) que designa números de ordenação para as mensagens vindas dos clientes, assim, toda réplica correta executa as requisições na ordem designada pelo USIG. A contribuição deste trabalho vem em termos de custo, resiliência e complexidade de gerenciamento, ficando mais próximo dos algoritmos de replicação tolerantes a faltas catastróficas. Entretanto, para a criação do serviço inviolável são necessárias modificações na máquina servidora que precisa possuir o *chip* TPM, em nível de *hardware*, tornando sua implementação mais complexa do que utilizar-se de virtualização e de compartilhamento distribuído de dados.

Em [Correia et al. 2004] é apresentado o *Trusted Timely Computing Base* (TTCB) com Trusted Multicast Ordering (TMO). Neste trabalho é discutido um sistema de replicação tolerante a intrusão baseado no conceito de replicação de máquina de estados. Com o uso de um *wormhole*, um componente inviolável distribuído que pode ser implementado localmente na máquina servidora ou diretamente em *hardware*, os autores conseguiram alcançar a resiliência $\lfloor \frac{n-1}{2} \rfloor$. O sistema possui duas redes para tolerar faltas, a primeira é assíncrona e se estabelece entre os clientes e os servidores (rede de *payload*), enquanto

que a segunda é síncrona e é utilizada para que os servidores conectem-se ao *wormhole* [Correia et al. 2002] TTCB (*Trusted Timely Computing Base*. O SMR, proposto no artigo, usa o serviço TMO (*Trusted Multicast Ordering*) implementado dentro do TTCB. Por estar dentro do TTCB, o TMO tem sua execução assegurada contra faltas maliciosas. Seu funcionamento é simples, o serviço designa números de ordenação para cada mensagem recebida dos clientes, então, todas as réplicas corretas utilizam esta ordem para executar as requisições. A abordagem usando TTCB e TMO é de suma importância na área de replicação de máquina de estados já que consegue atingir a melhor resiliência prática.

Por outro lado, a implementação do TTCB requer o uso de um *hardware* à parte, como pode ser visto em [Correia et al. 2002]. Este componente dificulta a aplicação prática, além disso, é necessário proteger este componente de *hardware* com algum componente em *software*, responsável por separar as operações do TTCB das operações do sistema operacional. Os autores fazem essa proteção através da modificação do *kernel* do RT-Linux. Esta abordagem tem as seguintes implicações:

1. Dependência da versão do *kernel* que pode estar desatualizada ou mais vulnerável, se comparado às versões mais recentes. Com o RegPaxos isto não é um problema, pois não há restrição quanto ao uso de sistema operacional, permitindo heterogenenidade [Obelheiro et al. 2005].
2. As bibliotecas de acesso ao TTCB precisam ser implementadas para cada linguagem de programação. Isto limita a diversidade, já que diferentes paradigmas precisam ser adaptados para serem utilizados com o TTCB. O RegPaxos provê acesso independente de linguagem, permitindo que o serviço seja implementado através de qualquer paradigma ou linguagem de programação.
3. O TTCB impõe limitações a um atacante através da remoção do acesso de super usuário no sistema operacional, protegendo apenas o ID/GID 0 (*root*). Técnicas do tipo *privilege escalation* podem ser empregadas para explorar falhas de *design* como *buffer, stack* ou *heap overflow* e erros de configuração. No RegPaxos, o isolamento é feito através do gerenciador de máquinas virtuais (*hypervisor*) da máquina virtual, portanto, um atacante consegue penetrar apenas na máquinas virtuais, deixando o sistema hospedeiro protegido.

Além disso, no TTCB, a separação entre as duas redes é feita pelas interfaces de rede, o que implica na necessidade de pelo menos duas placas de rede por servidor. No RegPaxos a separação é feita através do modo *bridge* da tecnologia de virtualização, diminuindo a quantidade de recursos de *hardware* necessários.

Tabela 1. Comparação entre os protocolos em ambientes livres de falhas

Protocolos Avaliados	***Propriedades e características***			
	# Réplicas	**# Máquinas físicas**	**# Passos de comunicação**	**Especulativo**
PBFT [Castro and Liskov 1998]	$3f+1$	$3f+1$	5	não
Zyzzyva [Kotla et al. 2008]	$3f+1$	$3f+1$	3	sim
TTCB [Correia et al. 2004]	$2f+1$	$2f+1$	5	não
A2M-PBFT-EA [Chun et al. 2007]	$2f+1$	$2f+1$	5	não
MinBFT [Veronese et al. 2011]	$2f+1$	$2f+1$	4	não
MinZyzzyva [Veronese et al. 2011]	$2f+1$	$2f+1$	3	sim
RegPaxos	$2f+1$	$2f+1$	3	não

3. Visão Geral do RegPaxos

3.1. Modelo de Sistema

O modelo de sistema adotado é híbrido [Veríssimo 2006], o que significa que existe variação, de componente para componente, em relação às suposições de sincronismo e presença/severidade de faltas e falhas [Correia et al. 2002, Veríssimo 2006]. Em nosso modelo, consideramos diferentes suposições para os subsistemas que executam no *host* e no *guest* das máquinas virtuais que compõe o sistema. Neste modelo, o conjunto $C = \{c_1, c_2, c_3, ...\}$ representa o número finito de processos clientes e $S = \{s_1, s_2, s_3, ..., s_n\}$ representa o conjunto de servidores contendo n elementos que implementam o serviço oferecido pelo sistema SMR. Cada servidor possui uma máquina virtual que contém apenas um sistema como *guest*. O modelo de falhas admite que um número finito de clientes e até $f \leq [\frac{n-1}{2}]$ servidores incorram em faltas de suas especificações, apresentando comportamento arbitrário ou bizantino [Lamport et al. 1982]: um processo faltoso pode desviar de suas especificações omitindo ou parando de enviar mensagens, ou ainda apresentar qualquer tipo de comportamento (malicioso ou não) não especificado. Entretanto, assumimos independência entre as faltas, em outras palavras, a probabilidade de um servidor incorrer em faltas é independente da ocorrência de faltas em outro servidor. Na prática, é possível atingir-se isto através do uso extensivo de diversidade (i.e. diferentes ambientes de *hardware/software*, sistemas operacionais, máquinas virtuais, bases de dados, linguagens de programação, etc) [Rodrigues et al. 2001].

Nosso modelo de sistema prevê o uso de duas redes de comunicação. A primeira, a rede de *payload* é assíncrona e é utilizada para transferência de dados da aplicação. Não existem suposições baseadas em tempo para a rede de *payload* e sua utilização se dá entre clientes e servidores para envio de requisições e respostas. A segunda rede é controlada (registradores compartilhados distribuídos) e é utilizada para que os servidores troquem mensagens do protocolo de acordo. Assumem-se as seguintes propriedades para este rede:

- é segura, isto é, resistente a qualquer possível ataque, falhando apenas por *crash*;
- é capaz de executar operações com delimitação temporal;
- provê apenas duas operações, leitura e escrita em registradores. Estas operações não podem ser afetadas por faltas maliciosas.

Assumimos que cada par cliente-servidor c_i, s_j e cada par de servidores s_i, s_j está conectado por um canal confiável com duas propriedades: se o remetente e o destinatário de uma mensagem são ambos corretos, então (1) a mensagem é eventualmente recebida e (2) a mensagem não é modificada no canal [Correia et al. 2004]. Na prática, estas propriedades podem ser obtidas com o uso de criptografia e retransmissão [Wangham et al. 2001]. *Message authentication codes* (MACs) são *checksums* criptográficos que servem ao nosso propósito, e usam apenas criptografia simétrica [Menezes et al. 1996, Castro and Liskov 2002]. Os processos precisam compartilhar chaves simétricas para fazerem uso de MACs. Neste artigo, assumimos que estas chaves são distribuídas antes do protocolo ser executado. Na prática, isto pode ser resolvido usando protocolos de distribuição de chaves disponíveis na literatura [Menezes et al. 1996].

3.2. Arquitetura do Ambiente

A arquitetura do sistema está representada na figura 1. As máquinas virtuais são os servidores que possuem a réplica do serviço. As máquinas físicas compõe uma abstração de

memória compartilhada emulada [Guerraoui and Rodrigues 2006] (veja 3.4.) aqui chamada de registradores compartilhados distribuídos (DSR - acrônimo para *Distributed Shared Register*). Cada máquina física possui seu próprio espaço dentro dos DSR, onde sua respectiva máquina virtual registra mensagens do tipo PROPOSE, ACCEPT ou CHANGE. Todos os servidores podem ler todos os registros, não importando quem o registrou.

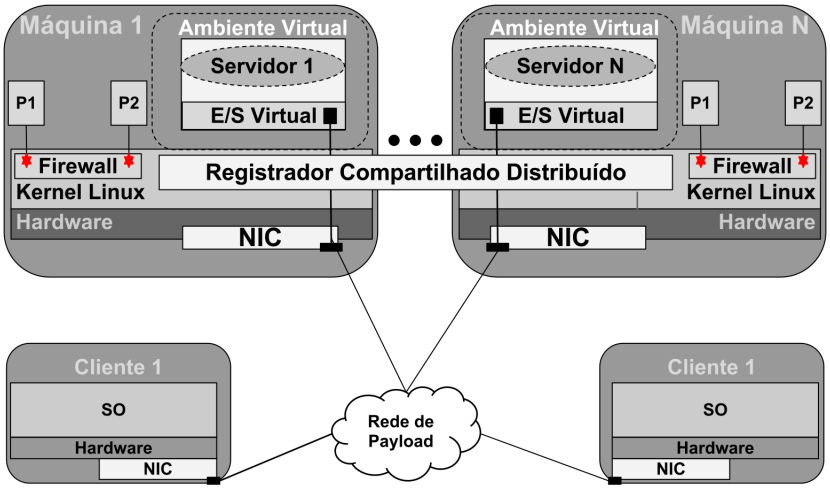


Figura 1. Visão geral da arquitetura

Nós assumimos que apenas as máquina físicas podem, de fato, conectar-se à rede utilizada pelos DSR. Isto implica que os registradores compartilhados distribuídos só estão acessíveis pelas máquinas físicas que hospedam (*host*) as máquinas virtuais. Transitivamente, os DSR não estão acessíveis através do *guest* das máquinas virtuais. Cada processo é encapsulado dentro de sua própria máquina virtual, assegurando isolamento. Todas as comunicações entre clientes e servidores acontecem em uma rede separada e, do ponto de vista do clientes, a máquina virtual é transparente. Isto significa que os clientes não reconhecem a arquitetura física-virtual. Cada máquina possui apenas uma interface de rede, um *firewall* e/ou o modo *bridge* são utilizados no *host* para assegurar a divisão entre as redes. Assumimos que as vulnerabilidades do *host* não podem ser exploradas através da máquina virtual. O gerenciador de máquinas virtuais (*hypervisor*) assegura este isolamento, garantindo que um atacante não tem meios para acessar o *host* através da máquina virtual. Isto é uma premissa em tecnologias de virtualização, como *VirtualBox, LVM, XEN, VMWare, VirtualPC*, etc. Nosso modelo assume que o sistema *host* é inacessível externamente, o que também pode ser garantido através do uso do modo *bridge* e/ou *firewalls* no sistema hospedeiro.

3.3. Tecnologia de virtualização

Dois dos maiores desafios deste trabalho, que são a disponibilização de um serviço confiável de registradores e a garantia do seu isolamento, são resolvidos através do uso de virtualização. Este serviço é implementado nas máquinas hospedeiras que permitem acesso controlado das réplicas. O uso de virtualização traz ainda outros benefícios, como permitir controle e monitoramento da comunicação entre as réplicas e a regeneração de réplicas faltosas, através do uso de imagens do sistema, o que pode ser implementado em trabalhos futuros.

A partir da tecnologia de virtualização é possível criar o serviço confiável de registradores mantendo-o isolado. Diversos gerenciadores de máquinas virtuais (*hypervi-*

sors) possuem mecanismos que podem ser utilizados para garantir o isolamento. No âmbito de testes deste trabalho utilizamos o Hypervisor VirtualBox 4.1.18 e sua funcionalidade de compartilhamento de diretórios afim de simplificar a implementação dos DSR. No entanto, qualquer *hypervisor* tipo 2 poderia ser utilizado e, inclusive, ressaltamos que o uso de diversidade na utilização de *hypervisors* é muito interessante, pois dificulta a exploração de vulnerabilidades por parte de um atacante. A segurança do *hypervisor* é essencial para obter isolamento, por isso alguns trabalhos estudaram como melhorá-la [Murray et al. 2008, Wang and Jiang 2010, Szefer et al. 2011]. Neste artigo, assumimos que um atacante não tem informações suficientes para determinar a arquitetura física-virtual, portanto, outras técnicas podem ser empregadas para garantir o isolamento e a segurança da máquina virtual. Técnicas como Blue Pill [Rutkowska 2006] impedem/dificultam a detecção de máquina virtual de maneira mais contundente.

3.4. Emulação de Memória Compartilhada: Registradores compartilhados distribuídos

Memória compartilhada emulada é uma abstração de registradores de um conjunto de processos que se comunicam através de troca de mensagens [Guerraoui and Rodrigues 2006]. Esta definição é realmente atraente, já que permite que a memória compartilhada seja construída utilizando-se qualquer tecnologia para compartilhamento de memória. Um memória compartilhada, emulada ou não, pode ser vista como um *array* de registradores compartilhados. Consideramos aqui a definição sob a ótica do programador. O tipo de registrador compartilhado especifica que operações podem ser efetuadas e os valores retornados por elas [Guerraoui and Rodrigues 2006]. Os tipos mais comuns são registradores de leitura/escrita. As operações de um registrador são invocadas pelos processos do sistema para troca de informações.

Neste trabalho criamos os registradores compartilhados distribuídos (DSR - acrônimo para *Distributed Shared Registers*), isto é, uma memória compartilhada emulada baseada em troca de mensagens, desenvolvida sobre uma rede controlada e por meio do uso de diretório compartilhados, com controle de acesso. Nós assumimos que a rede controlada só é acessada por componentes dos DSRs. Os DSRs são implementados nos *hosts* das máquinas virtuais dos sistemas e assume-se que o *hypervisor* assegura o isolamento entre *guests* e *hosts*. Os DSR são utilizados para implementar um serviço de *atomic multicast* que garante que todos os servidores corretos irão entregar as mesmas mensagens, em uma mesma ordem e, se o remetente é correto, todos os servidores irão garantir a entrega da mensagem enviada. Os DSRs são capazes de efetuar apenas duas operações (1) $read()$, que lê a última mensagem escrita nos DSR e (2) $write(m)$, que escreve a mensagem m nos DSR. Assume-se duas propriedades com relação às operações acima: (i) progressão (*liveness*) - a operação eventualmente termina; (ii) segurança (*safety*) - a operação de leitura sempre retorna o último valor escrito.

Para garantirmos isto, cada servidor possui um arquivo em que seu *guest* tem acesso para escrita e um segundo arquivo onde o mesmo possui acesso apenas para leitura. Todos os acessos são feitos por um único processo [Guerraoui and Rodrigues 2006]. O primeiro arquivo é espaço da réplica e nenhuma outra réplica pode escrever nele. O segundo arquivo representa o espaço de registradores das demais réplicas e é atualizado pelos DSR. O *hypervisor* fornece suporte para fazer com que um arquivo criado no *host* esteja acessível no *guest*, forçando as permissões de escrita e leitura.

Os DSR aceitam apenas mensagens tipadas, e existem apenas três tipos: (i) PRO-

POSE, (ii) ACCEPT e (iii) CHANGE. As mensagens sem tipo ou com tipos não permitidos são ignoradas. Todas as mensagens trocadas nos DSR são automaticamente identificadas, isto significa que, quando um servidor escreve uma mensagem, os DSR colocam o ID da réplica na mensagem, de modo que uma réplica não possa se passar por outra. Assumimos que a comunicação é feita através de *fair links* com as seguintes propriedades: se o remetente e o destinatário de uma mensagem são ambos corretos, então (1) se a mensagem é enviada infinitamente para um destinatário correto, então ela é recebida infinitas vezes; (2) existe um tempo T, tal que, se a mensagem é retransmitida infinitas vezes para um destinatário correto de acordo com um tempo t_0, então o destinatário recebe a mensagem pelo menos uma vez antes de $t_0 + T$; e (3) as mensagens não são modificadas no canal [Yin et al. 2003]. Esta suposição parece razoável na prática, uma vez que os DSR são síncronos e podem ser afetados apenas por faltas de *crash*, baseado no isolamento proporcionado pelo *hypervisor*.

4. Algoritmo

4.1. Visão geral do protocolo

O protocolo necessita que apenas um servidor seja líder, cuja responsabilidade é propor ordem de execução para as requisições de clientes. Os demais servidores são apenas réplicas do serviço. A escolha do primeiro líder é baseada em configuração. A mudança de liderança ocorre sempre que a maioria das réplicas ($f + 1$) concordam ser necessário. O protocolo se inicia quando um cliente requisita a execução de alguma tarefa nos servidores. O líder é responsável por ordenar e difundir mensagens dos clientes. A ordenação acontece quando o líder escreve nos DSR uma mensagem do tipo PROPOSE. Mensagens deste tipo incluem a mensagem original do cliente seguida de um número de ordenação para a mesma. Cada servidor delibera individualmente se deve ou não aceitar a proposta. Aceitar a proposta significa escrever nos DSR uma mensagem do tipo ACCEPT, esperar por $f - 1$ mensagens de aceitação (pois já possui a sua própria e a mensagem de PROPOSE, isto é $f - 1 + 2 \implies f + 1$), executar a tarefa na ordem estipulada e responder para o cliente com o resultado.

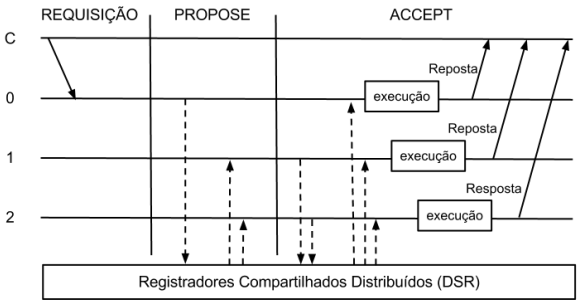


Figura 2. Fluxo da requisição à resposta

Como em outros sistemas tolerantes a faltas bizantinas [Kotla et al. 2008, Correia et al. 2004, Yin et al. 2003, Veronese et al. 2011, Castro and Liskov 2002], para lidar com o problema de um servidor malicioso s_j, que poderia deixar de difundir mensagens, o cliente espera por pelo menos $f + 1$ respostas de diferentes servidores por um tempo $T_{reenviar}$. Após $T_{reenviar}$ o cliente reenvia a requisição para todos os servidores. Uma réplica honesta (não líder) ao receber uma requisição correta, solicita uma

mudança de liderança. Se $f + 1$ servidores corretos solicitarem a mudança de líder, então a mudança ocorre e o protocolo progride. Entretanto, o sistema de *payload* é assumidamente assíncrono, portanto, não existem limitantes temporais para sua comunicação, não sendo possível definir um valor $T_{reenviar}$ "ideal". Em [Correia et al. 2004] é mostrado que o valor $T_{reenviar}$ envolve uma troca: se for alto demais, o cliente pode esperar tempo demais para ter sua operação executada; se for baixo demais, o cliente pode fazer o reenvio do comando sem necessidade. O valor deve ser selecionado considerando-se essa troca. Se o comando é reenviado sem necessidade, as duplicatas são descartadas pelo sistema.

Esta seção oferece uma descrição mais profunda do algoritmo. A sequência de operações do algoritmo é apresentada em um nodo líder e em um nodo não líder. Primeiramente considera-se a operação em caso normal (sem a presença de faltas) e posteriormente a operação sob a presença de faltas. O diagrama de fluxo da operação em caso normal pode ser visto na figura 2.

4.2. Operação em caso normal

1. O protocolo inicia-se quando um cliente c envia ao servidor líder a mensagem $\langle REQUEST, o, t, c_a, v\rangle_{\sigma ci}$ com a operação o que deseja que seja executada e o seu endereço c_a. O campo t é o *timestamp* da requisição para assegurar a semântica de apenas uma execução, em outras palavras, os servidores não executam uma operação do cliente c cujo *timestamp* não seja maior que último *timestamp* para o mesmo cliente. Esta política impede que uma mesma tarefa seja executada inúmeras vezes. O campo v é o vetor que armazena os MACs gerados pelo cliente para cada servidor. Os MACs são gerados utilizando-se a mensagem do cliente e as chaves que foram compartilhadas previamente entre o cliente e os servidores. Em função disto, cada servidor pode averiguar a integridade da mensagem, descartando-a se necessário.

Constants:
f : int // Maximum tolerated faults
T : int // Maximum waiting time for a proposal to be decided
Variables:
accepted : int // counter of acceptance for some ordering
upon receive $\langle REQUEST, o, t_j, c_a, v\rangle_{\sigma ci}$ from client
if $t_j \leq t_{j-1}$ *for* c_i *OR isWrong(* v *)* **then**
    return;
end
write($\langle PROPOSE, n, \langle REQUEST, o, t_j, c_a, v\rangle_{\sigma ci}\rangle_{\sigma si}$) into DSR;
accepted = waitForAcceptance(T);
if *accepted* $\geq f + 1$ **then**
    store $\langle PROPOSE, n, \langle REQUEST, o, t_j, c_a, v\rangle_{\sigma ci}\rangle_{\sigma si}$ in the execution buffer;
end
return;

Algoritmo 1. Algoritmo executado pelo líder.

2. Após receber a requisição do cliente e verificar sua integridade usando o MAC em v, o líder gera uma proposta e a escreve nos DSR. A mensagem $\langle PROPOSE, n_m, m\rangle_{\sigma s}$ gerada pelo líder possui em seu corpo m, que é a mensagem original do cliente, e n_m, que representa o número de ordenação para a mensagem. Vale ressaltar que, como foi discutido, os DSR automaticamente adicionam nas mensagens o identificador do servidor que as registrou. Após escrever a mensagem, o líder aguarda pelas mensagens de aceite de acordo com a proposta. Em paralelo, outras propostas podem ser ordenadas, a medida que chegam requisições.

Após receber f mensagens concordando com a proposta (pois a proposta mais f aceites constituem as $f + 1$ mensagens mínimas para o acordo) o líder guarda a mensagem e a executa na ordem estipulada. O resultado da execução é enviado para o cliente em uma mensagem $\langle REPLY, t, c_a, r_a, r \rangle_{\sigma si}$ que contém o endereço do cliente c_a, o *timestamp* t, o endereço da réplica r_a, e o resultado r. Este comportamento pode ser visto no algoritmo 1. Como foi discutido em 3., todas as mensagens trocadas nos DSR são eventualmente entregues se nem o destinatário, nem o remetente, tiverem sofrido falta de *crash*.

Constants:
f : int // Maximum tolerated faults
T : int // Maximum waiting time for a proposal to be decided.
Variables:
accepted : int // counter of acceptance for some ordering
upon read $\langle PROPOSE, n, \langle REQUEST, o, t, c_a, v \rangle_{\sigma ci} \rangle_{\sigma ss}$ from DSR
if *isValid(v) and isValid(n) and* $t_j > t_{j-1}$ *for* c_i **then**
    write($\langle ACCEPT, n, h_m \rangle_{\sigma si}$) into DSR;
    accepted = waitForAcceptance(T);
    if *accepted* $\geq f + 1$ **then**
        store $\langle PROPOSE, n, \langle REQUEST, o, t_j, c_a, v \rangle_{\sigma ci} \rangle_{\sigma si}$ in the execution buffer;
    end
else
    write($\langle CHANGE, h_m, s_s \rangle_{\sigma si}$) into DSR and bufferize;
end
return;

Algoritmo 2. Algoritmo executado pelo(s) não líder(es).

3. Sempre que a réplica r_j recebe uma proposta do líder, r_j a valida, o que significa que (i) é feita a verificação da integridade da mensagem do cliente usando o MAC em v, além da verificação de seu $timestamp$ e (ii) é feita a verificação de que nenhuma outra mensagem tenha sido ordenada com o mesmo n_m. Após serem feitas as verificações, se r_j aceitar a proposta, então é criada uma mensagem $\langle ACCEPT, n_m, h_m \rangle_{\sigma sj}$ que é escrita nos registradores. Os campos da mensagem representam o *hash* da mensagem do cliente h_m e o número de ordenação proposto pelo líder n_m. Após a escrita nos registradores, a réplica aguarda por $f - 1$ mensagens de aceitação (já que já possui dois aceites, a sua mensagem e a proposta do líder). Em paralelo, outras mensagens podem ser aceitas, à medida que propostas cheguem aos DSR. A réplica r_j executa as requisições na ordem em que foram aceitas e responde aos clientes com mensagens de REPLY. Este comportamento pode ser observado no algoritmo 2.
4. Após receber $f + 1$ respostas iguas entre si e de diferentes servidores, o cliente finalmente as aceita como corretas.

4.3. Operação sob a presença de faltas

A operação sob a presença de faltas implica na mudança de líder, portanto, fazemos uma breve explicação de como essa mudança se desenvolve.

Mudança de líder: durante a configuração do sistema, todas as réplicas recebem um número de identificação. Estes números são sequenciais e iniciados em zero. Todas as réplicas conhecem o identificador do líder L_{id} e o número total de réplicas no sistema N. Quando $f + 1$ réplicas corretas suspeitam do líder atual, elas simplesmente definem $L_{id} = L_{id} + 1$ como o próximo líder se $L_{id} < N$, senão, $L_{id} = 0$

Como foi discutido, a réplica r_j valida qualquer proposta que receba. Se por alguma razão a réplica decidir não aceitar a proposta ou se r_j receber uma mensagem correta de

um cliente e verificar que a mesma não foi proposta, r_j solicitará uma mudança de líder. O diagrama de fluxos da figura 3 exemplifica um dos casos de execução sob a presença de faltas.

1. O protocolo pode iniciar o modo faltoso de acordo com as duas maneiras abaixo:
   (a) Quando a réplica r_j recebe uma mensagem do tipo CHANGE, mas ainda não suspeita do líder, ela armazena a mensagem em seu *buffer* para utilizá-la futuramente, se necessário.
   (b) Quando a réplica r_j suspeita do líder por alguma mensagem m, então r_j escreve nos DSR e em seu *buffer* uma mensagem $\langle CHANGE, h_m, s_s \rangle_{\sigma sj}$ que contém o *hash* da mensagem h_m e o id s_s do líder do qual r_j suspeita.
2. A réplica r_j inicia uma busca em seu *buffer* de suspeita para encontrar $f+1$ mensagens relacionadas à mensagem m. Caso r_j encontre $f+1$ (incluindo ao seu próprio) identificadores para a mesma mensagem, então a réplica efetua a mudança de líder.
3. Se o identificador do líder for o da réplica r_j, então a réplica recomeça o processo de ordenação utlizando-se das mensagens de aceite escritas nos DSR para definir o estado atual do sistema. Recomeçando o processo de acordo com a última mensagem que foi aceita pela maioria. Se o identificador não for o da réplica r_J, então a réplica aguardará pela mensagem do novo líder.
4. Quando r_j receber uma proposta do novo líder, então ela limpará seu *buffer* e voltará o protocolo para a operação sem a presença de faltas.

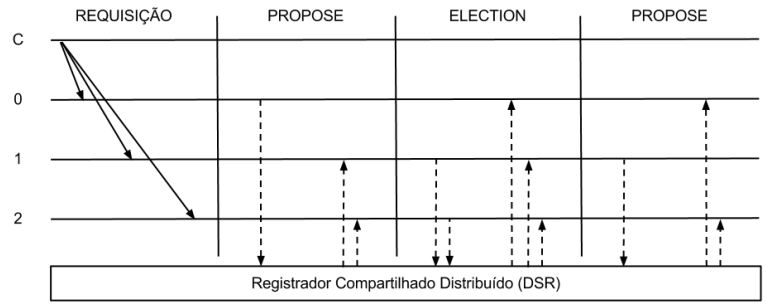
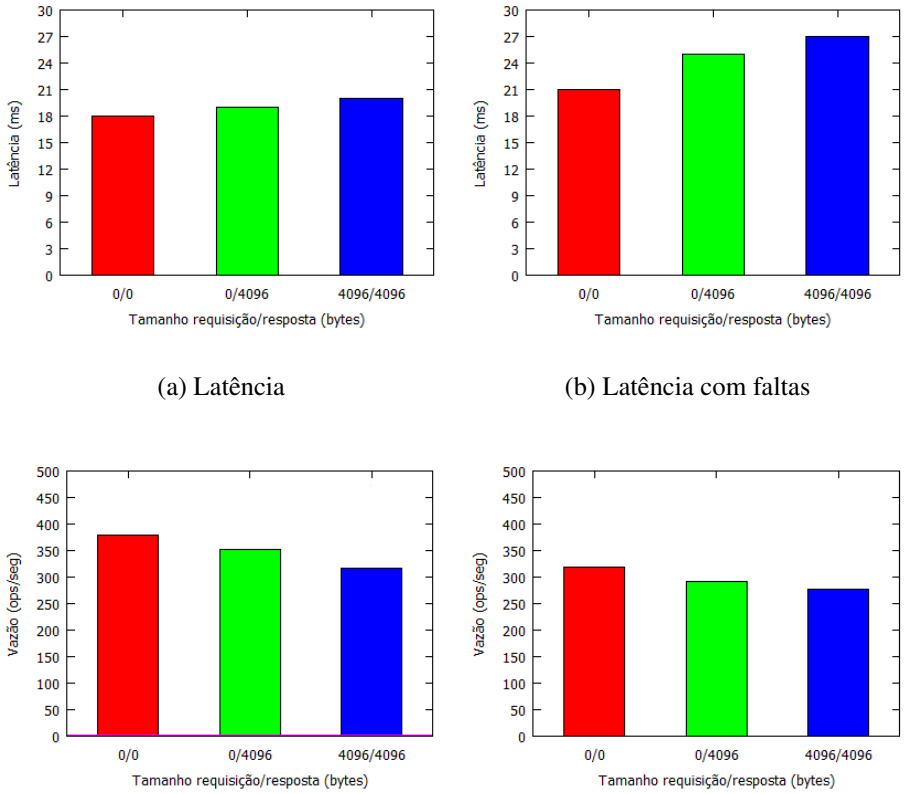


Figura 3. Fluxo de uma mudança de líder

5. Implementação e Avaliação de Desempenho

Os algoritmos foram implementados usando a linguagem Java, JDK 1.6.0. Os canais de comunicação foram implementados usando *sockets* TCP da API NIO. Os sistemas operacionais dos *hosts* das máquinas virtuais foram MacOSx Lion e Ubuntu 12.04. Nos *guests* das máquinas virtuais utilizou-se Ubuntu 12.04 e Debian 6 Stable. O *hypervisor* escolhido foi a VirtualBox. As métricas de avaliação escolhidas foram latência e vazão, dado que estas são as métricas largamente usadas para avaliar sistemas computacionais e representam a eficiência do sistema de uma maneira simples [Jian 1991].

Os valores foram obtidos através de *micro-benchmarks* com diferentes cargas. A latência foi obtida pela medida do *round-trip*. Nós medimos o tempo entre o envio e o recebimento de um grupo de mensagens. O raciocínio por trás do uso de *micro-benchmarks* é medir adequadamente o algoritmo sem influências externas. A fim de avaliar a capacidade do protocolo, executou-se com tamanhos diferentes de mensagens. A vazão representa a



(a) Latência

(b) Latência com faltas

(c) vazão em caso normal

(d) vazão com faltas

Figura 4. Desempenho verificado para o RegPaxos.

capacidade do sistema de processamento de mensagens por unidade de tempo, portanto, sendo medido o tempo que demora a receber a resposta de todas as réplicas.

Para avaliar o desempenho do algoritmo na ausência de faltas, executou-se o protocolo em condições normais enviando 10.000 requisições através de um único cliente com três cargas diferentes: 0/0kb, 0/4 kb e 4/4 kb. Com isto temos: uma requisição vazia e uma resposta vazia, uma requisição vazia e uma resposta de 4kb de tamanho e uma requisição de 4kb com uma resposta de 4kb. Para avaliar o algoritmo em um ambiente faltoso, as réplicas foram configuradas para que, quando assumissem o papel de líderes, enviassem uma entre dez respostas incorretas. As figuras 4(a) e 4(b) apresentam a latência para cada carga diferente. A latência foi obtida pela média entre as respostas de todas as requisições. Como podemos observar, a latência tem variações mínimas entre diferentes cargas. A vazão, representada pelas figuras 4(c) e 4(d), foram baseadas no cálculo do tempo total de 10.000 requisições

Dados comparativos entre o RegPaxos e o estado da arte em sistemas BFT são apresentados na tabela 1. Todos os dados consideram execuções na ausência de faltas. Os benefícios do uso do RegPaxos são visíveis quando comparados os números de passos de comunicação e quantidade de réplicas necessária. Nossa abordagem tem a melhor resiliência prática em termos de réplicas, juntamente com [Chun et al. 2007, Correia et al. 2004, Veronese et al. 2011], e também tem o mesmo número de passos que [Kotla et al. 2008, Veronese et al. 2011], mesmo não sendo especulativo. Ao evitar o envolvimento dos clientes, permitimos que um número finito de clientes incorra em faltas.

6. Conclusão

Ao explorar o uso de técnicas de memória compartilhada emulada (registradores compartilhados distribuídos) e virtualização, conseguimos propor uma rede inviolável simples que suporta o nosso algoritmo BFT. Mostrou-se que é possível implementar um algoritmo SMR confiável, com apenas $2f + 1$ réplicas usando tecnologias comuns, como virtualização e abstrações de compartilhamento de dados. Provou-se que é possível criar um modelo híbrido que, mesmo não sendo especulativo, é capaz de atingir o mesmo número de passos de comunicação sem a necessidade de modificar o *software* do servidor e/ou criar componentes de *hardware*. A tecnologia de virtualização é amplamente utilizada e pode proporcionar um bom isolamento entre o serviço confiável e o mundo exterior, e a utilização do DSR simplifica o progresso do protocolo. Além disso, foi possível reduzir o número de passos de comunicação, o que pode reduzir a latência.

Referências

Bessani, A. N., Correia, M., da Silva Fraga, J., and Lung, L. C. (2007). Decoupled quorum-based byzantine-resilient coordination in open distributed systems. In *NCA'07*, pages 231–238.

Castro, M. and Liskov, B. (1998). Practical byzantine fault tolerance. *Operating Systems Review*, 33:173–186.

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

Chun, B., Maniatis, P., Shenker, S., and Kubiatowicz, J. (2007). Attested append-only memory: Making adversaries stick to their word. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 189–204. ACM.

Correia, M., Neves, N., and Verissimo, P. (2004). How to tolerate half less one byzantine nodes in practical distributed systems. In *Reliable Distributed Systems, 2004. Proceedings of the 23rd IEEE International Symposium on*, pages 174–183. IEEE.

Correia, M., Neves, N., and Verissimo, P. (2012). Bft-to: Intrusion tolerance with less replicas. *The Computer Journal*.

Correia, M., Verissimo, P., and Neves, N. (2002). The design of a cots real-time distributed security kernel. *Dependable Computing EDCC-4*, pages 634–638.

Guerraoui, R. and Rodrigues, L. (2006). *Introduction to reliable distributed programming*. Springer-Verlag New York Inc.

Jian, R. (1991). The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling.

Kotla, R., Clement, A., Wong, E., Alvisi, L., and Dahlin, M. (2008). Zyzzyva: speculative byzantine fault tolerance. *Communications of the ACM*, 51(11):86–95.

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

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

Menezes, A., Van Oorschot, P., and Vanstone, S. (1996). *Handbook of applied cryptography*. CRC.

Murray, D. G., Milos, G., and Hand, S. (2008). Improving xen security through disaggregation. In *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 151–160. ACM.

Obelheiro, R. R., Bessani, A. N., and Lung, L. C. (2005). Analisando a viabilidade da implementação prática de sistemas tolerantes a intrusões. In *Anais do V Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg 2005*.

Reiser, H. and Kapitza, R. (2007). Vm-fit: supporting intrusion tolerance with virtualisation technology. In *Proceedings of the Workshop on Recent Advances on Intrusion-Tolerant Systems*.

Reiter, M. (1995). The rampart toolkit for building high-integrity services. *Theory and Practice in Distributed Systems*, pages 99–110.

Rodrigues, R., Castro, M., and Liskov, B. (2001). Base: Using abstraction to improve fault tolerance. *ACM SIGOPS Operating Systems Review*, 35(5):15–28.

Rutkowska, J. (2006). Introducing blue pill. *The official blog of the invisiblethings. org*, 22.

Schneider, F. (1982). Synchronization in distributed programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(2):125–148.

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

Stumm, V., Lung, L., Correia, M., da Silva Fraga, J., and Lau, J. (2010). Intrusion tolerant services through virtualization: a shared memory approach. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 768–774. IEEE.

Szefer, J., Keller, E., Lee, R. B., and Rexford, J. (2011). Eliminating the hypervisor attack surface for a more secure cloud. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 401–412. ACM.

Veríssimo, P., Neves, N., and Correia, M. (2003). Intrusion-tolerant architectures: Concepts and design. *Architecting Dependable Systems*, pages 3–36.

Veríssimo, P. E. (2006). Travelling through wormholes: a new look at distributed systems models. *SIGACT News*, 37(1):66–81.

Veronese, G., Correia, M., Bessani, A., Lung, L., and Verissimo, P. (2011). Efficient byzantine fault tolerance. *Computers, IEEE Transactions on*, pages 1–1.

Wang, Z. and Jiang, X. (2010). Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In *IEEE Symposium on Security and Privacy - SP'10*, pages 380–395. IEEE.

Wangham, M. S., Lung, L. C., Westphall, C. M., and da Silva Fraga, J. (2001). Integrating ssl to the jacoweb security framework: Project and implementation. In *Integrated Network Management'01*, pages 779–792.

Yin, J., Martin, J., Venkataramani, A., Alvisi, L., and Dahlin, M. (2003). Separating agreement from execution for byzantine fault tolerant services. *ACM SIGOPS Operating Systems Review*, 37(5):253–267.

HARP: Um novo protocolo para alta disponibilidade implementado em FPGA

Rômerson Deiny Oliveira, Daniel Gomes Mesquita, Pedro Frosi Rosa

Faculdade de Computação – Universidade Federal de Uberlândia (UFU)
Caixa Postal 593 – 38.408-100 – Uberlândia – MG – Brazil

romerson@mestrado.ufu.br, {mesquita,frosi}@facom.ufu.br

Abstract. *The network's downtimes has brought financial losses to Internet users and companies. This paper aims to show the developing of a new high availability protocol and how its validation was done. The existing protocols have two harmful situations, named No-Brain and Split-Brain conditions, that are algorithmic problems that attack the network availability. This paper will show HARP protocol and how it fixes its predecessors.*

Resumo. *Os períodos de indisponibilidade das redes de computadores têm causado grandes prejuízos às empresas e aos usuários da Internet. Este trabalho visa mostrar o desenvolvimento de um novo protocolo de alta disponibilidade e como ele foi validado. Os protocolos existentes apresentam situações conhecidas como Acéfalo e Cérebro Bipartido, que são problemas algorítmicos que afetam a disponibilidade das redes. Este trabalho apresentará o protocolo HARP e como ele corrige seus predecessores.*

1. Introdução

A Internet tornou-se um dos principais veículos para transações pessoais e empresariais nos últimos anos. Segundo a *International Telecommunications Union*, o número de usuários de Internet cresceu 389% entre 2001 e 2011 [ITU 2012], em números absolutos o total passou de 495 milhões para 2 bilhões e 421 milhões de usuários conectados. Acompanhando esta eclosão, cresce também o número de usuários com necessidade de suporte a algum tipo de disponibilidade.

Períodos de indisponibilidade podem causar prejuízos de alto valor econômico. A *CA Technologies* divulgou uma pesquisa dizendo que o tempo de inatividade custa às empresas norte-americanas coletivamente $ 26,5 bilhões em receita por ano [Technologies 2010]. Assim sendo, no intuito de deixar a Internet disponível o maior tempo possível, já há alguns anos diversas empresas e universidades abriram espaço para pesquisa e desenvolvimento sobre alta disponibilidade de rede.

Os mecanismos de alta disponibilidade são caracterizados por usarem soluções baseadas em redundância de hardware, softwares inteligentes e protocolos para identificação de falhas de sistemas [Kriens et al. 2006]. Eles funcionam por meio de elementos que aparecem para a rede como um único elemento abstrato chamado de elemento virtual [Hinden 2004]. Os elementos físicos operam na filosofia mestre/escravo, tendo sempre um nó como mestre (não mais e não menos que um) e os outros disponíveis como escravos. O protocolo de comunicação deve ser capaz de avisar periodicamente a existência de

um mestre na rede e deve também ser capaz de eleger um novo mestre no caso de falha do atual.

Entre os protocolos de alta disponibilidade existentes, o *Virtual Router Redundacy Protocol* (VRRP) [Hinden 2004] destaca-se por ser o padrão *de facto* para os equipamentos de alta disponibilidade. Em [Lopes Filho 2008], o autor mostra que situações de indisponibilidade foram percebidas em redes que operavam com o protocolo VRRP e [Hashimoto et al. 2010] aponta as condições de "acéfalo" e "cérebro partido" (seção 2) como as causas que afetavam diretamente seu funcionamento.

[Hashimoto et al. 2010] apresentou uma proposta de extensão do VRRP para contornar as condições de acéfalo e cérebro bipartido, modelada em Redes de Petri. A fim de implementar a proposta, devido ao seu nível de abstração elevado foi necessário refiná-la e definir os cinco elementos do protocolo [Holzmann 1991] para permitir a sua implementação. Neste cenário, este trabalho apresenta o protocolo *High Availability Router Protocol* (HARP), define seus elementos e mostra como ele corrige os seus predecessores. O processo de desenvolvimento do HARP e sua implementação em hardware reconfigurável estão descritas na seção 3.

O HARP é parte de um projeto maior que pesquisa sobre Internet do Futuro, liderado pelo grupo MEHAR [MEHAR 2012] e trata dos aspectos de requisitos de alta disponibilidade do projeto EDOBRA, que por sua vez visa ampliar a cobertura física da instalação experimental OFELIA no Brasil [EDOBRA 2012]. A versão do HARP tratada neste trabalho é a versão 1, desenvolvida para a arquitetura atual da internet. As versões compatíveis com IPv6 e abordagem *Clean Slate* para internet não são tratadas, mas são passíveis de desenvolvimento e prototipação, tendo em vista a plataforma reconfigurável adotada para implementar o HARP.

O artigo está organizado da seguinte forma: a Seção 2 trata do estado da arte e trabalhos relacionadas. Já a Seção 3 mostra o método de desenvolvimento do protocolo HARP e a sequência de passos para sua validação. A Seção 4 apresenta os cinco elementos do protocolo e a Seção 5 traz as considerações finais.

2. Fundamentação Teórica e Estado da Arte

Esta seção aborda fundamentos dos mecanismos de alta disponibilidade e as situações que afetam os protocolos. Em seguida, é feita uma revisão da literatura correlata a fim de situar como os trabalhos se relacionam a esta pesquisa.

2.1. Alta Disponibilidade

Alta disponibilidade refere-se capacidade de uma rede manter-se disponível próximo de 100% do tempo, evitando a perda de serviços por meio da redução ou gestão de falhas e minimizando o tempo de inatividade planejado para o sistema. É obtida através de um endereço virtual que é compartilhado entre dois ou mais equipamentos. Este endereço é definido como *gateway* padrão da rede para os nós internos. Um roteador virtual, por exemplo, é a abstração formada por um ou mais roteadores executando um protocolo de alta disponibilidade. Na ocasião de uma falha no equipamento principal (mestre), outro componente do grupo de alta disponibilidade (escravo) assume a tarefa de roteamento, utilizando o endereço IP virtual, correspondente a um MAC virtual. Desta forma, a fa-

lha fica imperceptível aos clientes locais, já que a comunicação permanece ininterrupta [Sonderegger et al. 2009].

A alta disponibilidade é um subconjunto de tolerância a falhas [Johnson 1988], dado que esta trata desde a redundância dos componentes até o gerenciamento da comunicação com um protocolo. Do ponto de vista do protocolo, são duas as principais causas que levam uma rede à indisponibilidade:

1. Condição de acéfalo é a condição em que o nó de uma infraestrutura de alta disponibilidade, correntemente no papel de mestre, se torna inoperante e nenhum outro nó (nós escravos) se habilita a assumir tal papel [Pereira Júnior 2010].
2. Condição de cérebro bipartido é a condição em que um ou mais nós, correntemente no papel de escravos, de uma infraestrutura de alta disponibilidade se alçam ao papel de nó mestre por julgarem que o atual nó mestre tornou-se inoperante [Pereira Júnior 2010].

Estas condições podem ser causadas por falha de interface ou por ataque de terceiros. Este trabalho trata da primeira causa.

2.2. Trabalhos Relacionados

A literatura apresenta várias publicações que se pode relacionar a esta. De um lado, tem-se o tangente a implementações em FPGA aplicadas a redes de computadores e aplicadas a tolerância a falhas no nível de hardware. De um outro, tem-se as pesquisas que envolvem os protocolos em níveis mais altos de abstração.

Em [Jiang and Prasanna 2012], são realizados testes com switches OpenFlow para gerenciamento de fluxo e encaminhamento de pacotes. Trata-se de um trabalho que explora o paralelismo abundante do FPGA [Brown and Rose 1996] para tratar a nova geração da classificação de pacotes e propõe melhorias neste quesito, no intuito de tornar o processo de classificação e encaminhamento de pacotes mais escalável e com menores perdas. Esta caracterísitica é esperada para as próximas versões do HARP.

Em [Casado et al. 2009], o autor mostra que a utilização de hardware especializado para o encaminhamento de pacotes é uma técnica eficiente. Ainda em [Casado et al. 2009], introduz-se a ideia do uso de processadores de rede mais flexíveis como uma forma de contornar a necessidade de refazer chips devido a mudanças nos protocolos ou por adicionar novas características a este hardware, já que o custo é uma fator limitante.

O trabalho desenvolvido por Straka et al [Straka and Kotasek 2009] apresenta uma metodologia de construção de sistemas tolerante a falhas baseada em FPGA. As arquiteturas baseiam-se tanto no sistema duplex como na técnica de redundância modular tripla para melhorar a detecção de falhas. Para este propósito, o uso de verificadores *on-line* é demonstrado. Também é mostrado como os parâmetros disponibilidade (por exemplo tempo médio entre falhas e taxa de recuperação) podem ser afetados pelo ambiente de funcionamento em que o sistema tolerante a falhas é implementado. O trabalho é focado em técnicas de replicação de hardware. O trabalho [Straka and Kotasek 2009] apresenta técnina para construção de elementos de hardware redundantes, enquanto o trabalho mostrado neste artigo cuida da comunicação entre os elementos.

O trabalho apresentado em [Lopes Filho 2008] investiga a suspeita de que o principal problema dos protocolos de alta disponibilidade seria a camada de transporte, devido ao uso de protocolos não orientados a conexão, o que levou à proposta de uma camada de transporte baseada no protocolo SCTP. No entanto, ele conclui que o problema não residia na camada de transporte e a hipótese passou a pairar sobre a camada de enlace, devido à transmissão de mensagens com falso positivo para as camadas superiores.

[Hashimoto 2009] atribui os erros não detectados pelos algoritmos de controle de erro da camada de enlace como causa para o problema da condição de cérebro bipartido e conclui pela necessidade de desenvolvimento de um novo protocolo de alta disponibilidade, ou a extensão de um já existente. Foi demonstrado que o autômato do VRRP é incompleto, pois ele não considera erros não detectáveis pela camada de enlace. O autor afirma que o problema encontrado no VRRP também se aplica aos protocolos CARP e HSRP. O trabalho apresenta uma modelagem em redes de Petri que especifica as perdas de mensagens de anúncio em função dos fenômenos da camada de Enlace.

[Pereira Júnior 2010] discute as condições concorrentes para as situações de acéfalo e cérebro bipartido e define uma especificação de serviço que compõe o projeto de um protocolo de alta disponibilidade. O trabalho apresenta suposições de um ambiente onde um serviço de alta disponibilidade deve operar e como os protocolos de alta disponibilidade podem resolver os desafios que surgem nesses ambientes.

Os trabalhos publicadas em [Lopes Filho 2008], [Hashimoto 2009] e [Pereira Júnior 2010] descrevem uma sequência de hipóteses e conclusões sobre os problemas que atacam os protocolos de alta disponinilidade. Os autores concluíram que os problemas residem nas condições de acéfalo e cérebro bipartido percebidas no protocolo VRRP. Este trabalho situa-se como a continuação das pesquisas desenvolvidas nos trabalhos citados.

3. Método de desenvolvimento

O processo de desenvolvimento do *High Availability Router Protocol* estabeleceu-se divido em três etapas. Partiu-se da caracterização e proposição de uma extensão do VRRP. Em seguida, houve o processo de especificação e desenvolvimento do HARP. Por fim, é mostrada a proposta de um sistema de avaliação para protocolos de alta disponibilidade. Os itens nesta seção explicam este processo evolutivo.

3.1. Caracterização do Problema

Situações de indisponibilidade foram percebidas no tráfego mantido por arquiteturas que operavam com o VRRP. Ao observar a situação, o fenômeno foi reproduzido e foram realizadas coletas e análises de tráfego de *Firewall* e *Intrusion Prevention Systems* operando em alta disponibilidade em ambientes de teste, devidamente estruturados no *Data Center* de uma empresa de telecomunicações [Lopes Filho 2008], [Hashimoto 2009]. As situações de acéfalo e cérebro bipartido ocorriam com a segunda sendo mais frequente. Nos dois casos as requisições de serviços não eram respondidas ou eram respondidas por mais de um equipamento, o que gerava uma sobrecarga impossível de ser gerenciada [Hashimoto et al. 2010].

O VRRP foi inicialmente lançado pela Cisco [Cisco 2012] e logo tornou-se padrão *de facto* para os equipamentos de alta disponibilidade. Em se tratando de alta

disponibilidade, há também a implementação *Common Address Redundancy Protocol* (CARP) [OpenBSD 2012], desenvolvida pela comunidade OpenBSD, e ainda os protocolos *Hot Standby Router Protocol* (HSRP) [Li et al. 1998] e *NetScreen Redundancy Protocol* (NSRP) [Kriens et al. 2006]. Segundo [Hashimoto 2009] o protocolo VRRP é mais utilizado que os outros e os problemas percebidos nele podem ser estendidos a todos os outros citados, em virtude de análise realizada sobre a máquina de estados de cada um deles.

Após estabelecer as causas da indisponibilidade e apontá-las como sendo problemas algorítmicos que atacam o protocolo, [Hashimoto et al. 2010] apresentou uma especificação abstrata, modelada em Redes de Petri e representada por um autômato para contornar as condições citadas. O autômato em [Hashimoto et al. 2010] não foi implementado e trata-se de uma descrição na mesma filosofia da especificação em Rede de Petri: a visão do comportamento do grupo de alta disponibilidade (elemento virtual).

Entretanto escapou-lhe a especificação do comportamento individual de cada um dos elementos físicos componentes do elemento virtual e, neste caso, fez-se necessária a especificação do autômato e dos outros quatro elementos de cada instância do protocolo [Holzmann 1991]. Foi necessário refinar a proposta de [Hashimoto et al. 2010] a fim de implementá-la e validar o funcionamento da extensão do VRRP.

Este trabalho apresenta as regras de comportamento para uma instância do HARP em um elemento físico, do ponto de vista da sua máquina de estados, o que não havia na proposta de [Hashimoto et al. 2010]. Elementos como parâmetros e condições para o processo de eleição a novo mestre, condições de entrada e saídas para transições entre estados de uma instância, especificação de serviço e análises dos casos de perdas de primitivas ainda eram necessários para uma implementação do protocolo. Esse processo de refinamento originou o HARP.

3.2. Desenvolvimento do *High Availability Router Protocol*

Nesta fase ocorreu efetivamente a especificação dos elementos do HARP, bem como a sua implementação em hardware. Os cinco elementos do HARP são apresentados detalhadamente na seção 4, enquanto as características de implementação são aqui tratadas.

A implementação em hardware permite a detecção de detalhes e correção de falhas que escapam ao projeto de protocolos em níveis elevados de abstração. No entanto, o custo de produção de um hardware específico é muito alto. Neste contexto, o uso de um dispositivo flexível faz-se necessário, sendo o FPGA o mais indicado para esta situação. O uso do FPGA permitiu que correções fossem feitas após cada prototipação do HARP, permitindo perceber erros do hardware e retornar à sua especificação para realizar as correções. Este processo de iterações pertimiu especificar o HARP e prototipar um hardware específico até que se chegasse à versão final. O uso de FPGA para implementar algoritmos de redes tem sido amplamente realizado por empresas e na academia, em virtude da sua adaptabilidade e flexibilidade, reduzindo o custo de produção e o *time-to-market* do projeto se comparados a um ASIC.

Uma plataforma foi montada para realizar a prova de conceito do HARP. Para tal, montou-se um esquema com três placas de prototipação DE2 simulando elementos de rede com conexões dedicadas entre si e em topologia estrela. Cada elemento de rede possui um FPGA Cyclone 2 [Altera 2006]. Os FPGAs foram conectados conforme a

figura 1. Cada FPGA recebe e envia dados a qualquer um dos outros através de conexão dedicada sobre um cabo *flat* de 36 vias conectado a um cabeçalho de expansão contido na placa de prototipagem. Cada solicitação de serviço é feita ao se pressionar um dos quatro botões de acionamento conectados ao FPGA. O meio comunicaçao foi divido em três canais (Figura 1) com doze vias cada um.

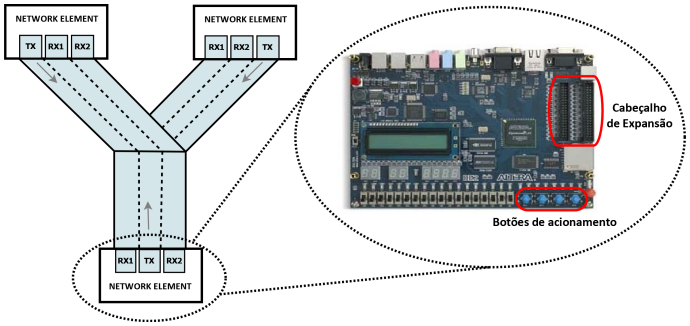


Figura 1. Esquema de conexão para a prova de conceito

Todos os serviços foram testados e, após uma sequência de reconfigurações do hardware, funcionaram corretamente. Dado que o HARP é baseado em *timeouts* (seção 4.4), como é o caso dos estados S2, S3, S4 e S6 da Figura 4, inserir atrasos entre estados e atender todos os possíveis casos de perdas de primitivas foram os principais benefícios da reconfigurabilidade do FPGA, que permitia revisar o HARP em cada implementação.

O HARP foi testado de forma que pudesse prever a perda de cada uma das primitivas reconhecidas no seu vocabulário, assim foi possível certificar de que o HARP conseguiria lidar com todas as situações de perdas. Esta análise foi feita paralelamente ao processo de implementação. Exemplos de situações de perdas são dadas na subseção 4.4.

Como dito anteriormente, a Figura 1 representa a prova de conceito dos elementos, que foi concluida com êxito. Entretanto, não trata do tempo de recuperação do sistema e nem traz dados sobre o tempo que cada primitiva leva para ser processada dentro de cada instância do HARP, isso será feito quando implementada a etapa de validação. A etapa de validação é independente da especificação do HARP e objetiva desenvolver um sistema capaz de testar qualquer protocolo de alta disponibilidade.

3.3. Sistema de Validação

A Figura 2 mostra o sistema de validação de protocolos de alta disponibilidade que desenvolvemos. A figura mostra um elemento virtual formada por três (ou mais) elementos de rede (FPGA) conectados entre si por uma via ethernet compartilhada. Esta mesma via está conectada a computadores pessoais responsáveis por gerar fluxo de informação.

O sistema é para testar protocolos de alta disponibilidade do ponto de vista das falhas de canal e interface de comunicação. Este é o principal caso gerador das condições de acéfalo e cérebro bipartido. Outro caso gerador é o ataque externo por invasores, o que não é tratado neste trabalho.

Cada um dos elementos de rede da Figura 2 é configurado na forma de um *System on Chip* (SoC) baseado no processador embarcado NIOS 2 [Altera 2012]. Um dos componentes do SoC é o hardware do protocolo de alta disponibilidade a ser testado. O

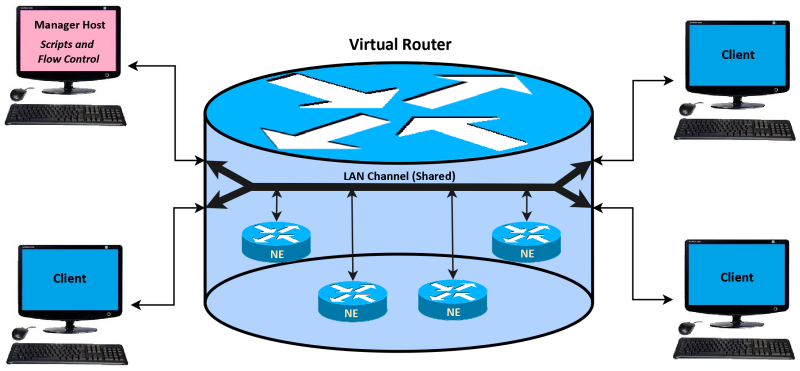


Figura 2. Sistema proposto para validação

sistema interage com uma aplicação em software que transmite dados via Ethernet ao FPGA através de um controlador operando na camada de enlace. Assim, o computador responsável por gerar o fluxo enviará mensagens aos outros computadores e o SoC as encaminhará.

Pode-se submeter qualquer protocolo de alta disponibilidade à validação, com implementação em hardware ou em software. Para implementações em hardware, inclui-se o protocolo como um módulo do SoC. Caso seja em software, a aplicação utliza o SoC apenas como plataforma. Uma camada de abstração de hardware foi criada para contornar os problemas de incompatibilidade do tamanho de primitivas na interface HW/SW entre diferentes protocolos.

4. HARP - Descrição dos Elementos

Os cinco elementos do protocolo são apresentados nesta seção, a saber: suposições sobre o ambiente, serviços, vocabulário, formatação e regras de procedimento. Por questões de espaço, os serviços e vocabulário serão apresentados em conjunto.

4.1. Suposições sobre ambiente

O HARP, como protocolo de alta disponibilidade, deve operar em elementos individuais de um grupo de equipamentos redundantes (Figura 2). Este grupo forma o elemento virtual e é visto pelo restante da rede como um único ponto de encaminhamento de pacotes. O HARP está projetado para trabalhar na camada três da arquitetura internet, juntamente com o protocolo IP.

4.2. Serviços e Vocabulário

Nesta subseção, tem-se a apresentação de todos os serviços oferecidos pelo HARP, de forma a contornar as condições de acéfalo e cérebro bipartido. As mensagens também são aqui introduzidas e é importante dizer que elas obedecem à taxonomia *Request/Indication* e *Response/Confirmation*. A Tabela 1 sumariza o conjunto dos serviços e mostra quais as mensagens são trocadas durante a execução de cada um deles. A seção 4.4 retoma esta discussão e detalha a execução de cada serviço, bem como a função de cada mensagem e seu devido uso.

Há ainda duas mensagens não citadas na Tabela 1, já que não são diretamente partes dos serviços, mas são apresentadas a seguir.

Tabela 1. Serviços do HARP

Nome	Mensagens	Descrição
Keep Alive (KA)	*KA Request (KA_REQ)*	Serviço não confirmado. Atua como *heartbeat* usado pelo mestre para avisar sua atividade. Monitorando estes *heartbeats*, os escravos do grupo de alta disponibilidade determinam quando um elemento mestre parou de funcionar.
Given Master (GM)	*GM Request (GM_REQ)* *GM Response (GM_RESP)* *GM Failed Request (GMFAIL_REQ)* *GM Ready Request (GMRDY_REQ)*	Serviço confirmado. Utilizado pelo mestre para indicar sua intenção de transferir o seu papel a um escravo específico. O endereço do escravo tem que constar na tabela de ativos do mestre.
Informe Node (INF)	*INF Request (INF_REQ)* *INF Response (INF_RESP)*	Serviço confirmado. Utilizado para indicar que um nó está se tornando membro do grupo de alta disponibilidade.
Remove Node (REM)	*REM Request (REM_REQ)* *REM Response (REM_RESP)*	Serviço confirmado. Utilizado por um escravo para indicar ao mestre do grupo de alta disponibilidade que ele está deixando a rede.
Check Brain (CB)	*CB Request (CB_REQ)* *CB Response Positive (CB_RESP+)* *CB Response Negative (CB_RESP-)*	Serviço confirmado. Utilizado por um nó escravo para se certificar de que não há mestre no grupo e evitar a situação do cérebro bipartido no processo de obtenção do novo mestre.

- *Active Slave Request* (ACTS_REQ): enviada pelo mestre, em *broadcast*, para atualizar a sua lista de escravos ativos;
- *Active Slave Response* (ACTS_REQ): enviada pelo escravo, ao mestre, para responder sua atividade.

4.3. Formato das Mensagens

Para atender às necessidades dos serviços apontados anteriormente e por operar na camada três da arquitetura Internet, a mensagem do HARP tem dez campos predefinidos para sua primeira versão, a operar com IPv4. As mensagens tem 128 bits de cabeçalho e não utilizam o campo de dados, que pode ser utilizado nas próximas versões e adaptações para Internet do futuro.

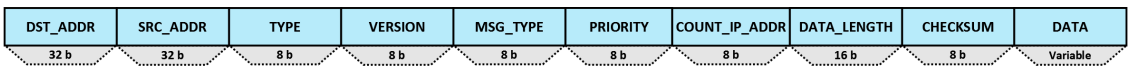


Figura 3. Formato da mensagem HARP

As mensagens HARP são essencialmente mensagens de controle, tendo por isso o cabeçalho maior que o campo de dados, em todos os casos. O formato das mensagens é ilustrado pela figura 3, que contém o identificador do campo e o seu comprimento em bits. O significado de cada campo pode ser conferido segundo a lista a seguir:

1. DEST_ADDR: IP de destino. Pode ser de um elemento ou *broadcast*;

2. SRC_ADDR: IP de origem. Especifica o elemento que enviou a mensagem;
3. TYPE: tipo do protocolo, i.e., número reservado à indentificação do HARP;
4. VERSION: versão do protocolo, para este caso será usada somente a vesão 1;
5. MSG_TYPE: indica qual o tipo de mensagem reconhecida pelo vocabulário do HARP que aquela primitiva está carregando;
6. PRIORITY: prioridade do nó emissor, atribuída pelo administrador de rede;
7. COUNT_IP_ADDR: em uma mensagem KA_REQ, indica o número de escravos ativos no grupo. Pode também indicar quantos endereços IP estão sendo carregados no campo DATA (em outras versões);
8. DATA_LENGTH: comprimento do campo de dados;
9. CHEKSUM: soma de verificação para detecção de erro;
10. DATA: campo de dados para parâmetros das mensagens, quando necessário.

Os endereços 0x0000 e 0xFFFF são reservados. O primeiro indica que não há endereço sendo transmitido, enquanto o segundo destina-se ao *broadast*. As próximas versões do HARP podem vir com mensagens de formatos diferentes para atender diferentes propostas de arquiteturas de redes.

4.4. Regras de Procedimentos

Cabe às regras de procedimentos explicar a dinâmica de troca de mensagens e o comportamento da máquina de estados (FSM) do HARP durante a execução dos procedimentos dos serviços. As condições de acéfalo e cérebro bipartido são evitadas ou contornadas segundo trocas de mensagens explicadas nesta seção. Para a condição de acéfalo, basicamente ela ocorrerá se o mestre atual sair de operação e é resolvida pelo processo Check Brain. Já a condição de cérebro bipartido pode ocorrer por diferendes situações de cadenciamento, que são exemplificadas nos itens subsequentes.

A Figura 4 representa a FSM do HARP e é a união de todos os autômatos parciais (por serviço), tanto para elementos emissores quanto para receptores. O restante desta seção explica as transições da FSM, ralacionando-as a cada serviço. O autômato é a descrição formal das transições explicadas e deve ser observado em cada item a seguir.

Keep Alive (KA): O HARP se inicia no estado *Idle* e baseado na prioridade cada nó irá de *Idle* a *Master* ou a *Slave*, tornando-se mestre ou escravo, respectivamente. Mestre tem prioridade zero, valores entre 1 e 255 são para escravos. Se um nó assume o papel de mestre ele inicia o serviço KA, enviando um KA_REQ a cada intervalo de tempo *t* para informar seu estado aos demais. O mestre utiliza o campo COUNT_IP_ADDR para informar o número de escravos ativos no grupo.

Inform Node (INF): Quando um nó escravo entra na rede, ele envia um INF_REQ ao grupo e aguarda um INF_CONF do mestre. O mestre deve incluí-lo em sua tabela de ativos. Um mestre não precisa enviar INF_REQ, a menos que ele transite ao estado de *Slave*.

Remove Node (REM): Um nó escravo envia ao mestre um REM_REQ para solicitar sua saída da rede. O mestre envia um REM_RESP ao solicitante autorizando a saídae a tualiza sua tabela. Recebendo confirmação, o nó pode deixar a rede.

Given Master (GM): Se houver necessidade de transferir a função de mestre, o processo GM é iniciado. Isso acontece quando é preciso realizar manutenção programada

de equipamentos, por exemplo. Para formalizar, o início do processo ocorre quando a *flag* GM_Start é setada no mestre, o que desencadeia o fluxo de mensagens do GM.

O mestre envia um GM_REQ a um escravo predeterminado e vai para o estado *Wait For Given Master Confirm* (WF_GM CONFIRM). Se ele recebe um GM_CONF, ele vai para o estado *Slave* e envia um GMRDY_REQ informando a conclusão do processo. Caso contrário, se ocorre o TO_1 [1] (intervalo *t*), ele envia o um GMFAIL_REQ informando erro no processo e volta ao estado *Master*.

Do lado do receptor, quando o escravo recebe o GM_IND, ele envia um GM_RESP e vai ao estado *Given Master Accepting* (GM ACCEPTING), esperando um intervalo para garantir que não há outro elemento de rede enviando mensagens *Keep Alive*. Se um GMRDY_IND é recebido, acontece a transição ao estado *Master*, mas se ocorre o TO_2, um KA_IND ou GMFAIL_IND, ele retorna ao estado *Slave*, evitando a condição de cérebro bipartido.

Dentro do contexto do GM, algumas situações são ilustradas para demonstrar a capacidade de recuperação do protocolo no caso de perdas de mensagens:

- Se a primitiva GM_REQ for perdida, então o escravo receptor não recebe o GM_IND e tampouco envia o GM_RESP, logo o mestre emissor deve voltar ao seu estado de origem antes que os outros escravos percebam a sua falta. Por isso, o TO_1 é menor que o TO_3 (*timeout* de não recebimento do KA_IND) para garantir que não haverá condição de acéfalo na rede.
- Já no caso de perda do GM_RESP, não haverá o envio da primitiva GMRDY_REQ, não permitindo que o receptor vá ao estado *Master*. O receptor, que estava no estado GM ACCEPTING, consequentemente voltará ao estado *Slave* pela ocorrência do TO_2[2], enquanto esperava o GMRDY_IND. Ele também deverá voltar ao estado *Slave* ao receber o GMFAIL_IND ou ao receber um KA_IND.
- Caso haja perda da primitiva GMFAIL_REQ não haverá problema, pois o TO_2 e o KA_IND contornam esta perda no estado *GM ACCEPTING*.
- Caso a primitiva GMRDY_REQ se perca no caminho, há a situação em que o mestre vai a *Slave* e o escravo continua em *Slave*, gerando a condição de acéfalo. Este é o pior caso quando se trata de perda de mensagens no processo GM, mas caso isso ocorra, o serviço *Check Brain* é iniciado automaticamente por algum escravo e um novo mestre será eleito.

Check Brain (CB): Se o escravo detecta a falta do mestre, o processo para eleger o novo mestre é iniciado. Este é um ponto crucial da especificação, pois garante a não ocorrência da condição de cérebro bipartido e resolve diretamente a condição de acéfalo. A *Check Brain Flag* (CB_Flag) é usada para inibir que o processo CB seja iniciado por algum escravo quando já tiver sido iniciado e não acabado por outro.

Um nó escravo percebe que há um intervalo de tempo TO_3[3] sem o recebimento

[1] TO_n representa *timeout* e é utilizado nos estados com tempo de permanência máximo pré-estabelecido

[2] O TO_2 e o TO_1 sao menores que o TO_3 e ambos valem *t*

[3] O TO_3 é o tempo necessário para entender que há falha do mestre e baseia-se no não recebimento de KA_IND. $TO_3 = (2 + a) \star t$, onde *a* é uma constante baseada na prioridade. Isso duminui a probabilidade de a falha do mestre ser percebida no mesmo momento por vários escravos. O TO_3 é maior que todos os outros TO's da FSM, garatindo que o Check Brain não seja iniciado durante a execução de um outro serviço.

do KA_IND e percebe ainda que a *flag* CB_flag está resetada, certificando-se de que o processo CB ainda não foi iniciado. O nó envia então um CB_REQ em *broadcast* para os outros escravos e vai ao estado *Wait For Check Brain Confirm* (WF_CB CONFIRM), para aguardar pela confirmação de outros escravos sobre a não existência do mestre na rede.

Caso o escravo *sender* receba uma confirmação positiva CB_CONF(+) ele volta ao estado *Slave*, resetando o contador do TO_3, que é naturalmente resetado a cada recebimento do KA_IND. O retorno ao estado *Slave* pode ocorrer também se houver o recebimento de um KA_IND (indicando que há mestre) ou caso nenhuma mensagem chegue e ocorra o TO_4[4].

Chegando um CB_CONF(-), entende-se que outro elemento escravo também não está recebendo mensagens do mestre. O emissor vai então ao estado *Master Election* para definitivamente certificar-se de que não há mestre na rede. Ele espera receber um total *b* de mensagens CB_CONF(-) proporcional ao número total de escravos no grupo. O valor *b* é dado pelo teto de $b = \frac{totaldeescravos}{2}$.

Se ele recebe as confirmações negativas esperadas, ele vai ao estado *Master* e envia KA_REQ aos outros escravos da rede. Ele deve então zerar o CB_flag e o contador *k* de CB_CONF (que espera atingir o valor *b*). Ao receber um KA_IND os nós receptores zeram todos os seus contadores e a CB_flag, continuando escravos e reconhecendo o novo mestre.

Do lado dos receptores, quando um CB_IND é recebido, ele armazena o endereço de origem e seta sua CB_flag, indicando que o processo CB foi iniciado e está em andamento. Então, ele vai para o estado *Search Master* para conferir quanto tempo decorreu desde o último KA_IND recebido. Caso haja no máximo *2t* ele envia resposta CB_RESP(+) e volta para o estado de *Slave*, pois para ele tudo está operando dentro da normalidade e certamente houve uma falha pontual do emissor.

Entretanto, se o receptor perceber no estado *Search Master* que não houve recebimento de KA_IND no último intervalo *2t*, ele envia um CB_RESP(-) atestando a falta do mestre e vai ao estado *Master Election* para atrasar sua volta e dar tempo para o emissor concluir o processo. Ele só deixa o estado *Master Election* para ir ao *Slave*, o que ocorre pelo recebimento de um KA_IND ou pelo TO_5, quando ele reseta seu CB_flag. O TO_5 é igual a *t* para emissor e *2t* para receptor.

Cabe destacar que apesar de o intervalo necessário para iniciar um CB seja TO_3, o intervalo *2t* já é suficiente para gerar respostas CB_RESP. Isso porque o escravo de menor TO_3 poderia perceber a falha e os outros ainda não terem atingido seus *timeouts* TO_3 (por causa da constante *a*) e deixariam o emissor sem resposta atualizada.

O uso de um TO_3 variável minimiza o risco de mais de um escravo iniciar o CB ao mesmo tempo. Ainda que isso ocorra, uma das mensagens atingirá o canal antes da outra, baseado no princípio de compartilhamento de canal (CSMA/CD). É a esta mensagem que os escravos responderão, pois eles ignorarão mensagens subsequentes por já terem setado a CB_flag. Este processo faz com que um dos escravos emissores volte ao estado *Slave*

[4]O TO_4 tem o valor de *t*. Quando ele ocorre o escravo volta ao seu estado inicial, resetando o TO_3 e a *flag* CB_flag, daí recomeça a esperar o KA_IND novamente e, depois disso, pode repetir todo o processo CB, caso necessário. Este procedimento garante que o escravo permanece escravo se, por exemplo, houver falha somente na interface dele, seja ela permanente ou temporária

quando ocorrer o TO_4 no estado WF_CB CONFIRM.

Para evitar mais uma vez a codição de cérebro bipartido, deve-se prever a situação em que o mestre fica indisponível um tempo suficientemente grande para a eleição de um novo mestre e depois disso volta a operar normalmente, deixando dois mestres no grupo. Para isso, ao receber um KA_IND os mestres envolvidos devem parar de enviar KA_REQ e irem ao estado de *Slave*. Isso evita o cérebro bipartido e, no máximo, gera a condição de acéfalo, que é resolvível com o processo *Check Brain*.

Ao se concluir um processo de *Given Master* ou *Check Brain*, o novo mestre envia um ACTS_REQ ao grupo para atualizar a sua tabela de ativos e, consequentemente, atualizar o grupo de quantos escravos estão inseridos. A tabela de ativos é sempre atualizada com o recebimento de um ACTS_CONF, INF_IND ou REM_IND.

Todos estes procedimentos descritos são executados por qualquer instância do protocolo e a qualquer tempo. Portanto, as regras são combinados e descritas formalmente em um único autômato capaz de gerenciar a comunicação.

4.5. Autômato Final

Combinando os autômatos dos serviços, é possível gerar uma FSM que expressa o comportamento de uma instância do HARP. A Figura 4 mostra todos os eventos e ações decorrentes disso. A descrição de todos os estados que compõem a FSM da figura já foram vistos na seção 4.4, bem como os eventos reconhecíveis para as transições entre eles. Esta FSM descreve o comportamento do HARP para qualquer um dos serviços oferecidos.

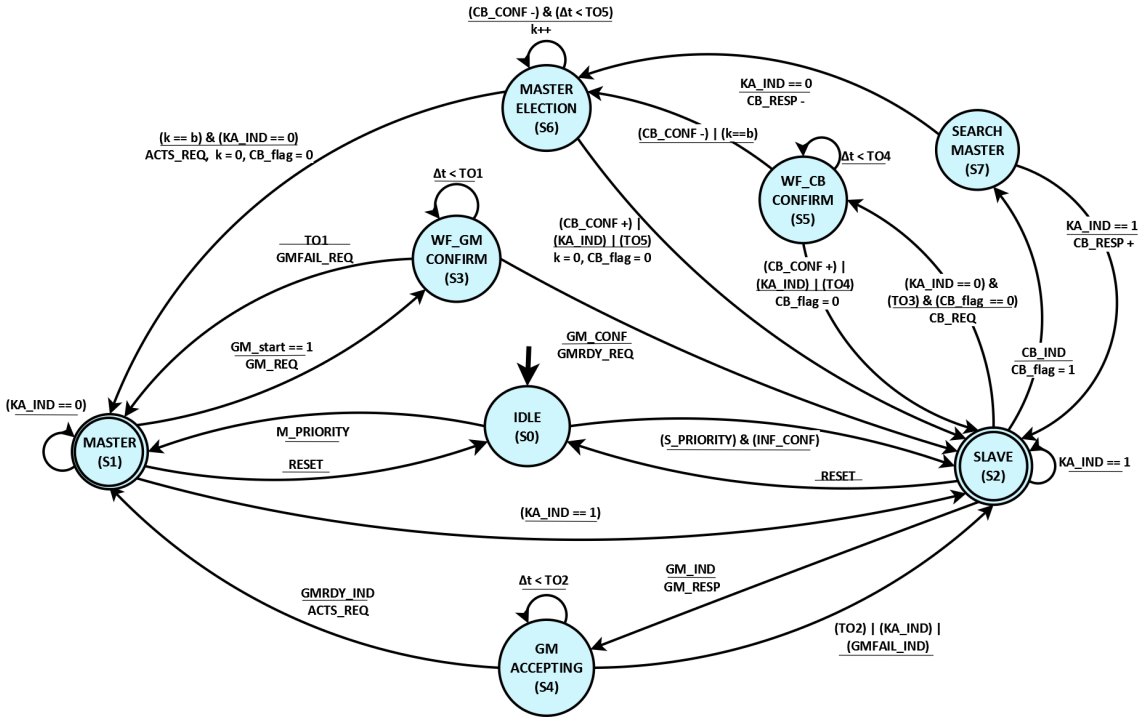


Figura 4. Autômato do HARP

Cabe salientar que o autômato apresentado na Figura 4 é bem formado e mantém as boas propriedades, a saber: é *k*-limitado, garantindo que a FSM é finita; não tem estados

sem sucessores, ou seja, livre de *deadlock*; é livre de *livelock* e há sempre uma sequência que leva ao estado inicial. Então, a modelagem foi feita no intuito de garantir a descrição completa dos cinco elementos e a demonstração das boas propriedades.

5. Considerações Finais

A principal contribuição deste artigo é apresentar o protocolo *High Availability Router Protocol* e como ele corrige os problemas algorítmicos que afetam os protocolos de alta disponibilidade existentes. Para a concepção do mesmo, partiu-se de uma proposta especificada em alto nível de abstração [Hashimoto et al. 2010], modelada em Rede de Petri e chegou-se a uma especificação completa e de mais baixo nível de abstração para o protocolo HARP, não deixando escapar questões necessárias à implemenção.

Os dois principais problemas que atacam a alta disponibilidade são as condições de acéfalo e cérebro bipartido. Estes problemas são resolvidos no HARP, considerando que o novo autômato provê estados e transições suficientes para cobrir estas possibilidades.

Um importante ponto mostrado ao longo do texto e principalmente ao se analisar o autômato do HARP é que as boas propriedades de uma FSM são mantidas: o HARP é k-limitado, livre de *livelock*, livre de *deadlock* e é reinicializável de qualquer estado. O autômato final tem oito estados e as transiçoes são controladas por um vocabulário bem definido.

A implementação em FPGA permitiu realizar atualizações com baixo custo, em virtude da sua reconfigurabilidade. Assim, foi possível realizar as mudanças necessárias após cada implementação até chegar a uma versão definitiva. Cabe lembrar que os parâmetros de tempo e desempenho não formam medidos nesta etapa, que foi destinada à demonstração de funcionamento do protocolo.

Este artigo apresenta resultados de um novo protocolo de rede. Tais resultados se concretizaram após um processo iterativo de proposições e testes em hardware reconfigurável. A partir de agora, podemos encaminhar o desenvolvimento de um módulo de hardware específico para tratar alta disponibilidade de rede e vislumbrar o engajamento deste módulo nas novas propostas de arquiteturas para Internet.

Referências

Altera (2006). De2 development and education board user manual. http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html.

Altera (2012). Nios 2 processor. http://www.altera.com/devices/processor/nios2/ni2-index.html.

Brown, S. and Rose, J. (1996). Fpga and cpld architectures: a tutorial. *Design Test of Computers, IEEE*, 13(2):42 –57.

Casado, M., Koponen, T., Moon, D., and Shenker, S. (2009). Rethinking packet forwarding hardware. In *ACM Workshops on Hot Topics in Networks*, volume VII, pages 1 –6.

Cisco, S. (2012). High availability. http://www.cisco.com.

EDOBRA (2012). Extending and deploying ofelia in brazil. http://www.mehar.facom.ufu.br/projects/ofelia-edobra.dot.

Hashimoto, G., Filho, E., Pereira, J., and Rosa, P. (2010). High avaliability: A long-term feature in network elements. In *Systems and Networks Communications (ICSNC), 2010 Fifth International Conference on*, pages 201 –206.

Hashimoto, G. T. (2009). Uma proposta de extensão para um protocolo para arquiteturas de alta disponibilidade. Dissertação de mestrado, Universidade Federal de Uberlândia.

Hinden, R. (2004). Virtual Router Redundancy Protocol (VRRP). RFC 3768 (Draft Standard).

Holzmann, G. J. (1991). *Design and validation of computer protocols*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

ITU (2012). Internet users. http://www.itu.int/ITU-D/ict/statistics/index.html.

Jiang, W. and Prasanna, V. (2012). Scalable packet classification on fpga. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(9):1668 –1680.

Johnson, B. W., editor (1988). *Design & analysis of fault tolerant digital systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Kriens, S., Cameron, R., Woodberg, B., and Madwachar, M. K. (2006). *Configuring Juniper Networks NetScreen & SSG Firewalls*. Syngress Publishing.

Li, T., Cole, B., Morton, P., and Li, D. (1998). Cisco Hot Standby Router Protocol (HSRP). RFC 2281 (Draft Standard).

Lopes Filho, E. (2008). Arquitetura de alta disponibilidade para firewall e ips baseada em sctp. Dissertação de mestrado, Universidade Federal de Uberlândia.

MEHAR (2012). Mondial entities horizontally addressed by requirements. http://www.mehar.facom.ufu.br/.

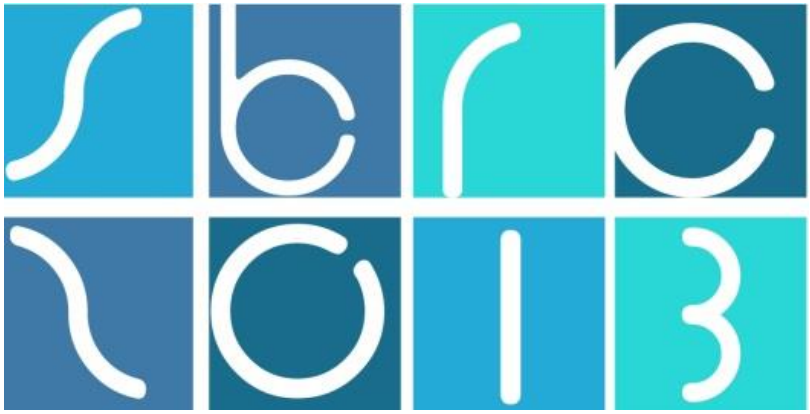
OpenBSD (2012). Pf: Firewall redundancy with carp and pfsync. http://www.openbsd.org/faq/pf/carp.html.

Pereira Júnior, J. E. (2010). Especificação de serviço e suposiçoes sobre o ambiente para um protcolo de alta disponibilidade. Dissertação de mestrado, Universidade Federal de Uberlândia.

Sonderegger, J., Blomberg, O., Milne, K., and Palislamovic, S. (2009). *Junos High Availability: Best Practices for High Network Uptime*. O'Reilly Media, Inc., 1st edition.

Straka, M. and Kotasek, Z. (2009). High availability fault tolerant architectures implemented into fpgas. In *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*, pages 108 –115.

Technologies, C. (2010). North american businesses lose $26.5 billion annually from avoidable downtime, according to new ca technologies study. http://www.ca.com/us/news/Press-Releases/na/2010/North-American-Businesses-Lose-26-5-Billion- Annually.aspx.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 4

Roteamento

Um novo Algoritmo de Roteamento para a Escolha da Melhor Entre as Menores Rotas

Iallen Gábio S. Santos[1], Gilvan Durães[2], William Giozza[3], André Soares[1]

[1]Departamento de Computação – Universidade Federal do Piauí (UFPI)
Teresina – PI – Brasil

[2]Laboratório de Computação Aplicada –Instituto Federal Baiano (IFBaiano)
Catu, BA – Brasil

[3]Departamento de Engenharia Elétrica (ENE) – Universidade de Brasília (UnB)
Brasília – DF – Brasil.

andre.soares@ufpi.edu.br

Abstract. *In all-optical networks the choice of route and wavelength is called Routing and Wavelength Assignment (RWA) problem. The majority of work on literature uses the Dijkstra algorithm with routing algorithm. However, a given pair of source-destination can have more than one shortest path. In this context, a solution of the 3MC problem try to choose the best of the shortest paths for a given network topology. In this paper is proposed a novel routing algorithm that aim to solve 3MC problem. Our algorithm achieved better performance than MMR and Dijkstra algorithms for all studies scenarios.*

Resumo. *No contexto de redes ópticas transparentes, a escolha de uma rota e de um comprimento de onda caracterizam o problema Routing and Wavelength Assignment (RWA). A maioria dos trabalhos na literatura utiliza algoritmos de menor caminho para o roteamento como o algoritmo de Djikstra. Entretanto, entre cada par origem destino de uma topologia pode existir mais de uma rota de menor caminho. Neste contexto o problema da escolha da Melhor Combinação entre as M Combinações de Menores Caminhos (3MC) visa à escolha da combinação de rotas de menor caminho que minimize a probabilidade de bloqueio em uma dada topologia. Neste artigo é proposto um novo algoritmo de roteamento para redes ópticas transparentes que visa solucionar o problema 3MC. O algoritmo proposto apresentou desempenho superior ao algoritmo Melhor entre as Menores Rotas (MMR) e ao algoritmo de Dijkstra (DJK) para todos os cenários estudados.*

1. Introdução

Com o aumento da demanda de tráfego e a exigência cada vez maior de QoS por parte de novas aplicações, a tecnologia de redes ópticas WDM é apontada como a principal alternativa para compor as redes de transportes.

Para o estabelecimento de um circuito óptico em uma rede WDM é necessário a escolha de uma rota e a alocação de um comprimento de onda. Em uma rede submetida a um tráfego dinâmico, a solução RWA se concentra em atender as solicitações de circuitos, minimizando a probabilidade de bloqueio das requisições futuras.

A maioria dos trabalhos na literatura tratam o problema RWA em duas fases, separando o problema do roteamento do problema de alocação de comprimento de onda. Diferentes algoritmos de roteamento e de alocação de comprimento de onda vêm sendo propostos na literatura para resolver esse problema [Durães et al. 2010] [Durães et al. 2009] [Lin et al. 2006] [Zang and Jue 2000] .

As estratégias de roteamento podem ser classificadas em três classes: a) roteamento fixo, b) roteamento fixo alternativo e c) roteamento adaptativo [Lin et al. 2006]. Os algoritmos da classe de roteamento fixo estabelecem, previamente, uma rota para cada par de nós origem e destido ($par(o, d)$) da rede. Nos algoritmos da classe de roteamento fixo alternativo, cada $par(o, d)$ possui mais de uma rota prevista podendo alternar entre elas já na fase de operação da rede. Os algoritmos da classe de roteamento adaptativo definem as rotas para cada $par(o, d)$ sob demanda de acordo com o estado da rede.

Em [Durães et al. 2009] os autores definem o problema da escolha da Melhor Combinação entre as M Combinações de Menores Caminhos (3MC) para algoritmos de roteamento fixo. Ainda em [Durães et al. 2009] os autores propõem o algoritmo Melhor Entre as Menores Rotas (MMR) como uma solução para o problema 3MC.

Este artigo propõe um novo algoritmo para a solução do problema 3MC, chamado Melhor Entre as Menores Rotas com Decisão por Similaridade (MMRDS). O algoritmo proposto é comparado com a solução MMR em termos de probabilidade de bloqueio, justiça e tempo de execução. O MMRDS apresentou uma melhora significativa em relação ao MMR nos cenários estudados.

As demais seções deste artigo estão organizadas da seguinte forma. A Seção 2 discute os trabalhos relacionados e apresenta as contribuições deste artigo. A Seção 3 descreve o problema 3MC. O algoritmo MMRDS é apresentado na Seção 4. Um estudo de avaliação que compara o desempenho dos algoritmos de Dijkstra (DJK), MMR e MMRDS em termos de probabilidade de bloqueio e justiça é realizado na Seção 5. Por fim, as conclusões são apresentadas na Seção 6.

2. Trabalhos Relacionados e Contribuições

O problema RWA pode ser considerado um dos grandes desafios no contexto das redes ópticas transparentes. Diferentes trabalhos tem estudado e proposto soluções de roteamento e alocação de comprimento de onda a fim de reduzir as taxas de probabilidade de bloqueio [Murthy and Gurusamy 2002][Chu et al. 2004].

Em [Birman 1996] é apresentado um algoritmo de roteamento adaptativo chamado *Least Loaded Routing* (LLR) que escolhe a rota que possui mais comprimentos de onda disponíveis em todos os enlaces. Em [Lin et al. 2006] é proposto um algoritmo de roteamento fixo alternativo para redes ópticas transparentes sem conversão de comprimento de onda. Nesse algoritmo é feita uma listagem de rotas disjuntas para cada par origem destino de acordo com informações da rede, a escolha da rota é feita priorizando as rotas com menor índice na lista construída.

Em [Zang and Jue 2000] foi realizado um estudo envolvendo diferentes abordagens para a resolução do problema RWA, este estudo destacou as diferentes classes de algoritmos de roteamento, os algoritmos de Dijkstra (DJK) e de Bellman-Ford foram classificados como algoritmos da classe de roteamento fixo.

Em [Durães et al. 2009] foi proposto um algoritmo da classe de roteamento fixo chamado Melhor entre as Menores Rotas (MMR). Através de simulações da rede este algoritmo busca identificar os enlaces mais sobrecarregados e distribuir a carga mantendo rotas com menor quantidade de saltos. O MMR apresentou um resultado de desempenho significativamente superior quando comparado aos demais algoritmos de roteamento avaliados no mesmo trabalho. Em [Durães et al. 2010] os autores compararam o MMR com o algoritmo LLR e mostraram o melhor desempenho do algoritmo MMR.

Os algoritmos da classe de roteamento fixo possuem menor complexidade, uma vez que as rotas são definidas previamente sem a necessidade da obtenção de informações do estado da rede. Por isso, um número significativo dos trabalhos que abordam o problema RWA em redes ópticas transparentes se baseia nesta classe de algoritmos [Durães et al. 2009].

A principal contribuição deste artigo é a proposta do algoritmo Melhor Entre as Menores Rotas com Decisão por Similaridade MMRDS. O MMRDS é uma solução que verifica a similaridade entre as rotas de menor caminho com o objetivo de realizar um melhor balanceamento de carga. Essa característica no MMRDS resultou em um melhor desempenho do MMRDS em relação ao MMR em termos de probabilidade de bloqueio e justiça, utilizando três topologias diferentes: EON, USA e Abilene. Além disso, foi realizado um estudo de avaliação de desempenho em relação ao tempo de execução para mostrar a viabilidade computacional do algoritmo proposto. Por último, é importante destacar que os resultados obtidos para os cenários com conversão total ficaram próximos aos resultados do modelo analítico [Chu et al. 2004], o que sugere uma validação dos experimentos de simulação.

3. Problema 3MC

Dada uma topologia de rede óptica com N nós, o número de pares de nós (origem, destino) é $N * (N - 1)$. Será utilizada a notação $par(o, d)$ para representar um par ordenado de nós com origem no nó o e destino no nó d.

Para fazer o planejamento de uma estratégia de roteamento fixo é necessário definir uma rota para cada $par(o, d)$. Dessa forma, são requeridas $R = N * (N - 1)$ rotas para uma topologia com N nós. A escolha dessas rotas representa uma solução de rota para esta topologia.

Para cada $par(o, d)$ podem existir mais de uma rota de menor caminho. Neste artigo estas rotas serão chamadas de Rotas Candidatas, o conjunto de rotas candidatas para um $par(o, d)$ será denotado por $RC_{par(o,d)}$ e a rota escolhida para este par será denotada por $r_{par(o,d)}$.

Existem M soluções diferentes para o planejamento das rotas fixas em uma determinada topologia de rede. Caso sejam consideradas apenas rotas de menor caminho, o cálculo de M, que representa o número de soluções possíveis, é dado pela Equação 1.

$$M = \Pi_{i=1,j=1}^{N,N} |RC_{par(i,j)}| \tag{1}$$

Em que $|RC_{par(o,d)}|$ é o número de rotas candidatas ao $par(i, j)$, com $i \neq j$. Note que todas as rotas candidatas possuem o menor número de saltos.

O problema 3MC consiste em identificar uma solução de rotas de menor caminho S_k com $1 \leq k \leq M$, tal que a combinação S_k resulte em um melhor desempenho em termos de probabilidade de bloqueio da rede.

4. Descrição do Algoritmo MMRDS

Esta seção apresenta o algoritmo MMRDS, proposto para encontrar uma solução para o problema 3MC.

O algoritmo MMR proposto em [Durães et al. 2009], funciona de forma iterativa, em que para cada iteração o algoritmo DJK é executado para encontrar uma solução de rota. Ao fim de cada iteração uma simulação é realizada para obter a probabilidade de bloqueio da rede e a utilização de cada enlace. Esses valores de utilização dos enlaces são utilizados como base para a alteração dos custos de cada enlace na próxima iteração. Com isso o algoritmo DJK poderá selecionar uma solução de rota diferente na iteração posterior.

Diferentemente do MMR, o MMRDS trabalha de forma não iterativa. O MMRDS escolhe uma rota de menor caminho para cada $par(o, d)$ de forma sucessiva, a escolha para cada $par(o, d)$ leva em consideração todas as escolhas feitas para $pares(o, d)$ anteriores, além disso, os pares que possuem maior similaridade(conceito detalhado a seguir) entre as rotas candidatas são sempre avaliados primeiro. Isso é realizado na tentativa de alcançar um melhor balanceamento da frequência de uso de cada enlace, isto é, o número de rotas que utilizam um mesmo enlace da topologia.

O MMRDS utiliza o conceito de similaridade entre rotas. A análise da similaridade é feita observando as rotas duas a duas. O cálculo da similaridade entre duas rotas a e b, rotas candidatas para um dado $par(o, d)$ é feito através da Equação 2.

$$Sml(a, b) = \frac{NE_{comum}(a, b)}{H}, \tag{2}$$

Em que H é o número de enlaces da rota a e $NE_{comum}(a, b)$ é o número de enlaces em comum entre as rotas a e b. Vale destacar que o número de saltos da rota a é igual ao da rota b.

Partindo do conceito de similaridade entre duas rotas, a medida de similaridade entre todas as rotas candidatas de um dado $par(o, d)$ é dada pela Equação 3.

$$Sml_{par(o,d)} = \frac{\gamma}{C^2_{|RC_{par(o,d)}|}} \tag{3}$$

Em que γ é o somatório da similaridade ($Sml(a, b)$) de todas as combinações de rotas candidatas do $par(o, d)$ tomadas duas a duas. $C^2_{|RC|}$ é o número de combinações de rotas candidatas do $par(o, d)$ tomadas duas a duas.

A Figura 1 ilustra o cálculo da similaridade entre as rotas candidatas para o $par(1, 4)$.

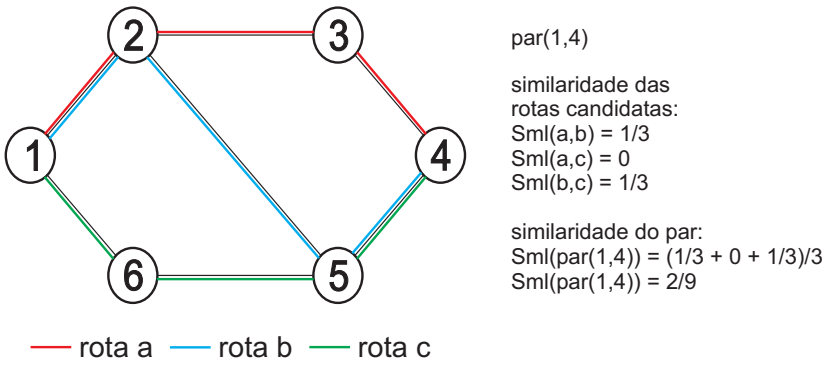


Figura 1. Exemplo do cálculo de similaridade.

Como pode ser observado na Figura 1, o $par(1,4)$ possui um conjunto composto de três alternativas de rotas de menor caminho. São elas: *rota a* = nó 1; nó 2; nó 3; nó 4, *rota b* = nó 1; nó 2; nó 5; nó 4 e *rota c* = nó 1; nó 6; nó 5; nó 4. No exemplo da Figura 1, a $Sml(a,b) = 1/3$. Isto porque as *rotas a* e *b* possuem apenas um enlace em comum (do nó 1 para o nó 2) e ambas possuem três saltos. $Sml(a,c) = 0/3$, pois as rotas não possuem enlaces em comum. $Sml(b,c) = 1/3$, pois as *rotas b* e *c* possuem apenas um enlace em comum (do nó 5 para o nó 4). Assim, para o exemplo da Figura 1 temos $Sml_{par(1,4)} = ((1/3) + (0/3) + (1/3))/3 = 2/9$.

O MMRDS pode ser dividido em duas etapas. Na primeira etapa é feita uma análise do nível de similaridade entre as rotas candidatas de cada $par(o,d)$. Na segunda etapa, cada $par(o,d)$ é tomado em ordem decrescente de similaridade, escolhendo uma das rotas candidatas ao $par(o,d)$.

A escolha da rota deve minimizar $\sum e_i$, em que e_i é o custo de cada enlace da rota. Em seguida é feito o incremento de uma unidade no custo de cada um dos enlaces pertencentes à rota escolhida.

Note que na escolha da rota para o primeiro $par(o,d)$, qualquer uma das rotas poderá ser escolhida. Isto pode acontecer porque todas as rotas tem a mesma quantidade de saltos e todos os enlaces iniciam com o mesmo custo.

A seguir é apresentado o Algoritmo 1, que detalha o funcionamento do MMRDS, considerando que as rotas candidatas para cada par de nós origem destino ($RC_{par(o,d)}$) já foram calculadas. O conjunto P contém os $pares(o,d)$ referentes à todas as combinações de nós de origem e destino. Por simplificação, a notação p representa um dado $par(o,d) \in P$.

Em uma determinada solução de rota, a frequência de uso de cada enlace pode ser definida pela quantidade de rotas que passam por este enlace. O MMRDS busca uma solução de rota na tentativa de encontrar o melhor balanceamento em termos de frequência de uso dos enlaces, evitando assim a formação de enlaces gargalos na rede.

Ao contrário do MMR, o MMRDS não é baseado em iterações e não necessita

Algoritmo 1 MMRDS.

inicializar o custo de todos os enlaces da topologia com valor 1;
while $P \neq \phi$ **do**
  encontrar $p \in P$ tal que a $Sml(p)$ seja máxima;
  encontrar $r_p \in RC_p$ tal que o somatório do custo dos enlaces de r_p seja mínimo;
  incrementar em uma unidade cada enlace pertencente a r_p;
  remover p de P;
end while

realizar simulações para obter a probabilidade de bloqueio com o objetivo de escolher a solução de rota. A execução do MMR é dependente do tempo das simulações para obtenção dos valores de utilização dos enlaces e da probabilidade de bloqueio.

5. Avaliação de Desempenho

O estudo de avaliação de desempenho apresentado nesta seção foi realizada com o auxílio da ferramenta de simulação TONetS [Soares et al. 2008].

A demanda de tráfego é composta por requisições de circuitos ópticos representados por $par(o, d)$. A carga de tráfego é distribuída uniformemente entre todos os $pares(o, d)$. A geração de requisições é um processo poissoniano de taxa média λ 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 40 comprimentos de onda em cada sentido. O algoritmo First-Fit é utilizado na alocação dos comprimentos de onda. Para cada simulação são realizadas 10 replicações com diferentes sementes de geração de variável aleatória. São geradas 100.000 requisições para cada replicação.

Os gráficos apresentam os intervalos de confiança calculados com um nível de confiança de 95%. Apesar de inseridos em todos os gráficos, os intervalos de confiança são imperceptíveis para alguns pontos, pelo fato do erro ser próximo de zero.

A Figura 2 ilustra as topologias das redes Abilene, EON e USA utilizadas nos estudos de avaliação de desempenho.

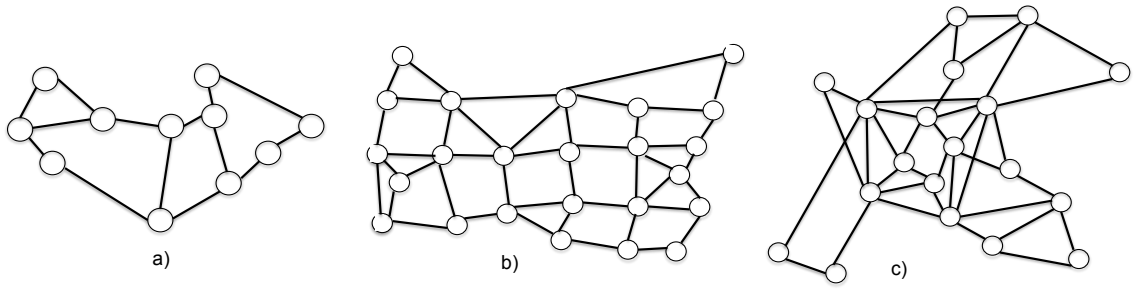


Figura 2. a) Topologia da rede Abilene. b) Topologia da rede USA. c) Topologia da rede EON.

As seções 5.1, 5.2 e 5.3 a seguir apresentam estudos de avaliação de desempenho considerando, respectivamente, a probabilidade de bloqueio, a justiça no atendimento das requisições dos diferentes $pares(o, d)$ e por último uma avaliação em termos do tempo de execução do algoritmo.

5.1. Probabilidade de bloqueio

Nesta seção, além dos resultados obtidos através de simulação utilizando a ferramenta TONetS, realizou-se também o cálculo de probabilidade de bloqueio para as três topologias estudadas com capacidade de conversão total. Para este cálculo foi utilizado o modelo analítico apresentado em [Chu et al. 2004], proposto para cenários de conversão total. Os gráficos expressos por curvas tracejadas representam os resultados obtidos com o modelo analítico. Apenas para esta seção, os resultados obtidos através de simulação são ilustrados exclusivamente através de pontos. Vale destacar que para o cenário com conversão total observou-se uma proximidade entre os resultados obtidos através de simulação e de modelagem analítica.

A Figura 3 ilustra o desempenho dos algoritmos DJK, MMR e MMRDS quando submetidos à topologia da rede Abilene com e sem conversão de comprimento de onda.

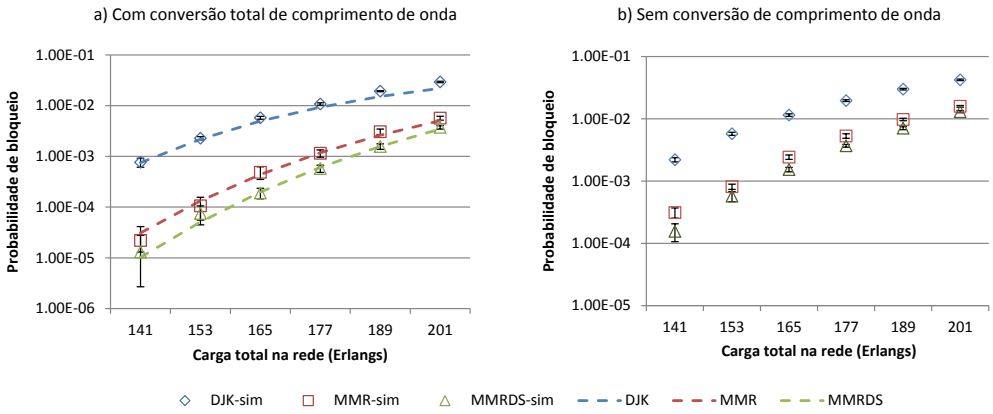


Figura 3. Probabilidade de bloqueio com e sem conversão total de comprimento de onda na topologia Abilene.

Para os cenários com e sem conversão o DJK apresentou desempenho bem inferior em relação aos algoritmos MMR e MMRDS, conforme ilustrado na Figura 3. Por isso, as próximas comparações serão restritas apenas aos algoritmos MMR e MMRDS. Para a topologia da rede Abilene, os algoritmos MMR e MMRDS apresentaram desempenho próximos, mas de maneira geral, o algoritmo MMRDS obteve a menor probabilidade de bloqueio para os dois cenários estudados, com e sem conversão de comprimento de onda.

As Figuras 4 e 5 apresentam uma comparação em termos de probabilidade de bloqueio entre os algoritmos MMR e MMRDS para as topologia EON e USA, respectivamente.

Conforme apresentado nas Figuras 3, 4 e 5, o algoritmo MMRDS obteve o melhor desempenho em termos probabilidade de bloqueio tanto no cenário com conversão total de comprimento de onda quanto no cenário sem conversão de comprimento de onda para as três topologias estudadas.

O ganho relativo do MMRDS em relação ao MMR em termos de probabilidade de bloqueio pode ser calculado segundo a Equação 4.

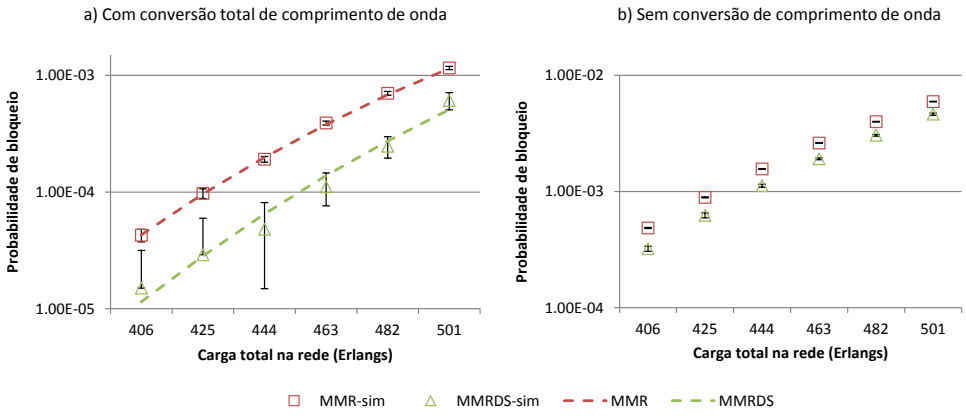


Figura 4. Probabilidade de bloqueio com e sem conversão total de comprimento de onda na topologia EON.

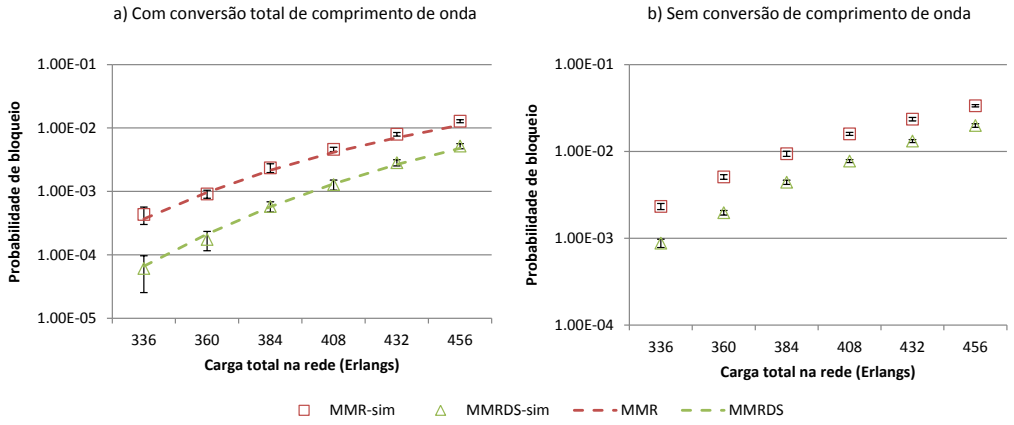


Figura 5. Probabilidade de bloqueio com e sem conversão total de comprimento de onda na topologias USA.

$$G = (PB_{MMR} - PB_{MMRDS})/PB_{MMR} \quad (4)$$

Os valores PB_{MMR} e PB_{MMRDS} são as probabilidades de bloqueio obtidas com os algoritmos MMR e MMRDS respectivamente. As Figuras 6a e 6b mostram o ganho relativo do MMRDS em relação ao MMR nas topologias Abilene, EON e USA, considerando cenários com e sem conversão total de comprimento de onda em termos de probabilidade de bloqueio.

As Figuras 6a e 6b evidenciam o ganho de desempenho do algoritmo MMRDS em relação ao algoritmo MMR considerando a probabilidade de bloqueio. Essas análises são feitas considerando a probabilidade de bloqueio obtida para os maiores valores de carga na rede observados nas Figuras 3, 4 e 5.

Observa-se que no cenário com conversão total o ganho relativo do MMRDS sobre

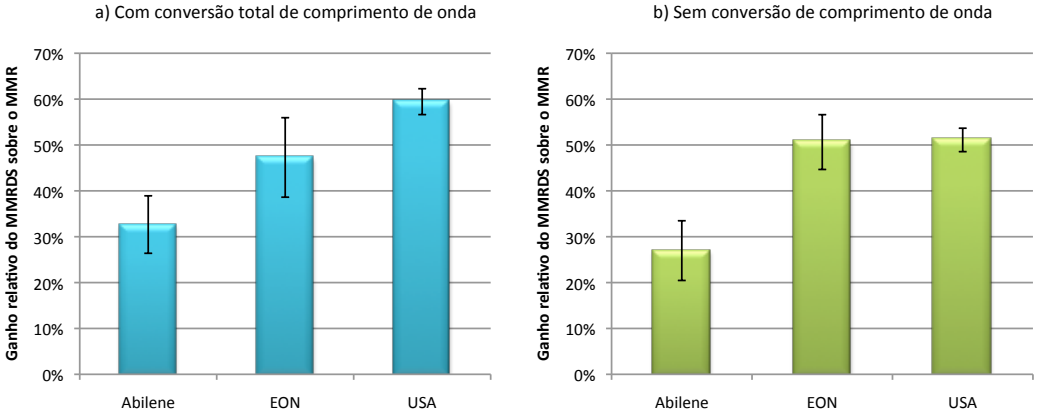


Figura 6. Ganho do MMRDS em relação ao MMR nos cenários sem e com conversão total de comprimento de onda.

o MMR é maior quando comparado aos ganhos obtidos no cenário sem conversão de comprimento de onda. O MMRDS obteve ganho médio próximo a 30% na topologia da rede Abilene e ganho médio de aproximadamente 50% para as topologias das redes EON e USA.

5.2. Justiça

Esta seção avalia o desempenho dos algoritmos DJK, MMR e MMRDS em termos da justiça no atendimento de requisições de circuitos ópticos para os diferentes $pares(o, d)$.

A Figura 7 mostra o desempenho em termos de probabilidade de bloqueio para cada um dos 110 $pares(o, d)$ da topologia da rede Abilene utilizando os algoritmos DJK, MMR e MMRDS.

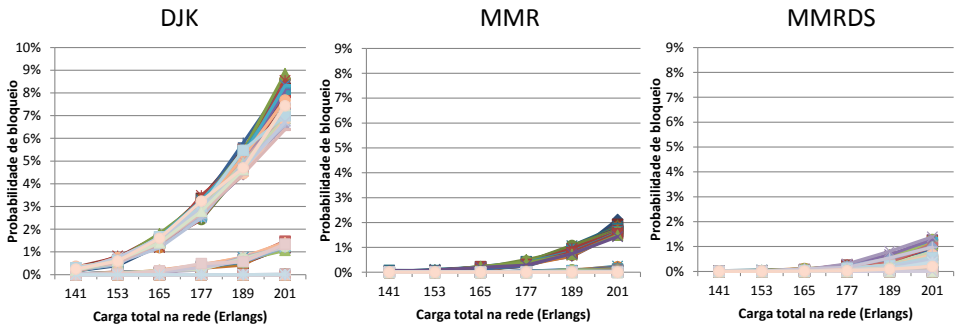


Figura 7. Probabilidade de cada um dos 110 $pares(o, d)$ da topologia da rede Abilene.

De maneira geral observa-se um espalhamento da probabilidade de bloqueio por $par(o, d)$ com o aumento da carga na rede. Além disso, nota-se um menor espalhamento da probabilidade de bloqueio por $par(o, d)$ quando o roteamento MMRDS é utilizado. O pico para o algoritmo MMRDS foi cerca de 0,015 de probabilidade de bloqueio, enquanto

o MMR e o DJK obtiveram picos de probabilidade de bloqueio iguais a 0,022 e 0,098 respectivamente.

Para uma melhor análise em termos de justiça será utilizada a Equação 5 proposta em [Soares et al. 2007], além disso será realizada uma anlálise do desvio padrão da probabilidade de bloqueio dos pares.

Considerando, neste contexto, injustiça como a diferença entre os desempenhos dos pares de nós com maior e menor $Pb_{(o,d)}$, a Equação 5 proposta em [Soares et al. 2007] será utilizada para medir a justiça no atendimento dos circuitos ópticos:

$$Fr = \frac{1 - (\max Pb_{(o,d)})}{1 - (\min Pb_{(o,d)})}, \tag{5}$$

em que $\max Pb_{(o,d)}$ e $\min Pb_{(o,d)}$ representam, respectivamente, a probabilidade de bloqueio dos pares de nós (o, d) com pior e melhor desempenho do conjunto de $N.(N-1)$ pares de nós. A medida de justiça é a razão entre os desempenhos do pares de nós com pior e melhor $Pb_{(o,d)}$.

As Figuras 8 e 9 apresentam o desempenho dos algoritmos DJK, MMR e MMRDS em termos de justiça (Fr) considerando as topologias das redes Abilene, USA e EON sem e com capacidade de conversão total.

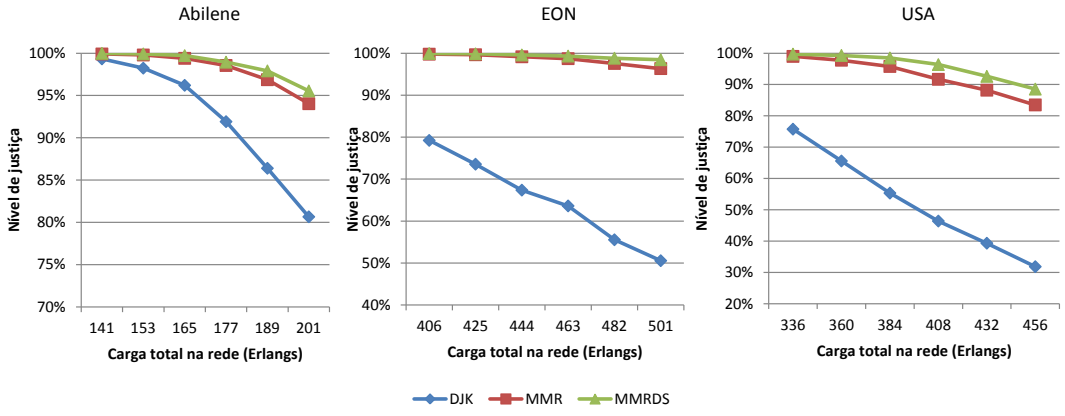


Figura 8. Desempenho em termos de justiça (Fr) dos algoritmos DJK, MMR e MMRDS aplicados as topologias das redes Abilene, USA e EON sem capacidade de conversão.

Comparando os cenários com e sem conversão de comprimento de onda observa-se um desempenho superior quando a rede faz uso da conversão total de comprimento de onda. Comparando o desempenho entre as topologias, EON e USA apresentam maior nível de justiça. Com relação ao algoritmos de roteamento, o MMRDS apresentou o melhor desempenho em todos os cenários. Entretanto, vale ressaltar que na topologia EON a diferença de desempenho entre o MMRDS e o MMR foi menor se comparada com as outras topologias. Ainda analisando as Figuras 8 e 9 nota-se que o DJK obteve um desempenho muito inferior aos algoritmos MMRDS e MMR.

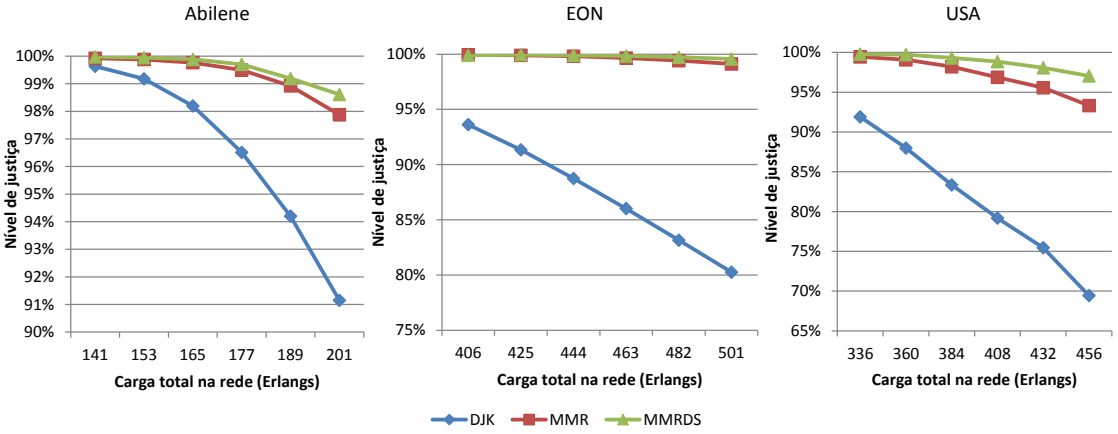


Figura 9. Desempenho em termos de justiça (Fr) dos algoritmos DJK, MMR e MMRDS aplicados as topologias das redes Abilene, USA e EON com capacidade de conversão.

As Figuras 10 e 11 ilustram o desempenho dos algoritmos MMR e MMRDS em termos do desvio padrão da probabilidade de bloqueio obtida por cada par(o,d), considerando as topologias das redes Abilene, USA e EON sem e com capacidade de conversão.

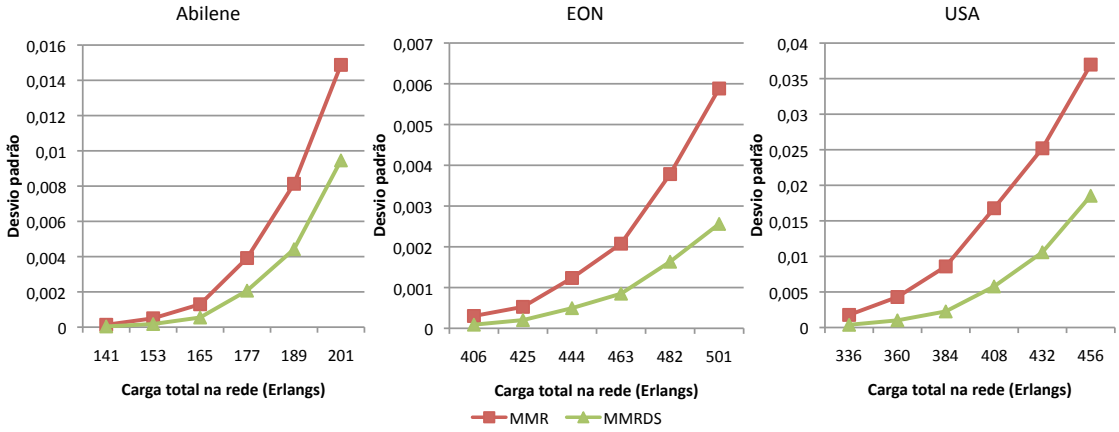


Figura 10. Desempenho em termos de desvio padrão da probabilidade de bloqueio dos pares(o,d) dos algoritmos MMR e MMRDS aplicados nas topologias das redes Abilene, EON e USA sem capacidade de conversão.

Com relação ao desvio padrão dos $pares(o,d)$ o algoritmo MMRDS também obteve desempenho superior ao MMR para todas as topologias e cenários observados.

Em termos de justiça e desvio padrão da probabilidade de bloqueio do $pares(o,d)$ o MMRDS apresentou um desempenho superior ao MMR. Vale destacar que além disso, para os mesmos cenários avaliados, o MMRDS obteve também o melhor desempenho em termo de probabilidade de bloqueio.

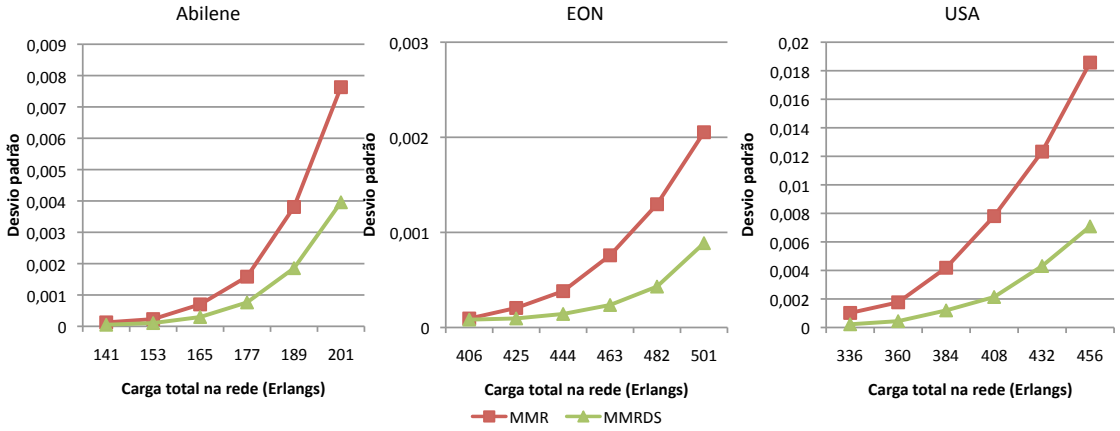


Figura 11. Desvio padrão da probabilidade de bloqueio dos pares(o,d) dos algoritmos MMR e MMRDS com capacidade de conversão.

5.3. Tempo de execução

Visando demonstrar a viabilidade computacional do MMRDS, foi feita uma avaliação dos tempos de execução dos algoritmos MMR e MMRDS. Foi utilizada a mesma metodologia de comparação apresentada em [Durães et al. 2009] para o algoritmo MMR. Para isso foram consideradas diferentes topologias com grau médio 3, variando o número de nós de 20 a 120.

A Figura 12 apresenta uma comparação em termos de tempo de execução dos algoritmos MMR e MMRDS.

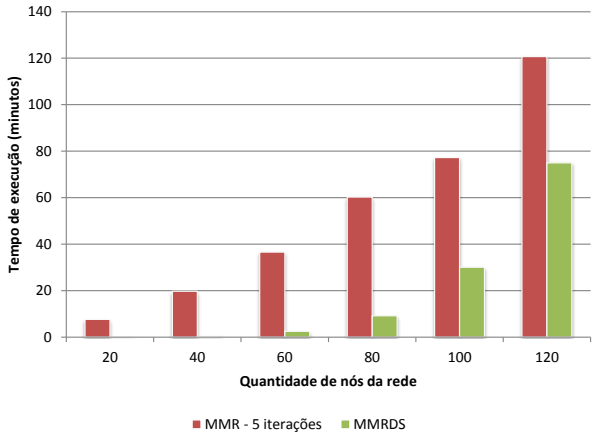


Figura 12. Tempo de execução dos algoritmos MMR e MMRDS.

O MMRDS possui tempo de execução significativamente menor não ultrapassando 80 minutos enquanto que o MMR, sob as mesmas condições, teve tempo de execução próximo a 120 minutos. Vale ressaltar que nesta comparação foram consideradas apenas 5 iterações para o algoritmo MMR. Entretanto, como se trata de algoritmos da classe de roteamento fixo, o tempo de execução de 120 minutos alcançados pelo algoritmo MMR com uma topologia de 120 nós não é crítico. Essa execução de algoritmos de roteamento fixo é feita apenas uma única vez, em uma fase de planejamento da rede. Essa

diferença em termos do tempo de execução pode ser explicada pelo fato do MMRDS não necessitar fazer simulações a cada iteração como ocorre com o MMR.

Com os estudos de avaliação de desempenho realizados neste trabalho observou-se um melhor desempenho do MMRDS quando comparado com o MMR e o DJK. O MMRDS foi superior para todas as topologias de redes estudadas com e sem conversão de comprimento de onda em termos de probabilidade de bloqueio, justiça. Acreditamos que esse melhor desempenho do MMRDS em relação ao MMR é justificado pelo fato do MMRDS investigar a similaridade das rotas no processo de escolha da rota de menor caminho.

6. Conclusão

Os algoritmos da classe de roteamento fixo são frequentemente utilizados nos trabalhos em redes ópticas transparentes. Isso se deve à baixa complexidade desse tipo de algoritmo, uma vez que não são necessários a troca e manutenção de informações sobre o estado da rede.

O problema 3MC consiste na escolha de uma combinação de rotas de menor caminho para cada par de forma a minimizar a probabilidade de bloqueio na rede. Este artigo apresentou o algoritmo MMRDS como solução para o problema 3MC, o algoritmo MMRDS se mostrou mais eficiente do que os algoritmos DJK e MMR em termos de probabilidade de bloqueio e justiça.

Vale ressaltar que as soluções de rota encontradas tanto pelo MMR quanto pelo MMRDS são soluções de rota subótimas para o problema do 3MC. Dessa forma, em trabalhos futuros serão investigadas novas estratégias para a resolução do 3MC como a utilização de metaheurísticas para a resolução do problema.

Referências

Birman, A. (1996). Computing approximate blocking probabilities for a class of all-optical networks. *Selected Areas in Communications, IEEE Journal on*, 14(5):852 –857.

Chu, X., Liu, J., and Zhang, Z. (2004). Analysis of sparse-partial wavelength conversion in wavelength-routed wdm networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1363 – 1371 vol.2.

Durães, G., Soares, A., and Giozza, W. (2009). A escolha da melhor entre as menores rotas em redes Ópticas transparentes. In *Simpósio Brasileiro de Redes de Computadores - SBRC*, pages 3–16.

Durães, G. M., Soares, A., Amazonas, J. R., and Giozza, W. (2010). The choice of the best among the shortest routes in transparent optical networks. *Computer Networks*, 54(14):2400 – 2409.

Lin, H.-C., Wang, S.-W., and Tsai, C.-P. (2006). Traffic intensity based fixed-alternate routing in all-optical wdm network. In *ICC 2006*, pages 2439 – 2446.

Murthy, C. S. R. and Gurusamy, M. (2002). Wdm optical networks - concepts, design and algorithms. *Prentice Hall PTR*.

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.

Soares, A., Giozza, W., and Cunha, P. (2007). Classification strategy to mitigate unfairness in all-optical networks. In *Networks, 2007. ICON 2007. 15th IEEE International Conference on*, pages 161 –165.

Zang, H. and Jue, J. P. (2000). A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks. *Optical Networks Magazine*, 1:47–60.

Enabling Information Centric Networks through Opportunistic Search, Routing and Caching *

Guilherme Domingues[1], Edmundo de Souza e Silva[1], Rosa Leão[1], Daniel S. Menasché[1]

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

Abstract. *Content dissemination networks are pervasive in todays' Internet. Examples of content dissemination networks include peer-to-peer networks (P2P), content distribution networks (CDN) and information centric networks (ICN). In this paper, we propose a new system design for information centric networks which leverages opportunistic searching, routing and caching. Our system design is based on an hierarchical tiered structure. Random walks are used to find content inside each tier, and gateways across tiers are used to direct requests towards servers placed in the top tier, which are accessed in case content replicas are not found in lower tiers. Then, we propose a model to analyze the system in consideration. The model yields metrics such as mean time to find a content and the load experienced by custodians as a function of the network topology. Using the model, we identify tradeoffs between these two metrics, and numerically show how to find the optimal time to live of the random walks.*

1. Introduction

In today's Internet, there is a strong demand for content dissemination networks, such as *social networks* (e.g. Facebook) and *video networks* (e.g. Youtube and RIO [de Souza e Silva et al. 2006]). To support this growing demand, two broad classes of solutions, supported by IP host-to-host communications, have been proposed: Content Delivery Networks (CDNs) and Peer-to-Peer Networks (P2P).

In CDNs, *dedicated servers* store all the information published. Copies of popular contents are placed close to the users within cache servers. Users are automatically and transparently re-directed to the most appropriate server by a central authority. Quality of Service (QoS) and advanced monitoring techniques are deployed, through proprietary solutions, to redirect each user requests (e.g., Akamai Networks). In CDNs, a set of *origin servers* store a copy of all published content. Popular contents are also stored in *cache servers* close to users, so as to minimize the service delay experienced by the requesters. Requests for content are sent through IP routers to the central authority. Content flows back to users, along the IP routers.

In P2P systems, *peers* act as both client and servers for contents. Peers' requests are sent to other peers, from where content fragments (*chunks*) can be retrieved. Bittorrent and Emule are examples of P2P systems. As soon as peers have downloaded all desired content chunks, they may either leave the system or remain as future providers for chunks (*seeders*). While seeders, P2P nodes can leave the system for different reasons (*e.g.*, closing their connection to other peers, power failures, mobility). Indeed, there are no

*This research was supported in part by grants from CNPq and FAPERJ.

guarantees on the time a seeder will remain present in a P2P system. If peers leave the system immediately after concluding their downloads, content might become unavailable which might lead to instability [Menasché et al. 2010]. Peers exchange information through IP routers, but in contrast to CDNs, there are no dedicated servers to cache content close to users.

According to [Ahlgren et al. 2012], global traffic in the Internet will increase by a factor of four from 2009 to 2014, approaching 64 exabytes per month in 2014, compared to approximately 15 exabytes per month in 2009. Global mobile data traffic is expected to double every year through 2014. Despite the tremendous success of CDNs and P2P systems, they present issues if we consider the *scalability* and *reliability* envisioned for next generation of content dissemination networks. With billions of mobile nodes claiming for contents, central decision policies at CDNs tend not to scale. In P2P networks, the absence of dedicated cache servers close to users, as well as the lack of guarantees for a peer to remain in the system after obtaining the desired content, hampers reliability.

The scalability and reliability challenges for massive distribution content gave rise to a new research area: Information Centric Networks (ICN) [Ghodsi et al. 2011]. In ICNs, users know the *name* of a content being searched, before issuing requests. All searched content is previously stored in the network through subscriptions to the network. Caching at all routers, also referred to as *universal caching*, is considered. Under universal caching, caching is provided to all users, and can be potentially implemented by all routers, being pervasive along the entire network. Figure 1 exemplifies this scenario. Questions on how to deploy efficient algorithms to explore *universal caching* and on how to define *inter-domain* routing policies give rise to important challenges.

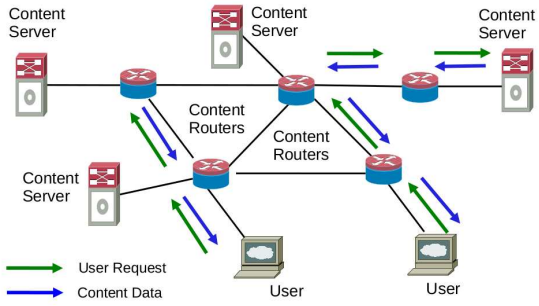


Figura 1. ICN scenario

Figure 2 shows some of the most important ICN features. In ICNs, content is published without any explicit destination address. Receivers subscribe to the network through a query (request) for named contents. Users' requests are forwarded by structured or non-structured topologies. When there is a matching between a query and a stored content, content is delivered to the subscribers. During delivery, content might be cached and replicated in the network. This strategy is known as *in-networking caching*. Content can be sent to subscribers over TCP/IP transport mechanisms over paths which are oblivious to the path taken by the requests. Alternatively, content can be sent in the reverse path of requests trails stored across the caches. We reference caches in ICNs henceforth as *cache-routers*.

In this article, our main contribution is a novel ICN architecture with routers dis-

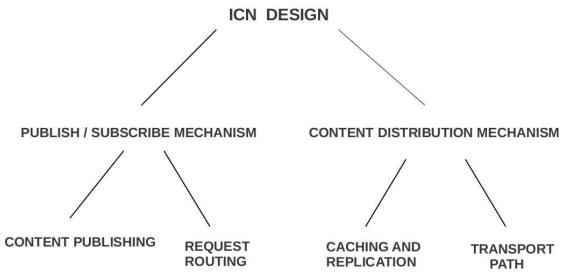


Figura 2. Publish/subscribe and content distribution mechanisms

posed along hierarchical tiers domains, without a global lookup service to map names to IP addresses to all content present in the network:

• **System design:** We propose an opportunistic search algorithm to be executed across hierarchical tiers. Random walks are used to opportunistically find replicas of the content inside a tier. If no replicas are found, requests are forwarded to another tier in the direction of publish area servers, which maintain fixed copies of the content. This way, we cope with the tradeoff between exploration of new routes to content replicas and exploitation of known routes to fixed replicas. In addition, we also propose an opportunistic caching policy using reinforced counters. The content placement and eviction policies are targeted towards self-tuned storage mechanism which must distribute content in the network to satisfy demands that vary over time.

• **Analytical model:** We propose an analytical model to evaluate ICNs inspired by reliability theory concepts. The model yields metrics such as the mean time to find a content and the load at the publishing areas as a function of the network topology and the time to live (TTL) of requests inside domains. Our model allows us to study tradeoffs in the choice of the TTL. Smaller values of TTL might lead to a reduction in the time to retrieve information at the cost of an increase in the system's load at publishing areas.

The remainder of this paper is organized as follows. Section 2 presents related work and a background on Information Centric Networks. Section 3 introduces the proposed ICN system design, Section 4 contains the analytical model and in Section 5 we present numerical results obtained with the proposed model. Section 6 concludes the paper.

2. Related Work

The literature on ICN can be broadly classified into structured architectures, such as [Koponen et al. 2007, Lagutin et al. 2010, 2nd NetInf Description 2010] and unstructured architectures, such as [Rosensweig et al. 2010, Kakida et al. 2011]. In what follows, we briefly describe some of the proposals.

Dona [Koponen et al. 2007] consists of an hierarchy of domains, wherein a resolution handler (RH) knows the IP location of all content published in descendent domains. RHs placed in the highest domain are aware of all the content published in the entire network. Psirp [Lagutin et al. 2010], Netinf [2nd NetInf Description 2010] and Mulitcache [Katsaros et al. 2011] are proposals that deploy Distributed Hash Tables (DHT), enabling multiple logical entities to share the knowledge of all content published in the

network, in a distributed fashion. Each content placed in the network is mapped to a hash key, with a hash key being associated with a single node of the DHT.

DHT structures are highly scalable, but may impose considerable overhead maintenance costs [Chen et al. 2008]. Whenever a node fails or becomes repaired, requests must be sent along the network to reassure the node location in the network. Neighborhood adjacencies must be reestablished and file-index databases related to a partition management must be resettled. Also, many unsolved security vulnerabilities are able to disrupt the pre-defined operation of DHTs nodes [Urdaneta et al. 2011]. At last, in a network composed of domains where providers care about administrative autonomy, the use of a global hash table is unfeasible [D'Ambrosio et al. 2011].

Jacobson et al. [Jacobson et al. 2009] propose a non structured geometry topology for routing requests. Published content is announced through routing protocols, composing routing tables supporting name aggregation. Requests are routed towards publishing areas leaving a trail called *bread crumbs*, so content follows the reverse paths set by the trails, when sent to users. As content flows to users, the bread crumbs are consumed. In general, content replicas will not be stored in the domain they were originally published. To be encountered, routes for replicas should be in the routing tables. Avoiding an explosion of the size of the routing tables becomes an important challenge.

To enhance the discovery of cached contents, Rosensweig et al. [Rosensweig et al. 2010] and Kakida et al. [Kakida et al. 2011] allow bread crumbs not to be consumed on the fly, when content traverses the network. This allows trails for previously downloaded contents to be preserved. A new user request that reaches a trail for a desired content will be sent to a cache-router indicated by the trail, before flowing to the publishing areas. Opportunistically downloading content from nearby cache-routers may considerably reduce the time users take to retrieve information. Nevertheless, the proposals encompassing the preservation of bread crumbs do not explicitly address how published information will be announced to fulfill the routing tables.

In this article, we avoid the drawbacks mentioned in this section, for both structured and unstructured geometry topologies. To this aim, we propose a novel architecture with cache-routers disposed across hierarchical domains. Users' requests flow across tiers towards the publishing areas. Random walks are issued to explore domains' vicinity, so as to allow opportunistic encounters with the desired content. That way, we avoid the drawbacks of structured geometries, as well as the problem of fulfilling routing tables which is common to the unstructured architectures discussed above.

The model presented in this paper is inspired by reliability metrics [de Souza e Silva and Muntz 1992]. The performance of random walks for search in unstructured hybrid peer-to-peer systems has been analyzed by Ioannidis and Marbach [Ioannidis and Marbach 2008]. Nonetheless, Ioannidis and Marbach focus on asymptotic results when the number of nodes in the network grows to infinity, and considered a single domain (tier). To the best of our knowledge, our work is the first to analyze the performance of random walks for search in multi-tiered architectures, accounting for the tradeoffs involved in the choice of the time to live of the random walks. Existing multicache multi-tier normal cache overlay network either consider global knowledge of content placement (CDN, P2P), or single path to the storage area (Cache Trees). None of these methods

explore the vicinity of a cache in a fully distributed way.

3. System Design

Our architecture design considers **logical** hierarchical *tiers* (also called *domains*) composed of *cache-routers* ($i.e.$, routers that can store content replicas). We consider N logical hierarchical tiers, in which tier 1 is the top level tier, and tier N constitutes the bottom level. Routers in tier N are the only ones connected to users. Users publish content in storage areas connected to tier 1 routers. The interaction among the publishing areas and tier 1 routers is better detailed in Figure 3. The figure displays a single publishing area in tier 1 and the logical hierarchy of cache-routers. Routers forward requests towards publishing areas which contain permanent copies of the content. Replicas may be cached in the cache-routers in the logical hierarchy. A strategy should be adopted to allow *opportunistic* encounters between requests and replicas in a best-effort manner as will be detailed below. Opportunistic encounters are probabilistic in nature.

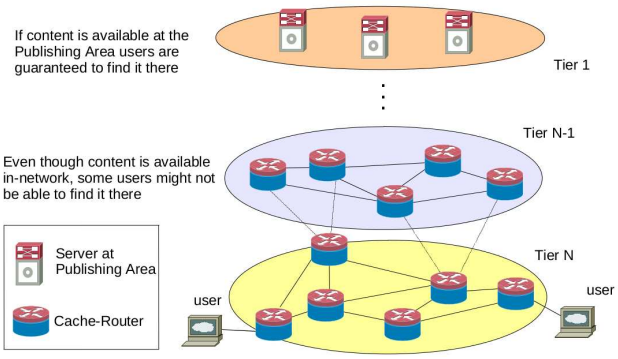


Figura 3. ICN content distribution mechanisms.

3.1. Content search: random walks and bread crumbs

When requests are issued by users the cache-routers forward them to the publishing areas. Each cache-router of tier i is logically connected to a subset of cache-routers in the same tier i and in the parent tier $i-1$. When a request first reaches a tier, a random walk is issued to find replicas of the content that may have been cached in that tier. The random walk lasts for at most T units of time, and only traverses cache-routers in the same level. That is, a counter is set to limit the amount of search time for a content *within a level*. If the desired content is not found when this counter expires, the router (say router k at level i) that holds the request at that time transfers it to level $i-1$, that is, to one of the routers in level $i-1$ router k is (logically) connected to. A new random walk will then be issued, but now at level $i-1$. If the content been searched is found at a cache-router of hierarchy j, it is sent to the user as it will be explained below. Otherwise, the request will be forwarded up in the hierarchy until the publishing area (tier 1) is reached where the content is guaranteed to be found (otherwise the content was never stored in the first place). Note that the search does not generate significant traffic. Each request from a user generates a single search message that performs a random walk at one tier at a time. The traffic due to this request message is negligible as compared to that generated by the content to be transferred.

As the requests are forwarded upwards across the hierarchy, a set of *b*ackward pointers is maintained. Those pointers are henceforth referred to as *bread crumbs*. When a content is located in the network, it is delivered to the users following the reverse path of the bread crumbs. As the content follows the path of *bread crumbs*, the trail is erased. Random walks within domains do not generate bread crumbs. Therefore, when content is delivered to users it does not follow the exact same path as the random walks. This means that random walks do not impose extra hops/side effects when content is delivered to users. We use a set of counters, named *reinforced counters* (RCs), to keep track of the load across the network for each content. The placement of replicas will be controlled using the reinforced counters, which will be used to decide which and when contents should be stored or evicted at each cache-router. Our approach is self-adaptive to users' loads.

3.2. Content placement: reinforced counters

Each request carries a hash key *inf* to identify the content being searched for. As mentioned above, a trail of bread crumbs is left at each cache-router traversed by the request. The trail includes a back pointer to the preceding router visited by the request (the bread crumb), the hash key *inf* to identify the request and an associated counter (called *reinforced counter*) that is incremented when the request arrives at a cache-router. The *reinforced counter* is used to determine whether the content associated with inf should be stored at the cache-router. Each cache-router keeps a reinforced counter rc(*inf*), associated to each hash key *inf*. Whenever a request for *inf* reaches a cache-router, rc(*inf*) is incremented by one. When content is downloaded through the reverse path left by the trails, bread crumbs are consumed but the reinforced counters are kept intact. Each cache-router periodically decreases the reinforced counter rc(*inf*) associated to *inf* and this period is a parameter to be set.

A reinforced counter rc(*inf*) for content *inf* at a cache-router has two thresholds: rc-up(*inf*) and rc-low(*inf*). If rc(*inf*) reaches the upper threshold rc-up(*inf*) the content *inf* is cached at that cache-router after it is found and when it traverses the cache-router towards to the user. Whenever rc(*inf*) reaches the lower threshold rc-low(*inf*) and if *inf* is cached at that cache-router, this content is immediately removed from the cache-router. We assume that cache-routers have enough storage capacity to store any content that they are required to maintain. The reinforced counters mechanism allows popular contents to be transferred from the publishing areas to the cache-routers. That way, the opportunistic discovery of popular content is favored at cache-routers. In contrast, if demand is not high enough to justify consumption of storage resources, reinforced counters will favor the eviction of the content.

We provide additional insights on the connection between reinforced counters and content placement. For simplicity of exposition, and without loss of generality, we assume that content is demanded by users at a given fixed rate. We then show that the reinforced counters fully determine content placement at the cache-routers. To this aim, we consider a fluid approximation, where content requests arrive according to a flow with a given intensity, and the reinforced counters are also approximated as being continuous. Let $\Lambda_{m,l}(\mathit{inf})$ be the total load for *inf* arriving from tier $l+1$ at the m-th cache-router at tier l. Let $\gamma(\mathit{inf})$ be the rate at which the reinforced counters rc(*inf*) are decremented. We assume that the rc(*inf*) are initialized as rc-low(*inf*), so that if $\Lambda_{m,l}(\mathit{inf}) = \gamma(\mathit{inf})$ then

content will not be stored at m.

Proposition 3.1. *Under the fluid approximation and an hierarchical tiered topology, a cache-router stores a copy of* inf *if and only if* $\Lambda_{m,l}(\mathsf{inf}) > \gamma(\mathsf{inf})$.

Proof: The proof is inspired by [Rosensweig et al. 2013, Lemma 1]. In an hierarchical network we define the direction of requests as *upstream* and the reverse as *downstream*. A cache-router is affected only by requests that pass through it, and requests pass through routers only upstream, so upstream cache-routers do not impact cache-router m through their requests. Content flows downstream, but only passes through m for requests that were issued by cache-router m. Therefore, the state and requests of upstream routers do not impact the state and requests issued by downstream routers.

We can determine if a cache-router will store a content in a bottom-up manner. First, consider the leafs of the tiered topology. At such cache-routers, if $\Lambda_{m,l}(\mathit{inf}) > \gamma(\mathit{inf})$ then rc(*inf*) will eventually reach rc-up(*inf*), and content will be stored and never evicted. If $\Lambda_{m,l}(\mathit{inf}) \leq \gamma(\mathit{inf})$, in contrast, rc(*inf*) will remain equal to its initial value rc-low(*inf*). rc(*inf*) will afterwards remain fixed, and the cache-router, marked. Once leafs are marked, we proceed upstream. Using the same argument as the one in the paragraph above, we determine the content placement at the next cache-router that has all its children marked, and so on, up to reaching routers at tier 1, where all content is stored. □

Given a workload, proposition 3.1 can be used to determine which domains across the network will hold a copy of each content, if we know the (probabilistic) choice by which a router of level i passes requests to routers of level $i-1$ at which it is logically connected to. In the remainder of this paper we will assume that the probability distribution of content over cache-routers is fixed and given for all cache-routers.

4. System Analysis

In this section we present the model analysis for a single domain (i.e., a logical tier). Our main goals are to derive the key metrics of interest, namely 1) mean time to find content and 2) the fraction of requests that hit the publishing area. These metrics are important to evaluate the performance of the proposed architecture. We consider a network of C cache-routers in a domain. Figure 4(a) illustrates a domain with five cache-routers. Assume that the desired content is stored at cache-routers 3, 4 and 5. Suppose that a request arrives from a downstream domain to cache-router 2. Then, cache-router 2 starts a random walk in the domain to find the content. Let T be the maximum time a request might spend in a domain before being redirected to the upstream domain in the hierarchy. T is also referred to as time to live, or TTL.

We consider a continuous time random walk in which the time between walker movements are exponentially distributed with rate ψ, with the associated infinitesimal generating matrix Q. Let $\Delta(i)$ be the out-degree of node i and ψ the walker rate. Then, all diagonal elements in Q are equal to $-\psi$, and $q_{ij} = \psi/\Delta(i)$ if there is a logical link from i to j. We use uniformization [de Souza e Silva and Gail 2001] to obtain the metrics of interest. Let P be the *uniformized matrix*, obtained from Q as $P = I + Q/\Lambda$, where Λ is a positive number greater than the maximum absolute value of the elements in the diagonal of Q. Λ is also referred to as the *uniformization rate*.

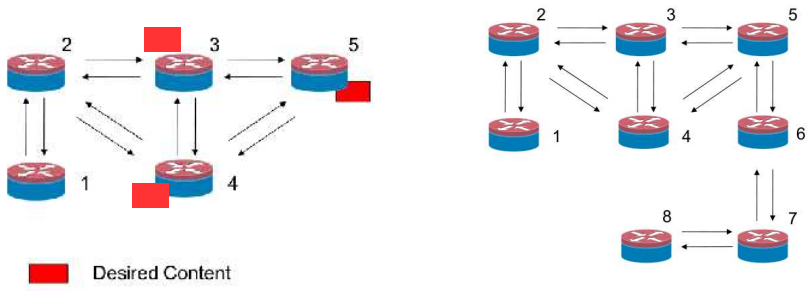
variable	description
C	number of cache-routers in the network
T	time to live
$H(T)$	time to hit content
$L(T)$	lifetime of random walk in a domain
$R(T)$	probability of not hitting content by T
T_0	time cost for accessing the publishing area
p	probability that content is in domain

Tabela 1. Table of notation. Time to live, T, is integer when considering discrete time random walks, and real when considering continuous time random walks.

Let $\Omega_P(k)$ be the probability that, after k transitions of the uniformized process, the random walk had not hit the desired content. To compute $\Omega_P(k)$ we construct a modified transition matrix $\widetilde{P}$, assuming that the desired content is placed in a subset of cache-routers of the domain. Let $\boldsymbol{\omega}$ be a column vector of size C where $\boldsymbol{\omega}_i = p_i$ and p_i is the probability that cache-router i stores the content, $\sum_{i=1}^{C} p_i = 1$. To facilitate the notation, we number the cache-routers where $p_i > 0$ with the highest indexes of matrix P. In Figure 4(a) they are numbered 3,4 and 5. Then, $\widetilde{P}$ is defined as

$$\widetilde{P} = P(I - (diag(\boldsymbol{\omega}))) \tag{1}$$

where $diag(\boldsymbol{\omega})$ is a matrix with the diagonal elements equal to vector $\boldsymbol{\omega}$ and all other elements equal to zero. $\widetilde{P}$ is a sub-stochastic matrix where the sum of the elements in line i is the probability of not finding the content in the domain, in one step, given that the random walk starts at cache-router i. The element (i, j) of $\widetilde{P}$ is the probability that a random walker that starts at cache i moves to cache j in one step and then a cache miss occurs at j.



(a) (b)

Figura 4. (a) Example of a domain with five cache-routers and (b) Example of a domain with eight cache-routers.

Let $\pi(0)$ be the initial state probability vector indicating from where the search starts in the domain. The i-th element of $\pi(0)$, $\pi_i(0)$, corresponds to the probability that the random walk starts at cache-router i. The (i, j) entry of matrix $\widetilde{P}^k$ characterizes the probability of visiting state j after jump k, given that the system starts at state i. Let $\boldsymbol{\upsilon}_{\widetilde{P}}(k)$ be a vector whose m-entry is the probability that after k steps a miss occurs at

cache-router m, given that the random walk initial state is $\pi(0)$. The following recursion can be used to compute $\boldsymbol{\upsilon}_{\widetilde{P}}(k)$,

$$\boldsymbol{\upsilon}_{\widetilde{P}}(k) = \boldsymbol{\upsilon}_{\widetilde{P}}(k-1)\widetilde{P}, \qquad k > 0 \tag{2}$$

where $\boldsymbol{\upsilon}_{\widetilde{P}}(0) = \pi(0)$. Then, $\Omega_P(k)$ is given by

$$\Omega_P(k) = \boldsymbol{\upsilon}_{\widetilde{P}}(k)\mathbf{U}, \qquad k \geq 0 \tag{3}$$

where $\mathbf{U}$ is a column vector with all elements equal to one.

Mean Time to Reach Content

Recall that T is the maximum time to live of a random walk in a domain, $T \in \mathbb{R}$. Let $R(T)$ be the probability that the content is not found by T units of time. Then, conditioning on the number of transitions in the interval $[0, T]$,

$$R(T) = \sum_{n=0}^{\infty} \varphi(n, T)\Omega_P(n) \tag{4}$$

where $\varphi(n, T) = \exp(-\Lambda T)(\Lambda T)^n/n!$ is the probability that n jumps occur in the interval $[0, T]$. The number of jumps in the interval $[0, T]$ is Poisson distributed as we uniformized Q.

Next, our goal is to compute the expected hitting time of a content, $E[H(T)]$. The expected hitting time of a content is the sum of two components. The first component characterizes the mean time to hit the content given that it is available in the domain, while the second component characterizes the mean time to hit the content if it is unavailable in the domain. In case the content is available in the domain, the *lifetime of the random walk* in the domain (i.e., the searching time in the domain), $L(T)$, is the minimum between the time until reaching the content and T. The mean of $L(T)$ is given by

$$E[L(T)] = \int_0^T R(t)dt \tag{5}$$

where $\lim_{T\to\infty} E[L(T)]$ is referred to as *mean time to exit* in the reliability literature, and can be evaluated using standard techniques [de Souza e Silva and Gail 2000].

Let p be the probability that the content is available in the considered domain. If the content is available, the probability that the content is not found by time T is $R(T)$. If the content is unavailable, the random walk will take time T inside the domain, and additional T_0 units of time to reach the publishing area. Therefore, the expected hitting time of a content is

$$E[H(T)] = (E[L(T)] + T_0 R(T))\, p + (T_0 + T)(1 - p) \tag{6}$$

Replacing (4) into (6), and exchanging the order of the integration and summation, yields, after simplification,

$$E[H(T)] = \left(\sum_{n=0}^{\infty} \left(1 - \sum_{m=0}^{n} e^{-\Lambda T} \frac{(\Lambda T)^m}{m!} \right) \Lambda^{-1} \Omega_P(n) + T_0 R(T) \right) p + (T_0 + T)(1 - p) \tag{7}$$

The analysis above easily handles discrete time random walks, where time between walkers movement is fixed.

5. Experimental Results and System Analysis

In this section we present numerical results for a few examples in order to illustrate our proposal and the existing tradeoffs in the choice of parameter values. In the development of section 3, the time intervals between the random walker steps can be either independent and exponentially distributed random variables or constant. For the numerical studies we consider constant step values.

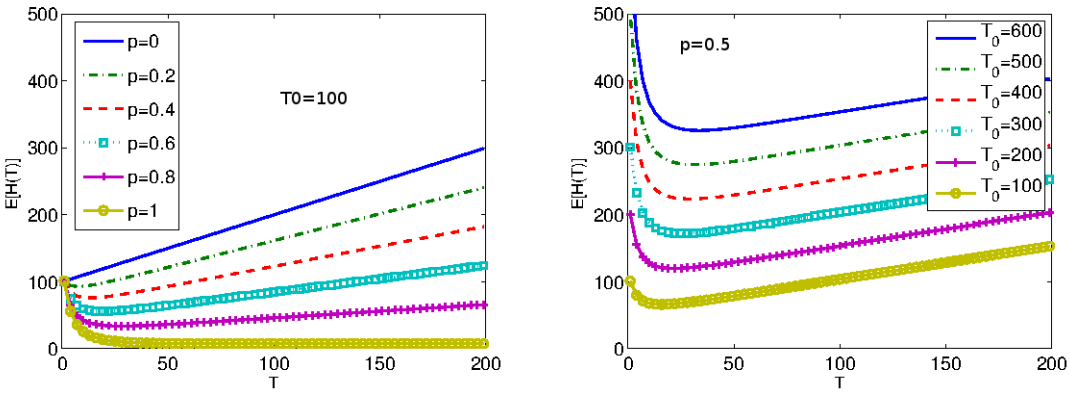
Our goals are to 1) show the impact of different system parameters on the mean time to find content and 2) evaluate the average load on publishing area servers, which impacts both the time to recover a content and the system's scalability. We analyze a simple two-tier setup wherein the publishing area servers have finite capacity. This simple architecture suffices to highlight the key points we want to emphasize in this work while keeping the model description short. From this scenario we also indicate how the evaluation could easily consider multiple logical tiers.

Our reference topology is that illustrated in Figure 4(a). In this topology, there is a single content that may be cached in one of the cache-routers $3, 4$ or 5 with equal probability of residing in any of these three routers, given it is in this logical tier. Therefore, $p_u = 1/3$ for $u = 3, 4, 5$. With probability $1 - p = 0.5$, the content does not reside in the tier and it takes and additional $T_0 = 100$ time units to find it in the publishing area. The parameter T, the maximum time the random walker is allowed to search for the content in a logical tier is varied between 1 and 200. The parameters values may vary according to the experimental goals.

5.1. Single Domain Case

We illustrate how the mean time to find the content, $E[H(T)]$, depends on different system parameters. Figure 5(a) shows $E[H(T)]$, obtained using (6), as a function of T, for different values of p varying between 0 and 1, with increments of 0.2. The topology and the content placement are kept fixed. Let $T^\star$ be the optimal value of T that minimizes $E[H(T)]$. As the probability that the content is available increases, $T^\star$ increases. Clearly, when $p = 0$, $T^\star = 0$, since if the content is not available in the tier it is better not to search for it. On the other hand, when the content is always available ($p = 1$), $T^\star \to \infty$. This is because, in this example, T_0 is large compared to the mean time to find the content in the tier ($E[H(\infty)]$). In this scenario, it will take less time, on average, for the random walker to find the content in the domain, as opposed to find it in the publishing area servers, if the walker is given enough time to search for the content in the domain (i.e., if T is very large). For values of p between $(0, 1)$, the optimum value $T^\star$ can be obtained from our model as shown in Figure 5(a).

For any given value of p, Figure 5(a) shows that $E[H(T)]$ first decreases and then increases, except for $p = 0$ and $p = 1$, when $E[H(T)]$ always increases and decreases, respectively. In any case, $E[H(T)]$ asymptotically grows at rate $1 - p$, as can be inferred from equation (6). The impact of T_0 (the time cost for accessing the publishing area servers) on $E[H(T)]$ is illustrated in Figure 5(b), which shows $E[H(T)]$ as a function of T, for $T_0 \in [100, 200, \ldots, 600]$. As T_0 decreases, there is a subtle decrease of the optimal time to find a content in the domain. This is because, as the time to access publishing area servers increases, it is beneficial to remain longer in the domain before forwarding the request to other domains.



(a) (b)

Figura 5. Effect of p and T_0 on the expected hitting time

Until now we analyzed the problem of content discovery from the clients perspective. The administrators of publishing areas, in turn, might worry about the traffic that is incurred in their servers. If content is available in the cache-routers in a tier, the load from that domain to the servers (assuming a single tier architecture) is proportional to $R(T)$, the probability that users do not find the content in the tier by time T. In more detail, let λ and Λ be the average user load (number of requests for a content per time unit) that arrives to a tier and the average load flowing from the tier to the next in the hierarchy (or from the tier to the publishing area), respectively. The number of requests per unit time served inside the domain is clearly $\lambda - \Lambda$ and $\Lambda = R(T)\lambda$. For the plots we consider $\lambda = 100$.

Figures 6(a) and 6(b) show how Λ varies as a function of T. Figures 6(a) considers the reference scenario, modified so that content may be available at caches 2, 3, 4 and 5 with probability 0.25 at each, given the desired content is in the domain. Figures 6(b), in turn, is generated from the topology with 8 caches shown in Figure 4(b), where content may be available at caches 5, 6, 7 and 8 with probability 0.25 at each, given the desired content is in the domain. The other parameters are the same as in the reference scenario.

Figures 6(a) and 6(b) indicate that there is a tradeoff between users interests and providers time-cost when choosing the optimal value of T. If T is selected so as to minimize $E[H(T)]$, the rate at which the publishing area servers are accessed, Λ might be considerably large. However, a slight increase in T and $E[H(T)]$ might yield significant reductions in Λ. Therefore, we may minimize $E[H(T)]$ constraining Λ to a given (relatively small) value, that is, we may be able to keep the load at the publishing area servers at a low value and yet avoiding a considerable increase in the time to retrieve the searched content.

Note that in Figure 6(b) Λ is constant for $T \leq 3$. This is because we are considering a discrete time random walk for the studies. In this case, the probability that the content is found inside the domain is zero if T is smaller than the distance between the source of the random walker and the cache-router that may have the content. In a continuous time random walk, in turn, $R(T)$ is strictly decreasing with respect to T.

We may summarize the above findings as follows. If content is available in a tier with high probability, it may be advantageous to increase the time the random walker is

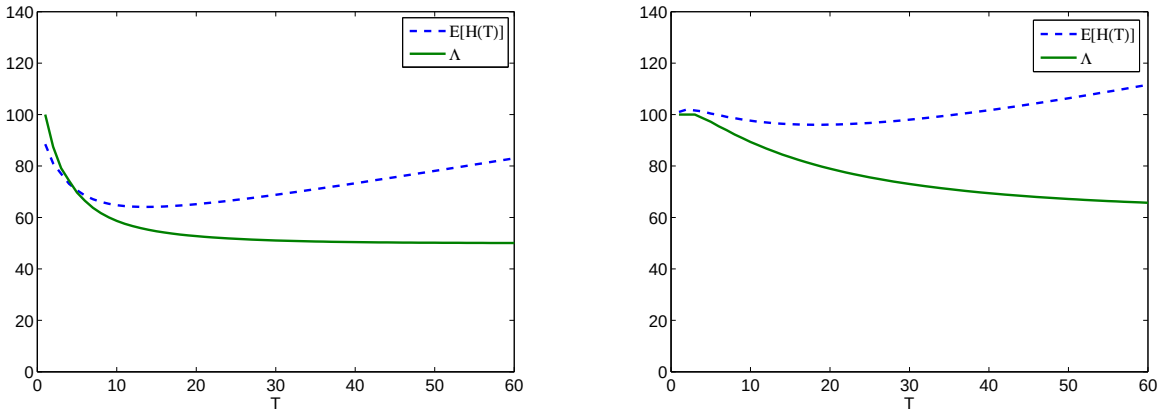


Figura 6. Probability of not hitting content by time T, $E[H(T)]$, and Λ, for (a) 5 cache topology, (b) 8 cache topology.

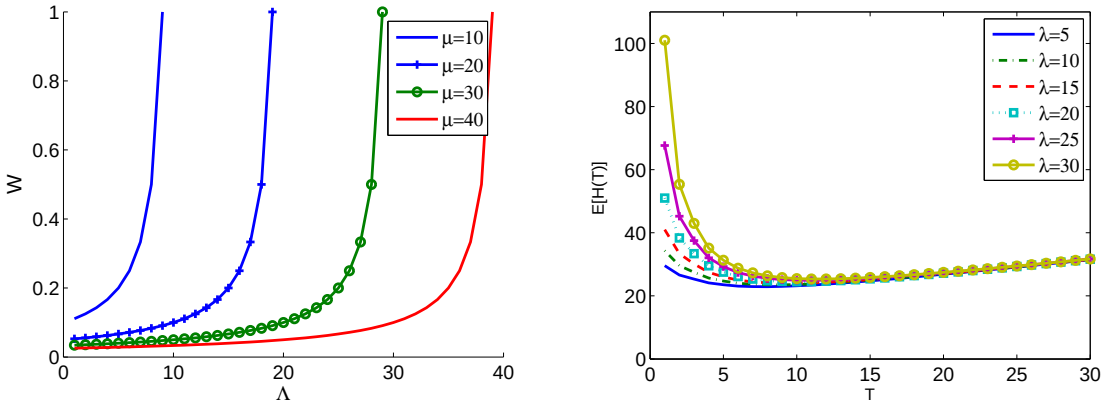
allowed to search for a content in the tier to reduce the load at the publishing area servers. On the other hand, the expected time to find the content may increase. Quantifying these tradeoffs is a contribution of our work.

5.2. Multiple Domain Case

In the previous section we assumed a constant time T_0 to retrieve a content from the publishing area servers. In what follows we consider a scenario where content that is not found in a domain is forwarded to publishing area servers with limited capacity. Therefore, the time cost to access the publishing area servers, T_0, will now be a function of the amount of traffic Λ that is directed to the publishing area servers (which in turn is a function of $R(T)$). To illustrate the impact of the load of users on the publishing area servers, let us a consider a simple M/M/1 model for the server, with associated service capacity μ. Then, the mean waiting time experienced by users accessing the server is $W = 1/(\mu - \Lambda)$. Figure 7(a) shows how W varies as a function of Λ. In what follows, for illustration purposes, we set $\mu = 40$ and $T_0 = 1000W$,

$$T_0 = 1000/(\mu - R(T)\lambda) \tag{8}$$

Figure 7(b) illustrates the impact of the service capacity on the mean waiting time experienced by users and shows how $E[H(T)]$ varies as a function of T. The qualitative behavior of $E[H(T)]$ is the same as the one observed previously, and the insights obtained in Section 5.1 about the tradeoff in the choice of T still apply. Note that an increase in $R(T)$, the probability of not finding the searched content in the tier (not shown in the figure) may degrade users performance, as it may lead to server overload. In particular, if T is small (e.g., $T = 1$), $E[H(T)]$ grows unbounded with λ. Due to space limitations we were only able to discuss a few examples aiming at describing our proposal and the existing design tradeoffs. Our approach, however, can handle multiple domains and user loads as we briefly sketch below. From the results of section 4 and given the user loads, we can obtain the domains that hold copies of the content. $R(T)$ is then used to obtain the loads at each domain as well as at the publisher. Finally, given the loads at the different domains, $E[H(T)]$ is obtained in a top down fashion, starting from the top and moving to the lowest level domains.



(a) (b)

Figura 7. (a) Average server processing time as a function of Λ, (b) mean time to find content, accounting for server capacity $\mu = 40$.

6. Conclusion

Information centric networks are gaining considerable attention from researchers and practitioners due to their scalability and robustness properties. Nonetheless, there are still important challenges in the design, modeling and analysis of such networks. From the design perspective, one of the challenges is to determine how to fill out the routing tables. From the modeling and analysis perspective, the challenges are associated to the large number of routers envisioned in ICNs. In this paper, we have proposed a novel design for ICNs that is built on top of random walks and hierarchical tiers (domains) to cope with the exploration *versus* exploitation tradeoff involved in content search.

Our model computes metrics such as mean time to find a content and evaluate existing tradeoffs to tune the parameters of the architecture we propose. In this present work we focus on the performance tradeoffs of the architecture. However, our analytical model can be extended to study the scalability of the proposed architecture with respect to others in the literature. For instance, as the user load for a content grows what is the overall increase in the expected time to find a content as compared to a P2P architecture. This work also opens additional avenues for future research. One such problem is to determine the impact of parallel random walks across tiers at the same level of the hierarchy as well as across different levels.

Referências

2nd NetInf Description (2010). Netinf. `http://www.4ward-project.eu/`.

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. (2012). A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36.

Chen, S., Zhang, Z., Chen, S., and Shi, B. (2008). Efficient file search in non DHT P2P networks. *Elsevier Computer Communications*, 31(2):304–317.

D'Ambrosio, M., Dannewitz, C., Karl, H., and Vercellone, V. (2011). MDHT: A hierarchical name resolution service for information-centric networks. In *Proc. of SIGCOMM workshop on ICN*, pages 7–12.

de Souza e Silva, E. and Gail, H. R. (2000). Transient Solutions for Markov Chains. In Grassmann, W., editor, *Computational Probability*, pages 44–79. Kluwer.

de Souza e Silva, E. and Gail, H. R. (2001). The Uniformization Method in Performability Analysis. In B. R. Haverkort, R. Marie, G. R. and Trivedi, K. S., editors, *Performability Modelling: Techniques and Tools*, chapter 3, pages 31–58. Wiley.

de Souza e Silva, E., Leão, R. M. M., Santos, A. D., Azevedo, J. A., and Netto, B. C. M. (2006). Multimedia Supporting Tools for the CEDERJ Distance Learning Initiative applied to the Computer Systems Course. In *22th ICDE World Conference on Distance Education*, pages 1–11.

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

Ghodsi, A., Schenker, S., Koponen, T., Singla, A., Raghavan, B., and Wilcox, J. (2011). Information-centric networking: Seeing the forest for the trees. In *Proc. of HOTNETS*, pages 1–6.

Ioannidis, S. and Marbach, P. (2008). On the design of hybrid peer-to-peer systems. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):157–168.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *Proc. of CoNEXT*, pages 1–12.

Kakida, M., Tanigawa, Y., and Tode, H. (2011). Breadcrumbs+ : Some extensions of naive breadcrumbs for in-network guidance in content centric networks. In *Proc. of IEEE IPSJ*, pages 376–381.

Katsaros, K., Xylomenos, G., and Polyzos, G. C. (2011). Multicache: An overlay architecture for information-centric networking. *Elsevier Computer Networks*, 55(4):936–947.

Koponen, T., Chawla, M., Chun, B. G., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. (2007). A data-oriented (and beyond) network architecture. In *Proc. of SIGCOMM*, pages 181–192.

Lagutin, D., Visala, K., and Tarkoma, S. (2010). Publish/subscribe for internet: PSIRP perspective. In *Emerging Trends from European Research, (Valencia FIA book 2010)*.

Menasché, D. S., Rocha, A. A., de Souza e Silva, E., Leão, R. M. M., Towsley, D., and Venkataramani, A. (2010). Estimating self-sustainability in peer-to-peer swarming systems. *Perf. Eval.*, 67(11):1243–1258.

Rosensweig, E., Kurose, J., and Towsley, D. (2010). Approximate models for general cache networks. In *Proc. of INFOCOM*, pages 1–9.

Rosensweig, E., Menasché, D., and Kurose, J. (2013). On the steady-state of cache networks. In *Proc. of INFOCOM, to appear*.

Urdaneta, G., Pierre, G., and Steen, M. V. (2011). A survey of DHT security techniques. *ACM Comp. Surveys*, 43(2).

RemapRoute: Reduzindo o custo do remapeamento de mudanças de roteamento na Internet

Ítalo Cunha[1] **Renata Teixeira**[2] **Darryl Veitch**[3] **Christophe Diot**[4]

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

[2]UPMC Sorbonne Universités & conseil Nationale de la Recherche Cientifique

[3]Department of Electrical and Electronic Engineering, University of Melbourne

[4]Technicolor

cunha@dcc.ufmg.br, renata.teixeira@lip6.fr, dveitch@unimelb.edu.au, christophe.diot@technicolor.com

Abstract. *Internet topology maps collected with traceroute may be incomplete or out-of-date because we cannot measure frequently enough to detect all routing changes without overloading the network. In this paper, we show that routing changes usually affect few routers. We exploit this property to design RemapRoute, a tool to locally remap Internet routing changes that probes only the few affected routers instead of the whole route. Our evaluation with trace-driven simulations and a deployment shows that RemapRoute significantly reduces the number of probes needed to remap routes, with no impact on remapping accuracy or latency. This reduction in remapping cost allows us to build more complete and up-to-date topology maps.*

Resumo. *Mapas topológicos da Internet coletados com traceroute podem estar incompletos ou desatualizados porque não podemos realizar medições em frequência suficiente para detectar todas as mudanças de rota. Neste artigo mostramos que mudanças de rota geralmente afetam poucos roteadores. Exploramos essa propriedade no RemapRoute, nossa ferramenta para remapeamento local de mudanças de roteamento que sonda apenas os poucos roteadores afetados ao invés de toda a rota. Nossa avaliação via simulação e um protótipo real mostra que RemapRoute reduz significativamente o número de sondas de remapeamento, sem comprometer a exatidão e a latência de remapeamento. Esta redução do custo de remapeamento nos propicia construir mapas topológicos mais completos e atualizados.*

1. Introdução

Sistemas para identificação de falhas na Internet coletam medições frequentes de rotas na rede [Duffield 2006, Dhamdhere et al. 2007, Kompella et al. 2007, Katz-Bassett et al. 2012]. De forma similar, redes de distribuição de conteúdo medem rotas na Internet para escolher o melhor servidor para atender uma requisição [Dilley et al. 2002]. Esses e outros sistemas medem rotas frequentemente na esperança de rastrear mudanças de roteamento à medida que elas acontecem.

Medições de rota na Internet são frequentemente coletadas com traceroute [Jacobson 1989, Augustin et al. 2007, Veitch et al. 2009], que envia sondas para identificar

uma sequência de roteadores entre uma origem e um destino. A banda disponível para enviar sondas é finita. Medir rotas para mapear a topologia requer um grande número de sondas e pode levar de vários minutos a alguns dias [Cunha et al. 2011a, Sherwood et al. 2008, Claffy et al. 2009]. É impossível medir com frequência suficiente para detectar todas as mudanças de roteamento sem sobrecarregar a rede. Consequentemente, mapas da topologia da Internet podem estar desatualizados ou inconsistentes pois mudanças de roteamento podem acontecer durante o processo de medição.

Nosso sistema DTRACK rastreia mudanças de roteamento na Internet para manter mapas da topologia da Internet mais atualizados [Cunha et al. 2011b]. O DTRACK separa as tarefas de detectar e remapear mudanças de roteamento. Para detectar mudanças, o DTRACK usa um processo de sondagem leve que combina duas ideias: (1) redirecionar sondas de caminhos estáveis onde mudanças de roteamento são improváveis para caminhos instáveis onde mudanças são mais prováveis; e (2) espalhar sondas de forma uniforme na rede para reduzir medições redundantes. Para remapear mudanças, o DTRACK usa o Paris traceroute [Augustin et al. 2007, Veitch et al. 2009] para medir a nova rota por inteiro. O Paris traceroute é uma versão moderna do traceroute capaz de identificar roteadores que fazem balanceamento de carga. Usamos o Paris traceroute por que é impossível inferir mudanças de roteamento de forma precisa sem informação sobre roteadores que fazem balanceamento de carga [Cunha et al. 2011a].

O DTRACK mantém um banco de dados com a última rota observada em cada um dos caminhos monitorados. Para detectar mudanças, o DTRACK envia uma sonda num ponto s' do caminho e compara a resposta da sonda com a última rota observada. Se a resposta da sonda for incompatível com a última rota observada, e.g., o endereço IP do roteador que enviou a resposta não pertence à última rota observada, uma mudança é detectada e o processo de remapeamento é disparado. Atualmente, o DTRACK remapeia o caminho por inteiro usando o Paris traceroute. Esta abordagem garante medição correta da nova rota, mas desperdiça várias sondas pois mudanças de caminho envolvem poucos roteadores (seção 3). Em particular, esta abordagem ignora duas informações disponíveis quando o processo de remapeamento é disparado: a última rota observada e o ponto s' onde a mudança foi detectada.

Neste artigo propomos RemapRoute, uma ferramenta para reduzir o custo do remapeamento de mudanças de roteamento na Internet (seção 4). Dadas a última rota observada e um ponto onde uma mudança foi detectada, o RemapRoute envia sondas em pontos estratégicos para localizar a mudança e remapeá-la localmente, sem desperdiçar sondas nos roteadores que não estão envolvidos na mudança. Nossa avaliação via simulação dirigida por dados reais mostra que RemapRoute reduz pela metade o custo do remapeamento de 88% das mudanças de roteamento em nossos dados (seção 5). A redução de custo é ainda maior em rotas longas ou com roteadores que fazem balanceamento de carga. Nossa avaliação de um protótipo do RemapRoute usando o PlanetLab confirma nossos resultados via simulação e demonstra a eficácia da ferramenta (seção 6). Sumarizando, neste artigo fazemos as seguintes contribuições:

- Caracterizamos mudanças de roteamento na Internet e mostramos que elas envolvem uma fração pequena dos roteadores numa rota (seção 3);
- Propomos métodos para localizar e remapear mudanças de roteamento que reduzem o desperdício de sondas (seção 4);

- Mostramos que nossa ferramenta reduz o custo de remapeamento de mudanças de rota via simulação e em cenários reais (seções 5 e 6).

A economia de sondas no processo de remapeamento de mudanças de roteamento aumenta o número de sondas disponíveis para mapeamento topológico. Podemos utilizar estas sondas para monitorar mais caminhos na Internet e melhorar a cobertura dos mapas da topologia, ou aumentar a frequência de sondagem e melhorar o rastreamento de mudanças de roteamento. RemapRoute é mais um passo na construção de mapas da topologia da Internet mais completos e consistentes.

2. Definições e fundamentos

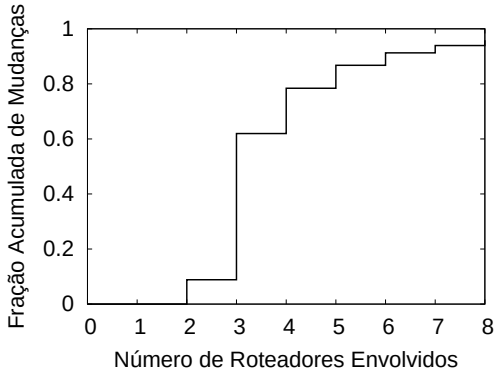
Seguindo a nomenclatura proposta por Paxson [Paxson 1997], chamamos de *caminho virtual* a conectividade entre uma origem e um destino na Internet. Em um dado momento, um caminho virtual é instanciado por uma *rota*. Devido a mudanças de roteamento, um caminho pode ser visto como um processo contínuo $C(t)$ que muda de uma rota a outra ao longo do tempo. Uma rota é composta de *saltos* (*hops*) que são instanciados por roteadores. Saltos são enumerados a partir da origem e nos referimos a um salto s numa rota $C(t)$ por $C(t)[s]$. Uma rota pode ser *simples* se ela tem apenas uma sequência de roteadores da origem ao destino; ou *ramificada* se ela tem roteadores que realizam balanceamento de carga e múltiplas sequências sobrepostas de roteadores da origem ao destino. Todos os saltos numa rota simples têm apenas um roteador e pelo menos um salto numa rota ramificada tem mais de um roteador.

Dadas duas medições consecutivas de um caminho nos instantes $C(t_i)$ e $C(t_{i+1})$, definimos uma *mudança de caminho* como uma sequência de saltos contíguos no novo caminho que altera o caminho antigo. Computamos mudanças de caminho minimizando o número de edições (adição, remoção e substituição de saltos) necessárias para transformar a nova rota na rota antiga. Definimos o *salto de divergência* s_d e o *salto de convergência* s_c de uma mudança como os saltos imediatamente anterior e posterior aos saltos editados pela mudança, respectivamente. Dizemos que o salto de divergência, o salto de convergência e todos os saltos entre eles estão *envolvidos* na mudança de roteamento. Exemplificando, se $C(t_i) = \{a, b, c, d, e, f, g\}$ e $C(t_{i+1}) = \{a, b, e, x, y, g\}$, temos uma mudança com $s_d = 1$ e $s_c = 2$ (remoção de c e d), e outra mudança com $s_d = 2$ e $s_c = 5$ (troca de f por x e y).

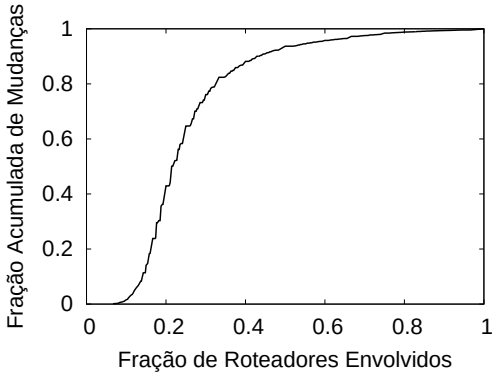
3. Caracterização de mudanças de caminhos

Nesta seção caracterizamos um conjunto de mudanças de caminho reais e verificamos que mudanças de caminho na Internet afetam poucos roteadores.

Implantamos o DTRACK [Cunha et al. 2011b] para medir caminhos em 72 monitores no PlanetLab por uma semana a partir de 4 de março de 2011. Cada monitor escolhe 1.000 destinos aleatoriamente de uma lista com 34.820 destinos alcançáveis escolhidos aleatoriamente na Internet. Em cada monitor, configuramos o DTRACK para rastrear mudanças nos caminhos escolhidos enviando 8 sondas por segundo, similar à taxa de sondagem do DIMES [Shavitt and Weinsberg 2009]. Em uma semana observamos 1.202.960 mudanças de caminho. Os caminhos medidos atravessam 7.315 sistemas autônomos e 97% dos sistemas autônomos de grande porte [Oliveira et al. 2010].



(a) Distribuição do número roteadores envolvidos em mudanças de caminho



(b) Distribuição da fração de roteadores de uma rota envolvidos em mudanças de caminho

Figura 1. Caracterização de mudanças de caminho na Internet

A figura 1(a) mostra a distribuição do número de saltos envolvidos em mudanças de caminho na Internet. O número de saltos envolvidos numa mudança de caminho é o mínimo de saltos que precisamos remapear para saber qual é a nova rota. Vemos que 78% das mudanças de caminho afetam quatro ou menos saltos, um número pequeno visto que a mediana do tamanho das rotas em nossos dados é 17 saltos. O tipo de mudança de caminho mais comum é quando o roteador de apenas um salto muda, resultando em mudanças que envolvem três saltos: o salto onde o roteador mudou mais os saltos de convergência e divergência. Notamos que 9% das mudanças de caminho envolvem dois saltos. Estas mudanças apenas removem saltos da rota antiga (a nova rota está contida na rota antiga) e os únicos saltos envolvidos são os saltos de convergência e divergência. Causas típicas de mudanças de caminho que envolvem dois saltos são falhas de conectividade na Internet e erros de medição onde é impossível medir os saltos atrás da falha ou erro.

A figura 1(b) mostra a distribuição da fração de saltos de um caminho envolvidos numa mudança, i.e., o número de roteadores envolvidos na mudança dividido pelo tamanho da nova rota, para todas as mudanças nos nossos dados. Vemos que 76% das mudanças de caminho envolvem menos de 30% dos saltos no caminho. Isso mostra o potencial do remapeamento local para redução de sondas de medição, se comparado com a abordagem atual de remapear o caminho por inteiro. A curva começa em $x = 0.066 = 2/30$ acontece porque uma mudança envolve pelo menos dois saltos e porque o DTRACK só mede os 30 primeiros saltos num caminho (o padrão do traceroute e do Paris traceroute [Jacobson 1989, Augustin et al. 2007]).

A figura 2 mostra a distribuição do número de sistemas autônomos envolvidos em mudanças de caminho. Convertemos endereços IPs medidos pelo DTRACK em sistemas autônomos combinando as tabelas de mapeamento do Team Cymru[1] e iPlane [Madhyastha et al. 2006]. Cada endereço IP que não aparece na tabela de mapeamento combinada é associado a um sistema autônomo que contém apenas um endereço IP. Essa é uma medida conservadora que pode sobrestimar o número de sistemas autônomos envolvidos em uma mudança. A figura 2 explica porque mudanças de roteamento envolvem poucos saltos. Mesmo fazendo a conversão de endereços IP para sistemas autônomos de forma conservadora, vemos que 60% das mudanças de caminho é interna a um sistema autônomo e apenas 7% envolve mais de dois sistemas autônomos.

[1]Team Cymru, IP to ASN Mapping, http://www.team-cymru.org/Services/ip-to-asn.html

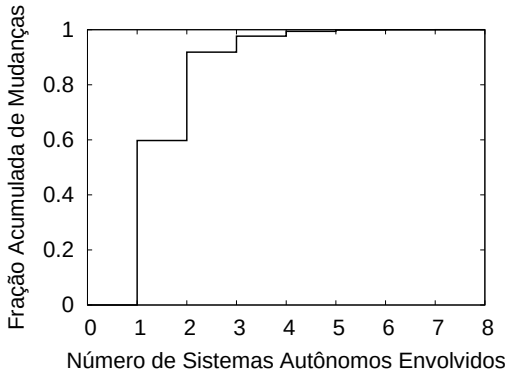


Figura 2. Distribuição do número de sistemas autônomos envolvidos em mudanças de caminho

4. REMAPROUTE

Agora apresentamos nosso algoritmo para remapeamento local de mudanças de caminho implementado no RemapRoute. O algoritmo recebe como entrada a rota antiga observada antes da mudança de roteamento $C(t_{i-1})$ e o salto s' do roteador onde a mudança foi detectada, $C(t_i)[s'] \neq C(t_{i-1})[s']$. Para remapear a mudança de rota, RemapRoute opera em duas etapas: primeiro ele localiza onde a mudança aconteceu (seção 4.1) e depois faz o remapeamento local da mudança (seção 4.2).

O processo de localização e remapeamento da mudança de rota envolve remapear saltos da rota atual e compará-los com saltos na rota antiga. Para remapear cada salto, o RemapRoute envia múltiplas sondas, modificando sistematicamente os campos do cabeçalho IP como o Paris traceroute [Augustin et al. 2007, Veitch et al. 2009], para identificar roteadores que fazem balanceamento de carga.

4.1. Localização da mudança de roteamento

O RemapRoute parte de um salto s' onde o roteador na rota atual é diferente do roteador na rota antiga, $C(t_i)[s'] \neq C(t_{i-1})[s']$. Se o roteador da rota atual no salto s' não pertencer à rota antiga, $C(t_i)[s'] \notin C(t_{i-1})$, então s' é um dos saltos envolvidos na mudança de caminho e podemos passar para a próxima etapa para remapear a mudança (seção 4.2).

Se o roteador da rota atual no salto s' pertencer à rota antiga em outro salto s'', $C(t_i)[s'] = C(t_{i-1})[s'']$, então temos uma mudança de roteamento em saltos anteriores a s' que adicionou ou removeu roteadores na rota. A figura 3 mostra um exemplo de uma mudança de roteamento onde o roteador d foi substituído por dois roteadores x e y. Uma sonda para o sétimo salto detecta uma mudança de roteamento pois a resposta da sonda vem do roteador $g = C(t_i)[7]$, que não é a resposta esperada na rota antiga, $h = C(t_{i-1})[7]$.

Para encontrar um roteador envolvido na mudança de caminho, i.e., um roteador na rota atual que não está presente na rota antiga, o RemapRoute faz uma busca binária no caminho. O RemapRoute inicializa $s_{\text{esq}} = 0$ e $s_{\text{dir}} = s'$. A cada iteração da busca, o RemapRoute remapeia o salto $s = (s_{\text{esq}} + s_{\text{dir}})/2$ e procura o roteador no salto s da rota atual na rota antiga. Se o roteador no salto s da rota atual não pertencer à rota antiga, $C(t_i)[s] \notin C(t_{i-1})$, a busca termina e passamos para a etapa de remapeamento. Se o

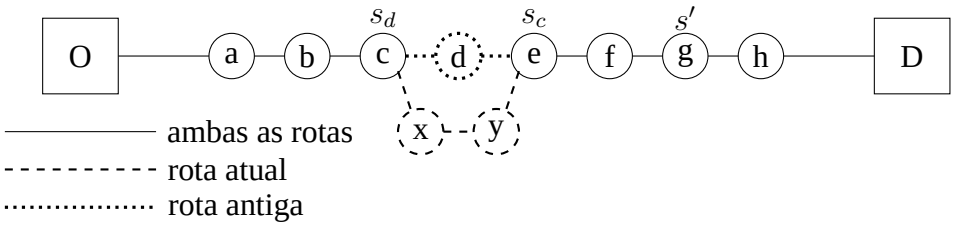


Figura 3. Exemplo de mudança de roteamento com adição de roteadores à rota.

roteador no salto s da rota atual estiver no salto s da rota antiga, $C(t_i)[s] = C(t_{i-1})[s]$, então a mudança está à direita de s e fazemos $s_{\text{esq}} = s$. Se o roteador no salto s da rota atual estiver em outro salto s'' na rota antiga, $C(t_i)[s] = C(t_{i-1})[s'']$, a mudança está à esquerda de s e fazemos $s_{\text{dir}} = s$.

Não temos como comparar a rota atual com a rota antiga se o roteador no salto s não responde a sondas. Neste caso tomamos a decisão conservadora de decrementar s e continuar a busca no salto anterior, sem alterar s_{esq} e s_{dir}.

Se uma mudança de caminho apenas remove saltos da rota antiga, então não existe salto na rota atual que não pertença à rota antiga. Neste caso, o processo de busca termina com $s_{\text{esq}} = s_{\text{dir}} + 1$ e nenhum remapeamento é necessário (seção 4.2).

4.2. Remapeamento local

O remapeamento local parte de um salto s onde o roteador na rota atual não pertence à rota antiga, $C(t_i)[s] \notin C(t_{i-1})$. O RemapRoute remapeia sequencialmente os saltos da rota atual posteriores a s até encontrar o salto de convergência $s_c > s$ com um roteador que pertence à rota antiga, $C(t_i)[s_c] \in C(t_{i-1})$. Se uma das rotas não chega até o destino, o salto de convergência pode não existir. Neste caso, o RemapRoute remapeia a rota atual até o destino ou até o último salto alcançável. Se o caminho não chega até o destino, o RemapRoute identifica o último salto alcançável após encontrar três saltos consecutivos que não respondem a sondas (igual o traceroute).

De forma similar, o RemapRoute remapeia sequencialmente os saltos da rota atual anteriores a s até encontrar o salto de divergência $s_d < s$ com um roteador que pertence à rota antiga, $C(t_i)[s_d] \in C(t_{i-1})$. No pior caso, a busca pelo salto de divergência termina na origem, que pertence a qualquer rota no caminho. Se o salto s_d não for idêntico nas duas rotas, $C(t_i)[s_d] \neq C(t_{i-1})[s_d]$, então existe outra mudança de roteamento anterior a s_d e voltamos à primeira etapa (seção 4.1), fazendo $s' = s_d$, para localizá-la e depois remapeá-la. Esse processo é realizado recursivamente até que não reste nenhuma mudança para remapear. O remapeamento dos roteadores nos saltos entre s_d e s_c precisa ser sequencial pois todos estão envolvidos na mudança de roteamento. Para mudanças de roteamento que apenas removem saltos, temos $s_d = s_{\text{esq}}$ e $s_c = s_{\text{dir}}$ e nenhum remapeamento é necessário.

4.3. Exemplo

Considere que a mudança de roteamento mostrada na figura 3 tenha sido detectada com uma sonda para o quinto salto no caminho. Temos $C(t_{i-1})[5] = f$ e $C(t_i)[5] = e$. Como $e \in C(t_{i-1})$, um nó foi inserido antes do quinto salto e o RemapRoute inicia uma

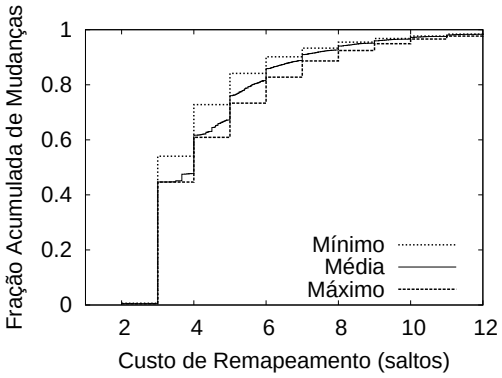


Figura 4. Custo de remapeamento com RemapRoute variando o salto s' de detecção da mudança.

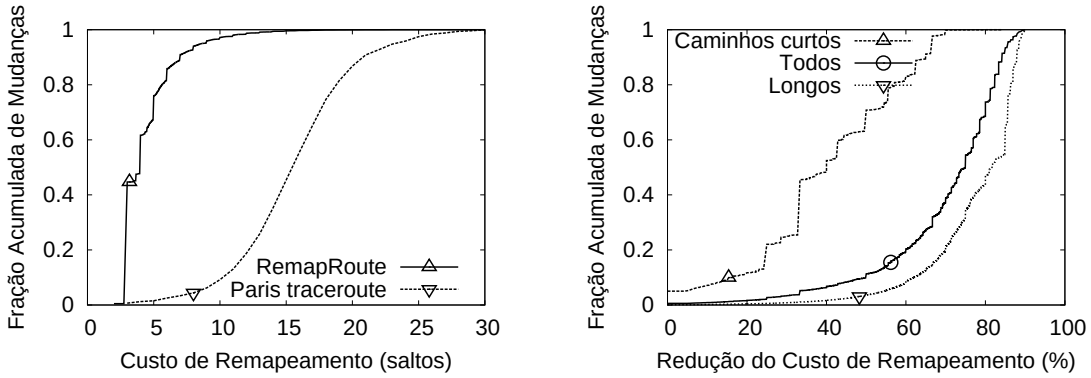
busca binária para identificar o local da mudança. O RemapRoute começa remapeando o segundo salto, onde $C(t_{i-1})[2] = C(t_i)[2] = c$, indicando que a inserção foi realizada entre o segundo e o quinto salto. O RemapRoute remapeia então o terceiro salto, onde $C(t_{i-1})[3] = d$ e $C(t_i)[3] = x$. Como $x \notin C(t_{i-1})$ o RemapRoute passa para a fase de remapeamento local da mudança, onde remapeia o quarto salto e termina.

5. Avaliação via simulação com dados reais

Nesta seção avaliamos o RemapRoute usando simulações dirigidas com dados reais. Utilizamos o mesmo conjunto de dados descrito na seção 3. Nosso foco é comparar o custo de remapear mudanças de caminho usando o Paris traceroute com o custo de remapear usando o RemapRoute.

O custo de remapeamento usando o RemapRoute varia de acordo com o salto s' onde a mudança é detectada. Calculamos o custo de remapeamento do RemapRoute para todos os saltos do caminho onde a mudança pode ser detectada. A figura 4 mostra a distribuição do custo de remapeamento de mudanças para o RemapRoute. Mostramos a distribuição dos custos mínimo, médio e máximo, calculados sobre todos os saltos envolvidos em uma mudança. Vemos que o custo mínimo e máximo em geral são muito próximos. Uma razão para este comportamento é que várias mudanças de caminho envolvem poucos roteadores, logo não existe muita variação em função do salto s' onde a mudança é detectada. No resto deste artigo usaremos o custo médio como representativo do custo de remapeamento com RemapRoute.

A figura 5(a) compara o custo de remapeamento do RemapRoute com o custo de remapeamento do Paris traceroute. Comparando com a figura 1(a) vemos que o RemapRoute frequentemente precisa sondar um número de saltos maior do que o número de saltos envolvidos numa mudança. Este aumento deve-se a mudanças de roteamento que precisam ser localizadas usando a técnica de pesquisa binária antes de serem remapeadas. Independente do processo de localização, o RemapRoute tem custo de remapeamento significativamente menor que o do Paris traceroute. Note que o custo de remapeamento do RemapRoute raramente é menor do que três saltos, mesmo com 9% das mudanças envolvendo apenas dois saltos (figura 1(a)). As mudanças que envolvem apenas dois roteadores apenas removem saltos da rota antiga, e precisamos localizar o salto onde a remoção aconteceu usando busca binária. A busca binária requer pelo menos três sondas a



(a) Custo de remapeamento (b) Redução relativa do custo de remapeamento

Figura 5. Comparação do custo de remapeamento entre RemapRoute e Paris traceroute.

não ser que a mudança seja detectada nos três primeiros saltos do caminho, o que acontece em 0,4% das mudanças em nossos dados.

A figura 5(b) mostra a distribuição da redução no custo de remapeamento quando usamos RemapRoute em vez do Paris traceroute. Calculamos a redução do custo de remapeamento como $(C_{\text{Paris}} - C_{\text{RemapRoute}})/C_{\text{Paris}}$, onde C_{Paris} é o número de saltos remapeados pelo Paris traceroute e $C_{\text{RemapRoute}}$ é o número de saltos remapeados pelo RemapRoute. A linha sólida, calculada para todas as mudanças de caminho em nossos dados, mostra que a redução no custo é significativa. O RemapRoute reduz pra menos da metade o custo de remapeamento de 88% das mudanças de caminho. As linhas tracejadas na figura 5(b) mostram a redução no custo para mudanças em rotas menores que 10 saltos (curtas) e maiores que 20 saltos (longas). Vemos que a redução no custo é mais acentuada para rotas longas, onde o Paris traceroute desperdiça sondas em vários roteadores que não estão envolvidos na mudança, e que o RemapRoute traz redução de custos para remapeamento mesmo em rotas curtas.

5.1. Erros de remapeamento

O RemapRoute remapeia mudanças num caminho dado um salto s' onde uma mudança foi detectada. Isso pode levar a inconsistências caso um caminho sofra duas ou mais mudanças disjuntas antes de detectarmos a primeira mudança. Por exemplo, um caminho $\{a, b, c, d, e, f, g\}$ pode mudar para $\{a, x, c, d, y, f, g\}$ entre duas medições consecutivas com Paris traceroute. Neste caso, podemos detectar duas mudanças nos saltos $s' = 1$ e $s'' = 4$. Infelizmente, dado s' ou s'', o RemapRoute remapeará apenas uma mudança.

Para avaliar a gravidade desse problema, a figura 6 mostra a distribuição do número de mudanças disjuntas para os pares de medições consecutivas dos caminhos em nosso conjunto de dados.[2] Vemos que 79% das medições consecutivas remapeiam apenas uma mudança disjunta, que o RemapRoute remapeará corretamente para qualquer salto s' onde a mudança for detectada. Apenas 3% das medições consecutivas remapeiam três ou mais mudanças disjuntas, indicando que o RemapRoute remapeará a nova rota corretamente na maioria dos casos.

[2]No nosso conjunto de dados só medimos caminhos com Paris traceroute quando uma mudança é detectada. Todos os pares de medições consecutivas remapeiam pelo menos uma mudança disjunta.

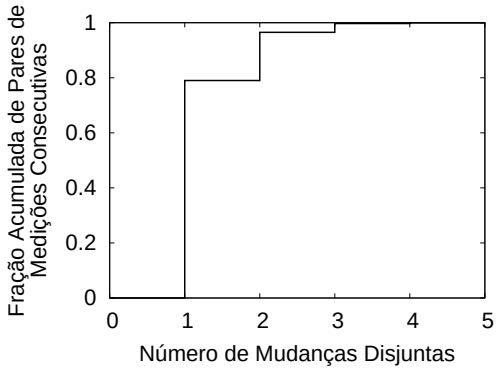


Figura 6. Distribuição do número de mudanças disjuntas entre duas medições com Paris traceroute.

Três outros fatores contribuem para minimizar o impacto de mudanças disjuntas. Primeiro, o RemapRoute pode remapear mudanças disjuntas anteriores ao salto de detecção s' caso ele as detecte durante o processo de remapeamento. Em particular, a probabilidade do RemapRoute remapear uma mudança disjunta anterior ao salto de detecção s' no nosso conjunto de dados é 43%. Segundo, uma mudança disjunta que não for remapeada quando executarmos RemapRoute será detectada e remapeada no futuro (assumindo que o processo de sondagem para detecção de mudanças de caminho seja capaz de detectar todas as mudanças possíveis). Qualquer mudança disjunta que não for remapeada causa apenas uma inconsistência temporária nos dados. Terceiro, a redução do custo de remapeamento obtida com RemapRoute pode ser utilizada para aumentar a frequência de sondagem para detecção de mudanças e reduzir a chance de ocorrerem duas mudanças em um caminho antes de detectarmos a primeira.

Outra limitação do RemapRoute é que o mecanismo de busca binária pode falhar quando a ordem relativa de dois saltos se inverte da rota antiga para a nova rota. Um exemplo extremo, mas ilustrativo, é uma mudança de $C(t_{i-1}) = \{a, b, c, d, e, f\}$ para $C(t) = \{a, e, d, c, b, f\}$. Apenas 0,9% das mudanças de caminho em nossos dados invertem a ordem relativa de saltos. Como inversão da ordem relativa de saltos é um evento raro, tomamos a decisão conservadora de remapear o caminho por inteiro (como o Paris traceroute) quando uma inversão é detectada durante o processo de remapeamento. Como mostram os resultados anteriores, essa limitação não compromete a utilidade do RemapRoute.

6. Avaliação com protótipo real no PlanetLab

Nesta seção avaliamos um protótipo do RemapRoute no PlanetLab. Implantamos o Paris traceroute e o RemapRoute em 140 nós PlanetLab e coletamos medições por 18 horas de 30 de Novembro de 2012. Cada nó executa o Paris traceroute para medir caminhos periodicamente. Como no conjunto de dados utilizado nas seções anteriores, cada nó monitora caminhos para 1.000 destinos escolhidos aleatoriamente de uma lista com 34.820 destinos alcançáveis na Internet. Quando duas medições consecutivas com Paris traceroute detectam uma mudança, executamos o RemapRoute para remapeá-la. Cada nó leva em média 7 horas e 42 minutos para medir os 1.000 caminhos. Devido à baixa frequência de sondagem, este conjunto de dados contém apenas 87.848 mudanças. Os caminhos medidos

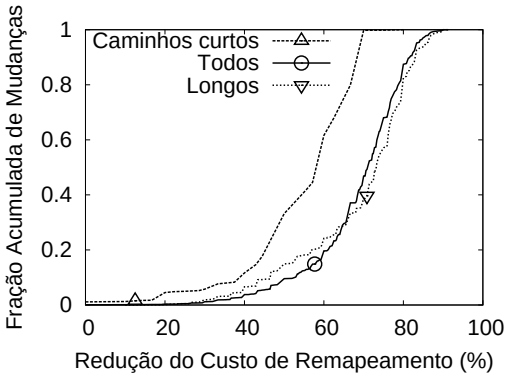


Figura 7. Redução do custo de remapeamento com RemapRoute em cenários reais.

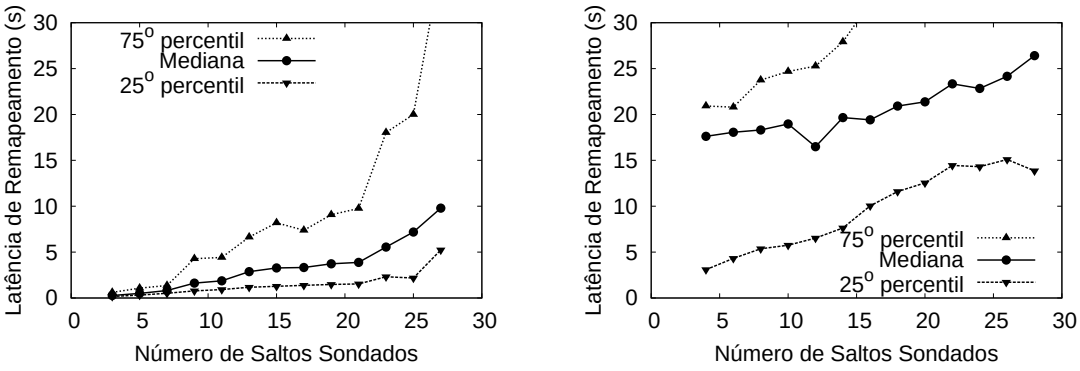
atravessam 7.143 sistemas autônomos e 95% dos sistemas autônomos de grande porte na Internet [Oliveira et al. 2010].

A figura 7 mostra a redução do custo de remapeamento quando usamos o RemapRoute em vez do Paris traceroute. Comparando com a figura 5(b), a redução média do custo de remapeamento no cenário real é quantitativamente similar aos resultados obtidos via simulação (linha sólida). Por exemplo, reduzimos pra menos da metade o custo de remapeamento de 90% das mudanças de caminho no cenário real.

A redução de custo para rotas curtas e longas é mais similar à redução de custo geral no cenário real que nas simulações. Em outras palavras, as linhas tracejadas na figura 7 estão mais próximas da linha sólida que na figura 5(b). Atribuímos essa mudança a três fatores: (i) o menor número de mudanças observadas no cenário real pode limitar a variedade de mudanças observadas; (ii) diferenças no conjunto de caminhos monitorados; e (iii) a diferente forma de detecção de mudanças (os dados utilizados nas simulações foram coletados com o DTRACK).

A figura 8(a) mostra os 25^o, 50^o e o 75^o percentis da latência de remapeamento em função do número de saltos sondados durante o processo de remapeamento. Avaliamos a latência de remapeamento por que o RemapRoute sonda saltos sequencialmente: a decisão do próximo salto a sondar depende do resultado do último salto sondado. O Paris traceroute, em contrapartida, poderia paralelizar a sondagem de saltos (apesar da implementação padrão não fazê-lo). Como a maior parte dos remapeamentos com RemapRoute requer sondagem de poucos saltos (figura 5(a)), a latência de remapeamento geralmente é menor que 5 segundos. A figura 8(b) mostra os 25^o, 50^o e 75^o percentis da latência de remapeamento com Paris traceroute no cenário real. Nosso objetivo não é comparar a latência de remapeamento do RemapRoute com Paris traceroute, pois a latência é diretamente afetada por decisões de implementação da ferramenta. Nosso objetivo é mostrar que a latência de remapeamento com RemapRoute é aceitável para uso em sistemas reais. Notamos ainda que um sistema de mapeamento topológico como o DTRACK pode executar o RemapRoute simultaneamente em caminhos diferentes caso mais de uma mudança seja detectada num curto intervalo de tempo.

Por último, avaliamos se o remapeamento com o RemapRoute é equivalente a utilizar o Paris traceroute para medir o novo caminho por inteiro. Para cada mudança



(a) RemapRoute (b) Paris traceroute

Figura 8. Latência de remapeamento em cenários reais.

observada no cenário real, comparamos os saltos remapeados pelo RemapRoute com a rota medida pelo Paris traceroute. Apenas 0,6% das medições com RemapRoute são diferentes das medições com Paris traceroute. A identificação de roteadores que fazem balanceamento de carga usando Paris traceroute ou RemapRoute é probabilística [Veitch et al. 2009]. Por exemplo, a configuração padrão do Paris traceroute e do RemapRoute identifica todos os roteadores que fazem balanceamento de carga numa rota com 95% de confiabilidade. Erros de medição são inevitáveis e causam diferenças entre remapeamentos independente da ferramenta utilizada. Outra causa para diferença nos remapeamentos são mudanças de roteamento que acontecem no intervalo entre a medição com Paris traceroute e a medição com RemapRoute.

Sumarizando, o remapeamento de mudanças de caminho na Internet com RemapRoute é tão preciso quanto remapeamento com Paris traceroute, reduz significativamente o número de sondas enviadas e tem pouco impacto na latência de remapeamento.

7. Trabalhos relacionados

Operadores podem utilizar mensagens dos protocolos de roteamento (e.g., OSPF e IS-IS) e arquivos de configuração dos roteadores para mapear a topologia de sua rede [Feamster and Balakrishnan 2005, Turner et al. 2010, Markopoulou et al. 2008]. Esta abordagem resulta em mapas completos e precisos da topologia, mas só está disponível para operadores de redes e é restrita a uma única rede. Para mapear múltiplas redes, podemos utilizar coletores públicos de mensagens BGP[3] para construir um mapa dos sistemas autônomos da Internet [Oliveira et al. 2010, Dhamdhere and Dovrolis 2008]. Infelizmente, o BGP não expõe todos os enlaces da rede e coletores públicos de mensagens BGP não cobrem todos os sistemas autônomos da Internet [Oliveira et al. 2010, Cohen and Raz 2006]. Neste trabalho tomamos a abordagem ortogonal de medir a topologia da rede no nível de roteadores usando medições ativas.

Pesquisa sobre mapeamento topológico no nível de roteadores usando medições ativas têm três objetivos principais: (i) aumentar a cobertura da Internet, (ii) aumentar a precisão da topologia construída e (iii) aumentar a frequência das medições.

[3]The University of Oregon Routeviews Project, http://www.routeviews.org
RIPE Routing Information Service, http://www.ripe.net/data-tools/stats/ris

A abordagem clássica para aumentar a cobertura da Internet é monitorar um grande número de caminhos. A plataforma Skitter/Ark da CAIDA [Claffy et al. 2009] tenta cobrir toda a Internet usando alguns monitores para medir caminhos para todos os prefixos $/24$ anunciados na Internet. O problema é que o Skitter/Ark demora de dois a três dias para coletar a topologia devido ao grande número de caminhos monitorados e limitações de banda nos monitores. Uma alternativa é dividir a carga de sondagem das medições entre vários monitores, como nos sistemas DIMES [Shavitt and Weinsberg 2009] e Ono [Choffnes et al. 2010]. Neste trabalho assumimos que o conjunto de monitores e destinos é fixo. Porém, utilização do RemapRoute é ortogonal ao conjunto de monitores e destinos. O RemapRoute pode ser usado por qualquer um dos sistemas acima para reduzir a sobrecarga de rede.

Técnicas para aumentar a precisão da topologia coletada tentam inferir mais informações sobre a rede do que medições tradicionais com traceroute. O Paris traceroute envia sondas adicionais variando sistematicamente os valores nos campos do cabeçalho IP para detectar todos os roteadores que fazem balanceamento de carga em uma caminho [Augustin et al. 2007, Veitch et al. 2009]. O RemapRoute detecta roteadores que fazem balanceamento de carga utilizando o mesmo algoritmo que o Paris traceroute. O DisCarte usa traceroute e sondas com a opção *Record Route* do protocolo IP ativada para coletar duas sequências relacionadas de roteadores em caminhos da Internet [Sherwood et al. 2008]. O DisCarte pós-processa essas sequências com ferramentas de aprendizado de máquina para combiná-las em uma topologia mais precisa. Estas e outras técnicas enviam sondas adicionais e aumentam o custo de mapeamento da topologia. O objetivo do RemapRoute é complementar: reduzir o custo do remapeamento de mudanças de roteamento, aumentando a disponibilidade de sondas para coleta de topologias mais precisas.

Nosso trabalho é mais relacionado com técnicas para aumentar a frequência de medições da topologia da Internet. Em geral, monitores têm banda de rede limitada para mapear a topologia. Reduzir o custo de cada medição da topologia aumenta diretamente a frequência com a qual medições podem ser coletadas. O RocketFuel, por exemplo, reduz o custo para mapear a topologia de um sistema autônomo alvo descartando caminhos que têm roteadores de ingresso e egresso no sistema autônomo alvo idênticos a outro caminho já medido [Spring et al. 2002]. Outra abordagem é reduzir o custo de medições escolhendo apenas um destino em cada sub-rede num sistema autônomo [Beverly et al. 2010]. O Doubletree reduz sondas redundantes nos roteadores próximos a monitores (compartilhados pelos caminhos partindo do monitor) e nos roteadores próximos a destinos (compartilhados pelos caminhos que terminam no destino) [Donnet et al. 2005]. O RemapRoute foi desenvolvido para reduzir o custo do remapeamento de mudanças no DTRACK, nosso sistema de mapeamento topológico [Cunha et al. 2011b]. O RemapRoute e o DTRACK complementam as técnicas existentes para redução do custo de mapeamento topológico. O DTRACK reduz sondas redundantes para roteadores próximos aos monitores como o Doubletree e é compatível com técnicas descritas acima para selecionar quais caminhos mapear.

8. Conclusões e trabalhos futuros

A manutenção de mapas completos e atualizados da Internet é difícil devido à grande quantidade de sondas necessárias e restrições de banda nos monitores. Neste trabalho propomos o RemapRoute, uma ferramenta para reduzir o custo de remapeamento de

mudanças de roteamento na Internet. Dadas a rota anterior a uma mudança de roteamento e um salto (*hop*) onde a mudança foi detectada, o RemapRoute (1) realiza uma busca binária pelo salto onde a mudança aconteceu e (2) faz o remapeamento local da mudança. Comparado com a abordagem atual de remapear a nova rota por inteiro usando traceroute, o RemapRoute reduz significativamente o número de saltos sondados para remapear mudanças de roteamento na Internet. Essa redução de saltos sondados aumenta a disponibilidade de sondas, potencializando a medição de mais caminhos ou aumento da frequência de medições. O remapeamento com RemapRoute é tão preciso quanto remapear o caminho inteiro com traceroute, e a latência de remapeamento é satisfatória para utilização do RemapRoute em sistemas reais. RemapRoute é mais um passo na construção de mapas da topologia da Internet mais completos e atualizados.

Como trabalho futuro, pretendemos integrar o RemapRoute no DTRACK e prover um serviço de mapeamento da Internet aberto para disponibilizar informações sobre a topologia para pesquisadores e aplicações. Queremos também reduzir ainda mais o custo de remapeamento de mudanças no DTRACK. Atualmente, o DTRACK remapeia cada caminho separadamente. Esta abordagem desperdiça sondas caso vários caminhos sejam afetados pela mesma mudança. Pretendemos desenvolver mecanismos para prever quais caminhos são afetados por uma mesma mudança e remapear apenas um deles.

Referências

Augustin, B., Friedman, T., and Teixeira, R. (2007). Measuring Load-balanced Paths in the Internet. In *Proc. IMC*.

Beverly, R., Berger, A., and Xie, G. (2010). Primitives for Active Internet Topology Mapping: Toward High-Frequency Characterization. In *Proc. IMC*.

Choffnes, D. R., Bustamante, F. E., and Ge, Z. (2010). Crowdsourcing Service-level Network Event Monitoring. In *Proc. ACM SIGCOMM*.

Claffy, K., Hyun, Y., Keys, K., Fomenkov, M., and Krioukov, D. (2009). Internet Mapping: from Art to Science. In *Proc. IEEE CATCH*.

Cohen, R. and Raz, D. (2006). The Internet Dark Matter - on the Missing Links in the AS Connectivity Map. In *Proc. IEEE INFOCOM*.

Cunha, I., Teixeira, R., and Diot, C. (2011a). Measuring and Characterizing End-to-End Route Dynamics in the Presence of Load Balancing. In *Proc. PAM*.

Cunha, I., Teixeira, R., Veitch, D., and Diot, C. (2011b). Predicting and Tracking Internet Path Changes. In *Proc. ACM SIGCOMM*.

Dhamdhere, A. and Dovrolis, C. (2008). Ten Years in the Evolution of the Internet Ecosystem. In *Proc. IMC*.

Dhamdhere, A., Teixeira, R., Drovolis, C., and Diot, C. (2007). NetDiagnoser: Troubleshooting Network Unreachabilities Usind End-to-end Probes and Routing Data. In *Proc. ACM CoNEXT*.

Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. (2002). Globally Distributed Content Delivery. *IEEE Internet Computing*, 6(5):50–58.

Donnet, B., Raoult, P., Friedman, T., and Crovella, M. (2005). Efficient Algorithms for Large-scale Topology Discovery. In *Proc. ACM SIGMETRICS*.

Duffield, N. (2006). Network Tomography of Binary Network Performance Characteristics. *IEEE Trans. on Inf. Theory*, 52(12):5373–5388.

Feamster, N. and Balakrishnan, H. (2005). Detecting BGP Configuration Faults with Static Analysis. In *Proc. USENIX NSDI*.

Jacobson, V. (1989). traceroute. Available at ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

Katz-Bassett, E., Scott, C., Choffnes, D. R., Cunha, I., Valancius, V., Feamster, N., Madhyastha, H. V., Anderson, T., and Krishnamurthy, A. (2012). LIFEGUARD: Practical Repair of Persistent Route Failures. In *Proc. ACM SIGCOMM*.

Kompella, R., Yates, J., Greenberg, A., and Snoeren, A. (2007). Detection and Localization of Network Blackholes. In *Proc. IEEE INFOCOM*.

Madhyastha, H., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., and Venkataramani, A. (2006). iPlane: an Information Plane for Distributed Services. In *Proc. USENIX OSDI*.

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.

Oliveira, R., Pei, D., Willinger, W., Zhang, B., and Zhang, L. (2010). Quantifying the Completeness of the Observed Internet AS-level Structure. *IEEE/ACM Trans. Netw.*, 18(1):109–122.

Paxson, V. (1997). End-to-end Routing Behavior in the Internet. *IEEE/ACM Trans. Netw.*, 5(5):601–615.

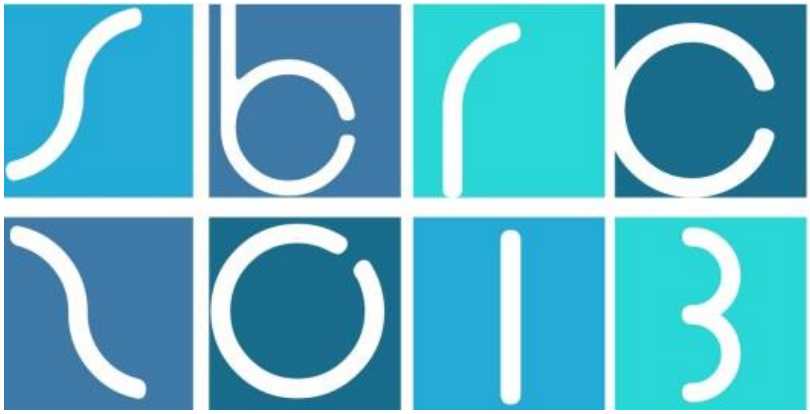
Shavitt, Y. and Weinsberg, U. (2009). Quantifying the Importance of Vantage Points Distribution in Internet Topology Measurements. In *Proc. IEEE INFOCOM*.

Sherwood, R., Bender, A., and Spring, N. (2008). DisCarte: a Disjunctive Internet Cartographer. In *Proc. ACM SIGCOMM*.

Spring, N., Mahajan, R., and Wetherall, D. (2002). Measuring ISP Topologies with Rocketfuel. In *Proc. ACM SIGCOMM*.

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*.

Veitch, D., Augustin, B., Friedman, T., and Teixeira, R. (2009). Failure Control in Multipath Route Tracing. In *Proc. IEEE INFOCOM*.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 5

Redes de Sensores sem Fio: Gerência e Operação

Localização 3D em Redes de Sensores Sem Fio Utilizando Veículo Aéreo não Tripulado

Leandro A. Villas[1,5], Daniel L. Guidoni[2], Azzedine Boukerche[3], Antonio A. F. Loureiro[4] e Jo Ueyama[5]

[1]Instituto de Computação – UNICAMP
[2]Departamento de Ciência da Computação – UFSJ
[3]PARADISE Research Laboratory – University of Ottawa
[4]Departamento de Ciência da Computação – UFMG
[5]Instituto de Ciências Matemáticas e de Computação – USP
(leandro,joueyama)@icmc.usp.br, guidoni@ufsj.edu.br
boukerch@site.uottawa.ca e loureiro@dcc.ufmg.br

Abstract. *A wireless sensor network (WSN) is designed to perform event detection, data collection, and reporting of such data to a monitoring station. In many cases, it is necessary to know the location of sensor nodes to relate the detection of the event at a specific location. However, the geographical location of the sensor nodes in most applications can only be set after their deposition in the area of interest. Therefore, for the sensor nodes to know their location, it is necessary to use specific algorithms to solve the problem of discovering the geographical position of sensor nodes. This work addresses the problem of 3D localization in WSNs using an unmanned aerial vehicle (UAV). The UAV is equipped with GPS and it flies over the monitoring area broadcasting its geographical position. Thus, the sensor nodes are able to estimate their geographical position without being equipped with GPS device. Simulation results show that using an UAV leads to a smaller error in the calculation of geographic location as compared to solutions presented in the literature.*

Resumo. *Uma rede de sensores sem fio (RSSF) é projetada para executar detecção de eventos, coleta de dados e o relato de desses dados para uma estação de monitoramento. Em muitos casos, é necessário saber a localização dos nós sensores para relacionar a detecção do evento à uma localização específica. Porém, a localização geográfica dos nós na maioria das aplicações só pode ser definida após a sua deposição na área de interesse. Logo, para os nós saberem a sua localização, é necessário a utilização de algoritmos específicos para resolver o problema de descoberta da posição geográfica dos nós sensores. Este trabalho aborda o problema de localização 3D em RSSFs utilizando um veículo aéreo não tripulado (VANT). O VANT é equipado com GPS e sobrevoa a área de monitoramento disseminando a sua posição. Dessa forma, os nós sensores são capazes de encontrarem a sua posição geográfica sem serem equipados com GPS. Resultados de simulação mostraram que a abordagem utilizando o VANT produz um erro menor no cálculo da posição geográfica em comparação a soluções propostas na literatura.*

1. Introdução

Uma rede de sensores sem fio (RSSF) [Akyildiz et al. 2002, Boukerche 2008, Romer and Mattern 2004, Villas et al. 2010] é projetada para detectar eventos de interesse de uma aplicação, coletar dados referentes a esses eventos e enviar um relato desses dados para uma estação de monitoramento. Tipicamente, a informação coletada e os nós sensores devem ser localizados no espaço para identificar a localização do evento de interesse. Este posicionamento é feito por um sistema de localização, que é uma parte fundamental das RSSFs. Esse sistema é usado não apenas na localização de eventos, mas também na identificação e correlação de dados recolhidos [Villas et al. 2011b], endereçamento dos nós [Heidemann et al. 2001], gestão e consulta de nós localizados em uma determinada região [Navas and Imielinski 1997], construção de mapas de energia [Mini et al. 2004], roteamento geográfico [Villas et al. 2011a], rastreamento de objetos [Kumar et al. 2000], e muitos outros algoritmos que de alguma forma fazem uso de informações geográficas.

A definição de um sistema de localização para nós sensores [Boukerche et al. 2008a, Boukerche et al. 2008b, Bulusu et al. 2004, Doherty et al. 2001, de Oliveira et al. 2009a, de Oliveira et al. 2009b, Rudafshani and Datta 2007, Savvides et al. 2001] é um pré-requisito necessário para que muitas aplicaões de RSSFs se tornem viáveis. O problema de localização em uma rede de sensores consiste em identificar a localização física (por exemplo, latitude, longitude e altitude) dos nós sensores. A importância de um sistema de localização em RSSFs surge a partir da necessidade de nomear os dados coletados [Heidemann et al. 2001] e eventos associados com o seu local de ocorrência [Intanagonwiwat et al. 2000]. Além disso, alguns algoritmos de roteamento usam informações de localização para melhorar seu desempenho como, por exemplo, o *Dynamic and Scalable Tree* [Villas et al. 2011a], que otimiza o processo de encontrar pontos de agregação de dados criando as rotas considerando as posições geográficas dos nós. No entanto, dependendo da precisão da informação de localização, tais rotas podem não alcançar os nós corretos para encontrar pontos de agregação.

Tipicamente, os sistemas de localização propostos para RSSFs usam uma abordagem recursiva [de Oliveira et al. 2009a], onde um nó estima a sua localização com base em informações de posição de três ou mais nós de referência (vizinhos que conhecem as suas posições). Uma vez que a sua posição é calculada, o nó torna-se uma referência (âncora) e transmite a informação com sua própria localização para auxiliar os outros nós na estimativa de suas posições. Essa abordagem apresenta alguns inconvenientes, como a propagação do erro de localização. Isto significa que o erro na estimativa de posição de um nó pode ser utilizado por outros nós para estimar as suas próprias posições, aumentando ainda mais o erro da posição estimada. Além disso, o nó deve ter pelo menos três vizinhos de referência para poder calcular a sua posição.

O objetivo deste trabalho é usar uma estratégia diferente, que não seja baseada em nós *beacons*[1] e que não gere a propagação do erro de sistemas de localização existentes. Assim, propomos um sistema de localização 3D diferente que apresenta três contribuições para a área de sistemas de localização em RSSFs. A principal contribuição é que todos os nós da rede são aptos a calcular a sua posição geográfica com alta precisão. Além disso, o algoritmo proposto é eficiente tanto para redes esparsas quanto densas, diferente da mai-

[1]Nós que conhecem a sua posição geográfica utilizando, por exemplo, um receptor GPS.

oria das soluções da literatura que apresenta alguns inconvenientes, como baixa precisão na estimativa de posição em cenários de rede esparsa. Por fim, o sistema de localização proposto reduz o custo da rede drasticamente, uma vez que necessita de apenas um nó (VANT) munido com receptor GPS.

Este trabalho esta organizado da seguinte forma. A próxima seção apresenta as definições de sistemas de localização em RSSFs. Os trabalhos relacionados são apresentados na seção 3. A seção 4 descreve o algoritmo proposto para o problema de localização 3D em RSSFs. A seção 5 apresenta os resultados de simulação. Finalmente, a seção 6 apresenta as conclusões e trabalhos futuros.

2. Sistemas de Localização em Redes de Sensores sem Fio

Uma RSSF é composta por n nós sensores, com raio de comunicação r_c e os mesmos são distribuídos em um campo tridimensional. A 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, isto é, a distância entre v_i e v_j é menor do que r_c;
- $w(e) \leq r_c$ é o peso da aresta $e = \langle i, j \rangle$, isto é, a distância entre v_i e vj.

A seguir, estão definidos alguns termos que podem ser utilizados para designar o estado de um nó sensor na rede.

Definição 1 (Nós Desconhecidos – D) *este termo refere-se aos nós da rede que não sabem a sua localização ainda. O principal objetivo de um sistema de localização é permitir que esses nós possam estimar suas posições.*

Definição 2 (Nós com Posições Estimadas – P) *esses nós eram inicialmente nós desconhecidos que conseguiram estimar suas posições usando o sistema de localização. O número de nós com posições estimadas e o erro de posição estimada destes nós são os parâmetros principais para determinar a qualidade de um sistema de localização.*

Definição 3 (Nós de Referência – R) *esses nós não precisam de um sistema de localização para estimar suas posições físicas. A localização desses nós é obtida por colocação manual ou por meios externos, como o GPS. Esses nós formam a base da maioria dos sistemas de localização para RSSFs.*

O problema da localização pode ser definido como segue:

Definição 4 (Problema de Localização) *dada uma rede $G = (V, E)$ e um conjunto de nós R e suas posições (x_r, y_r, z_r), para todo $r \in R$, queremos encontrar a posição (x_u, y_u, z_u) do maior número possível de $u \in D$, assim transformando os nós desconhecidos D em nós com posições estimadas P.*

A maneira mais simples para resolver o problema da localização em uma RSSF é munir cada nó sensor da rede com um receptor GPS. Esta solução apresenta algumas vantagens, como um erro de localização relativamente pequeno (2–15 m, dependendo do receptor GPS) e bastante preciso, uma vez que o erro seria semelhante para todos os nós sensores da rede. Porém, esta solução apresenta uma série de inconvenientes como aumento do tamanho dos sensores, falta de visada dos satélites e, por fim, aumento no

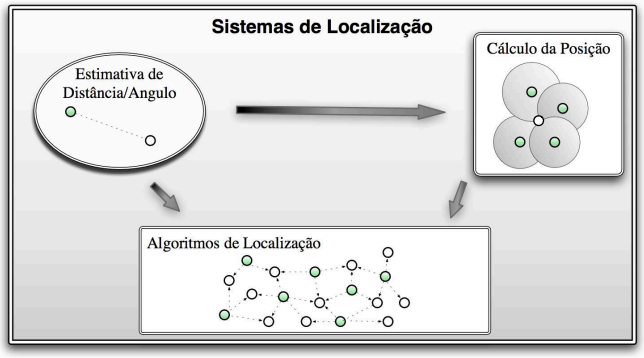


Figura 1. Sistemas de Localização divididos em três componentes distintos.

consumo de energia e do custo dos nós sensores. Esta solução torna-se impraticável para redes com centenas ou milhares de nós sensores, o que nos leva à necessidade de projetar sistemas de localização.

Os sistemas de localização podem ser dividido em três componentes distintos (veja a figura 1):

1. *Estimativa de Distância/Ângulo*: este componente é responsável por estimar as informações a respeito da distância e/ou ângulo entre dois nós. Técnicas utilizadas neste componente incluem indicador de intensidade do sinal recebido (RSSI), o tempo de chegada (ToA), o tempo diferença de chegada (TDoA), número de saltos, ou o ângulo de chegada (AoA).
2. *Cálculo da Posição*: este componente é responsável por calcular a posição de um nó com base na informação disponível sobre as distâncias/ângulos e posições dos nós de referência. Algumas técnicas usadas para calcular uma posição incluem trilateração, multilateração ou triangulação.
3. *Algoritmo de Localização*: este é o principal componente de um sistema de localização. Ele determina como a informação disponível será manipulada para permitir que a maioria ou todos os nós da rede de sensores sem fio possam estimar as suas posições.

O desempenho de um sistema de localização depende diretamente de cada um desses componentes. Além disso, cada componente tem seu próprio objetivo e métodos de solução.

3. Trabalhos Relacionados

Esta seção discute algumas abordagens da literatura para o problema de localização em RSSFs.

No Sistema de Posicionamento Ad Hoc – APS [Niculescu and Nath 2001], um reduzido número de nós *beacons* (por exemplo, três ou mais) é depositado junto com os nós que não conhecem suas posições geográficas. Em seguida, cada nó calcula a distância para os nós *beacons*, através de *multihops*. Uma vez que estas distâncias são estimadas, os nós podem calcular suas posições usando a triangulação. Três métodos de propagação de distância *hop by hop* são propostos: Dv-Hop, Dv-Distância, e a Euclidiana. No APS

Dv-Hop, os nós *beacons* iniciam a propagação de suas informações de posição. Todos os nós da rede recebem as informações de posição de todos os nós *beacons*, bem como o número de saltos para eles (nós *beacons*). Quando um nó *beacon* recebe a informação sobre a posição dos outros nós *beacon*, pode-se considerar que ele tem informações suficientes para calcular o tamanho médio de um salto com base na sua própria posição (em relação aos outros *beacons*), e também no número de saltos entre eles. Este último valor é então inundado de maneira controlada para toda a rede, como um mecanismo (fator) de correção. Quando um nó desconhecido recebe tal fator de correção, então ele é capaz de converter a sua distância entre os nós *beacons* do número de saltos (*hops*) para metros. Uma vantagem do APS é a de que o seu algoritmo de localização requer apenas um pequeno número de nós *beacon*para que possa funcionar. No entanto, a maneira como as distâncias são propagadas, em particular no Dv-Hop e no Dv Distância, assim como a forma como estas distâncias são convertidas do número de *hops* para metros (no Dv-Hop), resultam em cálculos de posicionamentos imprecisos, o que aumenta o erro de localização final do sistema.

No algoritmo de Estimativa de Posicionamento Recursivo – RPE [Albowicz et al. 2001], os nós estimam as suas posições com base em um conjunto de nós *beacons* iniciais (por exemplo, 5 a 10% dos nós). Nós *beacons* são cientes de suas posições (por exemplo, nós equipados com GPS). O algoritmo RPE é dividido em quatro fases. Na primeira fase, os nós *beacons* começam a transmitir a sua informação de posição, de modo que esses nós possam ser usados como nós de referência. Na segunda fase, um nó calcula a distância para os nós de referência, utilizando, por exemplo, o RSSI. Na terceira fase, o nó calcula a sua posição através da utilização, por exemplo de triangulação. Na fase final, o nó se torna uma referência e transmite a sua posição estimada para ajudar os seus vizinhos com as suas estimativas de posição. Uma vantagem deste algoritmo é que o número de nós de referência aumenta rapidamente, de tal maneira que a maioria dos nós possam calcular as suas posições. No entanto, esta técnica tem a desvantagem de propagar o erro de localização. Isto significa que a estimativa de posição imprecisa de um nó pode ser utilizada por outros nós para estimar as suas próprias posições, aumentando ainda mais a imprecisão. Além disso, o nó deve ter pelo menos três vizinhos de referência para poder calcular a sua posição.

No algoritmo de Localização com um Beacon Móvel – MBL [Sichitiu and Ramadurai 2004], o *beacon* móvel locomove-se no campo onde os nós sensores localizam-se e o nó *beacon* móvel transmite periodicamente mensagens com informações sobre a sua posição geográfica. Quando um nó sensor recebe três ou mais mensagens com as coordenadas do *beacon* móvel, então ele calcula a sua posição usando uma abordagem probabilística, com base nas coordenadas recebidas e nas estimativas de distância do RSSI. Uma vantagem deste algoritmo é que as estimativas de posição são calculadas com base no mesmo nó (*beacon* móvel), fazendo com que a média do erro de localização seja baixa e impedindo a sua propagação. Por outro lado, os autores consideram como nó *beacon* um robô, mas tipicamente os robôs terão dificuldade para se locomover em ambientes de difícil acesso. Além disso, o trabalho não considera o problema de localização em ambientes 3D.

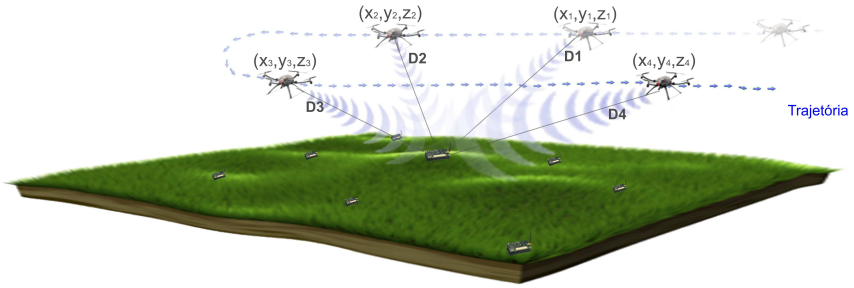


Figura 2. Sistema de localização proposto.

4. Sistema de Localização 3D Proposto

Esta seção apresenta a solução proposta para o problema de localização 3D utilizando um Veículo Aéreo Não Tripulado (VANT). Inicialmente, todos os nós sensores pertencem ao conjunto D e o VANT pertence ao conjunto R.

A figura 2 ilustra o sistema de localização 3D proposto que faz a utilização de um VANT. O VANT sobrevoa a área de monitoramento onde os nós foram depositados. Durante o vôo, o VANT faz um *broadcast* periódico da sua posição geográfica. Quando um nó recebe quatro ou mais mensagens contendo a posição do VANT, o nó pode calcular a sua posição. É importante ressaltar que para calcular a posição em um sistema de três dimensões, são necessários 4 pontos de referência, ao invés de 3 pontos de referência em um sistema de duas dimensões. A seguir, apresentaremos em detalhes todos os componentes do sistema de localização (ilustrados na figura 1) para o funcionamento do sistema de localização 3D utilizando VANT proposto, que são: Estimativa da Distância, Cálculo da Posição e Algoritmo de Localização. Após, apresentaremos a aplicabilidade do sistema de localização proposto.

4.1. Estimativa de Distância

Existem vários métodos para estimar a distância entre dois nós. O método mais utilizado é o *RSSI*, já que ele não necessita de nenhum *hardware* extra além do rádio transmissor/receptor embutido no nó sensor. Além disso, a técnica de *RSSI* não necessita de nenhuma mensagem de controle para que a estimativa da distância em relação ao nó que enviou seja feita, o que é observado nas outras técnicas de estimativa da distância descritas anteriormente. A figura 3 ilustra a força do sinal quando um nó envia uma mensagem considerando três dimensões. O nó envia um sinal com uma determinada força, que reduz a medida que o sinal é propagado. Neste caso, se o nó que recebeu o sinal está distante do nó que o enviou, a força do sinal recebido será baixa.

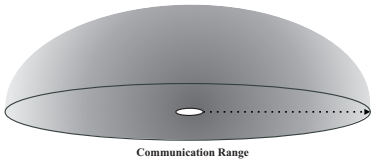


Figura 3. Decremento na força do sinal.

A tabela 1 exemplifica dois modelos de propagação de sinais conhecidos na literatura [Boukerche 2008]. Utilizando um modelo de propagação, a força do sinal pode ser

Modelo	Descrição	Fórmula
Free Space	Considera uma condição de propagação ideal, sem interferências ou obstáculos	$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L}$
Two-Ray Ground	Similar ao *Free Space*, mas considera a possibilidade de reflexão do sinal no solo	$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L}$

Tabela 1. Dois modelos de propagação.

convertida em distância. Dessa forma, quando um nó recebe uma mensagem, ele pode medir a força do sinal recebido e utilizar um modelo de propagação de sinal para encontrar a distância em relação ao nó que enviou a mensagem. Para ser mais realista com um sistema de localização 3D, neste trabalho, nós utilizamos o modelo de propagação de sinal *Two-Ray Ground*, que considera a possibilidade de reflexão do sinal no solo.

4.2. Cálculo da Posição

Para o nó calcular sua posição em um sistema com três dimensões, são necessários quatro pontos de referências. Esses pontos de referências são obtidos durante o vôo do VANT sobre a área de monitoramento. Além dos quatro pontos de referências, é necessário saber a distância para cada um desses pontos. Como descrito na seção anterior, isso pode ser feito medindo a força do sinal (RSSI) quando o VANT envia sua posição para os nós sensores. Quando o nó sensor possui quatro ou mais posições e suas respectivas distâncias, ele pode estimar a sua posição. O método mais utilizado para estimar a distância quando se tem quatro ou mais pontos de referência é a multilateração.

Utilizando a multilateração para estimar a posição de um nó, temos um sistema de equações (com no mínimo quatro equações) e três variáveis (x, y, z) O sistema de equações é construído a partir das posições recebidas e suas respectivas distâncias. O sistema de equações pode ser ilustrado da seguinte maneira, onde (x_i, y_i, z_i) e d_i, são respectivamente uma posição referência e a estimativa da distância para esse ponto de referência obtido através da técnica RSSI.

$$
\begin{aligned}
(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 &= d_1^2 \\
(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 &= d_2^2 \\
&\vdots \\
(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2 &= d_n^2
\end{aligned}
$$

O sistema de equações acima pode ser facilmente linearizado subtraindo a última equação, que é $(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2 = d_n^2$ das demais. Uma vez linearizado, o sistema de equações pode ser facilmente solucionado utilizando métodos de soluções de sistemas lineares. O método utilizado neste trabalho é o *least squares* [Golub and Loan 1996], devido a sua simplicidade e desempenho, sendo estes fatores importantes no projeto de soluções para redes de sensores sem fio. Como o sistema linear possui três variáveis, a solução do sistema linear representa a posição (x, y, z) do nó que executou o procedimento descrito.

4.3. Algoritmo de Localização

O algoritmo proposto é dividido em duas partes. A primeira se refere ao VANT e a segunda se refere ao cálculo da posição dos nós sensores. Como descrito anteriormente,

a figura 2 ilustra o funcionamento do algoritmo proposto. O VANT sobrevoa a área onde os nós sensores foram depositados. Durante o vôo, o VANT faz *broadcasts* periódicos informando para os nós sensores a sua posição geográfica. Quando um nó sensor recebe uma mensagem do VANT, ele calcula a sua distância em relação ao VANT utilizando a técnica de *RSSI*. Quando o nó sensor possui quatro ou mais posições recebidas do VANT (pontos de referência), o nó está apto a calcular a sua posição. Os algoritmos 1 e 2 descrevem essas etapas.

Neste trabalho, utilizamos o plano de vôo ilustrado na figura 4. O algoritmo 1 descreve o plano de vôo do VANT. As linhas 1 a 4 definem as variáveis utilizadas no algoritmo. A linha 4 escalona pela primeira vez o broadcast periódico da posição geográfica pelo VANT. Após $broadcastInterval$, a função $Timeout$ (linhas 12 a 17) será invocada. A função cria uma mensagem contendo a posição do VANT e faz o *broadcast* dessa mensagem. Após, o temporizador para um novo *broadcast* é escalonado. As linhas 6 a 11 executam o plano de vôo do VANT. Enquanto o VANT não chegou ao final da rota, ele recupera do plano de vôo o próximo ponto (coordenadas geográficas) no qual o VANT deve se deslocar. Após, o VANT se desloca para o ponto com uma determinada velocidade. É importante ressaltar que o *broadcast* periódico da posição é executado em paralelo com o deslocamento do VANT sobre a área monitorada.

Algorithm 1 Plano de Vôo do VANT

```
1:  broadcastInterval ← valor;                          \\ Intervalo entre cada broadcast da posicao
2:  direcao ← PosicaoInicial();                         \\ Posicao inicial do VANT
3:  velocidade ← velocidadeVANT;                        \\ Velocidade do VANT
4:  timer.schedule(broadcastInterval);                  \\ Timer para escalonar o broadcast periodico
5:  planoVoo();

6:  procedimento PLANOVOO()
7:      enquanto nao chegou ao final da rota faça
8:          direcao ← getNextPoint()                    \\ Proximo ponto no plano de vôo
9:          moveTo(direcao, velocidade)                 \\ Deslocamento do VANT
10:     fim enquanto
11: fim procedimento

12: procedimento TIMEOUT()
13:     (x_vant, y_vant, z_vant) ← getPosicaoVANT()
14:     mensagem ← (x_vant, y_vant, z_vant)
15:     broadcast(mensagem)                             \\ Enviando a posicao para os nos sensores
16:     timer.schedule(broadcastInterval)               \\ Escalonando um novo broadcast
17: fim procedimento
```

O algoritmo 2 descreve o algoritmo executado nos nós sensores para o cálculo de suas posições. A linha 1 define o conjunto referência, que são as posições geográficas do VANT juntamente com a distância relacionada com cada posição. A linha 2 define a variável que representará a posição calculada do nó sensor. Quando o nó sensor recebe uma mensagem do VANT, os seguintes passos são executados (linhas 3 a 9). O RSSI relacionado a mensagem recebida é convertido em distância utilizando a propagação de sinal *Two-Ray Ground* descrita anteriormente. Em seguida, o nó atualiza o conjunto referência com a mensagem contendo a posição do VANT juntamente com a distância em relação ao VANT. Se o conjunto referência for maior/igual a 4, o nó está apto a calcular a sua posição geográfica em três dimensões. Para isso, o conjunto referência é passado para o método $leastSquares$ (descrito na seção anterior) e, como retorno, é calculada a posição do nó sensor.

Algorithm 2 PositionComputation

```
1: conjuntoReferencia ← ∅                    \\ Conjunto contendo as posições de referências
2: myPosition ← ∅                            \\ Posição (x, y, z) do nó

3: procedimento HANDLEMESSAGE(mensagem, rssi)
4:     distancia ← convertRssiToDistance(rssi)
5:     conjuntoReferencia.add(mensagem, distancia)
6:     se conjuntoReferencia.size() ≥ 4 então
7:         myPosition ← leastSquares(conjuntoReferencia)
8:     fim se
9: fim procedimento
```

4.4. Aplicabilidade do Sistema de Localização Proposto

O sistema de localização proposto neste trabalho utiliza um VANT. Dessa forma, o VANT deverá percorrer toda a área onde os nós foram depositados para que todos os nós obtenham a quantidade de informação suficiente para o cálculo de sua posição. Uma consideração importante sobre esse esquema é o tempo para percorrer toda a área de monitoramento. A solução proposta neste trabalho irá gerar um atraso considerável para que todos os nós sejam capazes de calcular as suas posições. Entretanto, o algoritmo de localização será executado apenas uma vez durante todo o tempo de vida da rede[2], fazendo com que essa tarefa seja executada durante o seu *startup*. Dessa forma, não será gerado nenhum atraso desnecessário durante o seu funcionamento. Em contrapartida, como o VANT irá percorrer toda a área de monitoramento, todos os nós sensores serão capazes de calcular sua posição geográfica.

5. Avaliação de Desempenho

Esta seção avalia o desempenho do algoritmo proposto para o problema de localização 3D utilizando um veículo aéreo não tripulado em RSSF.

5.1. Metodologia

O algoritmo proposto nesse trabalho é comparado com o algoritmo Estimativa de Posicionamento Recursivo (RPE). A escolha do RPE deve-se ao seu bom desempenho em cenários 3D comparados com as outras soluções da literatura para o mesmo cenário, já que o algoritmo que utiliza um *beacon* móvel foi proposto para um sistema de localização 2D. A principal proposta da comparação é a avaliação do algoritmo proposto considerando as seguintes métricas: (i) Erro em metros no cálculo da posição geográfica; (ii) Quantidade de nós que não calcularam a sua posição geográfica. Para realizarmos as avaliações, variamos três parâmetros de rede, que são: (i) Quantidade de nós; (ii) Densidade e (iii) Erro no cálculo do RSSI. O erro no cálculo da posição foi computado utilizando a seguinte fórmula: $Erro = \frac{1}{n}\sum_{i=1}^{n} \mid PosicaoCalculada_i - PosicaoReal_i \mid$, onde n é a quantidade de nós sensores, $PosicaoCalculada_i$ é a posição calculada do i-ésimo nó e $PosicaoReal_i$ é a posição real do i-ésimo nó. O plano de voo utilizado é ilustrado na figura 4, enquanto os parâmetros de simulação são apresentados na Tabela 2. É importante observar que o raio de comunicação dos nós sensores e do VANT é de 50m. Isso foi feito para termos uma comparação justa com o algoritmo da literatura. Para termos resultados mais próximos de um ambiente real, utilizamos o modelo de propagação de sinal *Two-Ray Ground* e para estimar a distância em relação a força do sinal obtido, introduzimos um erro na técnica

[2]Neste trabalho consideramos uma RSSF estática.

de RSSI, que varia de 0 a 20% da potência do sinal recebido. Para calcularmos a área de sensoriamento (x, y), utilizamos a quantidade de nós (n) e o raio de comunicação r_c dos nós sensores. O relevo (terceira dimensão) para cada nó sensor é obtido através de um valor aleatório entre 0 e 10m. A altura do VANT é um valor aleatório entre 20 e 50m, que muda a cada nova direção durante o vôo sobre a área de monitoramento.

A avaliação foi realizada através de simulações utilizando o simulador SinalGo v.0.75.3 [Sinalgo 2008]. Cada simulação foi replicada 33 vezes utilizando diferentes sementes para a geração de números aleatórios. Em todos os resultados, as curvas representam os valores médios, enquanto que as barras de erros representam o intervalo de confiança de 95%.

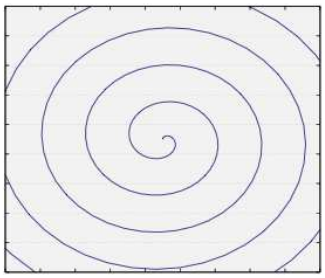


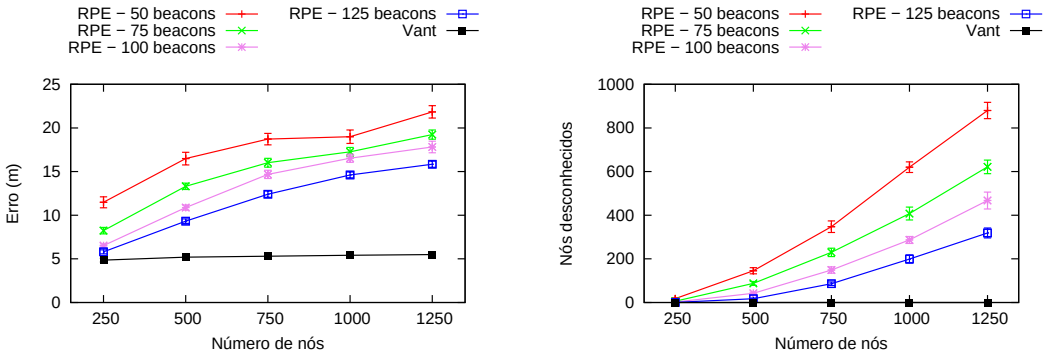
Figura 4. Plano de voo.

Parâmetros	Valores
Quantidade de nós	250, 500, 750, 1000 e 1250
Densidade (número médio de vizinhos)	20, 30, 40 e 50
Raio de comunicação dos nós sensores	50m
Raio de comunicação do VANT	50m
Velocidade do VANT	10m/s
Modelo de propagação	*Two-Ray Ground*
Erro do RSSI (%)	0, 5, 10 e 15 e 20
Quantidade de nós *Beacons* (RPE)	50, 75, 100 e 125
Área de sensoriamento (x e y)	$x = y = \frac{n \times \pi \times r_c^2}{Densidade}$
Relevo (z)	Valor aleatório entre 0 e 10m
Altura do VANT	Valor aleatório entre 20 e 50m
Intervalo de broadcast (VANT)	1/segundo

Tabela 2. Parâmetros de simulação.

5.2. Quantidade de Nós

Nós avaliamos o desempenho dos algoritmos para diferentes quantidade de nós na rede. A densidade foi fixada em 30 e o erro no cálculo do *RSSI* foi fixado em 5%. É importante ressaltar que, uma vez que a densidade da rede é de 30, ao aumentarmos a quantidade de nós também aumentamos a área de sensoriamento. A figura 5 ilustra os resultados obtidos para diferentes quantidade de nós.



(a) Erro (b) Nós desconhecidos

Figura 5. Quantidade de nós.

A figura 5(a) mostra o erro no cálculo da posição geográfica. O algoritmo de localização utilizando VANT proposto apresenta um erro pequeno nas posições estimadas. Observe que o erro não é afetado pela quantidade de nós na rede. Já o erro nas posições

estimadas pelo algoritmo RPE é grande (em média 3x maior que o erro das posições estimadas pelo algoritmo proposto), além disso, o erro no cálculo de posições estimadas pelo algoritmo RPE é afetado pela quantidade de nós na rede. Aumentando a quantidade de nós da rede, podemos observar que o erro na estimativa da posição também aumenta. Esse aumento do erro deve-se ao fato de muitos nós calcularem a sua posição baseado na posição calculada de outros nós. Dessa forma, o erro no cálculo se espalha pelos nós. Podemos observar que quando aumentamos a quantidade de nós *beacons* para o RPE, o erro no cálculo da posição geográfica pelo RPE diminui. Mas o principal problema com o aumento do número de nós *beacons*[3] é que eles são mais caros do que o resto dos nós sensores. Isto significa que, mesmo que seja apenas 10% dos nós *beacons*, o preço da rede aumenta aproximadamente em dez vezes. Outra observação é que, após os nós da rede estimarem sua posição geográfica, os nós *beacons* tornam-se inúteis, pois eles já não usam mais seus receptores (caro) de GPS.

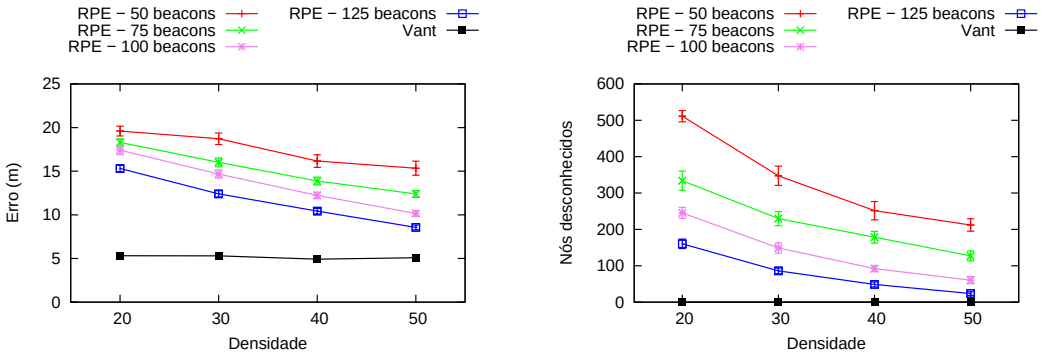
A figura 5(b) ilustra a quantidade de nós desconhecidos após a execução dos algoritmos. Um nó é considerado desconhecido em dois casos: (i) quando ele não recebeu informações suficientes para o cálculo da posição ou (ii) quando o erro da posição calculada for maior do que seu raio de comunicação. Podemos observar que, quando a rede possui poucos nós *beacons*, a quantidade de nós desconhecidos aumenta consideravelmente quando aumentamos a quantidade de nós da rede. Entretanto, para uma rede com poucos nós, a quantidade de nós desconhecidos é baixa. É importante observar que o sistema de localização proposto neste trabalho é capaz de fazer com que todos os nós da rede sejam aptos a calcular a sua posição geográfica, já que o VANT irá sobrevoar toda a área de monitoramento.

5.3. Densidade

Esta seção avalia o desempenho dos algoritmos para diferentes densidades de nós sensores na área de monitoramento. Para essa análise, fixamos a quantidade de nós sensores em 750 e o erro na estimativa da distância utilizando RSSI em 5%. A figura 6(a) mostra o erro no cálculo da posição para diferentes densidades. Pode-se observar que, quanto maior a densidade da rede, melhor é o desempenho do algoritmo RPE. Isso acontece devido ao fato de que quanto maior a densidade para uma mesma quantidade de nós, menor será a área de monitoramento. Como consequência, o erro no cálculo da posição geográfica não irá se espalhar para outros nós. A solução utilizando o VANT também não é afetada pela densidade da rede, uma vez que a troca de mensagem para o cálculo da posição acontece somente entre o VANT e os nós sensores e não entre nós sensores.

O mesmo padrão de resultados é encontrado na figura 6(b), que ilustra a quantidade de nós desconhecidos na rede após a execução dos algoritmos. Quando a densidade da rede aumenta, a quantidade de nós desconhecidos diminui como esperado. Além disso, a solução utilizando o VANT também não possui nenhum nó desconhecido após a execução do algoritmo.

[3]Um nó beacon pode ser de duas ordens de magnitude mais caro do que um nó desconhecido, devido ao seu receptor GPS.



(a) Erro (b) Nós desconhecidos

Figura 6. Densidade.

5.4. RSSI

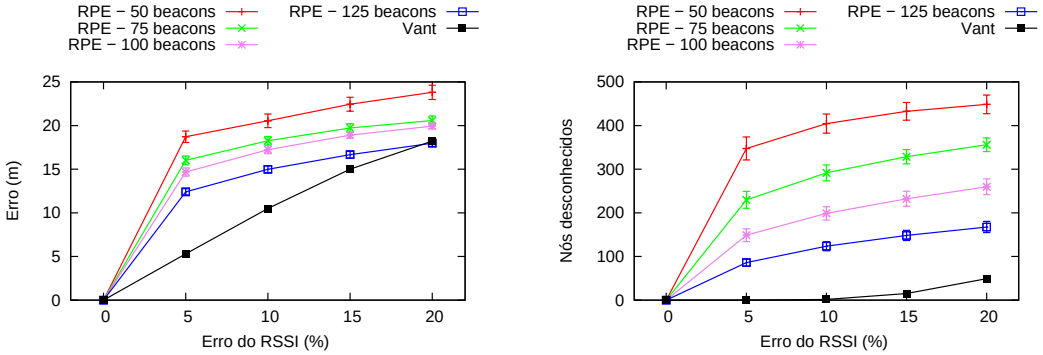
Variamos o erro no cálculo da distância utilizando a técnica RSSI para analisar o desempenho dos algoritmos. Para essa análise, fixamos a quantidade de nós sensores em 750 e a densidade da rede em 30. A figura 7 ilustra os resultados para essa avaliação. A figura 7(a) mostra o erro no cálculo da posição para diferentes erros no RSSI. Podemos observar que quando o erro na estimativa da distância é igual a zero, tanto o RPE quanto a solução proposta neste trabalho conseguem encontrar a posição exata dos nós sensores. A medida que o erro no cálculo do RSSI aumenta, todos os algoritmos aumentam o erro na estimativa da posição. Além disso, para erros no cálculo do RSSI inferiores a 15%, a solução proposta neste trabalho possui melhores resultados considerando qualquer quantidade de nós *beacons* pelo RPE. Entretanto, quando o erro na estimativa da distância é de 20%, a solução proposta neste trabalho possui o mesmo resultado quando a rede possui 125 nós *beacons* pelo RPE.

Considerando a quantidade de nós desconhecidos (figura 7(b)), quando o erro na estimativa da distância é igual a zero, a quantidade de nós desconhecidos após a execução dos algoritmos é zero, já que os nós calculam a sua posição de maneira exata. Entretanto, quando o erro na estimativa da distância aumenta, a quantidade de nós desconhecidos aumenta para todas as soluções avaliadas. É importante observar que mesmo o VANT sobrevoando toda a região de monitoramento, se o erro na estimativa da distância for muito alto, o nó pode calcular a sua posição com um erro superior ao raio de comunicação, fazendo ele ser um nó desconhecido.

6. Conclusões

O problema de localização em RSSFs é fundamental para diversas aplicações neste tipo de rede. Tipicamente, o erro nas posições estimadas pelas abordagens da literatura depende da quantidade de nós *beacons* depositados na rede e esses nós *beacons* aumentam drasticamente o custo da rede. Além disso, a maioria das abordagens propostas na literatura não considera o problema de localização 3D.

Neste trabalho, propomos um algoritmo para o problema de localização 3D em RSSFs que faz a utilização de um VANT, que funciona como um disseminar de posição



(a) Erro (b) Nós desconhecidos

Figura 7. RSSI.

para os nós sensores de alta qualidade em termos de precisão da informação disseminada. Os resultados obtidos mostram que o algoritmo proposto estima a posição dos nós desconhecidos com um erro pequeno. A eficiência do algoritmo proposto é independente do número de nós na rede, o que é um aspecto importante no caso da escalabilidade. Além disso, todos os nós sensores recebem informações suficiente para calcular sua posição.

Como trabalhos futuros pretendemos considerar diferentes planos de vôos, além de fazer experimentos em um ambiente real.

7. Agradecimentos

Os autores gostariam de expressar a sua gratidão pelo apoio concedido pelo CNPq, FAPESP para o INCT-SEC (processos 573963/2008-9 e 2012/12061-1), CAPES e FAPEMIG.

Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cyirci, 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 *Network Protocols, 2001. Ninth International Conference on*, pages 35 – 41.

Boukerche, A. (2008). *Algorithms and Protocols for Wireless Sensor Networks*. Wiley-IEEE Press.

Boukerche, A., de Oliveira, H. A. B. F., Nakamura, E. F., and Loureiro, A. A. F. (2008a). Localization Systems for Wireless Sensor Networks. In Boukerche, A., editor, *Algorithms and Protocols for Wireless Sensor Networks*, chapter 11, pages 307–340. John Wiley & Sons.

Boukerche, A., de Oliveira, H. A. B. F., Nakamura, E. F., and Loureiro, A. A. F. (2008b). Using Voronoi Diagrams to Scale a Localization System in Wireless Sensor Networks. *IEEE Wireless Communications Magazine*, 15:1–7.

Bulusu, N., Heidemann, J., Estrin, D., and Tran, T. (2004). Self-configuring localization systems: Design and experimental evaluation. *ACM Trans. Embed. Comput. Syst.*, 3(1):24–60.

de Oliveira, H. A. B. F., Boukerche, A., Nakamura, E. F., and Loureiro, A. A. F. (2009a). An Efficient Directed Localization Recursion Protocol for Wireless Sensor Networks. *IEEE Transactions on Computers*, 58(5):677–691.

de Oliveira, H. A. B. F., Boukerche, A., Nakamura, E. F., and Loureiro, A. A. F. (2009b). Localization in Time and Space for Wireless Sensor Networks: An Efficient and Lightweight Algorithm. *Performance Evaluation*, 66(3–5):209–222.

Doherty, L., pister, K., and El Ghaoui, L. (2001). Convex position estimation in wireless sensor networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1655 –1663 vol.3.

Golub, G. H. and Loan, C. F. V. (1996). *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, third edition.

Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., and Ganesan, D. (2001). Building efficient wireless sensor networks with low-level naming. *SIGOPS Oper. Syst. Rev.*, 35(5):146–159.

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*, MobiCom '00, pages 56–67, New York, NY, USA. ACM.

Kumar, S., Alaettinoglu, C., and Estrin, D. (2000). Scalable object-tracking through unattended techniques (scout. In *in: Proceedings of the 8th International Conference on Network Protocols (ICNP*.

Mini, R. A., Loureiro, A. A., and Nath, B. (2004). The distinctive design characteristic of a wireless sensor network: the energy map. *Computer Communications*, 27(10):935 – 945. <ce:title>Protocol Engineering for Wired and Wireless Networks</ce:title>.

Navas, J. C. and Imielinski, T. (1997). Geocast - geographic addressing and routing. In *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '97, pages 66–76, New York, NY, USA. ACM.

Niculescu, D. and Nath, B. (2001). Ad hoc positioning system (aps). In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 5, pages 2926 –2931 vol.5.

Romer, K. and Mattern, F. (2004). The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61.

Rudafshani, M. and Datta, S. (2007). Localization in wireless sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 51 –60.

Savvides, A., Han, C.-C., and Strivastava, M. B. (2001). Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 166–179, New York, NY, USA. ACM.

Sichitiu, M. and Ramadurai, V. (2004). Localization of wireless sensor networks with a mobile beacon. In *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, pages 174 – 183.

Sinalgo (2008). Simulator for network algorithms. Distributed Computing Group - ETH-Zurich.

Villas, L., Guidoni, D., Boukerche, A., Araujo, R., and Loureiro, A. (2011a). Dynamic and scalable routing to perform efficient data aggregation in wsns. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –5.

Villas, L. A., Boukerche, A., de Oliveira, H. A., de Araujo, R. B., and Loureiro, A. A. (2011b). A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, (0):–.

Villas, L. A., Guidoni, D. L., Araújo, R. B., Boukerche, A., and Loureiro, A. A. (2010). A scalable and dynamic data aggregation aware routing protocol for wireless sensor networks. In *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, MSWIM '10, pages 110–117, New York, NY, USA. ACM.

Sistemas de Monitoramento Passivo para RSSF – Soluções Existentes e uma Nova Proposta Energeticamente Eficiente

Fernando P. Garcia[1,2,3], José Neuman de Souza[1,2,a], Rossana M. C. Andrade[1,2,b]

Universidade Federal do Ceará (UFC)
[1]Mestrado e Doutorado em Ciência da Computação (MDCC)
[2]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
[3]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
fernandoparente@ifce.edu.br, neuman@ufc.br, rossana@ufc.br

Resumo. *Sistemas de monitoramento são importantes para depurar e analisar o funcionamento de uma rede de sensores sem fio (RSSF) em operação. No monitoramento passivo, uma rede de monitoramento adicional é implantada juntamente com a rede que deve ser monitorada (chamada de rede alvo). Esta rede de monitoramento captura e analisa os pacotes transmitidos pela rede alvo. Quando se deseja monitorar continuamente uma RSSF em um cenário real, um sistema de monitoramento passivo energeticamente eficiente é necessário, pois caso contrário a rede de monitoramento pode ter um tempo de vida bem menor do que a rede alvo. Neste artigo, inicialmente, nós identificamos, analisamos e comparamos os principais sistemas de monitoramento passivo propostos para RSSF. Durante as nossas pesquisas, não identificamos nenhum sistema de monitoramento passivo que se preocupasse em reduzir o consumo de energia da rede de monitoramento. Sendo assim, este artigo propõe um sistema de monitoramento passivo energeticamente eficiente para RSSF. Experimentos foram realizados na plataforma MicaZ com alguns módulos implementados do sistema e os resultados já demonstram a eficiência energética do sistema de monitoramento proposto.*

Abstract. *Monitoring systems are important for debugging and analyzing wireless sensor networks (WSN). In passive monitoring, a monitor network needs to be deployed in addition to the target network. This monitor network captures and analyzes packets transmitted by the target network. An energy-efficient passive monitoring system is necessary when we have to monitor a WSN in a real scenario over long periods; otherwise the lifetime of the network monitor can be shorter than the lifetime of the target network. In this paper, initially, we identify, analyze and compare the main passive monitoring systems proposed for WSN. During our research, we did not identify any passive monitoring system for WSN that aims to reduce the energy consumption of the monitor network. Therefore, we propose an energy-efficient passive monitoring system for WSN. Experiments were performed in the MicaZ platform with some modules and their results already show the energy efficiency of the proposed monitoring system.*

[a] Bolsista de produtividade D-1 do CNPq
[b] Bolsista de produtividade DT-2 do CNPq

1. Introdução

A miniaturização dos componentes eletrônicos e a evolução das tecnologias de comunicação sem fio têm estimulado o desenvolvimento e uso de Redes de Sensores Sem Fio (RSSF) em várias aplicações, tais como monitoramento ambiental, detecção sísmica, entre outras. Em geral, as RSSF são compostas por nós sensores de tamanho reduzido operados por baterias e que utilizam comunicação sem fio de pequeno alcance. Além disso, estas redes possuem severas restrições de consumo de energia, capacidade de processamento, capacidade de memória e largura de banda [Loureiro et al. 2003].

O monitoramento de uma RSSF em operação é importante para depurar e analisar o seu funcionamento. Utilizando-se um sistema de monitoramento, várias informações sobre o funcionamento da RSSF podem ser obtidas, tais como descoberta de topologia, morte e reinicialização de nós, nós isolados, *loops* de roteamento, perda de pacotes e latência da rede, entre outras [Ringwald and Romer 2007].

Em RSSF, o monitoramento da rede pode ser dividido em monitoramento ativo e monitoramento passivo. No monitoramento ativo são inseridas linhas de código na aplicação executada pelos nós sensores para obter informações sobre o funcionamento da rede. Neste caso, os pacotes de monitoramento são enviados juntamente com os pacotes de dados da rede alterando o comportamento e funcionamento da rede monitorada e consumindo os recursos desta rede. No monitoramento passivo, uma rede de monitoramento adicional é implantada juntamente com a rede que deve ser monitorada (**rede alvo)**. A rede de monitoramento captura e analisa os pacotes transmitidos pela rede alvo, não consumindo nenhum recurso da rede alvo. Portanto, quando é necessário reduzir a utilização de recursos da rede alvo, é melhor utilizar um sistema de monitoramento passivo. Pelas características das RSSF mencionadas anteriormente, é importante minimizar os recursos incluindo também aqueles utilizados pelos sistemas de monitoramento e, por isso, este trabalho foca em sistemas de monitoramento passivo, que não consomem nenhum recurso da rede alvo monitorada.

É importante ressaltar ainda que o tempo de vida de uma RSSF pode ser de até vários anos e nem todos os problemas aparecem durante as primeiras semanas após a implantação da rede [Hänninen et al. 2011]. Sendo assim, um sistema de monitoramento energeticamente eficiente é importante quando se deseja monitorar continuamente uma RSSF em um cenário real ("*in situ*"), pois caso contrário a rede de monitoramento pode ter um tempo de vida bem menor do que a rede alvo. Em [Liu et al. 2010], os autores descrevem a utilização de uma rede alvo para monitoramento de oceanos e ressaltam a importância de se monitorar esta rede através de um sistema de monitoramento passivo energeticamente eficiente.

Diante deste contexto, inicialmente, nós identificamos, analisamos e comparamos os principais sistemas de monitoramento passivo propostos para RSSF. Durante as pesquisas realizadas, não foi identificado nenhum sistema de monitoramento passivo para RSSF que se preocupasse em reduzir o consumo de energia da rede de monitoramento. Portanto, este trabalho propõe um sistema de monitoramento passivo energeticamente eficiente para RSSF, cujo principal objetivo é reduzir o consumo de energia da rede de monitoramento, e consequentemente prolongar o seu tempo de vida.

O restante deste artigo está organizado da seguinte forma: Na seção 2, os principais sistemas de monitoramento passivo propostos na literatura para RSSF são analisados e comparados. A seção 3 apresenta e discute o sistema de monitoramento proposto neste trabalho. Na seção 4 os detalhes de implementação de alguns módulos do sistema proposto são abordados. A seção 5 descreve os experimentos realizados com esses módulos, e apresenta e discute os resultados obtidos. As conclusões e trabalhos futuros são apresentados na seção 6.

2. Monitoramento passivo em redes de sensores sem fio

Após uma revisão bibliográfica em artigos publicados nas bibliotecas digitais *IEEE Xplore Digital Library* (http://ieeexplore.ieee.org/xplore) e *ACM Digital Library* (http://dl.acm.org) com a data de publicação a partir do ano de 2007 foram selecionados cinco sistemas de monitoramento passivo propostos especificamente para RSSF: SNTS [Khan et al. 2007], SNIF [Ringwald and Romer 2007], Pimoto [Awad et al. 2008], LiveNet [Chen et al, 2008] e PMSW [Xu et al. 2011].

2.1. SNTS (*Sensor Network Troubleshooting Suite*)

No SNTS [Khan et al. 2007], nós *sniffers* ouvem passivamente o canal de comunicação e coletam os pacotes enviados pelos nós da rede alvo. Ao capturar um pacote, o *sniffer* inclui um registro em sua memória não volátil (memória *flash*, por exemplo) com o conteúdo do pacote, e uma marca de tempo (*timestamp*) baseada no seu próprio *clock*. Após o período de captura dos dados, os *sniffers* são manualmente recolhidos e os registros dos pacotes capturados são transferidos para um computador (PC). Após os registros serem armazenados no PC, faz-se necessário ajustar o *timestamp* de cada registro, pois os *sniffers* não são sincronizados. Isto é feito da seguinte forma: antes de implantar a rede de monitoramento, cada *sniffer* tem o seu *clock* ajustado com o *clock* de um nó base. Ao final da captura dos dados, o *clock* de cada *sniffer* é comparado com o *clock* do nó base e os *timestamps* de seus registros são ajustados de acordo com a diferença entre estes *clocks*. Após o ajuste de *clock*, os registros duplicados são removidos. Para analisar os dados, os autores desenvolveram uma ferramenta utilizando técnicas de aprendizagem de máquina. Antes de executar a ferramenta é necessário configurar regras que definam o comportamento esperado da rede alvo.

O propósito do SNTS é ajudar o desenvolvedor de aplicações para RSSF a encontrar e resolver falhas em tempo de desenvolvimento. No entanto, torna-se inviável utilizar o SNTS para monitorar RSSF implantadas em cenários reais em que seja impraticável recolher os *sniffers*, como por exemplo, aplicações militares ou aplicações para monitoramento ambiental (e.g., florestas, oceanos, etc.).

2.2. SNIF (*Sensor Network Inspection Framework*)

Em [Ringwald and Romer 2007], os autores propõem um framework de inspeção passiva denominado SNIF. No SNIF, uma rede de monitoramento sem fio, denominada pelos autores de *deployment support network* (DSN), é implantada juntamente com a rede alvo. Os pacotes capturados pelos nós DSN são marcados com um *timestamp* e encaminhados até um computador onde são ordenados pelo *timestamp* e os pacotes

duplicados são removidos. Em seguida, os pacotes são decodificados de acordo com a descrição dos seus campos definida em um arquivo parametrizável. Após a decodificação, os pacotes são analisados utilizando uma árvore de decisão para inferir o status dos nós da rede alvo e encontrar possíveis falhas nesta rede. Finalmente, as informações obtidas são mostradas em uma interface gráfica desenvolvida pelos autores.

O SNIF não possui nenhum mecanismo de sincronização dos *clocks* dos nós DSN, podendo então ocasionar erros na ordenação dos pacotes capturados e na remoção de pacotes duplicados. Além disso, a falta de sincronização pode comprometer a análise e a precisão das informações de monitoramento obtidas.

2.3. Pimoto

No Pimoto [Awad et al. 2008], a rede alvo é subdividida em "ilhas de monitoramento". Em cada ilha de monitoramento é implantado um *sniffer*, que é responsável por capturar os pacotes enviados pelos nós da sua ilha e enviar estes pacotes diretamente para um gateway (computador) através de um rádio Bluetooth. O mesmo gateway pode receber os pacotes capturados de vários *sniffers*. O gateway inclui em cada pacote capturado o *timestamp* e o endereço do *sniffer*, e, em seguida, envia os pacotes capturados para um servidor central. O servidor analisa e mostra os pacotes capturados na ferramenta Wireshark [Wireshark 2012] utilizando um *plugin* desenvolvido pelos autores.

O Pimoto não possui nenhum mecanismo para analisar e inferir o comportamento da rede alvo. Os pacotes capturados podem ser apenas visualizados no Wireshark, e, portanto, toda a análise dos pacotes deve ser realizada pelo usuário. Além disso, a utilização do Pimoto pode ser inviável para RSSF com muitos nós distribuídos em uma área geográfica grande, pois neste caso é necessária uma infraestrutura composta por vários gateways interligados ao servidor.

2.4. LiveNet

Em [Chen et al, 2008] os autores propõem o LiveNet, um conjunto de ferramentas e técnicas para registrar o comportamento de uma RSSF. O LiveNet é constituído por três componentes principais: uma infraestrutura de monitoramento passiva composta por nós *sniffers* que coletam e armazenam os pacotes enviados pelos nós da rede alvo; um processo de *merging* que agrupa os pacotes coletados em um único *trace*; e um conjunto de algoritmos para analisar o *trace*. Enquanto a captura dos pacotes é realizada em tempo real, o *merging* e a análise do *trace* são realizados de forma *offline*.

No LiveNet, os pacotes capturados pelos *sniffers* podem ser armazenados em uma memória *flash* ou enviados para um computador através da porta serial. Desta forma, torna-se inviável utilizar o LiveNet para monitorar RSSF implantadas em cenários em que seja impraticável recolher as memórias *flash* ou utilizar uma rede cabeada para enviar os dados capturados, como por exemplo aplicações militares ou aplicações para monitoramento ambiental. Além disso, os *sniffers* são sincronizados apenas durante a implantação da rede de monitoramento, podendo então ocasionar erros no processo de *merging*, e, consequentemente, na fase de análise dos dados.

2.5. PMSW (*Passive Monitoring System in Wireless Sensor Networks*)

No PMSW [Xu et al. 2011], nós *sniffers* são implantados na área de monitoramento juntamente com os nós sensores da rede alvo. Cada *sniffer* captura os pacotes de dados e ACK (pacote de confirmação de recebimento) dos nós da rede alvo que estão na sua área de cobertura e envia os pacotes capturados para o seu *gateway*. Em alguns cenários pode ser necessário implantar *gateways* em diferentes partes da rede, pois o alcance do rádio dos *sniffers* é limitado. Ao receber os pacotes capturados por seus *sniffers*, o *gateway* cria um arquivo de *trace* local. Cada registro deste *trace* contém as informações de um pacote e um *timestamp* baseado no *clock* do *gateway*. Em seguida, cada *gateway* envia o *trace* gerado para um servidor através de uma rede TCP/IP.

O servidor recebe os *traces* gerados por todos os *gateways*, e faz o *merging* dos *traces* recebidos, gerando assim um único *trace* global. Após o *merging*, o servidor executa um algoritmo de inferência para inferir pacotes não capturados pelos *sniffers*. Em seguida, é realizada a análise do *trace* com o intuito de avaliar o desempenho e detectar eventuais falhas da rede alvo. Para a visualização das informações obtidas a partir do monitoramento, os autores desenvolveram uma ferramenta web.

Diferentemente dos demais trabalhos analisados, o PMSW implementa um algoritmo para inferir pacotes não capturados pela rede de monitoramento, gerando assim um *trace* com mais pacotes, e, consequentemente, obtendo informações mais precisas sobre o funcionamento da rede alvo. No entanto, são capturados apenas pacotes de dados e de confirmação (ACK), enquanto que pacotes de controle, tais como pacotes de roteamento e de eleição de *cluster*, não são capturados nem analisados. Além disso, o ajuste de *clock* dos *gateways* não é suficiente para garantir a remoção de todos os pacotes duplicados devido à latência da rede de monitoramento, pois dois *sniffers* podem capturar o mesmo pacote e enviar para o *gateway* utilizando rotas distintas, fazendo com que estes pacotes sejam recebidos pelo *gateway* em momentos distintos.

2.6. Análise comparativa

Nesta seção é realizada uma análise comparativa entre os sistemas de monitoramento passivo abordados neste artigo levando-se em consideração os seguintes aspectos:

- **Inferência -** capacidade do sistema em recuperar pacotes não capturados pela rede de monitoramento a partir dos pacotes capturados;
- **Modo de análise -** informa se a análise dos pacotes capturados pelo sistema é realizada de modo *online* ou *offline*;
- **Pacotes capturados -** relaciona quais tipos de pacotes são capturados pelo sistema;
- **Eficiência energética -** verifica se o sistema se preocupa em minimizar o consumo de energia dos nós da rede de monitoramento;
- **Mecanismo de sincronização** - descreve o mecanismo de sincronização utilizado pelo sistema para inserir a marca de tempo nos pacotes capturados;
- **Análise de eventos -** descreve quais informações de monitoramento são obtidas ao se analisar os pacotes capturados; e
- **Ferramenta de visualização -** tipo da ferramenta utilizada para a visualização das informações obtidas.

Tabela 1 – Comparação dos sistemas de monitoramento.

	Inferência	Modo de Análise	Pacotes Capturados	Eficiência Energética
SNTS [Khan et al. 2007]	Não	Offline	Dados + Controle	Não
SNIF [Ringwald and Romer 2007]	Não	Online	Dados + Controle	Não
PIMOTO [Awad et al. 2008]	Não	Online	Dados + Controle	Não
LiveNet [Chen et al, 2008]	Não	Offline	Dados	Não
PMSW [Xu et al. 2011]	Sim	Online	Dados + ACK	Não

Tabela 2 – Comparação dos sistemas de monitoramento (continuação).

	Mecanismo de Sincronização	Análise de Eventos	Ferramenta Visualização
SNTS [Khan et al. 2007]	Ajuste de clock	Diagnóstico de falhas na camada de rede	Desenvolvida pelos autores
SNIF [Ringwald and Romer 2007]	Nenhum	Diagnóstico de falhas *cross-layer*	Desenvolvida pelos autores
PIMOTO [Awad et al. 2008]	Ajuste de clock	Não possui	Wireshark
LiveNet [Chen et al, 2008]	Clock dos sniffers ajustados na implantação da rede	Não possui	Desenvolvida pelos autores
PMSW [Xu et al. 2011]	Ajuste de clock	Análise de falhas e desempenho	Desenvolvida pelos autores

As Tabelas 1 e 2 mostram um quadro comparativo dos sistemas de monitoramento discutidos neste artigo. As principais considerações obtidas observando-se os dados destas tabelas são:

i. Nenhum dos sistemas é energeticamente eficiente;
ii. Apenas o PMSW tem a capacidade de inferir pacotes não capturados;
iii. Nenhum dos sistemas implementa mecanismo de sincronização nos *sniffers* (nós da rede de monitoramento). O Livenet sincroniza os *sniffers* somente na implantação da rede. O SNTS, o Pimoto e o PMSW utilizam uma estratégia de ajuste de *clock*, contudo, esta estratégia não é tão precisa quanto à sincronização dos *sniffers*.
iv. O SNTS, o SNIF e o PWSW analisam os dados capturados com o intuito de detectar eventos de falha ou desempenho da rede monitorada, enquanto que o Pimoto e o Livenet apenas mostram os *traces* dos pacotes capturados; e
v. Apenas o Pimoto exibe as informações de monitoramento em uma ferramenta de gerência de rede utilizada pela comunidade (i.e., wireshark).

3. O Sistema de monitoramento proposto

Conforme mencionado na seção 1, um sistema de monitoramento passivo energeticamente eficiente é importante caso se deseje monitorar continuamente uma RSSF em um cenário real, pois caso contrário a rede de monitoramento pode ter um tempo de vida bem menor do que a rede alvo devido à má utilização da energia dos

sniffers. Então, nós propomos um sistema de monitoramento passivo para RSSF que reduz o consumo de energia da rede de monitoramento. Além disso, o sistema proposto implementa um mecanismo de sincronização nos *sniffers* e disponibiliza as informações de monitoramento através de um agente SNMP (*Simple Network Management Protocol*). A sincronização dos *sniffers* é importante para garantir a precisão dos *timestamps* dos pacotes capturados, possibilitando assim uma análise de dados mais precisa. A disponibilização das informações obtidas com o monitoramento através de um agente SNMP permite integrar o sistema proposto com ferramentas de gerência livres e gratuitas que suportam o protocolo SNMP, tais como Nagios e Net-SNMP.

A Figura 1 mostra a visão geral do sistema de monitoramento proposto, onde uma rede de monitoramento é implantada juntamente com a rede alvo. Um nó da rede de monitoramento, denominado de *sniffer*, captura em modo promíscuo os pacotes enviados por um ou mais nós da rede alvo, insere uma marca de tempo (*timestamp*) em cada pacote capturado, agrega os cabeçalhos (*headers*) de vários pacotes em uma mensagem de monitoramento e envia esta mensagem, através da rede de monitoramento, para o monitor local. O monitor local recebe as mensagens de monitoramento de vários *sniffers*, gera um arquivo de *trace* (*trace* local) com as informações dos pacotes capturados e envia o *trace* local, através de uma rede IP, para o monitor global. O monitor global recebe os *traces* de um ou mais monitores locais e gera um único *trace* (*trace* global), que é analisado para se obter diversas informações sobre a rede alvo.

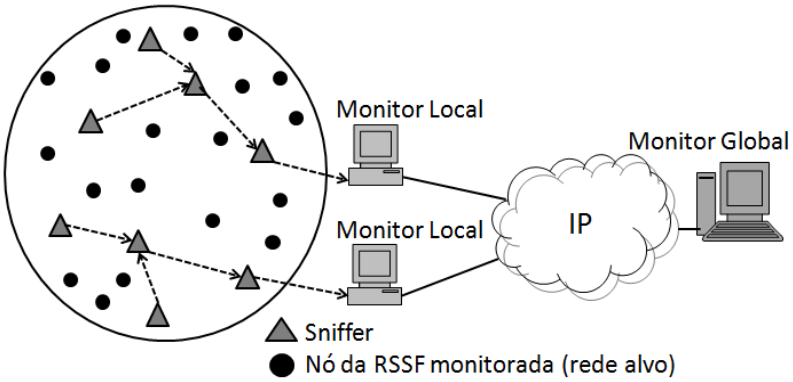


Figura 1 – Visão geral do sistema de monitoramento proposto.

A Figura 2 mostra o diagrama de atividades do sistema proposto, no qual os pacotes de um determinado nó da rede alvo são capturados por apenas um *sniffer* com o intuito de evitar a transmissão de pacotes duplicados e, consequentemente, reduzir o consumo de energia da rede de monitoramento. Para tanto, faz-se necessário implementar um mecanismo (**Eleição Sniffer**) para eleger quais nós da rede alvo terão seus pacotes capturados por quais *sniffers*. Este mecanismo de eleição é realizado pelos *sniffers* e pelo monitor local levando-se em consideração o RSSI (*received signal strength indicator*), que indica o nível de potência do sinal recebido. Este mecanismo de eleição é explicado de maneira detalhada na seção 4.1.

Ao **capturar um pacote** da rede alvo, o *sniffer* **insere um *timestamp*** neste pacote. Para garantir a precisão dos *timestamps*, faz-se necessário utilizar um mecanismo para a sincronização dos *sniffers* (**Sinc Sniffers**). Após capturar alguns pacotes, o *sniffer* pode utilizar um mecanismo para agregar os cabeçalhos (**Agrega *Headers***) destes

pacotes em uma mensagem de monitoramento para enviar para o monitor local. A agregação dos cabeçalhos tem como objetivo reduzir a quantidade de dados enviados pela rede de monitoramento, e consequentemente reduzir o consumo de energia desta rede. Neste caso, apenas as informações presentes nos cabeçalhos dos pacotes enviados pela rede alvo serão monitoradas. Entretanto, caso se deseje monitorar também os dados enviados pela rede alvo, basta não utilizar este módulo de agregação.

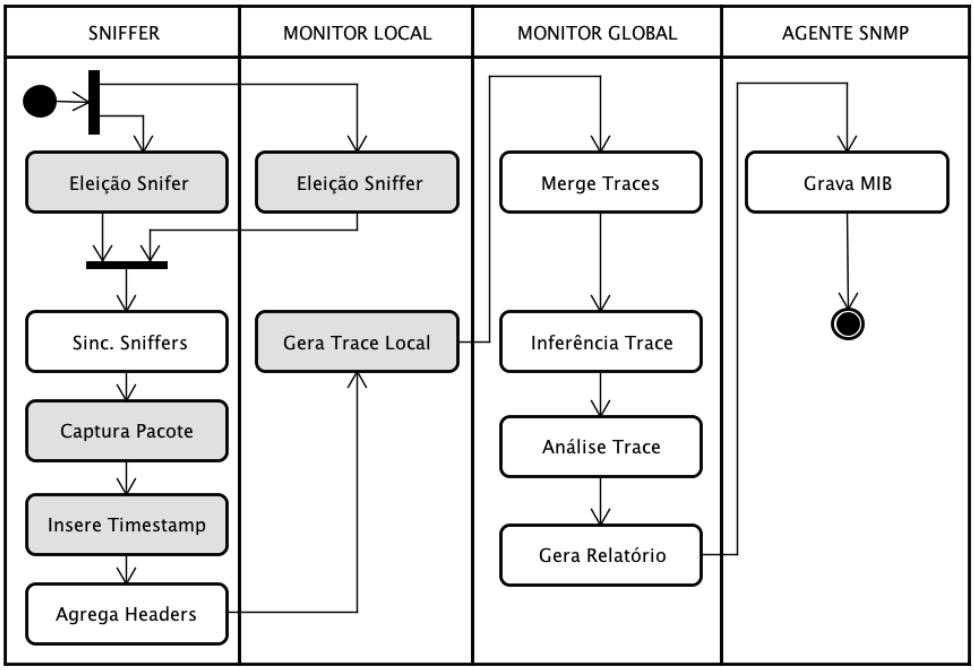


Figura 2 – Diagrama de atividades do sistema de monitoramento proposto.

O monitor local recebe as mensagens de monitoramento enviadas pelos *sniffers*, **gera o trace local** com todos os pacotes capturados e envia o *trace* para o monitor global através de uma rede IP. Em alguns cenários poderá ser necessário implantar monitores locais em diferentes partes da rede devido ao alcance limitado do rádio dos *sniffers*. Em um cenário mais restrito pode-se utilizar apenas um computador para desempenhar as funções de monitor local e de monitor global.

O monitor global recebe os *traces* enviados pelos monitores locais e faz o *merge* destes *traces* (**Merge Traces**), gerando um único *trace* global. A partir do *trace* global, o monitor global usa mecanismos de inferência (**Inferência Trace**) para inferir alguns pacotes que não foram capturados pelos *sniffers*. [Xu et al. 2011] propõem algoritmos para inferir pacotes não capturados pelos *sniffers*, que podem ser utilizados na implementação do sistema proposto.

Em seguida é realizada a **análise do trace** para se obter informações sobre a rede alvo (descoberta de topologia, perda de pacotes, morte e reinicialização de nós, etc.). Estas informações são utilizadas para **gerar um relatório** que será exibido para o usuário do sistema e são também **gravadas em uma MIB** (*Management Information Base*) por um **agente SNMP**. Desta forma, qualquer ferramenta de gerência que utilize o protocolo SNMP pode se comunicar com o agente SNMP e exibir as informações obtidas a partir do monitoramento da rede alvo.

O sistema proposto também possui um mecanismo para ligar e desligar (**on/off**) o monitoramento com o intuito de reduzir o consumo de energia dos *sniffers*. Para tanto, o usuário pode programar no monitor global os períodos de tempo em que a rede alvo será monitorada. O monitor global então envia um comando para cada monitor local para ligar ou desligar o monitoramento. O monitor local, por sua vez, envia este comando para os *sniffers*. Ao receber um comando *off*, o *sniffer* comuta para o modo *sleep*, reduzindo assim o consumo de energia em até 95% [Jurdak et al. 2008]. Periodicamente, o *sniffer* comuta para o modo de recepção para verificar se o monitor local enviou um comando *on*. Se receber um comando *on*, o *sniffer* permanece no estado de recepção e reinicia o processo de captura dos pacotes, e caso contrário retorna ao modo *sleep*.

Em resumo, o sistema de monitoramento passivo proposto neste trabalho tem como principal objetivo prolongar o tempo de vida da rede de monitoramento, podendo tornar possível o monitoramento de RSSF *in situ* durante um longo período de tempo ou até mesmo durante todo o seu tempo de vida.

4. Implementação de módulos do sistema de monitoramento proposto

Os módulos pintados com a cor cinza na Figura 2 (**Eleição Sniffer**, **Captura Pacote**, **Insere Timestamp** e **Gera Trace Local**) foram implementados, e serão discutidos e validados neste artigo para mostrar a eficiência energética do sistema de monitoramento proposto. O restante do sistema proposto ainda está em fase de desenvolvimento.

4.1. Sniffers

Nesta implementação, a rede de monitoramento utiliza como *sniffers* nós da plataforma MicaZ, desenvolvida pela *Crossbow Technology*. Esta plataforma foi escolhida por ser muito utilizada em aplicações de RSSF de uma maneira geral e por ser utilizada em outros trabalhos de RSSF do grupo de pesquisa ao qual este trabalho está vinculado ([Rocha et al. 2012] e [Cavalcante et al. 2012]). A aplicação de monitoramento embarcada nos *sniffers* foi desenvolvida utilizando a linguagem de programação nesC e executa sobre o sistema operacional TinyOS.

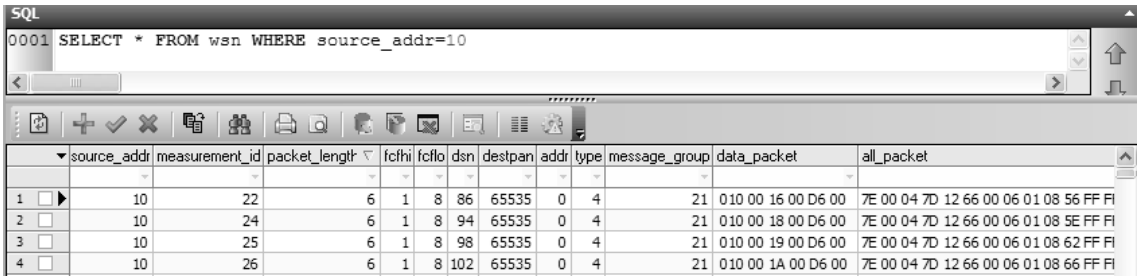
Após a implantação (*deploy*) da rede de monitoramento, os *sniffers* e o monitor local iniciam o mecanismo **Eleição Sniffer** para eleger quais nós da rede alvo terão seus pacotes capturados por quais *sniffers*. Este mecanismo é executado quando um *sniffer* captura pela primeira vez um pacote de um determinado nó da rede alvo e leva em consideração o nível de potência do sinal recebido (RSSI). Quando um *sniffer* $\mathbf{S_X}$ captura pela primeira vez um pacote de um nó **A** da rede alvo, ele envia uma mensagem de inclusão de um novo nó para o monitor local informando o endereço deste nó (**A**) e a potência do sinal recebido. Caso nenhum outro *sniffer* esteja capturando pacotes do nó **A**, o monitor local envia uma mensagem para $\mathbf{S_X}$ iniciar a captura dos pacotes enviados por A. No entanto, se já houver outro *sniffer* $\mathbf{S_Y}$ capturando pacotes do nó **A**, o monitor local analisa qual dos dois *sniffers* está recebendo os pacotes de **A** com maior potência de sinal. Caso $\mathbf{S_Y}$ esteja recebendo o sinal de **A** com maior potência do que $\mathbf{S_X}$, o monitor local envia uma mensagem para $\mathbf{S_X}$ informando que ele não deve capturar os pacotes de **A**. Porém, se $\mathbf{S_X}$ estiver recebendo o sinal de **A** com maior potência do que $\mathbf{S_Y}$, o monitor local envia uma mensagem para $\mathbf{S_X}$ capturar os pacotes de **A** e envia uma mensagem para

$\mathbf{S_Y}$ parar de capturar os pacotes de **A**. Desta forma, o mecanismo de eleição garante que apenas um *sniffer* captura os pacotes enviados por um determinado nó da rede alvo, evitando assim a transmissão de pacotes capturados redundantes através da rede de monitoramento.

Durante o processo de monitoramento, cada *sniffer* captura em modo promíscuo os pacotes enviados pelos nós da rede alvo que ele monitora, e que foram selecionados pelo mecanismo de eleição explicado anteriormente. Após capturar um pacote, o *sniffer* insere uma marca de tempo (*timestamp*) e envia o pacote capturado para o monitor local através da rede de monitoramento utilizando roteamento *multihop* (o pacote é encaminhado do nó de origem ao nó de destino passando por nós intermediários).

4.2. Monitor Local

A aplicação do monitor local foi implementada utilizando a linguagem de programação Java, por ser uma tecnologia multiplataforma e possibilitar que o mesmo código execute em diferentes sistemas operacionais (Linux, Windows, etc.). O monitor local comunica-se com a rede de monitoramento através de uma estação base. Conforme explicado na seção 4.1, o monitor local executa o mecanismo de eleição juntamente com os *sniffers*. Além disso, o monitor local recebe os pacotes enviados pelos *sniffers* e **gera o trace local** com os pacotes da rede alvo capturados. Nesta implementação, o trace local é armazenado em uma tabela do banco de dados MySQL [MySQL 2012]. O MySQL foi utilizado neste trabalho por ser um sistema de gerenciamento de banco de dados bastante difundido e ser um software livre com licença GPL (*General Public License*). Ao armazenar os pacotes capturados em um banco de dados, pode-se facilmente obter informações sobre a rede alvo utilizando cláusulas SQL (*Structured Query Language*). Para exemplificar, a Figura 3 exibe alguns pacotes enviados pelo nó da rede alvo cujo endereço (*source_addr*) é 10.


SQL

```
0001 SELECT * FROM wsn WHERE source_addr=10
```

	source_addr	measurement_id	packet_length	fcfhi	fcflo	dsn	destpan	addr	type	message_group	data_packet	all_packet
1	10	22	6	1	8	86	65535	0	4	21	010 00 16 00 D6 00	7E 00 04 7D 12 66 00 06 01 08 56 FF FI
2	10	24	6	1	8	94	65535	0	4	21	010 00 18 00 D6 00	7E 00 04 7D 12 66 00 06 01 08 5E FF FI
3	10	25	6	1	8	98	65535	0	4	21	010 00 19 00 D6 00	7E 00 04 7D 12 66 00 06 01 08 62 FF FI
4	10	26	6	1	8	102	65535	0	4	21	010 00 1A 00 D6 00	7E 00 04 7D 12 66 00 06 01 08 66 FF FI

Figura 3 – Exibição de pacotes capturados.

5. Experimentos

Esta seção descreve e analisa os experimentos realizados para validar os módulos do sistema de monitoramento proposto cujas implementações foram descritas na seção 4.

5.1. Descrição dos experimentos

Os experimentos foram realizados utilizando 28 nós MicaZ com sistema operacional TinyOS. A plataforma MicaZ possui como principais características: microprocessador

ATMEGA128L, 4KB de memória RAM, 128KB de memória ROM e transceptor de rádio frequência CC2420.

A Figura 4 mostra o cenário utilizado para a realização dos experimentos. A rede alvo é composta por 22 nós, sendo 21 nós sensores e 01 nó sorvedouro. Os nós sensores executam uma aplicação que a cada minuto mede a temperatura do ambiente e envia para o nó sorvedouro através de roteamento *multihop*. A área de dados (*payload*) dos pacotes enviados pelo nó sensor contém a temperatura medida e um contador que é incrementado a cada medição de temperatura. A rede de monitoramento é composta por 05 sniffers e 01 estação base. Os *sniffers* capturam os pacotes enviados pelos nós da rede alvo e enviam para a estação base. A estação base envia os pacotes recebidos dos *sniffers*, através de um cabo USB, para um notebook que executa a aplicação do monitor local. A aplicação do monitor local recebe os pacotes enviados pela estação base e armazena em um banco de dados MySQL. É importante ressaltar que a estação base tem como função principal intermediar o envio de pacotes entre os *sniffers* e o monitor local, e não captura nenhum pacote da rede alvo.

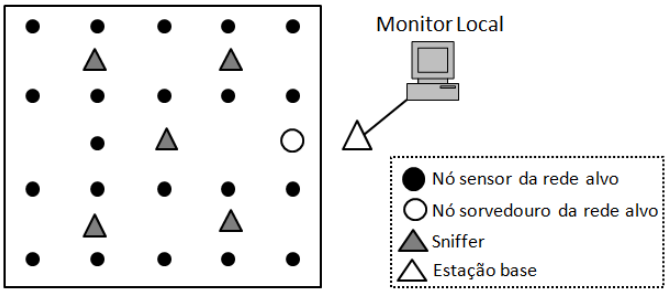


Figura 4 – Cenário utilizado nos experimentos.

Foram realizados dois tipos de experimentos: “Com Eleição” e “Sem Eleição”. No experimento “Com Eleição”, os sniffers executam a aplicação descrita na seção 4.1, onde é implementado o mecanismo de eleição proposto neste trabalho.

No experimento “Sem Eleição”, os sniffers não possuem nenhum mecanismo de eleição e capturam todos os pacotes dos nós da rede alvo que estão na área de cobertura dos seus rádios. Em seguida, os sniffers enviam os pacotes capturados para o monitor local, que então armazena no banco de dados. Vale ressaltar que esta é a estratégia utilizada por todos os sistemas de monitoramento descritos na seção 2.

Para a avaliação dos experimentos, foram definidas as seguintes métricas: quantidade de pacotes enviados pela rede alvo (*PenvAlvo*), quantidade de pacotes distintos capturados (*Pcapturados*), quantidade de pacotes não capturados (*PnãoCapturados*), quantidade de pacotes redundantes capturados (*Predundantes*) e energia consumida na transmissão dos pacotes capturados (*Et*).

No sistema proposto, assim como em todos os cinco sistemas analisados na Seção 2, os *sniffers* “escutam” todos os pacotes que trafegam nas suas interfaces de rádio. Portanto, a energia consumida na recepção de pacotes no sistema proposto é similar à energia consumida nos sistemas analisados, e por isso não foi utilizada como métrica de avaliação.

A quantidade total de pacotes enviados pelos 21 nós sensores é obtida através da

Equação 1, onde *contMediçãoInicial_i* e *contMediçãoFinal_i* são respectivamente o número da primeira e da última medição de temperatura realizada pelo nó *i*.

$$PenvAlvo = \sum_{i=1}^{21} (contMediçãoFinal\_i - contMediçãoInicial\_i + 1) \quad (1)$$

A quantidade de pacotes distintos do nó *i* capturados pela rede de monitoramento (*PcapturadosNó_i*) é determinada verificando-se quais pacotes do nó *i* existem no intervalo [*contMediçãoInicial_i*, *contMediçãoFinal_i*]. Logo, a quantidade total de pacotes distintos capturados é obtida através da Equação 2. Portanto, a quantidade de pacotes não capturados pela rede de monitoramento é determinada pela Equação 3.

$$Pcapturados = \sum_{i=1}^{21} PcapturadosNó\_i \quad (2)$$

$$PnãoCapturados = PenvAlvo - Pcapturados \quad (3)$$

Dois ou mais pacotes são considerados redundantes quando possuem o mesmo endereço do nó e mesmo contador de medição de temperatura. Portanto, a quantidade de pacotes redundantes do nó *i* (*PredundantesNó_i*) é determinada verificando-se quais pacotes deste nó possuem o mesmo contador de medição. Logo, a quantidade total de pacotes redundantes capturados é obtida através da Equação 4.

$$Predundantes = \sum_{i=1}^{21} PredundantesNó\_i \quad (4)$$

Para calcular a energia consumida pelos *sniffers* na transmissão dos pacotes foi utilizado o modelo de energia para sensores MicaZ definido em [Jurdak et al. 2008] e utilizado em [Rocha et al. 2012]. Neste modelo, a energia consumida na transmissão (*Et*) é determinada pela Equação 5, onde *Psent* é a quantidade de pacotes enviados, *Plength* é o tamanho do pacote em bytes, *TB* é o tempo gasto na transmissão de um byte, *It* é o valor da corrente elétrica no modo de transmissão e *V* é a tensão elétrica da bateria.

$$Et = Psent \times Plength \times TB \times It \times V \quad (5)$$

Os valores utilizados para *TB*, *It* e *V* foram 32 µS, 17.4 mA e 3 Volts, respectivamente. Estes valores foram obtidos no documento de especificação da plataforma MicaZ (*datasheet*), que também são iguais aos valores apresentados em [Rocha et al. 2012]. Nos experimentos realizados, cada pacote enviado pelos *sniffers* tem tamanho (*Plength*) de 23 bytes, sendo 07 bytes de *header* e 16 bytes referentes ao pacote enviado pelo nó da rede alvo. Substituindo-se estes valores na Equação 5, obtém-se a Equação 6.

$$Et = 38.42 \times 10^{-6} \times Psent \quad (6)$$

A quantidade de pacotes enviados pelos *sniffers* é determinada pela Equação 7. Então, a energia consumida pelos *sniffers* na transmissão dos pacotes capturados da rede alvo é determinada pela Equação 8.

$$Psent = Pcapturados + Predundantes \quad (7)$$

$$Et = 38.42 \times 10^{-6} \times (Pcapturados + Predundantes) \quad (8)$$

5.2. Resultados e discussão

Para cada tipo de experimento foram realizados 10 experimentos com duração de 15 minutos. Os resultados mostrados na Tabela 3 referem-se aos valores médios dos 10 experimentos realizados com intervalo de confiança de 95%.

Tabela 3 – Resultados dos experimentos.

Tipo de experimento	PenvAlvo	Pcapturados	PnãoCapturados	Predundantes	Et (mJ)
Com eleição	318 ± 2	298 ± 4	20 ± 3	0	11.44±0.16
Sem eleição	320 ± 5	306 ± 5	14 ± 1	403 ± 10	27.23±0.54

Pode-se observar na Tabela 3 que ao se utilizar o mecanismo de eleição proposto neste trabalho, a quantidade de pacotes redundantes capturados pela rede de monitoramento é zero, pois cada nó da rede alvo tem seus pacotes capturados por apenas um *sniffer*. Entretanto, 20 dos 318 pacotes enviados pela rede alvo não são capturados pela rede de monitoramento, o que corresponde a 6.28%.

Quando não é utilizado o mecanismo de eleição dos *sniffers*, a quantidade de pacotes não capturados é de apenas 4.38% (14 pacotes), pois o mesmo pacote pode ser capturado por mais de um *sniffer*, reduzindo assim a probabilidade de não capturá-lo. No entanto, foram capturados 403 pacotes redundantes, que corresponde a uma média de 1.26 (403/320) pacotes redundantes para cada pacote da rede alvo capturado.

Pode-se observar também na Tabela 3 que quando não é utilizado o mecanismo de eleição, cada *sniffer* consome, a cada 15 minutos, em média 27.23 mJ para a transmissão dos pacotes capturados. Quando o mecanismo de eleição é utilizado, este consumo de energia é de apenas 11.44 mJ. Isto significa uma redução de 58% da energia consumida pelos *sniffers* para a transmissão dos pacotes capturados, prolongando assim o tempo de vida da rede de monitoramento.

6. Conclusões e trabalhos futuros

Neste trabalho, inicialmente nós analisamos e comparamos cinco sistemas de monitoramento passivo propostos para RSSF: SNTS, SNIF, Pimoto, LiveNet e PMSW. Entretanto, nenhum destes sistemas de monitoramento passivo se preocupa em reduzir o consumo de energia da rede de monitoramento. Diante deste contexto, nós propomos um sistema de monitoramento passivo energicamente eficiente para RSSF. Neste artigo, nós apresentamos e validamos os módulos já implementados do sistema. Experimentos foram realizados na plataforma MicaZ e os resultados demonstraram que o mecanismo de eleição utilizado no nosso sistema reduz em até 58% a energia consumida pelos *sniffers* para a transmissão dos pacotes capturados, prolongando assim o tempo de vida da rede de monitoramento. Como trabalhos futuros teremos a implementação e a validação dos demais módulos do sistema proposto. Além disso, nós pretendemos alterar o mecanismo de eleição para levar em consideração, além da potência do sinal recebido, o nível de energia da bateria dos *sniffers* e a quantidade de nós monitorados por cada *sniffer*, com o intuito de balancear o consumo de energia dos *sniffers* e evitar que alguns *sniffers* tenham sua energia esgotada bem antes de outros.

Agradecimentos

Este trabalho é um resultado parcial do projeto UbiStructure financiado pelo CNPq (MCT / CNPq 14/2011 - Universal) sob o número de protocolo 481417/2011-7.

Referências

Awad, A., Nebel, R., German, R. and Dressler, F. (2008) "On the need for passive monitoring in sensor networks" In: IEEE Euromicro Conference on Digital System Design Architectures, Methods and Tools.

Cavalcante, M. T., Garcia, F. P., Andrade, R. M. C. (2012) "Avaliação de Desempenho de Mecanismos de Segurança para Redes de Sensores Sem Fio" In: XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), pp. 277-290.

Chen, B. R., Peterson, G., Mainland, G. and Welsh, M. (2008) "LiveNet: using passive monitoring to reconstruct sensor network dynamics" In: Distributed Computing in Sensor Systems, pp. 79-98.

Hänninen, M., Suhonen, J., Hamalainen, T. D. And Hannikainen, M. (2011) "Practical monitoring and analysis tool for WSN testing" In: IEEE Conference on Design and Architectures for Signal and Image Processing (DASIP), pp. 1-8.

Jurdak, R., Ruzzelli, A. G. and O'Hare, G. (2008) "Adaptive radio modes in sensor networks: How deep to sleep?" In: IEEE Communications Society Conference on Ad Hoc and Sensor Networks, pp. 386-394.

Khan. M. M. H., Luo, L., Huang, C. and Abdelzaher, T. (2007) "SNTS: sensor network troubleshooting suite" In: 3rd IEEE International Conference on Distributed Computing in Sensor Systems, Springer, Berlin.

Liu, Y., Liu, K. and Li, M. (2010) "Passive Diagnosis for Wireless Sensor Networks" In: IEEE ACM Transactions on Networking, Vol. 18, No. 4, pp. 1132-1144.

Loureiro, A. A. F., Nogueira, J. M. S., Ruiz, L. B., Mini, R. A. F., Nakamura, E. F. e Figueiredo, C. M. S. (2003) "Redes de Sensores Sem Fio" In: Simpósio Brasileiro de Redes de Computadores.

MySQL (2012) "MySQL", http://mysql.org. Novembro.

Ringwald, M. and Romer, K. (2007) "Deployment of sensor networks: problems and passive Inspection" In: Proceedings of the 5th Workshop on Intelligent Solutions in Embedded Systems, IEEE, New York.

Rocha, A. R., Pirmez, L., Delicato, F. C., Lemos, E., Santos, I., Gomes, D. G., Souza, J. N. (2012) "WSNs clustering based on semantic neighborhood relationships" In: Elsevier Computer Networks, vol. 56, pp. 1627-1645.

Wireshark (2012) "Wireshark", http://www.wireshark.org, Outubro.

Xu, X., Wan, J., Zhang, W., Tong, C. and Wu C. (2011) "PMSW: a passive monitoring system in wireless sensor networks" In: International Journal of Network Management, vol. 21, pp. 300-325.

Estimando o Nível de Segurança de Dados de Redes de Sensores sem Fio *

Alex Lacerda Ramos[1], **Raimir Holanda Filho**[1]

[1]Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR) – 60811-905 – Fortaleza – CE – Brasil

alex.lacerda@unifor.edu.br, raimir@unifor.br

Abstract. *The main purpose of Wireless Sensor Networks (WSN) is generating reliable data to their users. For that reason, sensor networks use a combination of different security mechanisms. However, in order to know whether these mechanisms are providing enough security for sensor data, it is interesting that users have access to a value that informs the security level provided by such mechanisms. Thus, users will be able to quickly decide on whether to use such data. This paper presents the Sensor Data Security Estimator (SDSE), a model to estimate the security level of data from sensor networks based on the existing security mechanisms deployed in them.*

Resumo. *O principal propósito das Redes de Sensores sem Fio (RSSF) é gerar dados confiáveis para seus usuários. Para isso, as redes de sensores utilizam uma combinação de diferentes mecanismos de segurança. No entanto, a fim de saber se estes mecanismos estão provendo segurança suficiente para os dados dos sensores, é interessante que os usuários tenham acesso a um valor que informe o nível de segurança proporcionado por tais mecanismos. Desse modo, os usuários serão capazes de decidir rapidamente sobre a utilização ou não desses dados. Este artigo apresenta o Sensor Data Security Estimator (SDSE), um modelo para estimar o nível de segurança dos dados de redes de sensores com base nos mecanismos de segurança existentes nelas.*

1. Introdução

O principal propósito das RSSF é capturar informações do mundo real e torná-las disponíveis para usuários interessados. Isso é feito pelos nós sensores que, de maneira distribuída, coletam e enviam informações para um dispositivo central, a estação base, para que os usuários possam acessá-las [Akyildiz et al. 2002]. Visto que os usuários de RSSF desejam utilizar informações confiáveis, é imperativo garantir a segurança dessas redes.

Em razão de suas características peculiares, para garantir segurança, as RSSF utilizam a combinação de diferentes mecanismos de segurança, cada um deles projetado para abordar vulnerabilidades específicas. Entretanto, somente a presença dos mecanismos de segurança não garante que os dados dos sensores sejam realmente confiáveis. É necessário portanto, medir o *nível de segurança* dos dados provido por esses mecanismos e informá-lo aos usuários, de modo que eles possam decidir sobre o uso desses dados.

*Esta pesquisa faz parte do Projeto Construindo Cidades Inteligentes [CIA][2] e Alex Lacerda Ramos é financiado por bolsa da CAPES.

Com esse propósito, este artigo apresenta o *Sensor Data Security Estimator* (SDSE), um modelo para calcular dinamicamente o *nível de segurança* de dados originados em RSSF. Para isso, o SDSE define *métricas de segurança* obtidas a partir dos mecanismos de segurança das RSSF. Os valores das métricas coletados a partir dos nós sensores são em seguida combinados para prover um valor global referente à confiabilidade dos dados. O *nível de segurança* representa portanto, a probabilidade dos dados de sensores serem seguros mesmo que a rede esteja sob ataque.

O cálculo do *nível de segurança* também pode ser usado para ajudar profissionais a tomar decisões sobre como avaliar diferentes mecanismos de segurança e modificar as configurações da rede a fim de aperfeiçoar a segurança [Ahmed et al. 2008].

O restante do artigo está organizado como segue. A Seção 2 discute os trabalhos relacionados. A Seção 3 descreve as *métricas de segurança* propostas e os *parâmetros* necessários para seu cálculo. A Seção 4 apresenta a estimativa do *nível de segurança*. A Seção 5 detalha o funcionamento do SDSE. A Seção 6 mostra as avaliações das *métricas* e do *nível de segurança* propostos. Por fim, a Seção 7 conclui o trabalho.

2. Trabalhos Relacionados

A maioria das propostas para estimativa de nível de segurança é baseada em vulnerabilidades e análise de riscos. Ahmed et al. [Ahmed et al. 2008] apresentam uma abordagem para medir nível de segurança baseada em métricas que quantificam vulnerabilidades de *serviços* de rede. Eles utilizam a norma *Common Vulnerability Scoring System* (CVSS) [Ahmed et al. 2008] para quantificar as métricas propostas e utilizam ferramentas automatizadas para identificar as vulnerabilidades existentes. Em seguida, eles combinam os valores das métricas para computar o nível de segurança global.

Frigault et al. [Frigault et al. 2008] propõem um esquema para medir segurança global de rede que utiliza grafos de ataque baseados em uma rede de *Bayes* para combinar os efeitos de todas as vulnerabilidades conhecidas do sistema. Eles usam métricas de probabilidades baseadas no CVSS e as combinam usando probabilidade condicional.

Li et al. [Li et al. 2011] apresentam um modelo estocástico para quantificar a segurança global de redes, de acordo com a capacidade dos mecanismos de segurança implantados e o grafo de vulnerabilidades subjacente.

Embora vulnerabilidades e análise de risco tenham sido abordadas recentemente para redes tradicionais, os trabalhos existentes não podem ser aplicados diretamente em redes de sensores, pois como a segurança de RSSF é uma área imatura, não existe muito sobre vulnerabilidades de serviços e ferramentas para identificá-las. Além disso, as RSSF dependem principalmente da *confiabilidade* e *resiliência* dos mecanismos de segurança para atenuar suas vulnerabilidades, que são *inerentes* e não necessariamente relacionadas a *serviços*, tais como canal de comunicação não confiável e exposição a ataques físicos.

Até o momento, pouquíssimos estimadores de segurança foram propostos para *redes sem fio*, principalmente para redes de sensores. Nesse contexto dinâmico das redes sem fio, Savola [Savola 2008] propõe um *framework* para estimar o nível de segurança de redes móveis sem fio baseado em métricas de segurança apropriadas. Contudo, em seu trabalho as métricas usadas não são mencionadas.

Com relação a RSSF, Ksiezopolski e Kotulski [Ksiezopolski and Kotulski 2005]

apresentam um modelo que usa vários parâmetros de protocolos de rede para estimar o nível de segurança global. No caso deste nível estar abaixo de certo patamar, a segurança do sistema é aperfeiçoada dinamicamente. No entanto, sua proposta supõe que existam nós centrais com maior proteção e mecanismos de validação para detectar incidentes em quaisquer lugares na rede.

Diferentemente dos trabalhos descritos acima, o SDSE define métricas que consideram a *resiliência* e a *confiabilidade* dos mecanismos de segurança, além de serem baseadas em informações que refletem o atual *estado de segurança* das redes, como será apresentado na Seção 3. Também é importante ressaltar que ao invés de estimar a segurança global da rede, este artigo considera somente os sensores pertencentes à rota pela qual os dados trafegaram até chegar à estação base, como será explicado na Seção 4.

3. Métricas de Segurança Propostas

Nesta seção, são apresentados os mecanismos de segurança considerados pelo SDSE e as métricas de segurança propostas para cada um desses mecanismos.

As métricas propostas avaliam os mecanismos de segurança instalados em uma rede de sensores e são calculadas na estação base pelo SDSE para cada nó sensor, a partir de parâmetros providos por esses mecanismos. Existem dois tipos de mecanismos de segurança para RSSF, os de prevenção e os detecção. Os mecanismos de prevenção evitam ou dificultam a execução de certos tipos de ataque, enquanto os mecanismos de detecção são responsáveis por identificar e isolar ataques que violaram os mecanismos de prevenção e passaram a realizar atividades maliciosas na rede.

Levando isso em conta, neste artigo são considerados quatro mecanismos de segurança, que juntos proporcionam uma solução de segurança completa para redes de sensores. Estes mecanismos são: *Criptografia* (prevenção), *Gerenciamento de Chave Criptográfica* (prevenção), *Sistema de Detecção de Intrusão* (detecção) e *Sistema de Gerenciamento de Confiança* (detecção). Vale ressaltar que o SDSE não é responsável por garantir a segurança da RSSF, ele apenas estima o nível da segurança provido pelos mecanismos já instalados na rede.

A seguir são descritas as métricas propostas para cada um dos mecanismos de segurança considerados.

3.1. Probabilidade de Força Criptográfica

Em uma Rede de Sensores sem Fio, adversários podem realizar ataques de força bruta nos algoritmos de criptografia e revelar as chaves secretas dos sensores. Neste tipo de ataque, todas as possíveis chaves são testadas até que se descubra a chave secreta procurada. É importante ressaltar que chaves inseguras podem ser utilizadas em RSSF devido à limitação de seus recursos ou por imperícia do próprio administrador da rede.

Para medir a capacidade de um sensor resistir à este tipo de ataque, definimos a métrica *Probabilidade de Força Criptográfica* (P_F), que varia de acordo com o tempo t, de acordo com a seguinte equação:

$$P_F(t) = \begin{cases} 1 - \dfrac{f \cdot t}{2^s} & \text{se } t \leq \dfrac{2^s}{f} \\ 0 & \text{caso contrário} \end{cases} \quad (1)$$

Onde s representa a força do algoritmo de criptografia, que é uma medida logarítmica do ataque computacional mais rápido conhecido para o algoritmo (medida em bits), t é o intervalo de tempo decorrido desde o momento da ativação da chave no sensor e f é a quantidade de chaves testadas por unidade de tempo.

Observe que P_F é calculada pelo complemento da probabilidade de uma chave ser descoberta, que por sua vez, é calculada pelo total de chaves que podem ser testadas até o tempo t (isto é, $f \cdot t$) dividido pelo total de chaves possíveis (2^s).

Existem diferentes forças criptográficas dependendo do algoritmo e do tamanho de chave utilizados. Normalmente, quanto maior o tamanho da chave (também medida em bits), maior sua força criptográfica. A força de um algoritmo de criptografia é sempre um valor menor ou igual ao tamanho de sua chave. Os valores de força criptográficas para algoritmos simétricos e assimétricos podem ser encontrados em relatórios anuais providos por organizações como NIST [Barker et al. 2012], entre outros.

Da mesma maneira, o valor de f pode ser determinado dependendo do algoritmo utilizado e corresponde ao maior *throughput* publicamente conhecido para dado algoritmo. Por exemplo, a máquina customizada *COPACOBANA RIVYERA* [SciEngines 2008] desenvolvida para quebrar o algoritmo DES (*Data Encryption Standard*) tem um *throughput* $f = 292$ bilhões de chaves por segundo e pode encontrar uma chave DES em menos de um dia [SciEngines 2008].

O SDSE é responsável por registrar e gerenciar os nomes dos algoritmos de criptografia e seus respectivos parâmetros f e s.

3.2. Probabilidade de Resiliência do Gerenciamento de Chave

Outra forma de descobrir chaves secretas é capturar fisicamente um sensor e violá-lo para ganhar acesso às chaves. O mecanismo responsável por abordar esse tipo de ataque é o *Gerenciamento de Chave*. Como os sensores compartilham chaves, a captura de um nó pode comprometer a segurança da comunicação entre outros nós. Portanto, a *Probabilidade de Resiliência do Gerenciamento de Chave* (P_R) calcula a capacidade de resistência do esquema de gerenciamento a um número x de nós capturados por adversários.

Desse modo, P_R representa a probabilidade de que um *link* de comunicação entre dois sensores quaisquer não capturados se mantenha seguro mesmo que x nós tenham sido capturados. Essa métrica é calculada da seguinte maneira:

$$P_R(x) = 1 - P_C(x) \tag{2}$$

Onde $P_C(x)$ representa a probabilidade de um *link* ser comprometido quando x nós são capturados e é definida pela probabilidade condicional $P_C(x) = P\{L_c|C_x\}$. Onde L_c é o evento de um *link* ser comprometido e C_x é o evento de x sensores serem capturados.

Para cada esquema de gerenciamento de chave, existe uma equação para calcular $P_C(x)$. Existem vários tipos de esquemas de gerenciamento para RSSF, como esquemas de chave única para toda a rede, esquemas de estabelecimento de chaves aos pares e esquemas de predistribuição de chaves. O SDSE é responsável por registrar e gerenciar os nomes dos esquemas de gerenciamento e suas respectivas equações de $P_C(x)$.

Por exemplo, no esquema de predistribuição de chave aleatória *q-composite* [Chan et al. 2003], antes da implantação da rede, cada nó escolhe k chaves de um *pool*

de chaves de tamanho $|K|$. Após a implantação, dois nós podem estabelecer um *link* seguro somente se tiverem pelo menos q chaves em comum, isto é, dois nós tem um *link* de comunicação seguro quando compartilham i chaves, sendo $q \leq i \leq k$.

Para o esquema *q-composite*, Chan et al. [Chan et al. 2003] mostraram que $P_C(x)$ é calculado pela seguinte equação:

$$P_C(x) = \sum_{i=q}^{k} \left(1 - \left(1 - \frac{k}{|K|}\right)^x\right)^i \frac{p(i)}{p} \tag{3}$$

Onde $p(i)$ é a probabilidade de um *link* ser estabelecido com i chaves e é definida por Chan et al. como: $p(i) = \frac{\binom{|K|}{i}\binom{|K|-i}{2(k-i)}\binom{2(k-i)}{k-i}}{\binom{|K|}{k}^2}$ e $p = p(q) + p(q+1) + \ldots + p(k)$, isto é, p representa a probabilidade de dois nós estabelecerem um *link* seguro.

Os valores de k, q e p são dados pelo próprio esquema de gerenciamento. A partir desses valores, é possível calcular o valor de $|K|$. Porém, para calcular $P_C(x)$, é necessário ainda obter o valor de x. Nesse caso, o valor de x pode ser facilmente obtido através do *Sistema de Detecção de Intrusão* presente na rede.

3.3. Probabilidade de Legitimidade

O Sistema de Detecção de Intrusão (IDS - *Intrusion Detection System*) de uma rede de sensores analisa o tráfego de rede em busca de atividades maliciosas e é capaz de identificar e isolar nós sensores que realizam ataques. Devido à natureza distribuída das redes de sensores, neste artigo são considerados os sistemas de detecção distribuídos e colaborativos [Farooqi and Khan 2009].

Existem vários IDSs distribuídos e colaborativos propostos para redes de sensores [Farooqi and Khan 2009]. Nestes mecanismos, cada nó sensor monitora sua vizinhança à procura de comportamento suspeito. Assim que uma atividade maliciosa é detectada, nós vizinhos trocam informações sobre o nó suspeito. Neste processo de colaboração, cada sensor vizinho de um nó suspeito deve indicar seu ponto de vista (v) em relação a esse nó, que pode ser $v = 1$ para indicar nó malicioso e $v = 0$ para indicar nó legítimo.

Ao final da colaboração, um nó é considerado pelo IDS como malicioso quando pelo menos m de seus N vizinhos indicaram que ele está realizando atividade maliciosa, isto é, indicaram $v = 1$. Detalhes dos métodos de detecção e processo de colaboração podem ser encontrados em [Farooqi and Khan 2009].

Para estimar a segurança provida por um IDS, definimos uma métrica chamada de *Probabilidade de Legitimidade* (P_L) que calcula a chance de um sensor ser legítimo. Essa métrica avalia a *confiabilidade* do IDS, pois é baseada nas taxas de falsos positivos e verdadeiros negativos de cada vizinho, isto é, taxas de acerto e erro do IDS causados por seu método de detecção. Além disso, essa métrica também avalia a *resiliência* do IDS a ataques ao seu processo de colaboração, pois considera o caso em que os nós vizinhos realizam falsas indicações por também estarem comprometidos. Ademais, como veremos a seguir, essa métrica é calculada de modo diferente, dependendo da *conclusão* do IDS sobre um nó em questão, o que reflete o atual *estado de segurança* da rede.

Considere P_f a *taxa de falsos positivos* de cada vizinho, N a quantidade de vizinhos do nó em questão e m a quantidade mínima de nós vizinhos necessária para que o IDS conclua que o nó em questão é malicioso. Considerando o caso em que o nó em questão tenha sido detectado como *malicioso* pelo IDS, definimos a *Probabilidade de Legitimidade* (P_L) como a probabilidade de pelo menos m vizinhos terem errado ao detectar o nó como malicioso, por ele tratar-se, na verdade, de um nó legítimo:

$$P_L = \sum_{j=m}^{N} \binom{N}{j} \left(\frac{P_f}{2}\right)^j \left(1 - \frac{P_f}{2}\right)^{N-j} \tag{4}$$

A taxa de falsos positivos P_f é calculada pela razão entre a quantidade de atividades normais incorretamente marcadas como intrusões e o número total de atividades normais da rede.

Observe ainda que P_f é multiplicado por $\frac{1}{2}$ para representar a probabilidade de resiliência da detecção. Para compreender a utilização de $\frac{1}{2}$, considere o caso em que os vizinhos podem estar comprometidos e, portanto, realizando falsas indicações. Assim, a probabilidade do vizinho estar comprometido (ou não) é de 50%. Desse modo, utilizamos $\frac{1}{2}$ no cálculo de P_L para representar a resiliência do IDS a falsas indicações de vizinhos.

A outra forma de calcular P_L é utilizada quando o IDS conclui que o nó é legítimo. Neste caso a chance do nó ser realmente legítimo é dado pela probabilidade do IDS ter acertado em sua conclusão. Portanto, para calcular P_L, definimos a seguinte equação:

$$P_L = \sum_{j=N-m+1}^{N} \binom{N}{j} \left(\frac{P_n}{2}\right)^j \left(1 - \frac{P_n}{2}\right)^{N-j} \tag{5}$$

Onde P_n é a *especificidade* ou *taxa de verdadeiros positivos* de cada nó vizinho, definida como a razão entre as atividades normais corretamente marcadas como normais e o número total de atividades normais da rede. Observe que na equação 5, para que o nó não seja classificado como malicioso, deve haver no mínimo $N - m + 1$ nós vizinhos indicando que o nó em questão não é malicioso, isto é, indicando $v = 0$.

Para o cálculo de P_L, o valor de m é fornecido pelo próprio IDS e o valor de N pode ser facilmente obtido pelo algoritmo de roteamento da rede, já os valores de P_f e P_n devem ser previamente estabelecidos a partir de análise estatística do comportamento do IDS ou simulações que testam o comportamento do IDS em um ambiente controlado. Geralmente, as taxas para determinados cenários e tipos de redes são dadas nos próprios artigos em que os IDSs são propostos. O SDSE é responsável por registrar e gerenciar os nomes dos IDSs e suas respectivas taxas P_f e P_n.

3.4. Probabilidade de Entrega

Os mecanismos de *Gerenciamento de Confiança* são usados para medir a confiabilidade de nós sensores de acordo com seu comportamento na rede. Confiança é um relacionamento de três partes, que pode ser expresso como "entidade A confia na entidade B para fazer X" [Shaikh et al. 2009]. Em redes de sensores, a parte "fazer X" normalmente se refere ao repasse (entrega) de pacotes e as entidades A e B referem-se aos a nós sensores.

Uma abordagem comum para se avaliar a confiabilidade de um nó b para um nó a é calcular a probabilidade de b entregar pacotes provenientes de a de maneira satisfatória. O valor de confiança de um nó pode ser calculado a partir de interações diretas ou recomendações de vizinhos confiáveis sobre um nó especifico. De modo geral, o valor de confiança de um nó a em um nó b pode ser expresso por: $T_{a,b} = \frac{S_{a,b}}{S_{a,b} + U_{a,b}}$. Onde $S_{a,b}$ é o número total de interações bem sucedidas do nó a com o nó b e $U_{a,b}$ é o número total de interações mal sucedidas do nó a com o nó b. Neste caso, o valor de confiança é um valor entre 0 e 1 (inclusive). Contudo, cada mecanismo de Gerenciamento de Confiança pode ter valores de confiança em diferentes intervalos, tais como $[-1, 1]$ e $[0, 100]$, além de poder utilizar uma forma modificada da equação $T_{a,b}$ acima.

Por exemplo, Shaikh et al. [Shaikh et al. 2009] propõem um mecanismo de Gerenciamento de Confiança chamado GTMS que calcula $T_{a,b}$ com a seguinte equação:

$$T_{a,b} = \left[100 \left(\frac{S_{a,b}}{S_{a,b} + U_{a,b}} \right) \left(1 - \frac{1}{S_{a,b} + 1} \right) \right] \tag{6}$$

Onde $[\cdot]$ é a função do inteiro mais próximo e $S_{a,b}$ e $U_{a,b}$ são calculados para um intervalo de tempo Δt. Observe que o valor de $T_{a,b}$ está no intervalo $[0, 100]$.

Para o SDSE, definimos a métrica *Probabilidade de Entrega* (P_E) como o valor de confiança fornecido pelo mecanismo de Gerenciamento de Confiança convertido para o intervalo $[0, 1]$, como mostra a seguinte equação de normalização:

$$P_E = \frac{T_{a,b} - T_{min}}{T_{max} - T_{min}} \tag{7}$$

Onde T_{min} e T_{max} representam respectivamente o valor mínimo e o valor máximo de confiança possíveis para um nó. No caso do GTMS, $T_{min} = 0$ e $T_{max} = 100$.

Essa métrica provê dinamicamente o atual *estado de segurança* de cada nó. Quanto maior for o valor de confiança de um nó, mais seguro ele será e, consequentemente, os dados que ele repassa.

É importante ressaltar que o SDSE não calcula o valor de confiança de um nó, pois isso já é feito pelo mecanismo de gerenciamento de confiança instalado na rede. Em nosso exemplo, o GTMS utiliza a equação 6 para calcular o valor de confiança e o SDSE apenas normaliza esse valor para o intervalo [0,1]. Portanto, o SDSE é responsável por registrar e gerenciar os nomes dos esquemas de gerenciamento de confiança e seus respectivos intervalos $[T_{min}, T_{max}]$.

Apesar da avaliação da entrega de pacotes também ser feita pelo IDS, o gerenciamento de confiança realiza esta tarefa de maneira mais precisa, visto que ele obtém maiores taxas de acerto por ser uma solução específica para abordar ataques à entrega de pacotes. Por isso, quando há um ataque relacionado à entrega de pacotes, mesmo que o IDS o detecte, nós utilizamos a métrica do IDS (P_L, eq. 5) como se não tivesse ocorrido esse tipo de ataque, visto que ele já foi abordado pelo gerenciamento de confiança. Por outro lado, para os inúmeros ataques que não se relacionam à entrega de pacotes, nós utilizamos a métrica P_L da eq. 4.

4. Cálculo do Nível de Segurança

O Nível de Segurança (SL - *Security Level*) é um valor no intervalo $[0, 1]$ atribuído para cada dado originado em um sensor para indicar o quão seguro esse dado é. Neste caso, *dado* de um sensor refere-se às leituras realizadas pelo sensor retornadas como resposta a uma consulta de dados solicitada na estação base.

Assim que uma resposta chega à estação base, o SDSE obtém os parâmetros dos mecanismos de segurança e do mecanismo de roteamento. Na próxima seção, discutiremos o modo utilizado pelo SDSE para obter os parâmetros dos mecanismos de segurança.

Após obtenção dos valores dos parâmetros, o SDSE calcula as métricas (definidas na Seção 3) para cada nó pertencente à rota pela qual a resposta de consulta passou até chegar à estação base. Depois, o SDSE utiliza os valores das métricas para calcular o SL.

Formalmente, para a rota $R = \{n_i \mid i = 1, 2, \ldots, q\}$ em questão, o SDSE inicialmente calcula para cada nó n_i pertencente à rota R, o grau de segurança (g_i) do nó n_i como o produto das métricas calculadas para este nó, visto que cada métrica corresponde a eventos independentes entre si, isto é, o valor de uma métrica não influencia o valor das outras. Esse cálculo é mostrado na seguinte equação:

$$g_i = P_F^i \times P_R^i \times P_L^i \times P_E^i \tag{8}$$

Dessa maneira, obtém-se o conjunto $G = \{g_i \mid i = 1, 2, \ldots, q\}$ contendo os graus de segurança de todos os nós da rota R. Como o nó com menor grau de segurança da rota representa o elo mais fraco e tem maiores chances de fazer com que o dado deixe de ser confiável, o SL é calculado como o mínimo valor de G, como mostra a seguinte equação:

$$SL = \min(g_1, g_2, \ldots, g_q) \tag{9}$$

Por fim, o SDSE atribui o SL calculado à respectiva resposta da consulta e a entrega aos usuários interessados.

5. Funcionamento do SDSE

Nesta seção, apresentamos todos os detalhes do funcionamento do SDSE, desde a geração da consulta pelo usuário, até a entrega da resposta juntamente com seu respectivo SL.

Antes de apresentarmos a consulta, é importante deixar claro que o SDSE possui previamente armazenados vários nomes de mecanismos de segurança e os respectivos valores de seus parâmetros estáticos, tais como os parâmetros de criptografia f (quantidade de chaves testadas por unidade de tempo) e s (força criptográfica), por se tratarem de valores fornecidos por fontes externas, tais como o NIST [Barker et al. 2012].

Além disso, como o SDSE fica instalado na estação base como uma aplicação do *TinyOS* [Karlof et al. 2004], sistema operacional dos nós sensores, ele é capaz de se comunicar com o TinyOS e obter o nome dos algoritmos instalados na rede. Com essa informação, ele pode ler sua base de dados e carregar na memória os parâmetros estáticos para os nomes de mecanismos fornecidos pelo TinyOS.

No entanto, o SDSE também precisa solicitar dos respectivos mecanismos de segurança, os seus parâmetros dinâmicos, tais como os parâmetros do gerenciamento de

chave q (número mínimo de chaves em comum para estabelecer um *link* seguro entre dois nós) e k (número de chaves armazenadas por cada nó). Para isso, o SDSE precisa se comunicar diretamente com esses mecanismos.

A comunicação direta entre o SDSE e os mecanismos da rede é feita através dos módulos e interfaces providos pelas aplicações executadas no *TinyOS*. Portanto, o SDSE verifica em sua base de dados quais métodos públicos são oferecidos pelos mecanismos da rede. De posse dessa informação, o SDSE está pronto para realizar chamadas aos métodos fornecidos pelas interfaces dos mecanismos e obter seus respectivos parâmetros dinâmicos quando necessário.

Entretanto, para que o SDSE consiga ter acesso aos parâmetros dos mecanismos, é preciso que esses mecanismos tenham um módulo na estação base capaz de fornecer tais informações. Essa é uma condição válida, visto que a maioria dos mecanismos de RSSF possui um módulo na estação base que gerencia seu funcionamento, como é o caso dos mecanismos de segurança abordados neste artigo (criptografia, gerenciamento de chave, detecção de intrusão e gerenciamento de confiança). Isto também se aplica aos algoritmos de roteamento, pois em vários deles a árvore de roteamento é gerada na estação base, como mostram Radi et al. [Radi et al. 2012]. Isto lhes permite conhecer todas as rotas da rede e a quantidade de vizinhos de cada sensor, como é o caso do roteamento *Directed Diffusion* [Radi et al. 2012], que é próprio para aplicações orientadas a consultas.

Dito isto, podemos apresentar o funcionamento do SDSE a partir do momento da geração da consulta. Para ilustrar, utilizaremos uma consulta no formato SQL que pode ser interpretada por aplicações como *TinyDB* e *Cougar* [Gehrke and Madden 2004].

Suponhamos então que um usuário W deseje que o nó n_6 envie algumas leituras para a estação base de 5 em 5 segundos, durante 1 minuto. Então ele realiza a seguinte consulta na estação base: ***SELECT temperatura, umidade, pressão, luminosidade FROM sensors s WHERE s.nodeid = 6 SAMPLE PERIOD 5s FOR 60s***.

Essa consulta é então disseminada pela rede até chegar a n_6. No momento em que este nó recebe a consulta, ele realiza as leituras necessárias e envia as respostas à estação base nos tempos determinados. Suponhamos ainda que as respostas enviadas por n_6 sejam encaminhadas pelos nós n_5, n_4, n_3, n_2 e n_1 até chegar à estação base. Neste caso, temos $R = \{n_1, n_2, n_3, n_4, n_5, n_6\}$.

Assim que cada resposta é recebida pela estação base, o SDSE inicia o processo para o cálculo do SL dessa resposta. Inicialmente, o SDSE solicita ao algoritmo de roteamento os $nodeids$ dos nós da rota R. Então, o SDSE realiza os quatro passos a seguir, para obter os parâmetros das métricas de segurança de cada nó de R.

Primeiro, o SDSE obtém o parâmetro t de cada nó e calcula a *Probabilidade de Força Criptográfica* (P_F), visto que ele já possui em memória os parâmetros estáticos f e s (equação 1) necessários para calcular P_F. Veja que os valores de f e s são os mesmos para todos os sensores da rede.

Segundo, o SDSE solicita ao mecanismo de detecção de intrusão a quantidade de nós capturados da rede (parâmetro dinâmico x) e calcula a *Probabilidade de Resiliência do Gerenciamento de Chave* (P_R), visto que ele já possui em memória os valores dos parâmetros estáticos k, q e p do gerenciamento de chave (que neste exemplo é o esquema

q-composite), necessários para calcular P_R (ver equações 2 e 3). Observe que os valores de k, q, p e x são os mesmos para todos os sensores da rede.

Terceiro, o SDSE obtém do mecanismo de detecção de intrusão sua conclusão em relação à cada nó de R. Para os nós indicados como maliciosos pelo IDS, o SDSE deve utilizar a *Probabilidade de Legitimidade* (P_L) para nós maliciosos (equação 4). Para os nós indicados como legítimos pelo IDS, o SDSE deve calcular a *Probabilidade de Legitimidade* (P_L) para nós legítimos (equação 5). Após descobrir qual equação deve ser utilizada, o SDSE solicita ao algoritmo de roteamento o valor do parâmetro dinâmico N (quantidade de vizinhos) de cada nó. Em seguida, o SDSE calcula a *Probabilidade de Legitimidade* (P_L), uma vez que ele já possui os valores dos parâmetros estáticos m, P_f e P_n. Observe que os valores desses parâmetros são os mesmos para cada sensor. Diferentemente dos valores do parâmetro N, que são diferentes para cada sensor.

Quarto, o SDSE obtém do gerenciamento de confiança (que neste exemplo é o esquema GTMS [Shaikh et al. 2009]), o valor de confiança (ver equação 6) de cada nó pertencente a R. Com esses valores, o SDSE calcula a *Probabilidade de Entrega* (P_E) de cada nó por meio da normalização desses valores para o intervalo $[0, 1]$ (equação 7).

Após calcular as métricas de segurança para cada nó, o SDSE segue para o cálculo do SL. Para isso, ele calcula o grau de segurança g_i (equação 8) de cada nó de R e em seguida, obtém o SL (equação 9), que é o menor valor g_i encontrado entre os nós de R.

Por fim, o usuário W recebe a resposta para sua consulta juntamente com seu respectivo *Nível de Segurança* (SL). Supondo-se que o nível de segurança calculado para essa consulta tenha sido $SL = 0.9$, o usuário W pode interpretá-lo da seguinte maneira: *a resposta para minha consulta possui 90% de chance de estar correta, isto é, de ser uma resposta confiável, que não tenha sofrido ataques nem tenha sido adulterada indevidamente.* Em outras palavras, ela é um dado que possui 90% de chance de ser seguro.

6. Análise das Métricas e do Nível de Segurança

Nesta seção, o comportamento dos valores das métricas e do nível de segurança é analisado de acordo com variados valores de parâmetros dos mecanismos de segurança.

6.1. Probabilidade de Força Criptográfica

Para mostrar o comportamento de P_F, escolhemos o algoritmo de criptografia RC5 [Karlof et al. 2004], por ser um algoritmo bastante conhecido e ser implementado pelo TinySec [Karlof et al. 2004] (arquitetura de segurança implementada no TinyOS).

O RC5 possui um tamanho de chave variável (0 a 2040 bits). A organização *Distributed.net* [Distributed.net 2002] possui projetos para quebra das chaves do RC5 que chegam à uma taxa $f = 394.254.429.396$ chaves por segundo. A Figura 1(a) apresenta os valores de P_F variando com o tempo t, para diferentes valores de força criptográfica s.

Observe na Figura 1(a) que quanto maior a força criptográfica de um algoritmo, mais seguro ele será. Por outro lado, à medida que o tempo passa, a *Probabilidade de Força Criptográfica* diminui, visto que mais chaves podem ser testadas, o que aumenta as chances de um adversário quebrar o algoritmo.

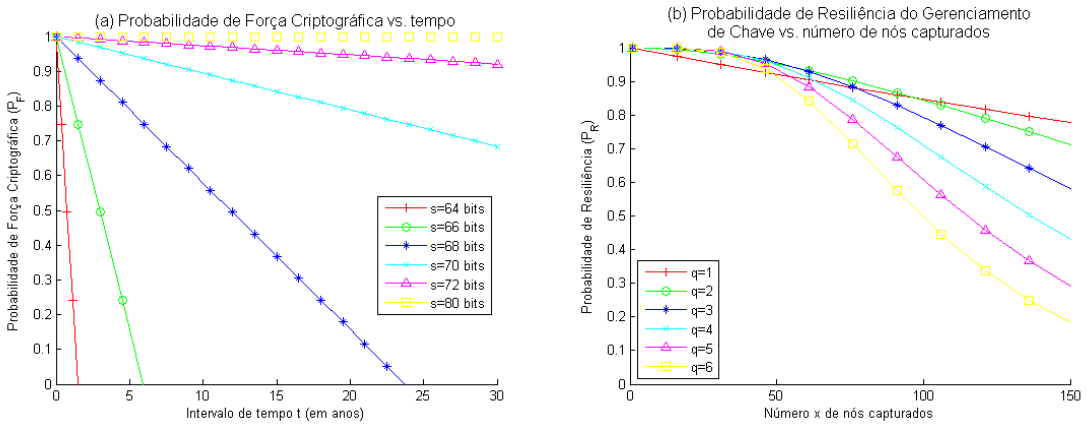


Figura 1. (a) Probabilidade de Força Criptográfica (P_F) em relação ao tempo t, para diferentes valores de s e taxa $f = 394.875.793.722$ chaves/segundo. (b) Probabilidade de Resiliência do Gerenciamento de Chave quando um adversário captura x nós da rede, para diferentes valores de q, para $k = 200$ e $p = 0,33$.

6.2. Probabilidade de Resiliência do Gerenciamento de Chave

Para apresentar o comportamento de P_R, escolhemos o esquema *q-composite* [Chan et al. 2003], por ser um mecanismo bastante difundido e que serve de comparação para vários outros mecanismos de gerenciamento de chave para RSSF.

A Figura 1(b) apresenta os valores de P_R quando x nós são capturados, para diversos valores de q (quantidade mínima de chaves necessárias para estabelecer um link seguro entre dois nós). Nesta figura, temos o número de chaves armazenada por cada nó $k = 200$ e a probabilidade de estabelecimento de um *link* seguro $p = 0,33$. A partir dos valores de k e p, o tamanho do *pool* de chaves $|K|$ é obtido para cada valor de q.

Na Figura 1(b), podemos perceber que até por volta de 50 nós capturados, a *Probabilidade de Resiliência* se mantém bastante próxima de 1, independentemente do valor de q. Entretanto, a partir de 50 nós capturados, P_R diminui de maneira mais significativa, além de ser menor para maiores valores de q.

6.3. Probabilidade de Legitimidade

Para mostrar o comportamento de P_L, escolhemos o IDS distribuído e cooperativo proposto por Krontiris et al. [Krontiris et al. 2009], por ser um IDS implementado e testado no *TinyOS* e por servir de base para vários outros IDSs colaborativos para RSSF.

A Figura 2(a) apresenta o comportamento de P_L para nós detectados como maliciosos pelo IDS e a figura 2(b) mostra o comportamento de P_L para nós apontados como não maliciosos pelo IDS. Nessas figuras, P_L é calculado para diferentes valores de P_f (taxa de falsos positivos de um nó) e P_n (taxa de verdadeiros negativos de um nó), respectivamente. A quantidade de vizinhos utilizada é $N = 10$.

Observe que na Figura 2(a), P_L diminui à medida que o parâmetro de consenso m aumenta, isto é, à medida que mais nós vizinhos são necessários para detectar um nó como malicioso. Isso acontece quando os valores de P_f são menores que 0.5, isto é, quanto mais nós forem necessários para detectar seu vizinho como malicioso, mais serão os casos em que as probabilidades baixas de P_f precisarão ocorrer. Isso também significa

que, quanto maior P_f, maior a *Probabilidade de Legitimidade*.

Já na Figura 2(b), como os valores de P_n são maiores que 0.5, a *Probabilidade de Legitimidade* aumenta à medida que m aumenta. Da mesma maneira, quanto maior P_n, maior a *Probabilidade de Legitimidade*. Observe ainda que grande parte dos valores de P_L da figura 2(b) é maior que os valores de P_L da figura 2(a), isto acontece porque um nó apontado como legítimo pelo IDS, possui muito mais chances de ser realmente legítimo do que um nó apontado como malicioso.

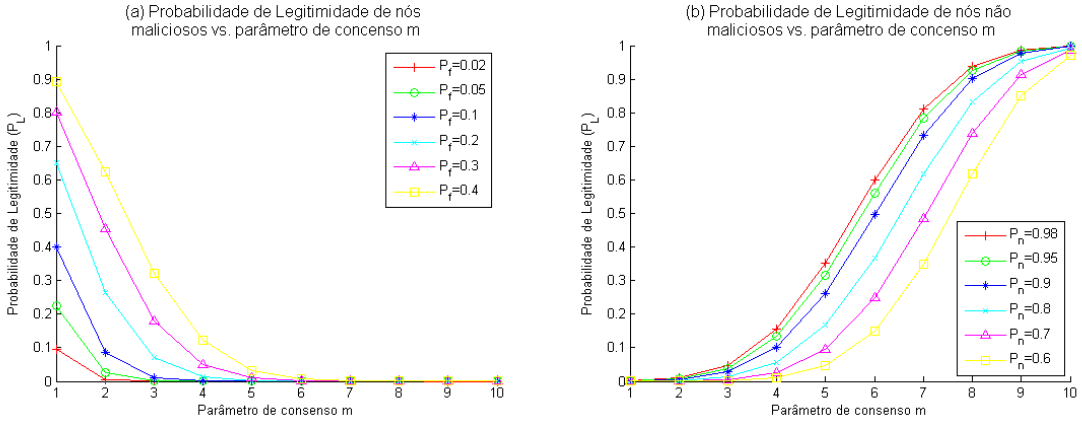


Figura 2. (a) Probabilidade de Legitimidade de nós maliciosos em relação a m, para diferentes valores de P_f e $N = 10$. (b) Probabilidade de Legitimidade de nós não maliciosos em relação a m, para diferentes valores de P_n e $N = 10$.

6.4. Probabilidade de Entrega

Para apresentar o comportamento de P_E, selecionamos o esquema de gerenciamento de confiança GTMS, por se tratar de um mecanismo bastante conhecido e estar entre os melhores mecanismos de gerenciamento de confiança para RSSF.

A Figura 3(a) apresenta os valores de P_E para variadas quantidades de interações bem sucedidas entre dois nós (S). Nesta figura, o eixo x representa a fração de interações bem sucedidas, isto é, a expressão $(\frac{S}{S+U})$ da equação 6.

Na Figura 3(a), vemos que a *Probabilidade de Entrega* aumenta à medida que a fração de interações bem sucedidas cresce. Além disso, quando o número de interações bem sucedidas é $S = 5$, P_E é bem menor do que para os outros valores de S, que possuem praticamente o mesmo P_E para frações de interações bem sucedidas entre 0 e 0.4. No entanto, para frações a partir de 0.4, P_E é ligeiramente maior para maiores valores de S.

6.5. Nível de Segurança

Para compreendermos o comportamento do *Nível de Segurança*, de acordo com as métricas de segurança utilizadas, a Figura 3(b) mostra como o *Nível de Segurança* varia considerando-se a quantidade de vizinhos de um nó (N) e o número x de nós capturados na rede. Para gerar a Figura 3(b), foram utilizados os mesmos algoritmos das subseções anteriores com os seguintes valores de parâmetros:

- **Criptografia - RC5.** $f = 394, 254, 429, 396$ chaves/s, $t = 3$ anos, $s = 80$ bits;
- **Ger. de Chave - *q-composite*.** $k = 200$ chaves, $q = 2$ chaves, $p = 0, 33$;

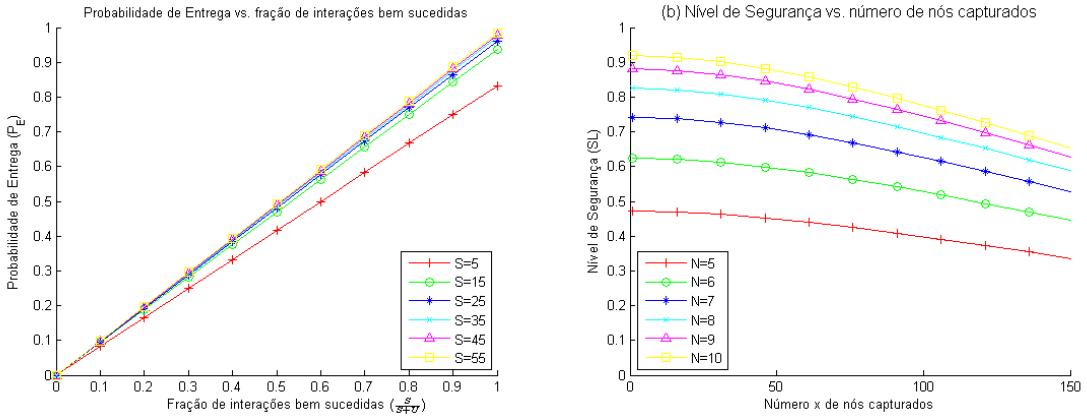


Figura 3. (a) Probabilidade de Entrega (P_E) em relação à fração de interações bem sucedidas ($\frac{S}{S+U}$). (b) Nível de Segurança de um nó para diferentes quantidades de vizinhos (N), quando um adversário captura x nós da rede.

- **IDS de Krontiris et al.** $m = N - 2$ nós, $P_n = 0,98$;
- **Gerenciamento de Confiança - GTMS.** $T = 0,98$.

Na Figura 3(b), é possível visualizar como o *Nível de Segurança* diminui à medida que a quantidade de nós capturados aumenta, o que já era de se esperar, visto que é o mesmo comportamento da *Probabilidade de Resiliência do Gerenciamento de Chave* (P_R), que é afetada pelo valor de x. Por outro lado, quanto maior a quantidade de vizinhos de um nó, maior será seu *Nível de Segurança*. Isto acontece porque um número maior de vizinhos de um nó gera uma maior *Probabilidade de Legitimidade* (P_L). Portanto, a partir dessa figura, podemos perceber como a alteração de parâmetros referentes a métricas distintas influenciam o valor final do SL.

7. Conclusões

Este artigo apresentou o *Sensor Data Security Estimator* (SDSE), um estimador do grau de confiabilidade dos dados de RSSF que utiliza métricas calculadas a partir de parâmetros dos mecanismos de segurança. Com a utilização do *nível de segurança*, os usuários podem tomar decisões informadas quanto ao uso dos dados dos sensores.

A avaliação apresentada neste artigo nos permitiu analisar o comportamento das métricas e do *Nível de Segurança* perante diferentes valores de parâmetros dos mecanismos. Isso permite que o usuário tenha conhecimento do que afeta a segurança dos dados de sua rede, além de permitir que profissionais de segurança utilizem o modelo proposto para selecionar valores de parâmetros adequados para garantir maior segurança.

Além disso, a proposta mostrou-se viável, uma vez que foram mostrados exemplos de como extrair efetivamente os parâmetros dos mecanismos. Atualmente a base de dados do SDSE está senso construída e já possui vários mecanismos, parâmetros e seus respectivos métodos de obtenção registrados.

Para avaliar a dinâmica do modelo proposto, estamos atualmente com um trabalho em andamento que inclui uma simulação detalhada do SDSE em uma rede com diferentes mecanismos de segurança implementados. Esperamos que estes resultados estejam disponíveis em breve em nosso próximo artigo.

Referências

Ahmed, M. S., Al-Shaer, E., and Khan, L. (2008). A Novel Quantitative Approach For Measuring Network Security. In *2008 IEEE INFOCOM - The 27th Conference on Computer Communications*, pages 1957–1965. IEEE.

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

Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. (2012). Recommendation for Key Management - Part 1 : General (Revision 3). *Technical Report*, 1(July):1–147.

Chan, H., Perrig, A., and Song, D. (2003). Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 197–. IEEE Computer Society.

Distributed.net (2002). distributed.net - RC5-72 Overall Project Stats.

Farooqi, A. and Khan, F. (2009). Intrusion detection systems for wireless sensor networks: A survey. In *Communication and Networking*, volume 56, pages 234–241. Springer Berlin Heidelberg.

Frigault, M., Wang, L., Singhal, A., and Jajodia, S. (2008). Measuring network security using dynamic bayesian network. In *Proceedings of the 4th ACM workshop on Quality of protection*, pages 23–30. ACM.

Gehrke, J. and Madden, S. (2004). Query processing in sensor networks. *IEEE Pervasive Computing*, 3:46–55.

Karlof, C., Sastry, N., and Wagner, D. (2004). Tinysec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, New York, NY, USA. ACM.

Krontiris, I., Benenson, Z., Giannetsos, T., Freiling, F. C., and Dimitriou, T. (2009). Cooperative intrusion detection in wireless sensor networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks*, pages 263–278. Springer-Verlag.

Ksiezopolski, B. and Kotulski, Z. (2005). On scalable security model for sensor networks protocols. In *22nd CIB-W78 Conference Information Technology in Construction (CIB- W78)*, pages 463–469.

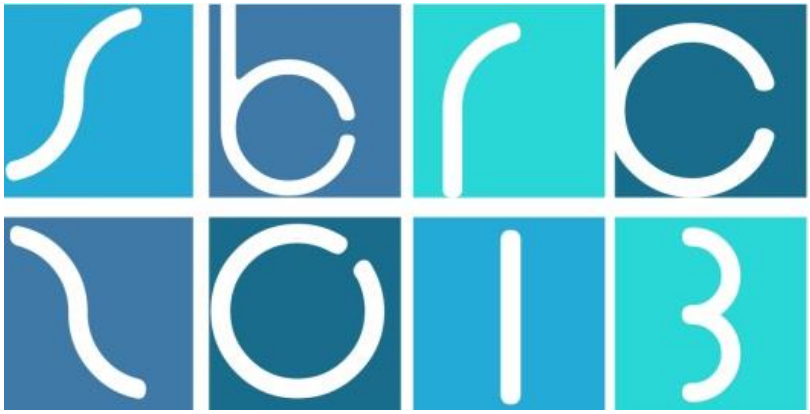
Li, X., Parker, P., and Xu, S. (2011). A stochastic model for quantitative security analyses of networked systems. *IEEE Trans. Dependable Secur. Comput.*, 8(1):28–43.

Radi, M., Dezfouli, B., Bakar, K. A., and Lee, M. (2012). Multipath routing in wireless sensor networks: Survey and research challenges. *Sensors*, 12(1):650–685.

Savola, R. (2008). Holistic Estimation of Security, Privacy and Trust in Mobile Ad Hoc Networks. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6, Damascus. Ieee.

SciEngines (2008). Break DES in less than a single day (COPACOBANA RIVYERA).

Shaikh, R. A., Jameel, H., d'Auriol, B. J., Lee, H., Lee, S., and Song, Y.-J. (2009). Group-based trust management scheme for clustered wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 20(11):1698–1712.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 6

Redes Móveis: Antenas e Múltiplos Canais

Mecanismo Conjunto de Atribuição de Canais e Roteamento para Redes em Malha Sem Fio de Múltiplos Rádios*

Celso Barbosa Carvalho[1], José Ferreira de Rezende[2]

[1]GPComp - ICET – Universidade Federal do Amazonas (UFAM)

[2]COPPE – Universidade Federal do Rio de Janeiro (UFRJ)

ccarvalho_@ufam.edu.br,rezende@land.ufrj.br

Abstract.

Resumo. *Este trabalho demonstra que a atribuição de múltiplos canais com diferentes larguras de banda aos enlaces de uma rede em malha sem fio IEEE 802.11 pode aumentar drasticamente a capacidade dessas redes, tudo isso para uma mesma quantidade de espectro disponível. No entanto, isso somente é possível em redes equipadas com múltiplos rádios e que utilizam mecanismos de atribuição de rádios e canais aos enlaces e de escolha de rotas que levem em conta essa atribuição. Levando isto em consideração, este artigo propõe um mecanismo conjunto de seleção da quantidade de rádios e da largura dos canais a serem utilizados em cada enlace, assim como uma métrica de roteamento que permite dimimuir a interferência entre esses enlaces. Os mecanismos propostos são avaliados através de simulações no ns-2 e comparados com outros mecanismos existentes na literatura, mostrando assim o aumento de capacidade obtido nessas redes.*

1. Introdução

As redes em malha sem fio (*Wireless Mesh Networks*-WMNs) possuem grande aplicabilidade e, como resultado é importante aumentar a máxima vazão fim-a-fim dos caminhos utilizadas pelos seus fluxos, também chamada de capacidade da rede ($bits/\mathrm{s}$) [Zhai and Fang 2006, Chimento and Ishac 2008].

A fim de aumentar a vazão das redes em malha sem fio, a maior parte dos trabalhos relacionados [De Couto et al. 2005] consideram o uso de canais de comunicação de largura fixa (ex: 20 MHz para a tecnologia IEEE 802.11). Entretanto, existem pesquisas que mostram que o desempenho das redes sem fio pode ser melhorado ao utilizar canais de comunicação de diferentes larguras (ex: 5, 10 e 20 MHz) [Chandra et al. 2008].

Ao utilizar canais de menor largura (ex: $5\mathrm{MHz}$), é possível aumentar a capacidade da rede quando o número de enlaces disputando o espectro é grande. Em primeiro lugar, a divisão do espectro em uma maior quantidade de canais ortogonais reduz a contenção dos enlaces. Em segundo lugar, aumenta-se a eficiência espectral da rede ao utilizar transmissões paralelas em canais de menor largura. Para exemplificar, 04 canais de $5\mathrm{MHz}$ podem transmitir paralelamente e em menor tempo 4 quadros, quando comparado com um único canal de $20\mathrm{MHz}$. No caso do canal de $20\mathrm{MHz}$, despende-se, serialmente, 4

*Este trabalho recebeu recursos da UFAM, FAPEAM, SECT/AM, CAPES e CNPq.

vezes os tempos de espera da camada MAC na transmissão. Em contrapartida, quando a quantidade de enlaces disputando o espectro é pequena, a melhor eficiência espectral é obtida ao utilizar os canais de maior largura [Carvalho and de Rezende 2010], uma vez que possuem maior capacidade [Yuan et al. 2007]. Outra vantagem dos enlaces estabelecidos em canais de menor largura trata-se deles possuírem maior alcance de transmissão. O alcance de transmissão de um enlace depende da mínima potência necessária para o receptor decodificar o sinal transmitido. Esta potência, chamada de sensibilidade mínima (S), é diretamente proporcional à largura do canal e, sendo assim, quanto menor a largura do canal, menor é o valor de S e consequentemente, maior o alcance de transmissão [Chandra et al. 2008, Carvalho and de Rezende 2010].

Em [Chandra et al. 2008], são realizados experimentos que mostram os efeitos de utilizar canais de diferentes larguras na vazão e alcance dos sinais transmitidos. No artigo, é desenvolvido um algoritmo de adaptação que determina a modulação e largura de canal de um enlace onde os nós são equipados com um rádio de transmissão.

Yuan et.al [Yuan et al. 2007] desenvolve um protocolo MAC (*Medium Access Control*) e algoritmo capazes de adaptar a largura de canal, frequência e tempo de transmissão de Rádios Cognitivos (CRs). Os CRs são equipados com 2 rádios, um para procurar os *white spaces* e outro para transmissão. Para isso, os autores propõem modificações ao MAC 802.11 para criar as mensagens do protocolo proposto.

Em [Carvalho and de Rezende 2010], é proposta uma métrica que gera valores utilizados para executar o roteamento e determinar a quantidade de rádios e largura de canal a serem utilizadas nos enlaces de uma WMN. No entanto, a quantidade de rádios é determinada através de um cálculo estático de capacidade e os valores da métrica não contabilizam interferências.

Neste artigo, trabalhamos em cenários de WMNs onde os roteadores podem adaptar a largura do canal de comunicação. Então, propomos uma métrica, implementada na camada de rede, e cujo objetivo é aumentar a capacidade fim-a-fim das rotas e, consequentemente, aumentar a capacidade da rede. Os valores da métrica são utilizados para se definir o roteamento dos fluxos, realizar a atribuição de canais, selecionar a largura de cada canal e escolher, em função da interferência existente, a quantidade de rádios de transmissão utilizados em cada enlace. De acordo com nossa pesquisa bibliográfica, este é o primeiro artigo a propor uma métrica cujos valores são utilizados para executar todas estas tarefas. Para avaliar a nossa proposta e comparar com outras métricas da literatura, utilizamos simulações no NS-2 nas quais é empregado o modelo de interferência físico [Augusto et al. 2010].

Para apresentar a pesquisa, o artigo é dividido nas seguintes seções. A Seção 2 apresenta trabalhos relacionados; A Seção 3 mostra a metodologia utilizada para simular canais com diferentes larguras no NS-2; Na Seção 4, apresentam-se a métrica e mecanismos propostos; Na Seção 5, a métrica e mecanismo propostos no artigo são avaliados e os resultados são comparados com os de outros trabalhos da literatura; Na Seção 6, são apresentadas as conclusões.

2. Trabalhos Relacionados

A métrica ETX (*Expected Transmission Count*) [De Couto et al. 2005] utiliza medidas das taxas de entrega de quadros no enlace direto d_f e reverso d_r. A equação $ETX =$

$1/(d_f \times d_r)$ representa a quantidade de transmissões necessárias para que em um enlace $e_{i,j}$, o quadro de dados seja recebido em j e o quadro de ACK seja recebido em i.

A métrica ETT (*Expected Transmission Time*) [Draves et al. 2004] objetiva estimar o tempo total, incluindo retransmissões, necessário para que um quadro seja transmitido e reconhecido em um enlace. Na equação $ETT = ETX \times \frac{S}{B}$, ETT é o valor da métrica para um enlace $e_{i,j}$, ETX é o valor da métrica ETX para o mesmo enlace, S e B representam, respectivamente, o tamanho e a taxa de transmissão de quadros.

A métrica WCETT (*Weighted Cumulative Expected Transmission Time*) [Draves et al. 2004] possui valor determinado para uma rota p, conforme equação $WCETT = (1 - \beta) \times \sum_{i=1}^{n} ETT_i + \beta \times \max_{1 \leq j \leq k} X_j$. O termo $\sum_{i=1}^{n} ETT_i$ representa a soma dos $ETTs$ dos enlaces da rota. O termo $\max_{1 \leq j \leq k} X_j$, retorna a soma dos tempos de transmissão nos enlaces da rota no canal j de maior tempo de ocupação. A variável β é um parâmetro com valor no intervalo $1 \leq j \leq \beta$, onde β com valor próximo de 1 favorece a escolha de rotas de maior capacidade e β com valor próximo de 0 determina a escolha de rotas de menor atraso.

A métrica EETT (*Exclusive Expected Transmission Time*) [Jiang et al. 2007] é calculada pela equação $EETT_l = \sum_{enlace\ i\ \in\ IS(l)} ETT_i$. O $EETT_l$ de um enlace l é resultado da soma do valor ETT_i dos enlaces i que fazem parte do Conjunto Interferente (*Interference Set*) do enlace l. Onde $IS(l)$ inclui o próprio enlace l. O valor da métrica para uma rota é dado pela soma dos $EETTs$ dos enlaces da rota.

A métrica B-MTM (*Burst per Medium Time Metric*) [Carvalho and de Rezende 2010] é calculada para um enlace físico ef[1] e possui valores dados pelo inverso da soma das capacidades de todos os enlaces e, que fazem parte do enlace físico.

3. Metodologia

Nesta seção apresentamos o modelo incorporado ao NS-2 para simular diferentes larguras de canal e os efeitos desta modelagem na vazão[2], capacidade[3] e alcance de transmissão de um enlace.

3.1. Canais de Diferente Larguras e seus Efeitos na Vazão e Capacidade dos Enlaces

Para verificar os efeitos da utilização de diferentes larguras de canal na vazão e capacidade dos enlaces, utilizamos o NS-2.33 com suporte a múltiplos canais e múltiplos rádios (MCMR) proposto em [Calvo and Campo 2007]. Utilizando o modelo MCMR, é possível simular, por exemplo, o cenário de um enlace estabelecido através de 01 ou mais canais de comunicação. Com o modelo MCMR cada nó sem fio possui 01 instância das camadas de aplicação, transporte e rede, uma ou mais instâncias das camadas de enlace e física, onde cada camada física está associada a um canal ortogonal.

[1]Conjunto de enlaces formado por um ou mais enlaces individuais e estabelecidos entre dois nós [Carvalho and de Rezende 2010]

[2]A taxa ($bits/\mathrm{s}$) através da qual nenhum dos quadros transmitidos é descartado pelo receptor" [Bradner 1991].

[3]A máxima quantidade de bits que pode ser transmitida de uma fonte e corretamente recebida pelo destino através de um enlace" [Chimento and Ishac 2008].

Além disso utilizamos e implementamos modificações no MAC 802.11g de [Telecomunicazioni 2007] e no agente NOAH [EPFL-IC]. Neste último foi implementado um esquema de escalonamento onde, em um nó transmissor, cada segmento recebido da camada de transporte é encaminhado, ciclicamente pela camada de rede, para uma das instâncias da camada de enlace.

No caso do MAC 802.11g de [Telecomunicazioni 2007], foram implementadas modificações para representar as diferenças nos tempos de transmissão de quadros em canais de diferentes larguras (ex: 5, 10 e $20\mathrm{MHz}$) e modulações da camada física OFDM (ex: m54, $\cdots$, m6) [IEEE 2007]. Estas diferenças nos tempos de transmissão dos quadros se refletem em diferenças de capacidade de um enlace estabelecido entre dois nós.

As variáveis das Equações (1) a (5), com excessão da variável β, são as que aparecem codificadas na extensão de [Telecomunicazioni 2007] e com o objetivo de representar o tempo total T de transmissão e reconhecimento de uma MPDU (*MAC Protocol Data Unit*) no 802.11g, camada física OFDM (*Orthogonal Frequency Division Multiplexing*). A variável β, das Equações (4) e (5), é um parâmetro com valor dado por $\beta = 20\mathrm{MHz}/w$, onde $w \in \{5, 10, 20\mathrm{MHz}\}$ é a largura do canal de comunicação. Na determinação de β, nota-se que a medida que a largura de canal w reduz, há um aumento nos valores dos tempos de transmissão t_{DATA} e t_{ACK}, respectivamente, de um quadro de dados e de reconhecimento (*Acknowledgement*) do MAC 802.11. A introdução do parâmetro β é a alteração necessária ao MAC 802.11g de [Telecomunicazioni 2007], de maneira que sejam representados os tempos de transmissão e reconhecimento de um quadro, de acordo com a largura de canal. As demais variáveis e constantes das Equações (1) a (5) são descritas a seguir: i) t_{CW}, da Equação (2), é o tempo da janela de contenção e $t_{slot} = 20\mu\mathrm{s}$ é o tempo de um *slot*; t_{DIFS}, da Equação (3), é o tempo de espera de um *Distributed Inter-Frame Space* e $t_{SIFS} = 10\mu\mathrm{s}$ é o tempo de um *Short Inter-Frame Space*; t_{DATA} e t_{ACK} representam os tempos de transmissão de um quadro de dados e de um quadro de ACK e aparecem detalhadas nas Equações (4) e (5). Nestas duas últimas Equações, $t_{Pr} = 20\mu\mathrm{s}$, $t_{sym} = 20\mu\mathrm{s}$ e o valor 22 estão relacionados com a camada física do OFDM e representam, respectivamente, o tempo de transmissão do preâmbulo de uma MPDU, o tempo de transmissão de um símbolo OFDM e, a soma dos bits dos campos de serviço (aplicações futuras) e *tail* (delimitador de fim de quadro). Nestas mesmas equações, R é a taxa de transmissão utilizada na modulação mR (ex:a modulação $m54$ possui R=54) e $6\mu\mathrm{s}$ é o valor de tempo chamado de *Signal Extension* com função de incluir tempo adicional de processamento ao demodulador. A variável R da Equação (5) assume sempre o valor 6, uma vez que o quadro de ACK é sempre transmitido na taxa básica do 802.11. Na Equação (4) em específico $L_{MAC} = 34bytes$ e L_{DATA} (tamanho variável) representam, respectivamente, o tamanho do cabeçalho MAC e quadro de dados da camada MAC. Na Equação (5) $L_{ACK} = 14bytes$ representa o tamanho de um quadro de ACK.

$$T = t_{CW} + t_{DIFS} + t_{DATA} + t_{SIFS} + T_{ACK} \quad (1)$$

$$t_{CW} = 8 \times t_{slot} \quad (2)$$

$$t_{DIFS} = 2 \times t_{slot} + t_{SIFS} \quad (3)$$

$$t_{DATA} = \beta \times \left[t_{Pr} + t_{sym} \cdot \left(\frac{22+8 \cdot (L_{MAC}+L_{DATA})}{4 \cdot R} \right) \right] + 6\mu\text{s} \quad (4)$$

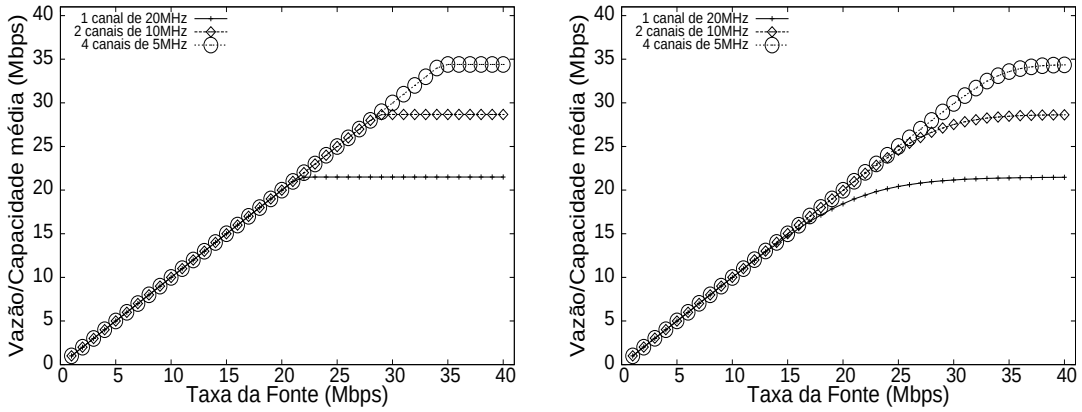
$$t_{ACK} = \beta \times \left[t_{Pr} + t_{sym} \cdot \left(\frac{22+8 \cdot (L_{ACK})}{4 \cdot R} \right) \right] + 6\mu\text{s} \quad (5)$$

Utilizamos o NS com as configurações comentadas para simular o cenário de um enlace entre dois nós A e B. Fixou-se em $m54$ a modulação utilizada para a transmissão do quadro de dados e o tempo de simulação possuia valor igual a 400s. Neste cenário variou-se a taxa da fonte CBR (*Constant Bit Rate*) de 1 até 40M$bits$/s, com o objetivo de observar o comportamento de vazão média e capacidade média do enlace, ao utilizar, respectivamente, 1, 2 ou 4 canais de 20, 10 ou 5MHz. Em cada uma das três configurações de quantidades de canais e larguras de canal simuladas, observa-se que o espectro ocupado (EO) possui valor igual a 20MHz. Simulamos duas configurações do cenário descrito e os resultados são apresentados nas Figuras 1(a) e 1(b). Na primeira configuração, o nó A utiliza fonte CBR, enquanto na segunda configuração, é utilizada uma fonte com intervalo entre geração de mensagens dado por uma função exponencial. Para ambos os tipos de fonte, as mensagens possuem tamanho de 2000 bytes.

Observa-se na Figura 1(a) que os canais de 20, 10 e 5MHz alcançam seus valores de capacidade quando a taxa da fonte CBR possui valor em torno de 22, 28 e 35M$bits$/s. Na Figura 1(b) observa-se comportamento similar, sendo que as curvas são mais suavizadas e a capacidade dos canais de 20, 10 e 5MHz é alcançada com valores em torno de 20, 30 e 35M$bits$/s de taxa da fonte exponencial. Estes resultados corroboram o que foi apresentado em [Carvalho and de Rezende 2010], onde é afirmado que para um mesmo valor de espectro ocupado, um conjunto de canais de menor largura possui maior capacidade que os canais de maior largura no IEEE 802.11g. Observamos em ambas as figuras que antes dos canais de 5, 10 ou 20MHz alcançarem suas capacidades, o valor de vazão é o mesmo para todos os canais e, portanto, as 03 curvas são coincidentes.

3.2. Canais de Diferente Larguras e seus Efeitos na Sensibilidade do Receptor

Na Figura 3.2 é mostrado como os valores de ruído de fundo (*Receiver Noise Floor*-RNF) e SINR (*Signal to Interference plus Noise Ratio*) vão se somando para determinar o valor de sensibilidade mínima de um rádio receptor para uma dada modulação. O nível de ruído representado pela variável RTN (*Receiver Thermal Noise*) é diretamente proporcional aos produtos da constante de Boltzman $K = 1.38 \cdot 10^{-23}$J/K, temperatura absoluta $T0 = 290k$ e largura do canal de comunicação w. Percebe-se, portanto que quanto menor o valor de w, menor é o valor de RTN. O próximo nível de ruído é representado pela variável $RNF(dBm)$, dada pela soma de RTN e da fígura de ruído NF que representa os ruídos gerados internamente pelo circuito receptor. O valor de sensibilidade mínima é dado pela soma de RNF e da razão sinal ruído $SINR_{Threshold}$ necessária para decodificar um sinal em uma certa modulação. Percebe-se na Figura 3.2 que quanto menor o valor da largura de canal, menor será o valor de sensibilidade, ou valor de potência necessária para decodificar um sinal em uma certa modulação e, sendo assim, maior poderá ser



(a) Fonte CBR (b) Fonte com intervalo exponencial

Figura 1. Vazão e capacidade de 1, 2 e 4 canais ortogonais de 20, 10 e 5MHz**, respectivamente.**

o afastamento entre transmissor e receptor de um enlace sem fio [Chandra et al. 2008, Carvalho and de Rezende 2010].

Nas simulações utilizamos o modelo de interferência físico [Augusto et al. 2010]. Conforme a Equação (6) é estabelecido que a probabilidade de recepção com sucesso de um quadro em um enlace e_{ij} é igual a 1, caso a razão entre a potência de recepção dos quadros do enlace ($Pr_{(i,j)}$ dada em W) e a soma das potências dos quadros de outros roteadores interferentes a j ($Pr_{(k,j)}$ em W), exceda ou seja igual ao limiar $SINR_{threshold}$. $SINR_{threshold}$ é o valor da relação entre sinal e interferência mais ruído, necessário para decodificar um sinal em uma dada modulação. Na Inequação (6), a variável $RNF_W = 10^{(RNF-30)/10}$, dada em W, possui valor que representa o valor de ruído de fundo percebido pelo receptor.

$$\frac{Pr_{(i,j)}}{\sum\limits_{k \neq i} Pr_{(k,j)} + RNF_W} \geq SINR_{threshold} \tag{6}$$

Utilizamos, ainda, durante as simulações o mecanismo de controle de taxa RA-SINR da extensão em [Telecomunicazioni 2007]. Este mecanismo transmite quadros em um enlace, mede o valor de SINR dos quadros recebidos e compara os valores de SINR e $SINR_{threshold}$, necessário para receber um quadro em uma dada modulação. Em função desta comparação, o mecanismo determina automaticamente a taxa de transmissão que possui maior taxa de entrega de dados.

Nas simulações utilizamos o cenário de um enlace AB, onde o roteador A permanece fixo no ponto $(0, 0)$ do plano cartesiano. O roteador B é móvel, inicia seu posicionamnto no ponto $(5, 0)$ e, a cada 30s, move-se 5m, terminando sua trajetória no ponto $(500, 0)$. O roteador A transmite quadros para B utilizando uma fonte CBR com taxa $10\mathrm{M}bits/\mathrm{s}$ e ambos os roteadores possuem um único rádio de comunicação. Utilizando o padrão de posicionamento descrito e perda de propagação no meio do tipo log-distância [Rappaport 2001] com expoente de perda de propagação $n = 2.86$, executou-se uma simulação para canais com larguras 5, 10 e 20MHz.

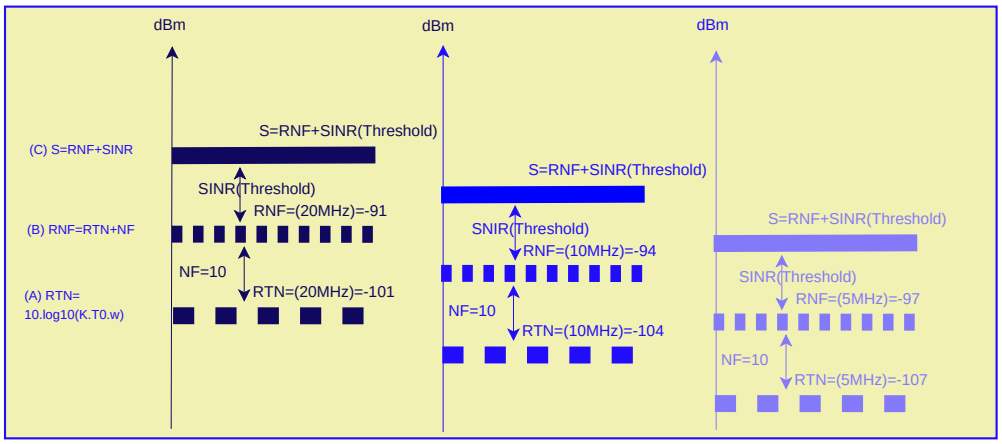


Figura 2. Sensibilidade para canais de 20, 10 e 5MHz.

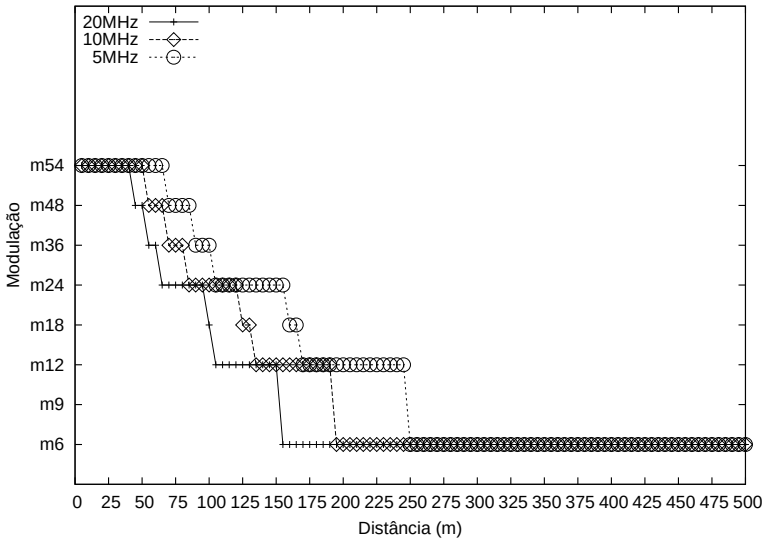


Figura 3. Distância $\times$ modulação

Observa-se na Figura 3 que com o aumento da distância entre A e B, menor é a taxa de transmissão no enlace para qualquer das larguras de canal simuladas. Na mesma figura, visualiza-se que para um dado valor de distância entre A e B, ao utilizar uma largura de canal mais estreita, emprega-se no enlace uma modulação capaz de tranferir maior quantidade de bits por símbolo (ex: modulação $m24$ transfere maior quantidade de bits por símbolo que a modulação $m18$), quando comparado aos canais de maior largura. Um exemplo pode ser observado para a distância de 75m. Nesta distância ao utilizar, respectivamente, as larguras de 20, 10 e 5MHz emprega-se no enlace as modulações $m24$, $m36$ e $m48$.

4. Métrica e mecanismos propostos

Nas subseções a seguir explicam-se os mecanismos e métricas propostos no artigo.

4.1. Divisão e utilização do Espectro Disponível (ED)

Tal como em [Li and Zhang 2009] chama-se de enlace, todo enlace individual e_{i,j,c^w} estabelecido entre um par de nós i, j no canal c^w de largura w. Chama-se de enlace físico ef_{i,j,cf^w}, ou somente ef, o conjunto formado por um ou mais enlaces individuais, estabelecidos entre os nós (i, j), em canais de largura w. O conjunto de canais de largura w através do qual um enlace físico pode ser estabelecido é chamado de canal físico cf^w. Para que um enlace físico seja estabelecido através de múltiplos canais é necessário que um par de nós i, j possua e disponibilize múltiplos rádios para se comunicarem. Para exemplificar o emprego de enlaces e canais físicos, consideremos o caso no qual o espectro disponível $ED = 40\text{MHz}$, 02 nós sem fio possuem 02 rádios de comunicação cada e podem se comunicar utilizando canais de 10MHz. Nesta situação, caso o par de nós decida utilizar apenas 01 rádio para se comunicar, ele utilizaria o ED, dividindo-o em 04 canais físicos de maneira a formar o conjunto $CF^w = \{cf^w = 1^{10}, cf^w = 2^{10}, cf^w = 3^{10}, cf^w = 4^{10}\}$. Caso o mesmo par de nós decidisse utilizar 02 rádios para comunicação no enlace, eles dividiriam o ED em 02 canais físicos $CF^w = \{cf^w = 1^{10}, cf^w = 2^{10}\}$, sendo cada canal físico composto por 02 canais individuais c^w de largura 10MHz (ex: $cf^w = 1^{10} = \{c^w = 1^{10}, c^w = 2^{10}\}$ *e* $cf^w = 2^{10} = \{c^w = 3^{10}, c^w = 4^{10}\}$).

4.2. Métrica MCWMR-BEETT

Na Equação (7) apresentam-se os cálculos de obtenção dos valores da métrica MCWMR-BEETT. A métrica é multi-objetivo, composta do produto de três variáveis α, β e γ, possui valor mínimo igual a 1.0 e quanto menor o seu valor, melhor é o resultado.

$$MCWMR - BEETT_{ef} = \alpha \cdot \delta \cdot \gamma \tag{7}$$

$$\alpha = \frac{Cap_{opt}}{Cap_{ef}} = \frac{(q_{Ref} \cdot L_{data} \cdot 8)/T}{\sum_{e \in ef} Cap_e} = \frac{(q_{Ref} \cdot L_{data} \cdot 8)/T}{\sum_{e \in ef} \left(\frac{8 * L_{data}}{EETT}\right)} \tag{8}$$

$$\delta = \max\left(\frac{|IS(ef)|}{|CF^w|}, 1\right) \tag{9}$$

$$\gamma = \max\left(\frac{ED}{EO}, 1\right) = \max\left(\frac{ED}{|IS(ef)| \times qR_{ef} \times w}, 1\right) \tag{10}$$

A variável α da Equação (7), objetiva escolher enlaces físicos de maior capacidade e é dada pela razão entre o valor de capacidade teórico Cap_{opt} de um enlace físico e a capacidade estimada Cap_{ef} para o mesmo enlace. Na Equação (8), qR_{ef} representa a quantidade de rádios utilizada no enlace físico ef, L_{data} é o tamanho do quadro e, T é o tempo de transmissão e reconhecimento de quadro em um enlace, determinado a partir da Equação (1) e que depende da modulação (utilizada a modulação m54 nos cálculos) e largura de canal $w \in \{5, 10\ e\ 20\text{MHz}\}$. A capacidade efetiva do enlace físico Cap_{ef} é dada pela somatória das capacidades Cap_e dos enlaces individuais e que fazem parte de ef. Cap_e é dada na Equação (8), onde a variável $EETT$ é o valor medido da métrica $EETT$ para todo enlace $e \in ef$.

O termo δ das Equações (7) e (9) é uma relação entre a quantidade de enlaces físicos interferentes ao enlace ef, representada por $|IS(ef)|$ e a quantidade de canais físicos existentes, dado pela conjunto $|CF^w|$, obtidos ao se dividir o espectro disponível

ED em um conjunto CF^w de canais físicos. Esta razão tem como objetivo escolher enlaces físicos com menor quantidade de outros enlaces físicos interferentes. O menor valor do termo δ é 1, indicando que um enlace possui no mínimo ele mesmo como interferente.

Por fim, o termo γ, das Equações (7) e (10), representa a razão entre o espectro disponível ED e o espectro ocupado EO, caso todos os enlaces físicos que fazem parte do conjunto $IS(ef)$ para o qual deseja-se calcular o valor da métrica, optem por empregar enlaces físicos com as mesmas características que ef. Na Equação (10), $|IS(ef)|$ é a quantidade de enlaces físicos do conjunto $IS(ef)$, qR_{ef} é a quantidade de rádios utilizadas no enlace físico ef e w é a largura de canal empregada neste mesmo enlace físico. O valor 1, dentro da função max do termo γ representa que todo o ED foi ocupado.

4.3. Determinação de Rotas

Na Equação (11) utilizou-se $BEETT_{ef}$ para representar o valor da métrica para um enlace físico ef e $BEETT_{Ro}$ para representar o valor da métrica para uma rota Ro.

$$BEETT_{Ro} = (1-\beta) \times \sum_{ef \in Ro} BEETT_{ef} + \beta \times \max_{ef \in Ro} BEETT_{ef} \tag{11}$$

Tal como na métrica WCETT, β é um parâmetro ajustável com valores entre zero (0) e um (01). Na avaliação da métrica MCWMR-BEETT utilizou-se o valor 0.5 para β. O termo $\sum_{ef \in Ro} BEETT_{ef}$ representa a soma dos valores da métrica de todos os enlaces da rota Ro. O termo $\max_{ef \in Ro} BEETT_{ef}$ representa o máximo valor da métrica para os enlaces que fazem parte da rota Ro. O primeiro termo tem como objetivo reduzir o número de saltos, uma vez que quanto maior este número, maior é o produto tempo$\times$frequência consumido pelos enlaces da rota e, consequentemente maior é o produto tempo$\times$frequência negado a outros enlaces. O segundo termo da Equação (11) tem o objetivo determinar a escolha de enlaces de maior capacidade.

Para determinar o valor da métrica de uma rota desenvolveu-se uma versão modificada do algoritmo de Dijkstra que recebe uma matriz $|V| \times |V| \times |CF|$ de métricas. Portanto, antes de determinar os valores da matriz de métricas, determina-se primeiro o tamanho da dimensão $|CF|$, onde CF é o conjunto de canais físicos e é dado por $CF = \cup_{w=1}^{W} CF^w$. A determinação do conjunto de canais físicos de largura w é realizado, conforme explicado na Seção 4.1.

5. Avaliação de Desempenho

Utilizou-se o NS-2.33 com o agente NOAH [EPFL-IC] e com as extensões MCMR de [Calvo and Campo 2007] e 802.11g de [Telecomunicazioni 2007]. Além disto, acrescentamos as seguintes funcionalidades ao simulador.

- Roteamento através de múltiplos rádios e canais e, transmissão cíclica de quadros em um enlace estabelecido através de múltiplos rádios e canais;
- Tempos de transmissão de quadros dependentes da largura de canal;
- Interferência entre canais com espectro sobreposto. Como exemplo, um $ED = 40\text{MHz}$ pode ser dividido em 2 canais $c1^{20}$ e $c2^{20}$ de 20MHz e, ainda, 4 canais de $c1^{10}, c2^{10}, c3^{10}$ e $c4^{10}$ de 10MHz. O canal $c1^{20}$ possui espectro sobreposto aos canais $c1^{10}$ e $c2^{10}$;

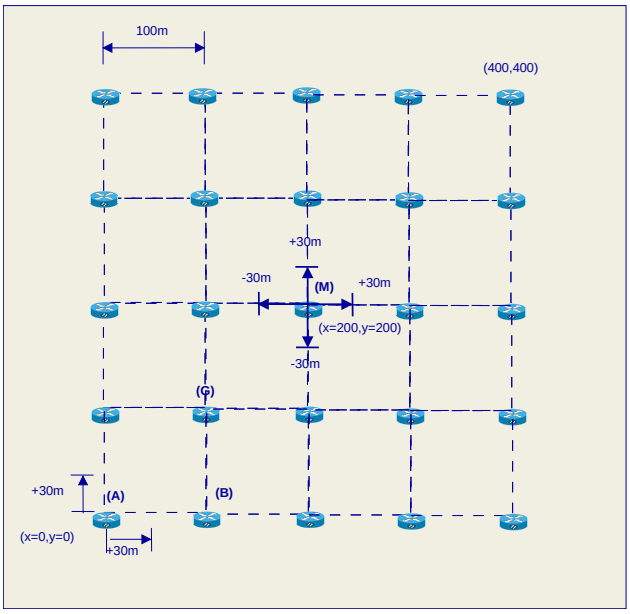


Figura 4. Cenário de avaliação

- Sensibilidade mínima dependente da largura de canal utilizada pelo rádio receptor;
- Perda de propagação log-distância;
- Métricas de roteamento ETX, ETT, WCETT, EETT, MTM e B-MTM.

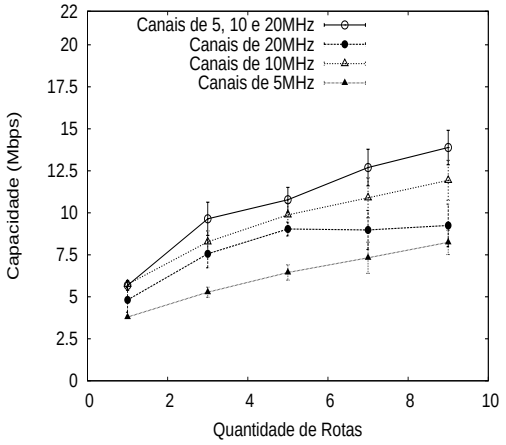
Na avaliação utilizou-se o cenário da Figura 4, de uma grade 5×5 com espaçamento entre linhas de 100m e 25 nós. Em cada rodada de simulação, os nós variam em $\pm 30\text{m}$, em ambos os eixos X e Y, o seu posicionamento em relação a intersecção das linhas e colunas. Nesta variação evita-se que os nós da borda da grade utrapassem as coordenadas com valor 0 e 400, em ambos os eixos X e Y. A variação de $\pm 30\text{m}$ no posicionamento permite a existência de distâncias mínima $d_{MIN} = 40\text{m}$ (ex: nós A e B) e máxima $d_{MAX} = 226\text{m}$ (ex: nós G e M) de separação entre vizinhos de um (01) salto. Conforme observado na Figura 3, estes valores de distância entre roteadores permitem que a modulação utilizada em um enlace possa variar da $m54$ até a $m6$ para qualquer das larguras de canal simuladas. O objetivo do cenário em grade é simular a disposição de roteadores de uma rede em malha de um campus e a variação de posicionamento tem o intuito de simular o posicionamento de nós devido a existência de obstáculos.

Utilizou-se $ED = 60\text{MHz}$ e parâmetro $E_{MAX} = 20\text{MHz}$ que estabelece o máximo valor de espectro que um enlace físico pode ocupar. Cada roteador foi equipado com 4 rádios de comunicação e 1 rádio adicional utilizado para executar medidas e transmitir sondas nos canais de diferentes larguras. Durante as simulações foram admitidas $k = \{1, 3, 5, 7, 9\}$ demandas que geravam mensagens de tamanho 1000 $bytes$ e que foram estabelecidas entre pares de roteadores distintos. As simulações possuíram duração de 220s, cada nova demanda foi admitida a cada 12s. Desta maneira, na configuração onde foram simuladas 9 demandas, a última foi admitida no tempo $12\text{s} \times 9 = 108\text{s}$. A cada nova demanda admitida executou-se o algoritmo de Dijkstra, para determinar a nova rota. Nos tempos de simulação igual a 120s e 220s iniciou-se e terminou-se, respectivamente, a transmissão de dados e, em seguida, executaram-se as medidas de desempenho de cada métrica. Utilizou-se fonte CBR com taxa de geração de mensagens igual a menor taxa

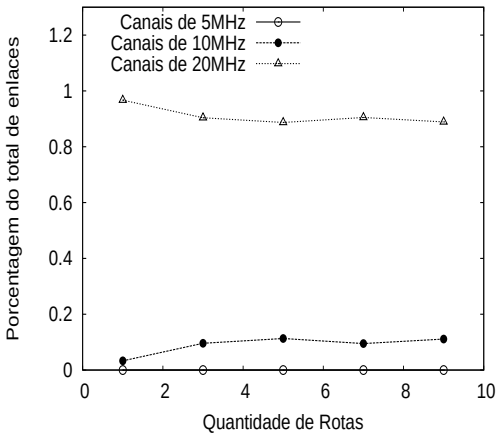
de transmissão dos enlaces da rota. Tal informação de taxa foi obtida na camada MAC e repassada periodicamente para a camada de aplicação. Executaram-se 30 rodadas de simulação e calculou-se a média dos resultados com intervalo de confiança de 95%.

Nas Figuras de 5(a) a 8(a), são apresentados os resultados de capacidade para cada métrica avaliada, em função da quantidade de demandas/rotas k admitidas na rede. Conforme observado nestas figuras, executaram-se simulações onde cada métrica escolheu dentre os canais de largura 5, 10 e 20MHz. Além destas, executaram-se simulações onde existia somente canais na largura de 5 ou 10 ou 20MHz. Nas Figuras de 5(b) a 8(b) são apresentados, para as simulações onde coexistem canais de 5, 10 e 20MHz, os valores de percentual do total de enlaces estabelecidos em cada largura de canal, em função da quantidade de demandas admitidas.

Na Figura 5(a), observa-se que os maiores valores de capacidade para a métrica WCETT são obtidos ao selecionar entre canais de 5, 10 ou 20MHz. Nesta situação e conforme pode ser observado na Figura 5(b), os enlaces estabelecidos utilizam a largura de 20 ou de 10MHz. Isto porque os valores da métrica WCETT determinam a seleção de enlaces que com menor tempo de transmissão fim-a-fim na rota e enlaces que utilizem canais com menor tempo de ocupação.



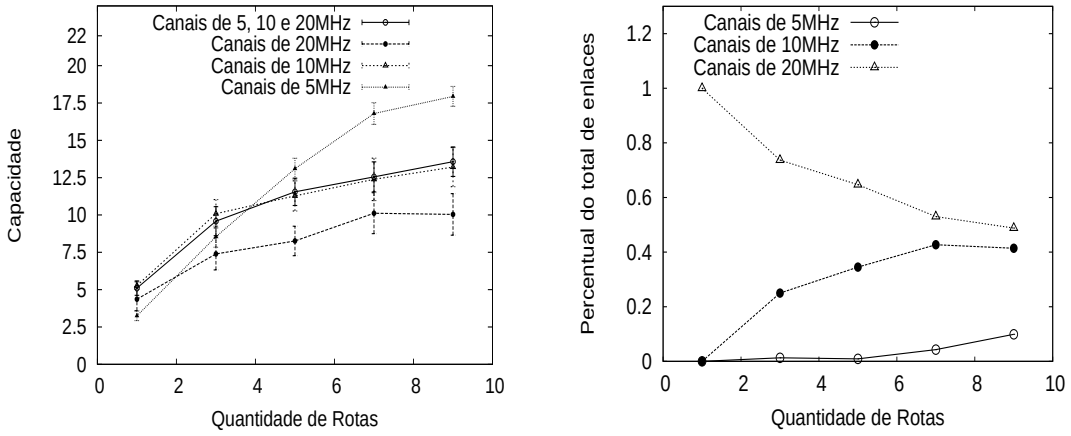
(a) Capacidade das Rotas.



(b) Percentual de enlaces em cada largura de canal.

Figura 5. Resultados da métrica WCETT

Na Figura 6(b), observa-se que são escolhidos enlaces nas larguras de 10 e 20MHz quando utiliza-se a métrica EETT para selecionar a largura de canal. Isto porque os valores da métrica determinam a seleção de enlaces em canais com menor tempo de transmissão, como é o caso dos enlaces que utilizam canais de 10 e 20MHz. Com esta escolha de canais, visualiza-se na Figura 6(a) que são obtidos valores de capacidade próximos da capacidade ao utilizar somente a largura de canal de 10MHz. Na Figura 6(a), nota-se que a partir de $k = 5$ há um aumento da disputa dos enlaces pelos canais e, assim, a maior capacidade é obtida ao dividir o ED em uma maior quantidade de canais de 5MHz. A métrica $EETT$ não é capaz de contabilizar a interferência entre canais com espectro parcialmente sobreposto e, sendo assim, os valores da métrica não contabilizam que com o aumento da contenção é preferível utilizar canais de 5MHz.

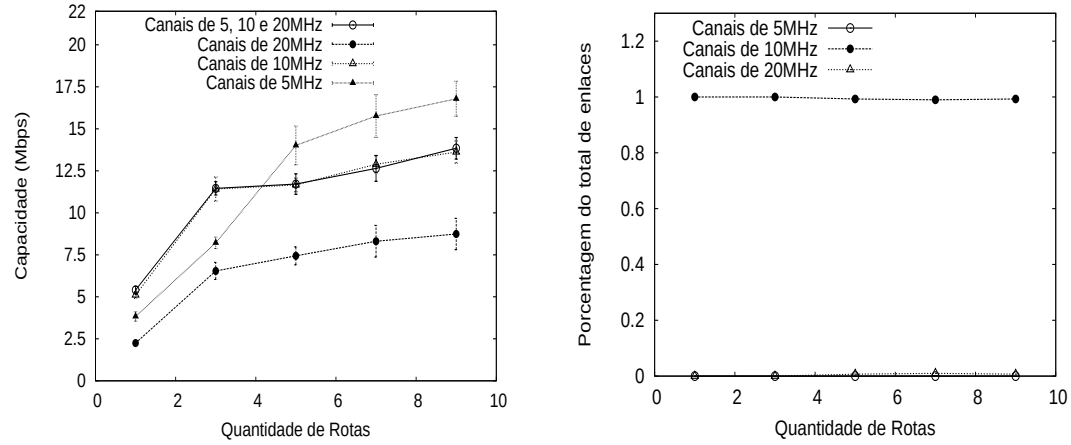


(a) Capacidade das Rotas.

(b) Percentual de enlaces em cada largura de canal.

Figura 6. Resultados da métrica EETT

Na Figura 7(a), nota-se que ao utilizar os valores da métrica B-MTM para selecionar dentre canais de largura de 5, 10 e 20MHz, obtém-se valores de capacidade similares aos obtidos somente com canais de largura 10MHz. Isto porque para todas as quantidades de rotas simuladas, a métrica gera valores que determinam a escolha de canais de largura 10MHz, uma vez que estes canais s ao os que oferecem maior capacidade para um enlace que emprega dois rádios de saída. A utilização de canais de 10MHz pode ser constatada na Figura 7(b). Percebe-se, entretanto na Figura 7(a), que com o aumento da quantidade k de rotas e, consequente aumento da disputa pelos canais, que a largura de 5MHz é a que oferece maior capacidade ao utilizar a métrica.



(a) Capacidade das Rotas.

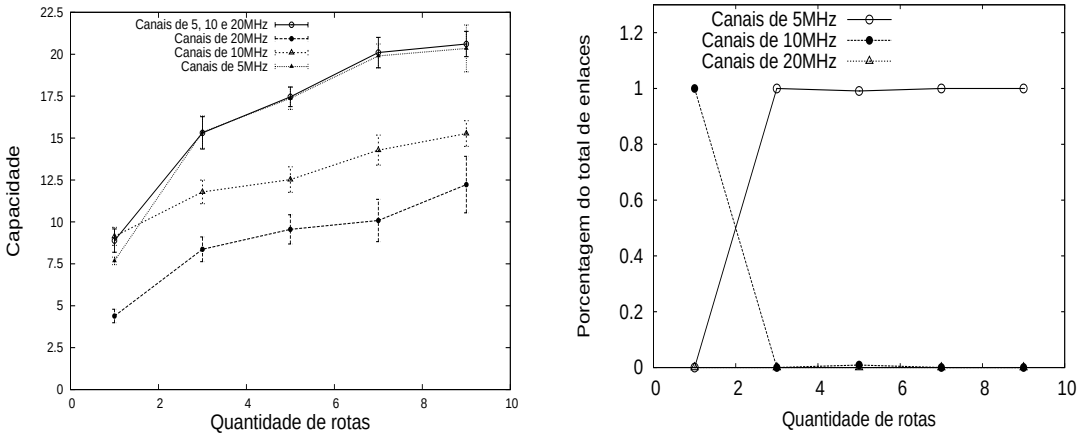
(b) Percentual de enlaces em cada largura de canal.

Figura 7. Resultados da métrica B-MTM

Nota-se na Figura 8(a) que os maiores valores de capacidade são obtidos ao utilizar os canais de largura 5MHz. Esta largura de canal oferece uma maior quantidade de canais ortogonais, dentro dos quais a utilização dos valores da métrica tende a distribuir

os enlaces igualitariamente. Ao utilizar os valores da métrica para selecionar entre as larguras de 5, 10 e 20MHz, percebe-se que há um aumento na capacidade da rede. Na Figura 8(b) percebe-se que a métrica utiliza as larguras de 5 e 10MHz para disponibilizar os maiores valores de capacidade.

Para exemplificar os ganhos de capacidade obtidos através do uso da métrica MCWMR-BEETT, compara-se, com as demais métricas, os valores de capacidade obtidos quando $k = 9$. Nesta condição, a maior capacidade ocorre quando as métricas WCETT e MCWMR-BEETT selecionam dentre as larguras de 5, 10 ou 20MHz e, quando as métricas EETT e B-MTM utilizam somente a largura de canal de 5MHz. Na situação comentada, os valores de capacidade obtidos através do uso das métricas WCETT, MCWMR-BEETT, EETT e B-MTM possuem, respectivamente, os valores 13.9, 20.6, 17.9 e 16.7$\mathrm{M}bits/\mathrm{s}$. Neste caso, a métrica MCWMR-BEETT oferece ganho de mais de 15% quando comparada com a EETT que é a métrica que oferece o segundo maior valor de capacidade. Comparando as mesmas métricas EETT e MCWMR-BEETT com $k = 1$, e em situação similar a do exemplo anterior, onde a primeira métrica utiliza somente a largura de canal de 5MHz e a segunda métrica seleciona dentre todas as larguras de canal disponíveis, há um ganho 160% para a métrica proposta.



(a) Capacidade das Rotas.

(b) Percentual de enlaces em cada largura de canal.

Figura 8. Resultados da métrica MCWMR-BEETT

6. Conclusão

Neste artigo propôs-se e avaliou-se, através de simulações no NS-2, a métrica denominada MCWMR-BEETT, em cenários onde existiam diferentes larguras de canal. Nas avaliações comparou-se os resultados da proposta MCWMR-BEETT com diferentes métricas para WMNs. Conforme resultados obtidos, observa-se que com a utilização da métrica MCWMR-BEETT foi possível aumentar a capacidade das redes MCMR-WMNs nos cenários estudados.

Referências

Augusto, C., Carvalho, C., da Silva, M., and de Rezende, J. (2010). Impacto do roteamento no escalonamento de enlaces em redes em malha sem fio. In *XXVIII Simpósio Brasileiro de Redes de Computadore*, SBRC, pages 189–202.

Bradner, S. (1991). *Benchmarking Terminology for Network Interconnection Devices*. IETF.

Calvo, R. A. and Campo, J. P. (2007). Adding multiple interface support in ns-2. Report, University of Cantabria, Spain. http://personales.unican.es/aguerocr/files/ucMultiIfacesSupport.pdf - último acesso em 21/09/2011.

Carvalho, C. B. and de Rezende, J. F. (2010). Roteamento em redes em malha sem fio ieee 802.11 com adaptação de largura de canal. In *XXVIII Simpósio Brasileiro de Redes de Computadores (SBRC 2010)*, pages 161–174.

Chandra, R., Mahajan, R., Moscibroda, T., Raghavendra, R., and Bahl, P. (2008). A case for adapting channel width in wireless networks. volume 38, pages 135–146, New York, NY, USA. ACM.

Chimento, P. and Ishac, J. (2008). *Defining Network Capacity*. IETF.

De Couto, D. S. J., Aguayo, D., Bicket, J., and Morris, R. (2005). A high-throughput path metric for multi-hop wireless routing. *Wireless. Networks*, pages 419–434.

Draves, R., Padhye, J., and Zill, B. (2004). Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, MobiCom, pages 114–128.

EPFL-IC. No ad-hoc routing agent (noah). `http://icapeople.epfl.ch/widmer/uwb/ns-2/noah`. - último acesso em 30/01/2012.

IEEE (2007). Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard 802.11.

Jiang, W., Liu, S., Zhu, Y., and Zhang, Z. (2007). Optimizing routing metrics for large-scale multi-radio mesh networks. In *International Conference on Wireless Communications, Networking and Mobile Computing - WiCom*, pages 1550 –1553.

Li, L. and Zhang, C. (2009). Optimal channel width adaptation, logical topology design, and routing in wireless mesh networks. volume 2009. Hindawi Corporation.

Rappaport, T. (2001). *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Telecomunicazioni, D. G. (2007). dei80211mr library for the network simulator 2. http://telecom.dei.unipd.it/pages/read/58/ - último acesso em 19/12/2011.

Yuan, Y., Bahl, P., Chandra, R., Chou, P. A., Ferrell, J. I., Moscibroda, T., Narlanka, S., and Wu, Y. (2007). Knows: Kognitiv networking over white spaces. In *Proceedings of IEEE DySPAN 2007*.

Zhai, H. and Fang, Y. (2006). Impact of routing metrics on path capacity in multirate and multihop wireless ad hoc networks. In *Proceedings of the IEEE International Conference on Network Protocols*, pages 86–95.

Uma Avaliação do Uso do Canal de Controle Pelo IEEE 802.11 Em Ambiente de Múltiplos Canais

Marcos Fagundes Caetano, Bruno Figueira Lourenço, Jacir Luiz Bordim

[1]Departamento de Ciência da Computação – Universidade de Brasília (UnB)
Caixa Postal 4466 – 70.910-900 – Brasília – DF – Brazil

{caetano, brunofigueira, bordim}@cic.unb.br

Abstract. *Wireless networking enhanced with multiple transmitting channels has been considered to improve the utilization of the eletromagnetic spectrum and to provide ways to accommodation the growing demand for connectivity. In this scenario, the use of a common control channel to coordinate the stations and networking resources is usually employed. This work evaluates the control channel capacity and its impact on the overall system throughput when multiple channels are available. To this end, an analytical model is proposed to evaluate the saturation conditions of the control channel. Empirical results are also provided and compared with the analytical model. The results show that the empirical results are consistent with the analytical model. In particular, it is shown that the ratio of eight data channels per control channel attains the best results.*

Resumo. *O uso de múltiplos canais para a comunicação sem fio tem sido objeto de pesquisa recente. Redes de rádio que utilizam múltiplos canais, como exemplo Rádios Cognitivos, buscam o melhor aproveitamento do uso do Espectro Eletro Magnético e a acomodação da crescente demanda por conectividade. Neste cenário, diversos mecanismos de acesso ao meio utilizam um canal exclusivo, conhecido como Canal de Controle, para a coordenação das estações e dos recursos da rede. Neste trabalho, realizamos a avaliação do impacto do uso do Canal de Controle quando utilizado em um mecanismo de acesso a múltiplos canais baseado no padrão IEEE 802.11. Apresentamos um modelo analítico para a avaliação do mecanismo proposto, bem como a sua implementação em ambiente de simulação. Os resultados apresentados demonstram que o modelo simulado encontra-se condizente com os resultados analíticos. Dentre os resultados obtidos, a relação de oito canais de dados para um canal de controle apresenta a menor taxa de desperdício de recursos, o que permite a construção de mecanismos de acesso ao meio mais eficientes e menos onerosos.*

1. Introdução

Atualmente, o IEEE 802.11 [Committee 1999] é o padrão de redes WLAN largamente utilizado. O aumento no uso deste tipo de tecnologia tem impactado diretamente no nível de interferência e de saturação do meio de comunicação. Este problema impulsionou o surgimento de diversas áreas de pesquisa, bem como o desenvolvimento de novas técnicas. Dentre as soluções propostas para a minimização do problema de interferência e de saturação do canal de comunicação, destacam-se as que utilizam múltiplos canais de dados para a comunicação [Domenico et al. 2012].

O uso de múltiplos canais de dados vem ganhando força com o surgimento de redes de rádios cognitivos [Zhao and Morales-Tirado 2012]. Neste contexto, as estações buscam mapear as oportunidades (canais ociosos) de acordo com as necessidades de comunicação das estações da rede, de forma a maximizar o uso dos canais de comunicação. Conforme exposto em [Domenico et al. 2012], a coordenação das estações e dos canais de dados pode ser feita com o uso exclusivo de um canal, denominado canal de controle. O uso do canal de controle permite as estações trocarem informações sobre o uso dos canais de dados, facilitando assim, a sua gerência.

Diversos trabalhos utilizam o conceito do canal de controle como forma de gerenciamento da comunicação nos canais de dados. Dos trabalhos que utilizam múltiplos canais para comunicação, destacam-se os que são baseados no padrão IEEE 802.11. Li *et al.* [Li et al. 2003] e Zhao *et al.* [Zhao et al. 2003] associam aos quadros RTS/CTS a informação de qual canal de dados será utilizado na comunicação. So *et al.* [So and Vaidya 2004] focam no problema de consumo de energia, evitando que as estações troquem de canais constantemente. Cordeiro *et al.* [Cordeiro et al. 2006] propõem uma forma de decremento diferenciado do contador de *backoff* como forma de minimizar as colisões. Choi *et al.* [Choi et al. 2003] propõem a alocação dos canais de dados por um período variável de tempo, limitando, ao canal de controle, a transmissão de quadros de confirmação (Ack).

A comunicação utilizando múltiplos canais acarreta em perda da informação transmitida no canal de controle durante os períodos de transmissão de dados. Para tentar resolver este problema, alguns autores propõem a utilização de múltiplos rádios para comunicação e o monitoramento simultâneo [Nasipuri et al. 1999], [Hung et al. 2002] e [Xu et al. 2005]. Um outra abordagem é a utilização de apenas um rádio, resolvendo este problema no próprio mecanismo de acesso ao meio. Uma abordagem comum é forçar cada estação a recolher informações suficientes antes de tentar comunicar [So and Vaidya 2004] e [Hung et al. 2002]. Outros autores, propõem a modificação do esquema de modulação dos rádios. Kwon *et al.* [Kwon et al. 2009] propõem o uso de OFDMA (*Orthogonal Frequency Division Multiple Access*) como forma de permitir o monitoramento e a transmissão em paralelo. Joa-Ng *et al.* [Joa-Ng and Lu 1999] propõem a utilização do mecanismo de modulação CDMA (*Code Division Provides Multiple Access*), associando os códigos de espalhamento a representação de canais de dados.

Dos trabalhos relacionados acima, nenhum considera as limitações físicas e lógicas envolvendo a utilização do canal de controle no processo de gerenciamento dos canais de dados. Todos assumem a utilização deste tipo de abordagem sem se preocupar com o impacto gerado na vazão total do sistema, no atraso médio, na quantidade de colisões e no desperdício de recursos. Este trabalho apresenta uma avaliação do uso do canal de controle em um ambiente de múltiplos canais. Até onde é de conhecimento destes autores, este é o primeiro trabalho feito na área. Para isso, utilizou-se como base o mecanismo de acesso ao meio CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) definido pelo padrão IEEE 802.11 [Committee 1999]. O mecanismo CSMA/CA foi modificado para o contexto de múltiplos canais, seguindo o conjunto interseção das funcionalidades comuns identificas nos trabalhos correlatos apresentados. O modelo matemático para um único canal proposto por Bianchi [Bianchi 2000] é estendido para múltiplos canais e utilizado na análise deste trabalho. Por fim, um simulador de

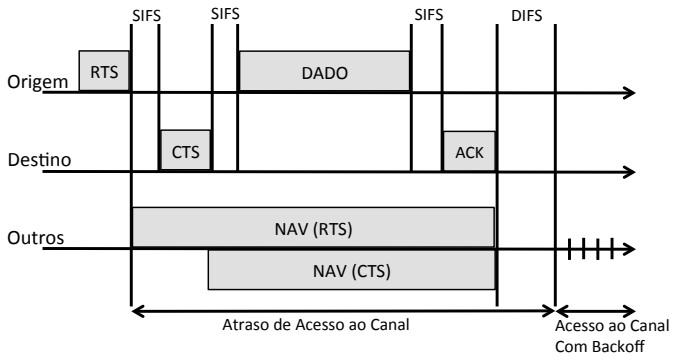


Figura 1. Funcionamento do Padrão IEEE 802.11.

eventos discretos foi implementado e validado com o modelo analítico proposto. O ambiente de simulação foi utilizado para a avaliação aprofundada dos aspectos associados ao mecanismo proposto. Resultados preliminares demonstram que o canal de controle atinge o seu limite de saturação com 8 canais de dados. Outro ponto observado é que o problema conhecido como *Problema do Terminal Escondido Em Ambiente de Múltiplos Canais* [So and Vaidya 2004] impacta diretamente na taxa de ociosidade do canal de controle, o que diminui a vazão final do sistema, mesmo em situações em que existe recursos disponíveis. Este problema é minimizado com o aumento da quantidade de nós na rede. Todavia, o aumento no número de estações implica no aumento do atraso médio do sistema, o que pode tornar proibitiva a sua utilização por aplicações críticas. Para contornar este problema, propomos um modelo de inferência de canais ociosos como forma de diminuir a taxa de ociosidade sem aumentar o atraso médio do sistema.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 será apresentado o funcionamento do protocolo IEEE 802.11 DCF. O protocolo IEEE 802.11 DCF aplicado ao contexto de múltiplos canais será discutido pela Seção 3. O modelo analítico deste protocolo é abordado na Seção 4. Informações sobre o ambiente de simulação e a apresentação de resultados serão abordados pela Seção 5. Por fim, as conclusões e trabalhos futuros serão abordados pela Seção 6.

2. IEEE 802.11 DCF

Nesta seção, apresentaremos uma breve introdução sobre o funcionamento no modo DCF (*Distributed Coordination Function*) padronizado pelo IEEE 802.11. Maiores informações podem ser obtidas em [Committee 1999].

No modo distribuído de operação, toda estação que deseja transmitir dedica-se ao monitoramento do canal para identificar o seu estado corrente. Se o estado do canal estiver ocioso, por um período de tempo igual a DIFS (*Distributed Interframe Space*) tempo, a estação inicia o processo de transmissão. Caso durante este período a estação verifique que o canal encontra-se ocupado, a abordagem adotada será aguardar por um intervalo de tempo pseudoaleatório antes de tentar utilizar novamente o canal. Desta forma, a estação tenta minimizar a probabilidade de colisão. Este procedimento é conhecido como CA (*Collision Avoidance*) e faz parte do mecanismo de acesso ao meio CSMA/CA (*Carrier Sense Multiple Access*) implementado pelo IEEE 802.11 DCF Mode.

O valor que representa o intervalo de tempo em que uma estação não tentará trans-

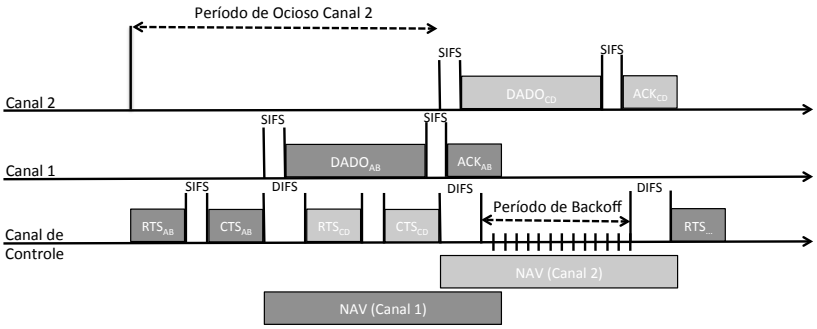


Figura 2. Padrão IEEE 802.11 aplicado ao cenário de Múltiplos Canais.

mitir no canal é conhecido como *valor de backoff* e é obtido pela execução do algoritmo exponencial de *backoff*. Após o canal ficar ocioso por DIFS tempo, a estação começa a decrementar o seu *valor de backoff*, interrompendo este processo toda a vez em que o canal ficar ocupado e retomando o seu decremento após o canal ficar DIFS tempo ocioso. O decremento ocorre na unidade de *unidades de tempo* e é representada por σ.

Com objetivo de realizar a alocação do espaço aéreo e diminuir a quantidade de colisões, o padrão IEEE 802.11 prevê o uso de quadros de controle conhecidos como RTS (*Request to Send*) e CTS (*Clear to Send*). A Figura 1 apresenta a representação do funcionamento do protocolo. Após verificar que o canal ficou ocioso por DIFS tempo, a estação origem envia o quadro RTS notificando a estação de destino que deseja estabelecer comunicação. Após SIFS (*Short Interframe Space*) tempo, a estação de destino irá responder com um quadro CTS. A partir deste momento, as demais estações que escutaram um dos dois quadros irão ficar em silêncio durante todo o período em que a comunicação irá ocorrer, para isso, elas atualizam o valor do seu NAV (*Network Allocation Vector*). Após receber o quadro CTS, a estação de origem irá aguardar SIFS tempo para iniciar a transmissão do quadro que contém os dados. Ao receber o quadro com os dados, a estação de destino irá aguardar SIFS tempo para iniciar o envio do quadro ACK (*Acknowledge*), confirmando assim o recebimento correto do quadro de dados. A partir deste ponto, o processo é reiniciado e as estações passam a ter o mesmo comportamento descrito. Colisões são detectadas quando a sequência de eventos descrita não ocorre. A estação não recebe um CTS, por exemplo, após o envio de um RTS. Neste caso, todas as estações envolvidas irão executar o algoritmo exponencial de *backoff*.

3. IEEE 802.11 DCF no Contexto de Múltiplos Canais

A seção anterior apresentou uma breve introdução sobre o mecanismo IEEE 802.11 em modo de operação distribuída. Nesta seção será apresentado o nosso mecanismo para múltiplos canais baseado nas características comuns dos trabalhos correlacionados. Este mecanismo foi utilizado como base para a proposta do modelo analítico para múltiplos canais a ser discutidos na próxima seção. As características comuns dos trabalhos apresentados estão realcionadas abaixo:

- uso de um canal exclusivo para a coordenação das estações;
- modificação dos quadros RTS/CTS para incluir o canal a ser utilizado;
- mapeamento do uso dos canais de dados através dos quatros RTS/CTS;
- perda do estado dos canais durante os períodos de comunicação;

- transmissão do quadro de dados e o de confirmação (ACK) no canal de dados;

As características relacionadas foram reunidas no mecanismo de acesso a múltiplos canais, cujo funcionamento encontra-se representado pela Figura 2. Todas as estações por padrão encontram-se no canal de controle, ausentando-se somente durante o período de transmissão dos dados. Os quadros RTS e CTS foram modificados para incluírem a informação do canal de dados escolhido pela estação transmissora. Desta forma, as demais estações tomam conhecimento do estado dos canais de dados a medida em que quadros RTS e CTS são transmitidos no canal de controle. Somente as estações que conhecem um ou mais canais de dados ociosos é que podem comunicar.

Diferentemente do mecanismo de acesso ao meio apresentado na Seção 2, o processo de alocação do espaço aéreo define em qual canal de dados as informações serão transmitidas. No exemplo apresentado pela Figura 2, as estações A e B definem que a transmissão será realizada no canal de dados 1. Neste momento, as demais estações atualizarão os seus respectivos NAVs para que este canal não seja utilizado durante o período da transmissão anunciada. Todavia, como existe a disponibilidade de um segundo canal de dados, as estações C e D irão solicitar o seu uso. É importante ressaltar que durante o processo de alocação do canal de dados 2, as estações A e B não tomaram conhecimento deste processo, pois ambas encontravam-se no canal de dados 1. Sendo assim, quando uma estação retorna ao canal de controle após transmitir, o único canal que ela pode considerar livre é aquele que ela acabou de liberar. Caso não consiga alocar este canal, a estação permanecerá no canal de controle por um tempo suficientemente grande para tomar conhecimento de outros canais livres.

Na próxima Seção iremos apresentar o modelo teórico definido para avaliação do IEEE 802.11 em ambiente de múltiplos canais.

4. Modelo Teórico

O modelo teórico a ser apresentado é baseado no trabalho proposto por Bianchi [Bianchi 2000], o qual é aplicado ao contexto de um único canal. Nesta seção, apresentamos o nosso modelo analítico para o mecanismo IEEE 802.11 em modo distribuído para multiplos canais.

Preliminares

Seja n o número de estações de uma rede de um salto, k a quantidade de canais de dados e c o canal de controle. Onde k e c apresentam a mesma taxa de transmissão T_{rx}, $n >> k$ e c é dividido em *unidades de tempo*, definida por u. Cada unidade de tempo u possui um tamanho variável, que está associado a uma certa probabilidade de ocorrer. Neste trabalho, assume-se que cada estação sempre tem informação para transmitir. Os valores possíveis de ocorrer estão relacionados aos seguintes eventos do canal de controle:

- *Canal Ocioso* ($u = \sigma$)
- *Transmissão bem Sucedida* ($u = T_s$)
- *Transmissão com Colisão* ($u = T_c$)

Os valores que u pode assumir são constantes e são definidos por:

Tabela 1. Parâmetros publicados em [Committee 1999] e utilizados para a obtenção dos resultados numéricos e de simulação.

Descrição	Valor
Carga útil do pacote (FRAME)	8184 bits
Atraso de propagação (δ)	1 μs
unidade de tempo ociosa (σ)	50 μs
Taxa de Bits dos Canais (T_{rx})	1 Mbps
DIFS	128 μs
SIFS	28 μs
RTS	288 μs
CTS, ACK	240 μs
W	16
m	6

$$\begin{cases} T_s = DIFS + RTS + SIFS + CTS + (2 * \delta) \\ T_c = DIFS + RTS + \delta \\ T_{DATA} = SIFS + \frac{FRAME}{T_{rx}} + SIFS + ACK + (2 * \delta) \end{cases}$$

T_{DATA} é o intervalo de tempo suficientemente grande para acomodar a transmissão de informação útil em um canal de dados. A Tabela 1 apresenta os valores das constantes apresentadas acima. No canal de controle, os quadros colidem de forma constante e independente com probabilidade p. O comportamento do algoritmo exponencial de *backoff* é modelado por τ, que representa o "desejo" de uma estação querer transmitir em dado momento [Bianchi 2000]. Seja W o tamanho da janela, $W = CW_{min}$, m o "estágio máximo de *backoff*", tal que o tamanho máximo da janela seja $CW_{max} = 2^m \times W$, o valor de τ pode ser obtido por:

$$\tau = \frac{2(1-2p)}{(1-2p)(W+1) + pW(1-(2p)^m)}. \quad (1)$$

A probabilidade p está associada o fato de, em dado momento, duas ou mais estações transmitirem. A premissa assumida de que os quadros colidem de forma constante e independente com probabilidade p, implica que cada transmissão "vê" o sistema em um mesmo estado. Quando o sistema estabiliza-se, cada estação passa a transmitir um quadro com probabilidade τ [Bianchi 2000], o que resulta na definição:

$$p = 1 - (1-\tau)^{n-1}. \quad (2)$$

A probabilidade de pelo menos uma transmissão ocorrer, em um dado momento, é definida por:

$$P_{tr} = 1 - (1-\tau)^n. \quad (3)$$

A probabilidade de uma transmissão ser bem sucedida é dada pela probabilidade de exatamente uma estação e definida por:

$$P_s = \frac{n\tau(1-\tau)^{n-1}}{P_{tr}}. \quad (4)$$

Estimativa da Vazão Máxima em Múltiplos Canais

O objetivo do modelo a ser apresentado é calcular a vazão máxima dos canais de dados. Nesse sentido, faz-se necessário calcular a taxa em que cada estação chega ao canal de dados. Para simplificação da análise, no limite, assume-se que o canal de controle encontra-se em um dos seguintes estados:

- **Estado 1:** Todos os k canais de dados estão ocupados;
- **Estado 2:** Existe exatamente um canal de dados ocioso e $n - k + 1$ estações no canal de controle disputando o seu acesso.

Como $T_{DATA} > T_s$, durante o período de transmissão em um canal de dados, o canal de controle é utilizado para alocação dos canais de dados disponíveis. Neste caso, em condições adequadas, o sistema irá alcançar o *Estado 1*. Quando todos os canais de dados estiverem ocupados, o canal de controle ficará ocioso. Neste caso, um canal de dados será liberado a cada $\frac{T_{DATA}}{k}\mu s$. Após a liberação de um canal de dados, o sistema alcança o *Estado 2*. Neste ponto, se o canal de dados for negociado em um espaço de tempo menor que $\frac{T_{DATA}}{k}\mu s$, o sistema atingirá novamente o *Estado 1*. Caso esta condição não seja respeitada, um segundo canal de dados será liberado e mais uma estação chegará ao canal de controle. A restrição $n >> k$ faz com que a condição $\frac{T_{DATA}}{k}\mu s$ ocorra um número maior de vezes. Sendo assim, para simplificação do modelo, assume-se que a condição é sempre respeitada. Neste caso, no limite, existe um canal de dados ocioso, $n - k + 1$ estações disponíveis e, em um dado momento, um dos três eventos abaixo podem ocorrer no canal de controle:

- **Evento 1:** Nenhuma estação deseja transmitir. Este estado pode acontecer quando todas as estações encontram-se decrementando os seus contadores de *backoff*. A probabilidade deste evento ocorrer é representada por $(1 - P_{tr})$ e o tamanho da unidade de tempo u é $\sigma(1 - P_{tr})$;
- **Evento 2:** Intervalo de tempo gasto no processo de alocação do canal de dados. Conforme a premissa definida, um canal de dados será liberado em um tempo inferior ou igual a $\frac{T_{DATA}}{k}$. Neste caso, o tempo compreendido entre a negociação do canal de dados e outra estação chegar ao canal de controle não será maior que $\max(\frac{T_{DATA}}{k}, T_s)$. A probabilidade deste evento ocorrer é representada $P_{tr}P_s$ e o tamanho da unidade de tempo u é $P_{tr}P_s \max\left(\frac{T_{DATA}}{k}, T_s\right)$;
- **Evento 3:** Duas ou mais estações desejam transmitir. Neste caso, haverá colisão. A probabilidade deste evento ocorrer é representada pela probabilidade de ocorrer um transmissão e ela não ser bem sucedida, $P_{tr}(1 - P_s)$. O tamanho da unidade de tempo u é $P_{tr}(1 - P_s)T_c$.

Com base nos três possíveis eventos definidos para o canal de controle, é possível calcular o tamanho médio de uma unidade de tempo escolhida ao acaso.

$$S = \sigma(1 - P_{tr}) + P_{tr}P_s \max\left(\frac{T_{DATA}}{k}, T_s\right) + P_{tr}(1 - P_s)T_c. \quad (5)$$

De acordo com a análise apresentada, o único evento em que existe uma estação deixando o canal de controle é o *Evento 2*. Desta forma, podemos representar a taxa em que uma estação chega a um canal de dados, pela equação:

$$\lambda = P_{tr} P_s. \tag{6}$$

Fazendo uma analogia com teoria de filas, temos que o "tempo de serviço" de um canal de dados pode ser representado por $\frac{T_{DATA}}{S}$. Ou seja, gasta-se T_{DATA}/S para um canal transmitir apenas um pacote de dados. Neste caso, o tráfego oferecido é representado por $\frac{\lambda T_{DATA}}{S}$ e a taxa de ocupação dos canais de dados é calculada pela Equação:

$$O = \frac{\lambda T_{DATA}}{kS}. \tag{7}$$

5. Ambiente de Simulação e Resultados

Nesta seção iremos apresentar os resultados do modelo analítico discutido na Seção 4, bem como questões relacionadas ao comportamento do método de acesso a múltiplos canais proposto na Seção 3. Por fim, discutiremos os resultados obtidos com o simulador implementado.

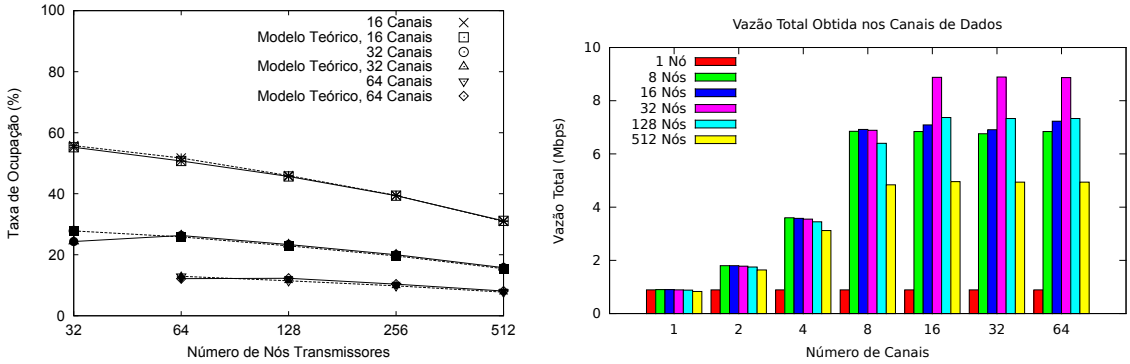
Com objetivo de realizar uma avaliação aprofundada do mecanismo de acesso proposto, um simulador de eventos discreto foi implementado na linguagem de programação C++. A validação do simulador implementado foi realizada mediante a comparação dos resultados empíricos, provenientes do simulador, com o resultados obtidos a partir do modelo analítico apresentado.

Os parâmetros de simulação utilizados encontram-se publicados na Tabela 1. O número de estações transmissoras foram variadas de tal forma que n assumiu os seguintes valores *1, 2, 4, 8, 16, 32, 64, 128, 256 e 512*. É importante ressaltar que por motivos de simplificação, cada estação transmissora possui a sua estação receptora correspondente a qual não faz parte do conjunto de estações transmissoras. A quantidade de canais de dados foi variada de tal forma que k assumiu os seguintes valores *1, 2, 4, 8, 16, 32 e 64*. Cada valor obtido é o resultado de uma média de 10 simulações, cujo intervalo de confiança é de 95% com uma variação média de menos de 1%.

5.1. Taxa de Ocupação

O gráfico apresentado na Figura 3(a) representa a taxa média de ocupação dos canais de dados (Equação 7) a medida em que o número de estações da rede cresce. Conforme discutido na Seção 4, estamos considerado o cenário em que exite $n - k + 1$ estações transmissoras no canal de controle, sendo $n \geq k$. Caso contrário, não seria possível obter a vazão máxima do sistema.

Observando a Figura 3(a), verifica-se que o modelo analítico proposto encontra-se condizente com os resultados obtidos pelo simulador implementado. É possível verificar que o aumento na quantidade de estações transmissoras e de canais de dados têm um impacto negativo na taxa de ocupação dos canais de dados e consequentemente na vazão total do sistema. Este comportamento está diretamente relacionado com a capacidade física do canal de controle e melhor discutido nas próximas subseções.



(a) Taxa de Ocupação (b) Vazão Total

Figura 3. Avaliação dos canais de dados quando há variação na quantidade de estações que desejam transmitir.

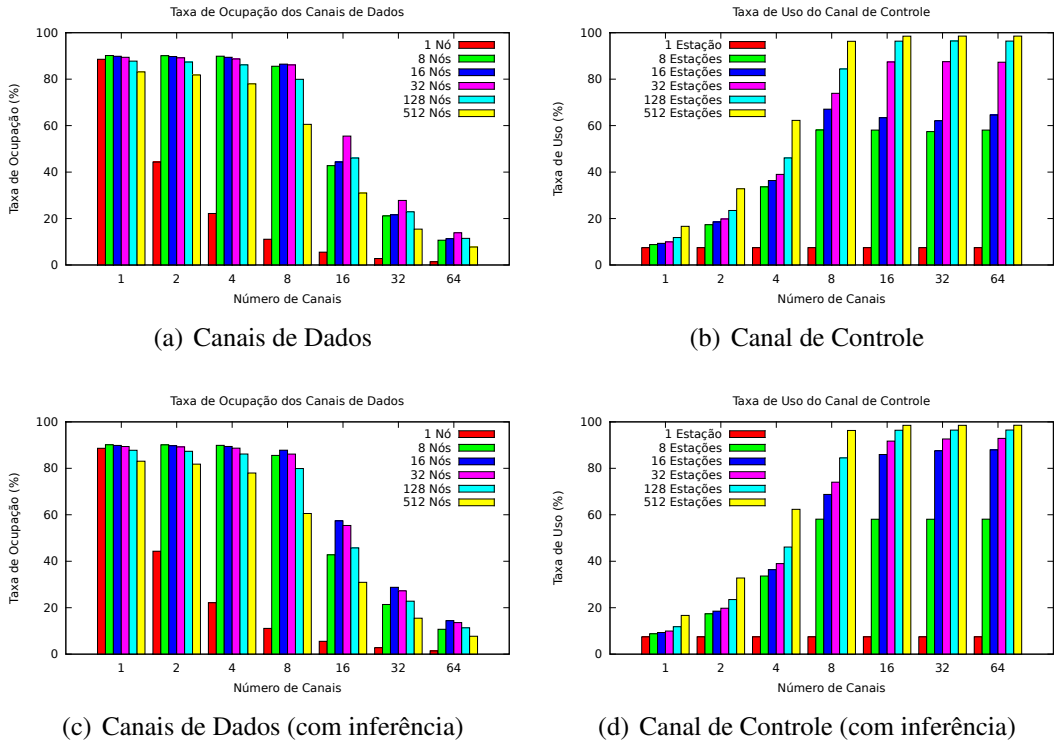
5.2. Avaliação Capacidade do Canal de Controle

Conforme discutido, diversos trabalhos utilizam o canal de controle como meio para a coordenação entre os recursos disponíveis e as estações que desejam comunicar. Nesta seção é apresentado uma investigação quanto o uso deste recurso no processo de comunicação entre as estações da rede.

A Figura 3(b) apresenta a vazão total do sistema quando a quantidade de canais de dados aumenta. Intuitivamente, acredita-se que quanto maior for a quantidade de canais de dados disponíveis, maior será a vazão total do sistema. Contudo, os resultados demonstram que este comportamento não acontece. É possível verificar que para 16 estações transmissoras, a vazão inicia com 1.8 Mbps, utilizando 2 canais, aumentando para 3.58 Mbps com 4 canais e 6.92 Mbps com 8 canais. A partir deste ponto, para qualquer quantidade de estações, mesmo dobrando-se o número de canais de dados a vazão total obtida não cresce na mesma proporção. Para qualquer quantidade de estações, chega-se um ponto em que a vazão total agregada estabiliza-se e aumentar a quantidade de recursos não significa aumentar a vazão total do sistema. Verifica-se ainda que o aumento da quantidade de estações implica na diminuição da vazão total do sistema. Para 16 canais, a vazão máxima obtida com 32 estações é 8.88 Mbps. Este valor cai para 7.37 Mbps com 128 estações e 4.94 Mbps com 512 estações.

A relação de desperdício torna-se evidente quando analisamos os resultados apresentados pela Figura 4(a). A figura apresenta a taxa de ocupação dos canais de dados quando a quantidade de canais cresce. Analisando o exemplo anterior, 16 estações transmissoras e 8 canais de dados, aumentando-se em 100% a quantidade de canais, a taxa de ocupação cai em 49%. Nesse caso, saímos de 86.77% de ocupação para 44.31%, ou seja, com 16 canais de dados, passamos a ter 55.69% de ociosidade desses recursos. Fica claro que, para este cenário, o gerenciamento da quantidade excessiva de recursos disponíveis tornou-se um problema para o mecanismo de acesso ao meio.

A Figura 4(b) apresenta a taxa de uso do canal de controle. Esta taxa corresponde a todo o tempo, durante o período de simulação, em que o canal de controle encontrou-se ocupado. Conforme pode ser observado, para 16 ou mais estações transmissoras e a partir de 8 canais de dados, a taxa de uso do canal de controle é superior a 59% de uso.



(a) Canais de Dados (b) Canal de Controle

(c) Canais de Dados (com inferência) (d) Canal de Controle (com inferência)

Figura 4. Comparação entre a Taxa de Ocupação vs a Taxa de Uso.

Cruzando as informações dos dois últimos gráficos, destaca-se o caso específico de 16 estações e 16 canais de dados, onde é observado que a taxa de ocupação do canal de dados é de 44.31% (Figura 4(a)), sendo que a taxa de uso do canal de controle é de 60% (Figura 4(b)). Mesmo existindo recursos disponíveis as estações no canal de controle não os estão utilizando.

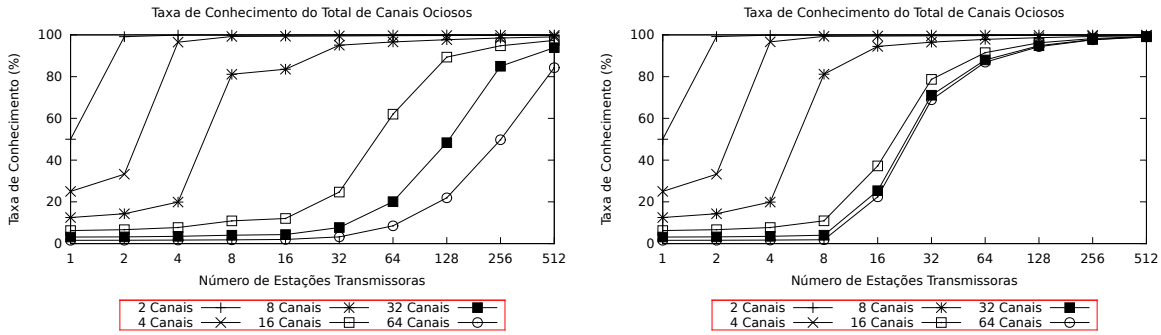
São dois os problemas identificados para esse comportamento. O primeiro está relacionado com a perda do estado dos canais de dados quando uma estação retorna ao canal de controle. Conforme já discutido, ao retornar ao canal de controle, a estação conhece somente o estado do canal que ela acabou de liberar. De acordo com o protocolo, para poder transmitir novamente, esta estação deverá aguardar no canal de controle por quadros RTS e CTS, caso não tenha conseguido retornar ao canal liberado, de forma a tomar conhecimento de quando um próximo canal de dados ficará disponível. No cenário observado, a estação retorna ao canal de controle e passa a ficar aguardando por um período muito maior que o necessário, já que existe canal de dados disponível, contudo, ela não tem este conhecimento. O segundo problema está relacionado com o algoritmo exponencial de *backoff*. Mesmo existindo canais de dados disponíveis, as estações presentes no canal de controle encontram-se decrementando os seus contadores de *backoff* e assim, o recurso é desperdiçado. Estes dois problemas serão abordados com maiores detalhes nas próximas subseções.

5.3. Perda do Estado dos Canais de Dados

O problema de perda do estado dos canais de dados também é conhecido na literatura como *problema do terminal escondido em ambiente de múltiplos canais*

Tabela 2. Resultados para 16 canais de dados e 16 estações.

Qt de Estações	Frequência	τ
5	0.15	0.076149
6	0.43	0.069677
7	0.29	0.064275
8	0.11	0.059719
9	0.02	0.055832



(a) Taxa de Conhecimento (b) Taxa de Conhecimento (com inferência)

Figura 5. Taxa de conhecimento do total de canais ociosos quando a quantidade de estações cresce.

[So and Vaidya 2004]. Este problema pode acarretar em colisões, caso as estações tenham uma política agressiva e queiram selecionar um canal para comunicar sem o conhecimento real do seu estado. Caso as estações tenham uma política conservadora, a perda do estado dos canais também irá gerar desperdícios de recursos, já que as estações aguardarão o envio de novos RTS e CTS para tomar conhecimento de quando os canais ficarão liberados, o que pode acarretar em muitos casos, em esperas desnecessárias.

Todavia, esse comportamento do mecanismo de acesso ao meio pode ser aprimorado. Quando uma determinada estação retorna ao canal de controle, os canais de dados alocados serão liberados, conforme discutido na Seção 4, a uma taxa de $\max\left(\frac{T_{DATA}}{k}, T_s\right)$. Esta informação nos permite inferir que a partir do momento em que uma estação retorna ao canal de controle, passados T_{DATA} tempo, os canais que não foram alocados via RTS/CTS, pelas demais estações, podem ser considerados ociosos e devem ser disputados pela estação. Esta abordagem é definida, por este trabalho, como sendo o *método de acesso a múltiplos canais com inferência* e permite a diminuição do tempo de espera no canal de controle, por estações que aguardam a liberação de recursos.

Nesse contexto, a Figura 5(a) apresenta o grau de conhecimento da quantidade de canais de dados ociosos que cada estação tem, no momento em que consegue selecionar um canal de dados para comunicação. O gráfico nos mostra que quanto maior for a quantidade de canais de dados, maior deverá ser a quantidade de estações para que o grau de conhecimento individual cresça. Este comportamento ocorre pois o aumento no número de estações implica no aumento do tempo médio de permanência da estação no canal de controle e, consequente, no aumento na quantidade de quadros RTS/CTS recebidos. A Figura 5(b) apresenta o mesmo gráfico, contudo, o modo de inferência do

mecanismo de acesso encontra-se habilitado. É possível verificar um aumento acentuado no grau de conhecimento dos canais de dados, principalmente quando a quantidade de canais é maior que 8. Em nosso exemplo, para 16 canais e 16 estações, saímos de um cenário de 12% (Figura 5(a)) de conhecimento para para 38% (Figura 5(b)).

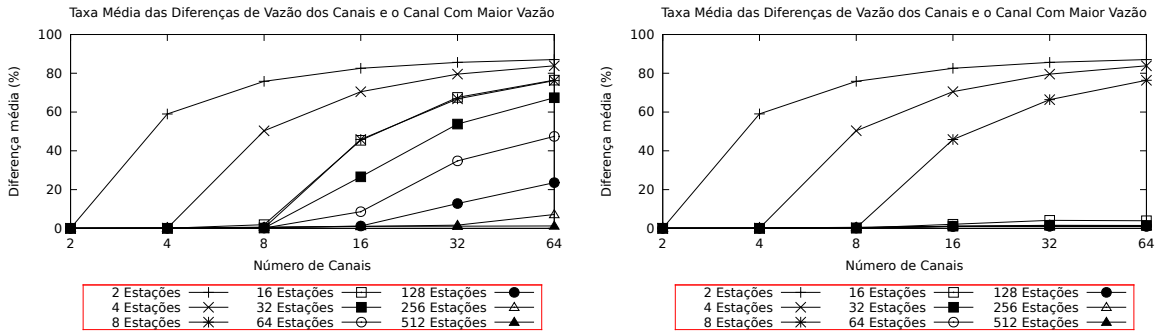
Este comportamento tem um impacto direto na taxa de ocupação dos canais de dados e no uso efetivo do canal de controle. Na Figura 4(c) apresentamos a taxa de ocupação dos canais de dados e na Figura 4(d) a taxa de uso do canal de controle quando o modo de inferência do mecanismo de acesso encontra-se habilitado. É possível verificar que houve uma melhora em alguns casos. No exemplo considerado, 16 canais e 16 estações, saímos de uma taxa de ocupação de 44.31% (Figura 4(a)) para 57.39% (Figura 4(c)). Consequentemente, a taxa de uso do canal de controle subiu de 60% (Figura 4(b)) para 86.96% (Figura 4(d)).

Retomando a discussão feita no parágrafo final da Subseção 5.2, foi possível verificar que, no caso analisado, 16 estações e 16 canais, o uso de inferência para minimizar o problema de perda do estado dos canais acarretou em uma melhora de quase 23% na taxa de ocupação dos canais de dados. Contudo, mesmo após a melhora do uso dos canais de dados, verifica-se ainda 13.04% de ociosidade do canal de controle. Conforme já explicado, este problema está relacionado com o algoritmo exponencial de *backoff*. Ou seja, as estações encontram-se em contenção decrementando os seus respectivos contadores de *backoff*. Para verificarmos esta afirmação, precisamos calcular a quantidade de tempo total referente as unidades de tempo u que tiveram tamanho σ. Para isso, a equação abaixo calcula P_{fo} que representa a probabilidade de se ter, em um dado momento, $u = \sigma$.

$$P_{fo} = \sum_{i=1}^{n} P_{cc_i}(1 - \tau_i)^i, \tag{8}$$

Onde P_{cc_i} representa a probabilidade de i estações estarem no canal de controle e τ_i representa o valor de τ calculado para a quantidade de estações igual a i. Como não há nenhum estudo na literatura que identifique a distribuição de probabilidade que modele a população de estações presentes no canal de controle, retiramos do simulador esta informação. A Tabela 2 apresenta a quantidade de estações relacionadas com a frequência em que estiveram presentes no canal de controle durante o período total de simulação. Utilizando os valores da tabela, verificamos que a probabilidade de termos uma unidade de tempo ociosa no canal de controle é de $P_{fo} = 0,641026$. Para a simulação, foram considerados no canal de controle um universo de 100.000 unidades de tempo. Sendo assim, a quantidade total de tempo em que o canal de controle ficou ocioso, é representado por $100.000 * P_{fo} * \sigma$. O Tempo total de simulação, para este cenário, foi de $24.522.519$ μs. Calculando o percentual de tempo em que o canal de controle ficou ocioso, temos: $\frac{100.000 * P_{fo} * \sigma}{24.522.519,4}$ = 13.07%.

O percentual de conhecimento da quantidade de canais ociosos disponíveis (Figura 5(a)) gera um impacto direto na taxa de ocupação individual de cada canal. Isso acontece pois ao conhecer uma fração menor do universo de canais disponíveis, cada estação tende a acessar sempre o mesmo conjunto de canais. Este comportamento leva o desbalanceamento no uso dos recursos disponíveis. As Figuras 6(a) e 6(b) apresentam a taxa média das diferenças de vazão entre os canais de dados e o canal com maior vazão.



(a) Diferença Média (b) Diferença Média (com Inferência)

Figura 6. Avaliação quanto ao uso homogêneo dos canais de dados.

O objetivo destes gráficos é demostrar o impacto no sistema do uso desbalanceado dos recursos de comunicação. Na Figura 6(a), é possível verificar que a medida em que a quantidade de recursos aumenta, aumenta também a discrepância média de vazão entre o canal com maior vazão e os demais canais de dados. Este problema é resolvido, Figura 6(b), para $k \leq n$, quando o modelo de inferência encontra-se habilitado. Para o caso $k > n$, como a quantidade de recurso é maior que a quantidade de estações, cada estação tende a sempre voltar para o canal que acabou de liberar. Isso ocorre pois a estação não permanece tempo suficiente no canal de controle para tomar conhecimento, por inferência ou RTS/CTS, da existência de outros canais ociosos.

Isso encerra essa abrangente avaliação sobre o IEEE 802.11 em múltiplos canais, bem como evidencia o impacto do uso de canal de controle no desempenho do padrão 802.11.

6. Conclusão

Neste trabalho foi apresentado uma avaliação do uso do canal de controle em ambiente de múltiplos canais. Utilizou-se como base para este trabalho o mecanismo de acesso ao meio definido pelo padrão IEEE 802.11 [Committee 1999]. As modificações propostas no padrão seguiram o conjunto interseção das funcionalidades apresentas pelos trabalhos correlados apresentados. Para análise do protocolo, o modelo matemático de Bianchi [Bianchi 2000] foi estendido. Um simulador de eventos discretos foi implementando, o que permitiu o aprofundamento nas análises do mecanismo de acesso. Resultados preliminares demonstram que o canal de controle atinge o seu limite de saturação com 8 canais de dados. Outro ponto observado é que o problema conhecido como *Problema do Terminal Escondido Em Ambiente de Múltiplos Canais* [So and Vaidya 2004] impacta diretamente na taxa de ociosidade do canal de controle, o que diminui a vazão final do sistema, mesmo havendo recursos disponíveis. Este problema foi minimizado com a proposta do modelo de inferência incorporado a proposta. Como trabalho futuro, propõem-se a investigação da distribuição de probabilidade que modela o comportamento da população de estações transmissoras no canal de controle.

Agradecimentos

Este trabalho foi parcialmente financiado pelo CNPq.

Referências

Bianchi, G. (2000). Performance analysis of the ieee 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547.

Choi, N., Seok, Y., and Choi, Y. (2003). Multi-channel mac protocol for mobile ad hoc networks. In Socity, I., editor, *IEEE 58th Vehicular Technology Conference*, volume 2, pages 1379–1382.

Committee, L. M. S. (1999). Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Technical report, IEEE Computer Society.

Cordeiro, C., Challapali, K., and Birru, D. (2006). Ieee 802.22: An introduction to the first wireless standard based on cognitive radio. *Journal of Communications*, 1(1):38–47.

Domenico, A. D., Strinati, E. C., and Benedetto, M.-G. D. (2012). A survey on mac strategies for cognitive radio networks. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 14(1):21–44.

Hung, W.-C., Law, K. E., and Leon-Garcia, A. (2002). A dynamic multi-channel mac for ad hoc lan. In *21st Biennial Symposium on Communications*, pages 31–35.

Joa-Ng, M. and Lu, I.-T. (1999). Spread spectrum medium access protocol with collision avoidance in mobile ad-hoc wireless network. In *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM*, volume 2, pages 776–783.

Kwon, H., Seo, H., Kim, S., and Lee, B. G. (2009). Generalized csma/ca protocol for ofdma systems: Protocol design, throughput analysis, and implementation issues. In *IEEE Transaction on Wireless Communications*, volume 8, pages 4176–4187.

Li, J., Haas, Z. J., Sheng, M., and Chen, Y. (2003). Performance evaluation of modified ieee 802.11 mac for multi-channel multi-hop ad hoc networks. *Journal of Interconnection Networks*, 4(3):345–359.

Nasipuri, A., Zhuang, J., and Das, S. R. (1999). A multichannel csma mac protocol for multihop wireless networks. In *IEEE Wireless Commnications and Networking Conference WCNC*, volume 3, pages 1402–1406.

So, J. and Vaidya, N. H. (2004). Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '04, pages 222–233, New York, NY, USA. ACM.

Xu, C., Liu, K., Yuan, Y., and Liu, G. (2005). A novel multi-channel based framework for wireless ieee 802.11 ad hoc networks. In *International Conference on Wireless Communications, Networking and Mobile Computing*, volume 2, pages 812–815.

Zhao, Y. and Morales-Tirado, L. (2012). Cognitive radio technology: Principles and practice. In *International Conference on Computing, Networking and Communications*, pages 650 – 654.

Zhao, Y., Xiang, Y., Xu, L., and Shi, M. (2003). A multi-channel medium access control protocol for multicast in mobile ad-hoc network. In Socity, I., editor, *The 14th IEEE 2003 International Symposium on Personal, Indoor and Mobile Radio Communication Proceedings*, volume 2, pages 1639–1643.

Atenuando o Problema de Surdez de Antenas em Comunicações Direcionais com a Utilização de Tones na Reserva de Canal

Lucas de Melo Guimarães[1], **Jacir Luiz Bordim**[1]

[1]Departamento de Ciência da Computação – Universidade de Brasília (UnB)
Campus Universitário Darcy Ribeiro
Caixa postal 4466 – 70.910-900 – Brasília – DF – Brasil

{lucasmg,bordim}@cic.unb.br

Abstract. *The use of directional antenna in support of MANET operations has been considered a promising alternative to improve space division, throughput and reduce interferences. However, the use of directional antenna has a number of challenges to be overcome such as deafness problem. This work proposes a deafness avoidance technique based on traffic flow information coupled with a promising channel reservation technique which is based on pulse and tone signals. This technique was evaluated through simulations using the EXata network simulator. When compared against other protocols and deafness avoidance mechanisms, the proposed technique was able to increase fairness up to* 76% *and individual flow throughput up to* 350%*.*

Resumo. *O uso de antenas direcionais em redes móveis ad hoc (MANETs) tem sido considerado uma alternativa para melhorar a utilização do espaço aéreo, a vazão e reduzir interferências. Apesar de prover um melhor uso do espaço aéreo, a utilização de antenas direcionais ainda possui alguns problemas, tais como o de surdez. É neste contexto que este trabalho propõe uma técnica de atenuação do problema de surdez baseada nas informações dos fluxos de tráfego aliada a uma técnica inovadora de reserva de canal baseada em sinais pulse e tone. Quando comparado com outros trabalhos, a técnica proposta obteve ganhos de até* 350% *em vazão por fluxo e* 76% *em justiça nos cenários avaliados.*

1. Introdução

Na última década, houve um grande aumento no uso das redes sem fio e no desenvolvimento de suas tecnologias. As redes sem fio são aquelas que utilizam o espectro eletromagnético para a transmissão de dados entre dispositivos fixos ou móveis. As principais frequências utilizadas pelas redes locais (LAN – *Local Area Network*) que fazem uso da tecnologia de rede sem fio fazem parte da chamada banda ISM (*Industrial, Scientific, and Medical*), isto é, uma banda livre destinada a uso não comercial nas áreas industrial, científica e médica. Dentre as tecnologias que permitiram este avanço está o padrão IEEE 802.11 (*WiFi*), que tinha como objetivo principal permitir a conectividade sem fio entre dispositivos de uma LAN. O padrão IEEE 802.11 permite ainda o funcionamento de redes *ad hoc*, ou seja, aquelas redes que não necessitam de uma elemento central preexistente

como um ponto de acesso. Padrões como o IEEE 802.11 utilizam protocolos de controle de acesso ao meio, ou seja, protocolos MAC (*Medium Access Control*) tais como o CSMA/CA (*Carrier-Sense Multiple Access with Collision Avoidance*) [IEEE 2007].

O principal objetivo de um protocolo MAC é permitir que vários nós de uma rede utilizem de maneira eficiente, um canal por eles compartilhado, conforme [Kumar et al. 2006]. Por sua vez, no que concerne a redes sem fio, a eficiência de um protocolo MAC é caracterizada como sendo proporcional a quantidade de transmissões que ocorrem com sucesso entre nós em um dado intervalo de tempo [Ramanathan 2004].

De maneira geral, os estudos correlatos a redes sem fio, partem da premissa de que os dispositivos da rede se comunicam utilizando antenas omnidirecionais [Mohapatra and Krishnamurthy 2005]. Vale ressaltar que uma antena omnidirecional irradia os sinais de rádio em todas as direções. No entanto, a utilização de antenas omnidirecionais associado a um mecanismo de acesso ao meio que realiza a reserva de espaço aéreo através de pacotes de controle RTS/CTS (*Request to Send*/*Clear to Send*) impõe severas restrições a utilização do espaço aéreo. Estas restrições, por sua vez, refletem na vazão e no atraso [Bordim et al. 2010]. Por isso, a comunidade científica vem explorando alternativas para aumentar a vazão e reduzir atraso nestes ambientes. Uma destas alternativas é o emprego de antenas direcionais/setoriais [Krishnamurthy and Krishnamurthy 2005]. Uma antena direcional possui a capacidade de direcionar o feixe de transmissão para o setor da antena que maximiza a potência do sinal de interesse [Ramanathan 2004]. Entre os benefícios do uso de antenas direcionais, pode-se citar: redução de interferência; aumento do alcance da transmissão e da qualidade do sinal; melhoria do reuso espacial. Para o melhor aproveitamento destes benefícios, o protocolo de acesso ao meio utilizado deve ser capaz de explorar de modo eficiente as características da antena direcional. Como pode ser constatado em [Choudhury et al. 2002, Takata et al. 2009, Bordim and Nakano 2010, Amiri Sani et al. 2010], a tarefa de desenvolver um protocolo de acesso ao meio eficiente para antenas direcionais não é simples devido aos problemas de *surdez de antena* e *terminal exposto/escondido*. É neste contexto que a proposta deste trabalho está inserida. Mais precisamente, este trabalho propõe uma nova técnica de atenuação do problema de surdez baseada nas informações dos fluxos de tráfego aliada a uma técnica inovadora de reserva de canal baseada em sinais *pulse* e *tone*. Para a validação da técnica proposta, foram realizadas simulações utilizando o simulador de redes EXata [Scalable Network Technologies 2011]. Quando comparado com outros trabalhos, a técnica proposta obteve ganhos de mais de 350% em vazão por fluxo e até 76% em justiça (*fairness*) nos cenários avaliados.

O restante do texto está organizado da seguinte forma: na Seção 2 é apresentada uma revisão teórica acerca do padrão IEEE 802.11 e de comunicações direcionais, dando especial ênfase às técnicas de tratamento do problema de surdez existentes. Na Seção 3, é apresentada a técnica proposta neste trabalho que visa atenuar o problema de surdez. Em seguida, na Seção 4, é especificado o ambiente de simulação utilizado, bem como são apresentados os resultados obtidos. Por fim, na Seção 5, retoma-se uma discussão acerca dos aspectos mais importantes abordados ao longo deste trabalho, destacando suas contribuições, bem como possíveis trabalhos futuros.

2. Padrão IEEE 802.11 e Comunicações Direcionais

Esta seção apresenta uma revisão teórica no que diz respeito a antenas direcionais e omnidirecionais, explicando um pouco sobre o padrão IEEE 802.11 [IEEE 2007] e conceitos associados ao padrão em questão. Além disso, são apresentados os diversos benefícios provenientes da utilização de antenas direcionais. Por fim, expõe-se uma explicação detalhada do problema de surdez de antenas.

2.1. Padrão IEEE 802.11

No que diz respeito aos ambientes que utilizam transmissões omnidirecionais, um protocolo de acesso ao meio que visa resolver os problemas de terminal escondido é o padrão IEEE 802.11. Devido à grande extensão e nível de detalhe presentes no padrão IEEE 802.11, serão explicadas somente as técnicas utilizadas no padrão que estão diretamente relacionadas com este trabalho, em especial técnicas utilizadas pelos protocolos de acesso ao meio projetados para funcionar em ambientes que se utilizam de transmissões direcionais. As referidas técnicas são: reserva de canal e detecção de portadora virtual.

2.1.1. Reserva de Canal

Consiste em reservar o recurso antes de usá-lo, evitando colisões de pacotes de dados. Para realizar a reserva de canal, utilizam-se pacotes de controle como os pacotes *Request to Send* (RTS) e *Clear to Send* (CTS). Tais pacotes carregam informações sobre a duração da comunicação, dentre outras. A Figura 1 ilustra uma reserva de canal seguida de início de troca de pacotes entre os nós A e B. Na Figura 1, o nó A envia um RTS de maneira que qualquer outro nó que escute o RTS fique sem mandar mensagens por um período tal que a transmissão entre A e B acabe. Após receber o RTS, o nó B responde com uma mensagem CTS. O argumento análogo vale também para o CTS, isto é, qualquer outro nó que escute o CTS deve ficar sem mandar mensagens por um período tal que a transmissão entre A e B acabe. Com isso, o nó A pode enviar o pacote de dados para B com menor risco de sofrer colisões. Assim, a comunicação entre A e B flui normalmente com as trocas de pacotes DATA e ACK. Apesar das vantagens do uso dessa técnica de reserva de canal, o envio das mensagens de controle RTS e CTS aumenta consideravelmente a latência da rede [Bordim et al. 2010]. Além disso, tal técnica minimiza a colisão entre pacotes de dados, no entanto pode ocorrer colisão entre pacotes de controle. Por isso, foi proposto o protocolo RCA [Shih et al. 2009] (RTS *collision avoidance*) que antes do envio do RTS envia um pulso (*pulse*) na camada física e aguarda uma resposta (sinal *tone* na camada física) de maneira que esse tempo de envio do *pulse* é bem menor do que o de RTS. Dessa forma, sem aumentar muito a latência da rede, consegue-se evitar colisões de RTS [Shih et al. 2009]. Ainda com o intuito de reduzir a latência introduzida pelo uso dos quadros RTS/CTS na reserva de canal, foi proposta a técnica DPTCR (*Directional Pulse/Tone Based Channel Reservation*) que utiliza um *4-way handshake* com os sinais *pulse/tone* ao invés dos quadros RTS/CTS para a reserva de canal. Devido ao uso de *pulse/tone*, o DPTCR obtém ganhos de até 30% de vazão quando comparado com o esquema de RTS/CTS [Guimarães and Bordim 2012].

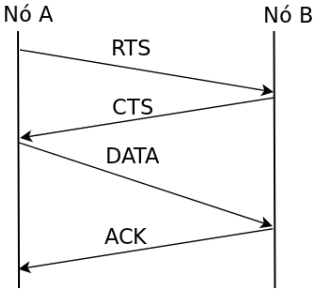


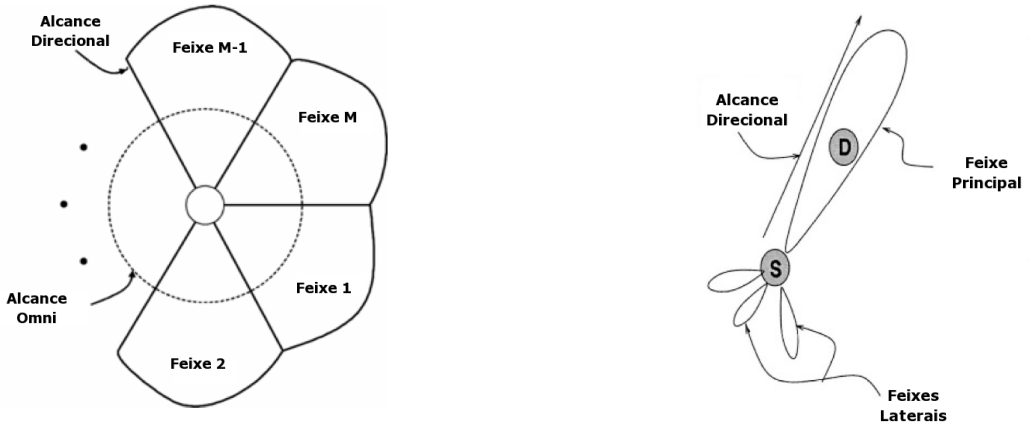
Figura 1. *4 way handshake* do padrão IEEE 802.11 no modo DCF, quando o nó A quer enviar dados ao nó B.

2.1.2. Detecção de Portadora Virtual

Para diminuir a quantidade de colisões, pode-se verificar se o canal está ocioso ou ocupado em um dado momento. Este processo é denominado de detecção de portadora. Tal funcionalidade está intimamente relacionada à camada física e necessitaria de *hardware* que possibilitasse a detecção em questão. Na prática, tal solução possui elevado custo e por isso realiza-se a detecção de portadora de maneira "virtual" [Gast 2002]. A detecção "virtual" é implementada usando uma estrutura denominada vetor de alocação de rede (*Network Allocation Vector* – NAV). O NAV é constituído de um contador que indica por quanto tempo o meio ainda estará ocupado. O NAV é atualizado utilizando informações embutidas nas mensagens de controle nos protocolos de acesso ao meio (RTS e CTS). Após o estabelecimento do NAV, ele vai sendo decrementado até que possua valor nulo. Quando isso ocorre, significa que o meio está ocioso. Utilizando o NAV é possível verificar o estado do canal de modo "virtual", reduzindo as colisões em ambientes onde as transmissões ocorrem de modo omnidirecional [IEEE 2007]. Neste contexto, duas propostas usadas em cenários ligeiramente distintos, introduziram a realização da detecção direcional de portadora virtual (DNAV) que nada mais é do que uma versão direcional do NAV [Choudhury et al. 2002] [Takai et al. 2002]. Estas propostas são: a técnica *Directional Virtual Carrier Sensing* (DVCS) [Takai et al. 2002] e o protocolo *Directional MAC Protocol* [Choudhury et al. 2002]. Tais propostas funcionam de maneira bem semelhante, sendo ambas baseadas no padrão IEEE 802.11. A principal diferença entre elas é que a técnica DVCS não requer conhecimento prévio da posição dos nós vizinhos e suporta interoperabilidade entre nós com antenas direcionais e omnidirecionais. É importante mencionar que a grande maioria dos protocolos atualmente projetados para antenas direcionais são baseados no padrão IEEE 802.11 com a utilização de DVCS. Terminada essa explicação preliminar de conceitos, será apresentada uma revisão de características, benefícios e desafios da utilização de antenas direcionais.

2.2. Comunicações Direcionais

Conforme citado na Seção 1, o uso de antenas direcionais possibilita diversas vantagens com relação a reuso do espaço aéreo, menor interferência entre os nós, melhor intensidade e qualidade do sinal enviado [Choudhury et al. 2002, Ramanathan 2004]. Isso ocorre uma vez que pode-se direcionar o feixe de transmissão da antena para uma direção específica. Porém, isto não é possível em antenas cuja transmissão ocorre apenas em modo omnidirecional, pois nesse tipo de antena os sinais eletromagnéticos são irradiados em todas as direções. O comportamento descrito pode ser observado na Figura 2(a). Enquanto as



(a) *Switched Beam.* (b) *Steerable Beam.*

Figura 2. Tipos de antena direcionais mais utilizadas na literatura, adaptado de [Krishnamurthy and Krishnamurthy 2005] e [Takata et al. 2009]

.

antenas direcionais podem direcionar seu feixe, as omnidirecionais irradiam o sinal para todas as direções. O ganho relativo a reuso do espaço aéreo fica ainda mais evidente quando se observa que as antenas direcionais irradiam sinal em uma área bem menor do que as antenas omnidirecionais para realizar a mesma comunicação, conforme ilustrado na Figura 2(a). Devido a esse maior reuso do espaço aéreo, há uma diminuição nas perdas por interferência. Quando bem explorado, o aumento da quantidade de transmissões simultâneas reduz o atraso médio nas comunicações da rede, o que exemplifica o relevante potencial da utilização de antenas direcionais. Há ainda ganhos na qualidade do sinal que chega no destino com maior intensidade e ganhos na segurança da comunicação, pois quanto menor a área irradiada com sinais eletromagnéticos, mais difícil se torna interceptar uma comunicação [Basagni et al. 2004]. É ainda importante mencionar que na literatura correlata geralmente se considera a utilização de um dos seguintes tipos de antenas direcionais [Krishnamurthy and Krishnamurthy 2005, Ramanathan 2004]:

1. *Switched Beam*: Consiste de n setores fixos, onde cada um aponta em uma direção e cada setor cobre $360/n$ graus, conforme ilustrado na Figura 2(a).
2. *Steerable Beam*: Permite direcionar o feixe para qualquer direção de interesse. Uma vantagem dessa antena é o fato dela ser mais flexível no que concerne ao ajuste do feixe da antena. Um exemplo de antena deste tipo está disponível na Figura 2(b).

Entretanto, conforme exposto na Seção 1, a utilização de antenas direcionais também impõe limitações e incorre em problemas tais como de surdez de antenas. A seguir, será apresentada uma explicação mais detalhada acerca deste problema.

2.2.1. Problema de Surdez de Antenas

Conforme descrito em [Li et al. 2005], um nó que utiliza uma antena direcional é considerado "surdo" em todas as direções com exceção da direção do seu feixe de recepção principal, ou seja, ele escuta somente na direção do feixe. Essa característica juntamente com

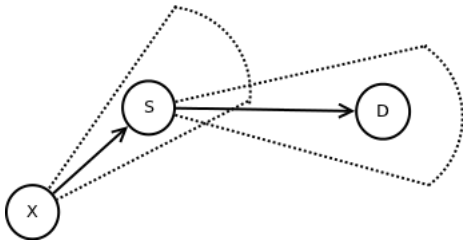
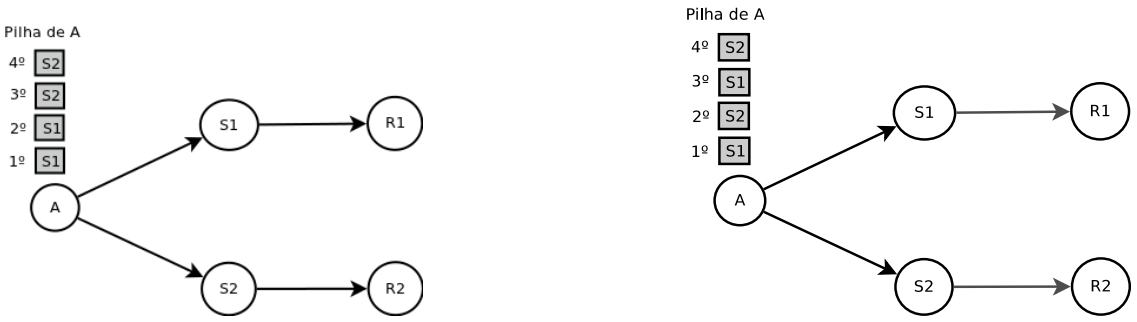


Figura 3. Cenário que ilustra o problema de surdez.



(a) Surdez atenuada (b) Surdez não atenuada

Figura 4. Cenários do protocolo RI-DMAC

o fato das antenas serem *half-duplex* provoca o chamado problema de surdez de antenas. Este problema foi identificado e denominado de *deafness* por [Choudhury et al. 2002]. A Figura 3 ilustra o problema em questão.

Considere o cenário ilustrado na Figura 3, onde todas as transmissões são consideradas direcionais. O nó X deseja se comunicar com o nó S. No entanto, o nó S está transmitindo dados para D, estando assim "surdo" para X. A situação descrita gera uma considerável perda de vazão e aumento no tempo de transmissão para cada pacote que X vai transmitir. É importante mencionar que quando o problema de surdez é recorrente em um mesmo nó, essa situação é denominada de surdez persistente (*persistent deafness*) [Takata et al. 2006].

O problema de surdez pode reduzir significativamente o desempenho da rede causando perda de pacotes devido a elevado tempo de espera, atrasos elevados, injustiça e até situações específicas de *deadlock*, conforme [Choudhury and Vaidya 2005] e [Li et al. 2005]. Por isso, torna-se evidente a necessidade de se minimizar os efeitos do problema de surdez de antenas. A seguir, serão brevemente descritos alguns protocolos MAC e técnicas relativas a comunicações direcionais.

2.2.2. Técnicas e Protocolos para Comunicações Direcionais

O DVCS é uma técnica de detecção de portadora utilizada por protocolos MAC para detectar a disponibilidade do meio em uma direção específica, quando estes fazem uso de antenas direcionais na camada física. Ao contrário de outros estudos realizados com antenas direcionais, o DVCS não requer conhecimento prévio da posição dos nós vizinhos e suporta interoperabilidade entre nós com antenas direcionais e omnidirecionais. Vale

ressaltar que o DVCS é uma técnica e não um protocolo, podendo ser utilizada ou adaptada para funcionar com diversos protocolos de camada MAC. Na proposta da técnica em questão [Takai et al. 2002], apresenta-se o DVCS funcionando em conjunto com o padrão IEEE 802.11 utilizando-se antenas direcionais na camada física. Por isso, quando da comparação de protocolos MAC, este trabalho considera a técnica DVCS utilizada em conjunto com o padrão IEEE 802.11.

Há diversas propostas para tratamento do problema de surdez como as apresentadas por [Bordim and Nakano 2010, Choudhury and Vaidya 2005, Subramanian and Das 2010]. Entretanto, tais propostas não avaliam nem obtém justiça (*fairness*) na distribuição dos recursos aos fluxos incidentes aos nós. Portanto, estas propostas não serão avaliadas neste trabalho. Uma outra proposta que avalia e obtém justiça em alguns casos é o protocolo *Receiver-initiated Directional MAC* (RI-DMAC) [Takata et al. 2006]. Para isso, o RI-DMAC utiliza uma abordagem de protocolo baseada em dois modos de operação: modo *sender-initiated* (SI) que é o modo padrão, baseado no padrão IEEE 802.11; modo *receiver-initiated* (RI) que é o modo secundário, sendo habilitado somente quando um nó detecta que o transmissor está sendo afetado pela surdez de antenas. Nesse modo, utiliza-se um quadro de controle do tipo *Ready to Receive* (RTR) para tratar o problema de surdez detectado. Esse tipo de quadro foi introduzido pelo protocolo MACA-BI [Talucci et al. 2002], de maneira a iniciar a transmissão do pacote pelo nó receptor.

Para um nó detectar se algum outro está sofrendo o problema de surdez por conta de uma comunicação direcional em andamento, cada nó insere informações de próximo pacote de sua fila IP. Portanto, no RI-DMAC, para que um nó consiga uma boa predição da ocorrência de surdez, os pacotes incidentes a esse nó devem estar em sequência na fila IP, conforme o cenário ilustrado na Figura 4(a). Em cenários onde essa premissa não ocorre, como na Figura 4(b), o RI-DMAC não atenua os efeitos do problema de surdez. Neste contexto, surge uma outra técnica que visa atenuar o problema de surdez que é o DVCS-DA (*Directional Virtual Carrier Sensing with Deafness Avoidance*) [Dias et al. 2012]. O DVCS-DA por sua vez se baseia em informações dos fluxos incidentes ao nó, ao invés de se basear em informações de próximo pacote. Dessa forma, o DVCS-DA consegue atenuar os efeitos do problema de surdez em cenários de fluxos intercalados, provendo assim uma melhora no que tange a justiça (*fairness*) na distribuição da vazão dos fluxos incidentes a um nó [Dias et al. 2012]. Entretanto, para prover esses ganhos relativos a justiça, tanto o DVCS-DA como o RI-DMAC apresentam uma vazão total da rede menor do que quando se utiliza apenas o DVCS. Para atenuar o problema de surdez e evitar essa queda na vazão total da rede, este trabalho propõe uma nova técnica de tratamento do problema de surdez baseada no DVCS-DA [Dias et al. 2012] e no DPTCR [Guimarães and Bordim 2012] que será melhor explicado na seção a seguir.

3. DPTCR-DA: Técnica de Atenuação do Problema de Surdez de Antenas Baseada em Sinais *Pulse/Tone*

Nesta seção, é apresentada a técnica DPTCR-DA (*Directional Pulse/Tone Based Channel Reservation with Deafness Avoidance*) que busca amenizar o problema de surdez, se utilizando de uma diferente forma de reservar o canal. O DPTCR-DA utiliza um mecanismo para predizer a ocorrência do problema de surdez e posteriormente tratá-lo utilizando transmissões iniciadas pelo receptor com auxílio de um sinal *tone*. O mecanismo

em questão baseia-se em informações locais de um nó sobre os fluxos incidentes a este para predizer a ocorrência de surdez, tal como ocorre com a técnica DVCS-DA, descrita por [Dias et al. 2012].

3.1. Mecanismo de Detecção da Ocorrência de Surdez

O mecanismo de detecção da ocorrência do problema de surdez determina dentre os fluxos incidentes ao nó qual está sofrendo do problema de surdez a mais tempo, ou seja, qual nó vizinho está esperando há mais tempo para enviar um quadro de dados com sucesso. Por isso, a detecção de surdez depende do instante de recepção do último quadro e do limiar de surdez esperado (T_i) para cada fluxo i, onde i representa o índice do fluxo. A detecção é feita verificando-se, para cada fluxo i incidente ao nó, há quanto tempo um quadro não chega desse fluxo, ou seja, verificando-se o tempo de espera de recepção desse fluxo (E_i).

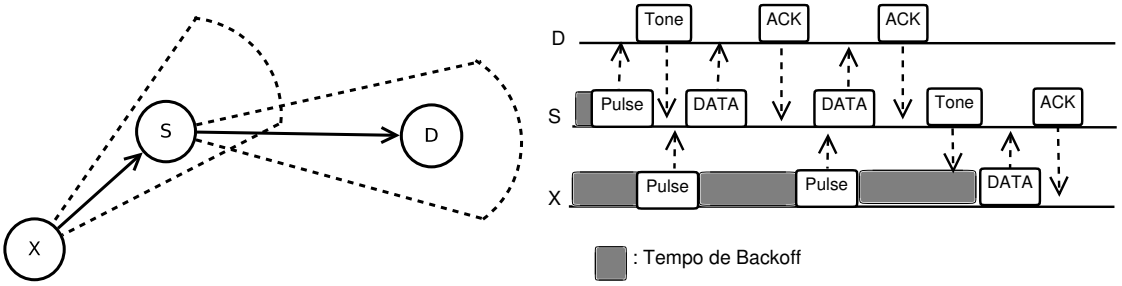
Um fluxo i é identificado como aquele que está sofrendo do problema de surdez se o tempo de espera E_i for superior a um limiar de detecção de surdez (T_i), isto é, caso $E_i > T_i$. Caso mais de um fluxo esteja sofrendo do problema de surdez, o fluxo com maior tempo de espera (E_i) será o escolhido.

Note que o limiar de surdez é uma função que depende do valor esperado do intervalo entre pacotes (I^i_{esp}). Seja α um fator de correção, o limiar de surdez é definido como: $T_i = \alpha * I^i_{esp}$. Para cada fluxo i, o fator de correção (α) pode ser estimado com base nas informações da tabela de fluxos ativos do nó ou de outras maneiras, conforme descrito por [Dias et al. 2012].

3.2. Comportamento do Protocolo

Assim como ocorre com o RI-DMAC e o DVCS-DA, o DPTCR-DA possui dois modos de operação: o **modo padrão** que utiliza comunicações iniciadas pelo transmissor (*sender-initiated*) e o **modo de tratamento de surdez** que utiliza comunicações iniciadas pelo receptor (*receiver-initiated*). As comunicações iniciadas pelo receptor são utilizadas somente para tratar o problema de surdez, quando este é detectado.

O **modo padrão** de operação do DPTCR-DA é baseado na técnica de reserva de canal denominada DPTCR, descrita por [Guimarães and Bordim 2012]. Dessa forma, a reserva de canal passa a utilizar sinais *pulse* e *tone* enviados de modo direcional ao invés dos quadros RTS e CTS visando obter ganhos de vazão, uma vez que o tempo de transmissão de sinais *pulse* e *tone* é inferior ao dispendido para transmitir quadros RTS e CTS [Guimarães and Bordim 2012]. Para o correto funcionamento da reserva de canal é necessário o uso do mecanismo de predição de origem e duração do sinal. O mecanismo usado no DPTCR-DA é o mesmo descrito por [Guimarães and Bordim 2012]. No mais, o modo padrão de operação se assemelha ao DVCS, com as exceções explicadas a seguir. Da mesma forma descrita por [Dias et al. 2012], após a correta transmissão de um quadro de dados, o nó transmissor realiza o já descrito procedimento de detecção de surdez. Em caso de detecção do problema de surdez, o nó transmissor vai para o modo tratamento de surdez. No que diz respeito ao nó receptor, a mudança reside no fato dele tratar com os sinais *pulse* e *tone* a ele direcionados, ao invés de pacotes RTS e CTS o que se torna possível com o uso do mecanismo de predição de origem e duração do sinal, descrito por [Guimarães and Bordim 2012]. No caso de receber um *pulse*, ocorre o *4-way handshake* (Pulse-Tone-DATA-ACK), conforme ilustrado na Figura 5(b). Caso um sinal



(a) Topologia exemplo do problema de surdez. (b) Fluxo de Sinais/Mensagens do DPTCR-DA

Figura 5. Funcionamento do DPTCR-DA, adaptado de [Dias et al. 2012] e [Guimarães and Bordim 2012].

tone seja recebido, é verificado se o nó que o enviou é o destino do pacote atual que se encontra no *buffer* da camada MAC. Nesse caso, o pacote é enviado e espera-se por um quadro de confirmação ACK.

O **modo de tratamento de surdez** é ativado somente no caso em que o procedimento de detecção de surdez indicar que um nó vizinho está sofrendo do problema de surdez. Seu comportamento consiste em enviar um *tone* para o nó que está sofrendo do problema de surdez, como ilustrado na Figura 5(b). Neste caso, o processo de início de comunicação ocorre em três vias (Tone-DATA-ACK). A seguir, serão descritos alguns importantes detalhes do modo de tratamento de surdez. O sinal *tone* enviado para iniciar uma comunicação em três vias é utilizado para atualizar o DNAV dos nós vizinhos. A duração de tempo da transmissão do sinal *tone* (Y_{tone}) enviada nesse caso é tal que: $Y_{tone} = Y_{sync} + \lceil \log_2 P_{sz} \rceil$, onde Y_{sync} é um tempo de sincronização e P_{sz} é o tamanho predito do pacote de dados pelo mecanismo de detecção da ocorrência de surdez. Essa duração do *tone* é necessária para o correto funcionamento do mecanismo de predição da origem e duração do sinal, descrito em [Guimarães and Bordim 2012]. Tal mecanismo é necessário para o correto funcionamento do DNAV no DPTCR-DA. Vale ressaltar que o tempo de sincronização de $5\mu s$ é suficiente para um sinal *tone* [Shih et al. 2009]. É importante mencionar que não ocorre retransmissão de *tones* no modo tratamento de surdez.

Um exemplo que ilustra o funcionamento dos dois modos de operação do DPTCR-DA está disponível na Figura 5. Neste cenário, assume-se que há os seguintes fluxos de dados: $X \rightarrow S$ e $S \rightarrow D$. Assume-se ainda que todos pacotes e sinais são enviados no modo direcional e que alguns pacotes de cada fluxo já foram transmitidos com sucesso, ou seja o nó S tem conhecimento dos dois fluxos de dados. No início, o nó S espera o seu tempo de *backoff* e então tenta realizar uma comunicação iniciada pelo transmissor (**modo padrão**) enviando um *pulse* para D. O nó D verifica que aquele *pulse* é para ele e responde com um *tone*, fazendo com que S envie pacotes de dados para D que por sua vez confirma o recebimento de tais pacotes por meio de mensagens ACK. Enquanto isso, X tenta também realizar uma comunicação iniciada pelo transmissor (**modo padrão**) com S por meio do envio de sinais *pulse*. Por S estar com o feixe de antena direcionado para D, S não processa os sinais enviados por X, causando um aumento exponencial no tempo de *backoff* do nó X. Essa situação persiste até que S verifica que $E_j > T_j$, onde j corresponde ao índice do fluxo $X \rightarrow S$. Isso significa que S verificou que X possui pacotes pendentes para S (identificados como surdez para S). Então, S entra no

modo tratamento de surdez e envia um sinal *tone* para X, de maneira a começar uma comunicação iniciada pelo receptor. Logo ao receber o sinal *tone*, o nó X verifica que aquele *tone* é para ele, cancelando o seu tempo de *backoff* e enviando a S um quadro de dados. Dessa forma, a comunicação entre X e S continua normalmente. Note que por meio das comunicações iniciadas pelo receptor é possível estabelecer uma maior justiça (*fairness*) na vazão dos fluxos, conforme será melhor explicado na Seção 4. Concluída a apresentação do DPTCR-DA, serão apresentados resultados que validam sua relevância.

4. Resultados

Nessa seção, avaliou-se o desempenho do DPTCR-DA comparando-o com outros dois protocolos de acesso ao meio: o padrão IEEE 802.11 DCF em sua versão direcional acoplado à técnica DVCS; o protocolo RI-DMAC. Com este fim, utilizou-se a versão 2.0 do simulador EXata. O EXata é o antigo simulador Qualnet com mais funcionalidades, englobando também as funcionalidades do simulador GloMoSim [Scalable Network Technologies 2011]. É importante mencionar que os 3 ambientes de simulação mencionados são frequentemente utilizados na literatura correlata.

Nas simulações realizadas, foram considerados dois cenários distintos, ilustrados na Figura 6. O primeiro cenário (Figura 6(a)) visa validar os ganhos obtidos pelo DPTCR-DA quando opera somente no modo padrão, uma vez que em tal cenário não ocorre surdez. O segundo cenário (Figura 6(b)) é referente a uma situação de fluxos alternados como fora mencionado na Seção 2. Embora tenha-se realizado simulações para o DPTCR-DA em diversos outros cenários, estes resultados não serão apresentados por limitação de espaço. Entretanto, os cenários apresentados já validam o impacto positivo do uso do DPTCR-DA em comunicações direcionais. Para cada fluxo de dados presente nos cenários apresentados, utilizou-se uma taxa de transmissão constante CBR (*constant bit rate*), que foi variada durante as baterias de simulações modificando-se o intervalo entre pacotes. Assumiu-se uma taxa de transmissão do canal de 2Mbps e tamanho de pacote de 1024 *bytes*. Não considerou-se mobilidade dos nós, uma vez que os trabalhos correlatos não avaliam tal prisma, considerando que a mobilidade deve ser tratada pelas camadas superiores. Assumiu-se o uso de antenas do tipo setorial (*switched beam*) com 8 setores de 45° cada. Para o DPTCR-DA, utilizou-se um limiar fixo tal que: $T_i = 1,3 \cdot I^i_{esp}\ \forall i,\ i \geq 0$. Este limiar foi o utilizado com base nos estudos apresentados em [Dias et al. 2012].

Utilizou-se como métrica a vazão individual de cada fluxo existente nos cenários ilustrados na Figura 6. Embora a vazão seja uma métrica comumente usada na literatura para se comparar a eficiência de dois protocolos, esta não é uma métrica adequada para identificar ou medir o problema de surdez [Choudhury et al. 2002]. Por isso, utilizou-se também como métrica o índice de justiça de Jain [Jain et al. 1984]. O índice de Jain provê valores no intervalo $[0, 1]$, de maneira que um valor alto de índice de justiça representa uma maior justiça na distribuição dos recursos do meio. Seja th_i a vazão do fluxo de índice i e n a quantidade de fluxos da rede, o índice de justiça (índice de Jain) pode ser calculado com base na seguinte equação [Jain et al. 1984]:

$$\frac{(\sum_{i=1}^{n} th_i)^2}{n \cdot \sum_{i=1}^{n} th_i^2} \tag{1}$$

Cada um dos resultados apresentados é fruto da média de 10 simulações. A seguir,

(a) Cenário 1 (b) Cenário 2

Figura 6. Cenários Avaliados.

Tabela 1. Comparação de vazão e justiça dos fluxos no cenário 1.

Técnica	Fluxo $1 \rightarrow 2$ (Kbps)	Fluxo $1 \rightarrow 2$ (Kbps)	Índice de Justiça
DVCS	$1440,8621$	$1433,8156$	$0,999994$
DPTCR-DA	$1576,7975$	$1567,8636$	$0,999992$

serão apresentados os cenários utilizados, bem como os resultados obtidos para cada um deles.

4.1. Cenário 1

O cenário 1 possui quatro nós e dois fluxos, indicados respectivamente pelos círculos e setas na Figura 6(a). Este cenário tem por objetivo avaliar o funcionamento dos protocolos quando não ocorre situação de surdez. Como em casos que não ocorre surdez o RI-DMAC é igual ao DVCS, seu resultado neste cenário foi omitido. Desta forma, comparou-se o DPTCR-DA em seu modo padrão com o DVCS. A comparação em questão ocorreu quando se tem um intervalo de geração de pacotes igual a 1ms, uma vez que tal intervalo já é suficiente para saturar o canal de 2 Mbps com pacotes de 1024 *bytes*. Os resultados obtidos para vazão individual de cada fluxo estão apresentados na Tabela 1.

Quanto aos resultados, pode-se observar que o DPTCR-DA provê ganhos de aproximadamente 9% de vazão tanto para o fluxo $1 \rightarrow 2$ quanto para o fluxo $3 \rightarrow 4$, o que mostra que o modo padrão de funcionamento do DPTCR-DA é mais eficiente do que o DVCS. Esse ganho ocorre pois o DPTCR-DA realiza a reserva de canal usando sinais *pulse* e *tone*, ao invés de quadros RTS e CTS. Conforme descrito em [Guimarães and Bordim 2012], essa forma de reservar o canal possui um impacto bastante positivo nas comunicações direcionais. Tal impacto positivo faz com que o uso de *pulse/tone* possua uma vazão teórica maior do que o uso de RTS/CTS. É importante ainda mencionar que esse ganho teórico varia de acordo com o cenário e pode ser de até 42%, segundo [Guimarães and Bordim 2012]. Com relação ao índice de justiça, os resultados apontam uma justiça ligeiramente melhor para o DVCS, embora os resultados sejam bastante próximos. Todavia, este cenário não é muito relevante para a aferição de índice de justiça entre técnicas, uma vez que neste cenário não há ocorrência de surdez. Estas medições de índice de justiça são mais relevantes no cenário 2 (Figura 6(b)), onde a surdez ocorre.

Tabela 2. Vazão dos Fluxos e índice de justiça para o cenário 2.

Técnica/ Protocolo	I_g (ms)	Vazão do Fluxo $1 \rightarrow 2$ (Kbps)	Vazão do Fluxo $2 \rightarrow 3$ (Kbps)	Vazão do Fluxo $1 \rightarrow 4$ (Kbps)	Vazão do Fluxo $4 \rightarrow 5$ (Kbps)	Índice de Justiça
DPTCR-DA	6	$311,1105$	$725,7313$	$313,5964$	$730,1281$	$0,8624$
DPTCR-DA	5	$357,5906$	$651,6369$	$357,9443$	$647,7588$	$0,9225$
DPTCR-DA	**4**	$\mathbf{418,4422}$	$\mathbf{578,1706}$	$\mathbf{418,5690}$	$\mathbf{576,2043}$	$\mathbf{0,9752}$
RI-DMAC	6	$66,8295$	$1321,9703$	$66,5679$	$1323,8066$	$0,5503$
RI-DMAC	5	$68,6008$	$1321,3789$	$68,5395$	$1321,6764$	$0,5517$
RI-DMAC	4	$69,3540$	$1319,3879$	$68,0835$	$1322,2799$	$0,5519$
DVCS	6	$67,4337$	$1324,8763$	$65,8449$	$1328,8588$	$0,5501$
DVCS	5	$70,1337$	$1322,2012$	$69,2376$	$1323,0414$	$0,5525$
DVCS	4	$67,0369$	$1325,5814$	$68,2906$	$1324,5164$	$0,5509$

4.2. Cenário 2

Conforme ilustrado na Figura 6(b), o cenário 2 possui cinco nós e quatro fluxos, onde os nós 2 e 3 ficam surdos para o nó 1. Este cenário tem por objetivo mostrar que o DPTCR-DA atenua o problema de surdez quando há fluxos alternados tendo como origem um mesmo nó, nesse caso o 1. Os resultados para índice de justiça e vazão individual estão disponíveis na Tabela 2. Estes resultados variam o intervalo de geração de pacotes nos valores de 4, 5 e 6ms.

Os resultados mostram que o protocolo RI-DMAC apresenta um comportamento idêntico ao protocolo DVCS, como esperado. Nos resultados de vazão individual e de índice de justiça, os valores dos protocolos RI-DMAC e DVCS são praticamente iguais. O RI-DMAC bem como o DVCS possuem uma distribuição injusta da vazão entre os fluxos, conforme mostrado na Tabela 2. Este comportamento decorre do problema de surdez que acontece nos nós 2 e 4, causando uma baixa vazão nos fluxos $1 \rightarrow 2$ e $1 \rightarrow 4$ e uma alta vazão nos demais. Conforme descrito na Seção 2, essa situação ocorre porque nesse cenário os pacotes da fila IP do nó 1 estão intercalados. Portanto, nesse caso, o RI-DMAC não envia informações de próximo pacote, consequentemente não atenuando o problema de surdez.

Contudo, a técnica proposta (DPTCR-DA) consegue atenuar o problema de surdez e distribuir de maneira mais justa a vazão entre os fluxos. Isso pode ser observado na Tabela 2, onde se verifica uma distribuição mais igual da vazão entre os fluxos. Note que a vazão dos fluxos $1 \rightarrow 2$ e $1 \rightarrow 4$ é mais do que $4,5$ vezes maior (350% maior) do que quando se usa o DVCS ou o RI-DMAC. Quanto aos valores de índice de justiça, o DPTCR-DA obteve resultados melhores que o RI-DMAC e o DVCS em todos os casos, chegando a se obter um índice de justiça até 76% maior. Dessa forma, espera-se ter validado o impacto positivo do uso do DPTCR-DA em comunicações direcionais, uma vez que tal técnica atenua o problema de surdez como mostra esse cenário, além de aumentar a vazão teórica em casos onde a surdez não ocorre, conforme mostrado no cenário 1. Por fim, será apresentada uma breve conclusão deste trabalho.

5. Conclusão

Este trabalho apresentou o contexto em que se propõe a utilização de antenas direcionais, mostrando suas vantagens bem como os problemas decorrentes de seu uso. Dentre es-

ses problemas, um possui destaque especial: o problema de surdez. Por isso, o presente trabalho apresentou uma breve revisão do estado da arte acerca dos protocolos e técnicas existentes para se atenuar o problema de surdez. Então, esse trabalho apresenta a proposta de uma nova técnica de tratamento do problema de surdez (DPTCR-DA) baseada nas informações dos fluxos de tráfego para predizer a ocorrência de surdez, bem como baseada em uma inovadora maneira de se reservar o canal por meio do uso de sinais *pulse/tone*. O DPTCR-DA foi validado com o auxílio do simulador de redes EXata. Quando comparado às demais técnicas, o DPTCR-DA apresenta ganhos de mais de 350% em vazão por fluxo e até 76% em justiça. Por fim, com relação a trabalhos futuros, identificou-se que seria relevante realizar uma melhor parametrização do limiar T visando uma melhora no desempenho da técnica proposta.

Agradecimentos

Este trabalho foi parcialmente financiado pelo CNPq e CIC/DPP/UnB.

Referências

Amiri Sani, A., Zhong, L., and Sabharwal, A. (2010). Directional antenna diversity for mobile devices: characterizations and solutions. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 221–232. ACM.

Basagni, S., Conti, M., Giordano, S., and Stojmenović, I. (2004). *Mobile ad hoc networking*. Wiley-IEEE Press.

Bordim, J., Barbosa, A., Caetano, M., and Barreto, P. (2010). IEEE802. 11b/g standard: Theoretical maximum throughput. In *Networking and Computing (ICNC), 2010 First International Conference on*, pages 197–201. IEEE.

Bordim, J. and Nakano, K. (2010). Deafness resilient mac protocol for directional communications. *IEICE TRANSACTIONS on Information and Systems*, 93(12):3243–3250.

Choudhury, R. and Vaidya, N. (2005). Deafness: A MAC problem in ad hoc networks when using directional antennas. In *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pages 283–292. IEEE.

Choudhury, R., Yang, X., Ramanathan, R., and Vaidya, N. (2002). Using directional antennas for medium access control in ad hoc networks. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 59–70. ACM.

Dias, P., Guimarães, L., Nunes, L., Caetano, M., and Bordim, J. (2012). Proposta de tratamento do problema de surdez de antenas em comunicações direcionais. In *XXX Simpósio Brasileiro de Telecomunicações*, pages 540–544. Sociedade Brasileira de Telecomunicações.

Gast, M. (2002). *802.11 wireless networks: the definitive guide*. O'Reilly Media.

Guimarães, L. and Bordim, J. (2012). Utilização de pulse/tone como mecanismo de reserva de canal em comunicações direcionais. In *XXXVIII Conferencia Latinoamericana en Informática*, pages 515–523. Centro Latinoamericano de Estudios en Informática.

IEEE (2007). IEEE Standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. IEEE Standard 802.11, Institute of Electrical and Electronics Engineers.

Jain, R., Chiu, D., and Hawe, W. (1984). *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation.

Krishnamurthy, P. and Krishnamurthy, S. (2005). Use of smart antennas in ad hoc networks. In *AD HOC NETWORKS: technologies and protocols*, volume 1, page 197. Springer.

Kumar, S., Raghavan, V., and Deng, J. (2006). Medium Access Control protocols for ad hoc wireless networks: A survey. *Ad Hoc Networks*, 4(3):326–358.

Li, G., Yang, L., Conner, W., and Sadeghi, B. (2005). Opportunities and challenges for mesh networks using directional antennas. In IEEE, editor, *Proc. First IEEE Workshop Wireless Mesh Networks (WiMesh'05)*, pages 106–116.

Mohapatra, P. and Krishnamurthy, S. (2005). *AD HOC NETWORKS: technologies and protocols*. Springer.

Ramanathan, R. (2004). Antenna beamforming and power control for ad hoc networks. In *Mobile ad hoc networking*, volume 1, pages 139–173. Wiley-IEEE Press.

Scalable Network Technologies (2011). Simulador exata. *http://www.scalable-networks.com/exata/*.

Shih, K., Liao, W., Chen, H., and Chou, C. (2009). On avoiding rts collisions for ieee 802.11-based wireless ad hoc networks. *Computer Communications*, 32(1):69–77.

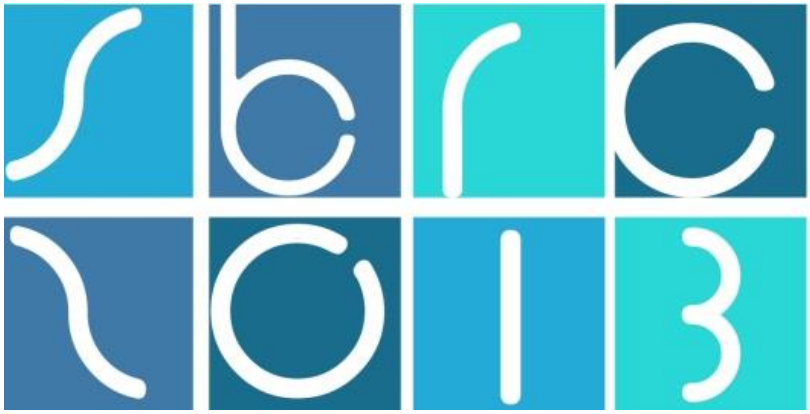
Subramanian, A. and Das, S. (2010). Addressing deafness and hidden terminal problem in directional antenna based wireless multi-hop networks. *Wireless Networks*, 16(6):1557–1567.

Takai, M., Martin, J., Bagrodia, R., and Ren, A. (2002). Directional virtual carrier sensing for directional antennas in mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 183–193. ACM.

Takata, M., Bandai, M., and Watanabe, T. (2006). A receiver-initiated directional MAC protocol for handling deafness in ad hoc networks. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 9, pages 4089–4095. IEEE.

Takata, M., Bandai, M., and Watanabe, T. (2009). RI-DMAC: a receiver-initiated directional MAC protocol for deafness problem. *International Journal of Sensor Networks*, 5(2):79–89.

Talucci, F., Gerla, M., and Fratta, L. (2002). MACA-BI (MACA by invitation)-a receiver oriented access protocol for wireless multihop networks. In *Personal, Indoor and Mobile Radio Communications, 1997.'Waves of the Year 2000'. PIMRC'97., The 8th IEEE International Symposium on*, volume 2, pages 435–439. IEEE.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 7

Sistemas Distribuídos

Estratégias de Obtenção de um Item Máximo em Computação por Humanos

Jeymisson Oliveira, Lesandro Ponciano, Nazareno Andrade, Francisco Brasileiro

[1]Departamento de Sistemas e Computação
Universidade Federal de Campina Grande – UFCG
Campina Grande – PB – Brasil

jeymisson.oliveira@ccc.ufcg.edu.br, lesandrop@lsd.ufcg.edu.br,

{nazareno,fubica}@dsc.ufcg.edu.br

Abstract. *Human computation is a distributed system that orchestrates the work of a group of workers willing to solve relatively simple tasks, whose solutions, once grouped, allow us to solve a computational problem that could not be resolved satisfactorily by using today's machine computation systems. In this work, we address a recurring problem in Human Computation, which is to identify a maximum item in a set of items candidates. We evaluate three algorithms in state-of-the-art for selecting a single candidate for each comparison (2-Max, tournament selection and tournament max) and we propose a multiple elimination strategy that allows workers to eliminate pairs of candidates in a single comparison. In an analytical study, we show that the proposed strategy can increase the efficiency of the algorithms 2-Max and tournament max in terms of the number of tasks required to achieve the maximum item. The analytical study is expanded in an experimental study where we assessed the accuracy of workers when the multiple elimination strategy is used.*

Resumo. *Um sistema de Computação por Humanos orquestra o trabalho de um grupo de trabalhadores dispostos a resolver tarefas relativamente simples, cujas soluções, uma vez agrupadas, resolverão um problema computacional que não poderia ser resolvido de forma satisfatória com os sistemas atuais. Neste trabalho, tratamos de um problema recorrente em Computação por Humanos que é identificar um item máximo em um conjunto de itens candidatos. Avaliamos três algoritmos no estado-da-arte que permitem selecionar um único candidato a cada comparação (2-Max, tournament selection e tournament max) e propomos uma estratégia de eliminação múltipla que permite eliminar pares de candidatos em uma única comparação. Em um estudo analítico mostramos que a estratégia proposta permite aumentar a eficiência dos algoritmos 2-Max e tournament max em termos do número de tarefas necessárias para se obter o item máximo. O estudo analítico é ampliado em um estudo experimental, no qual avaliamos a acurácia dos trabalhadores com uso da estratégia de eliminação múltipla.*

1. Introdução

Existem diversos problemas que os algoritmos e computadores atuais ainda não são capazes de resolver de forma satisfatória [Quinn and Bederson 2011,

Quinn and Bederson 2009, Ahn 2005]. Esses problemas incluem compreensão de linguagem natural, recuperação de informação em imagens e tarefas ligadas à criatividade. Seres humanos, por sua vez, possuem habilidades que permitem resolver esses problemas de forma rápida e precisa [Ahn 2005, Eickhoff and de Vries 2011]. Visando tirar proveito dessas habilidades, tem sido desenvolvido um novo modelo de computação distribuída que utiliza da cognição e da capacidade de raciocínio de seres humanos para resolver problemas que os computadores ainda não podem resolver de forma satisfatória. Esse modelo é denominado Computação por Humanos (*Human Computation*).

Um sistema de Computação por Humanos pode ser visto como 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 dispostos a resolver tarefas relativamente simples, cujas soluções, uma vez agrupadas, resolverão um problema computacional que não poderia ser resolvido de forma satisfatória com os sistemas dotados de processadores convencionais [Ahn 2005]. Esses sistemas também são utilizados para execução de aplicações híbridas, nas quais algumas tarefas são executadas de forma mais eficiente por processadores de silício e outras por humanos [Schall et al. 2008, Dustdar and Truong 2012].

As aplicações de computação por humanos geralmente são divididas em pequenas tarefas descritas como micro tarefas (*microtasks*) ou tarefas que requerem inteligência humana (HIT, do inglês, *Human Intelligence Task*) [Little 2011]. Cada tarefa pode ser executada de forma independente por um trabalhador, e a agregação dos resultados provê a solução do um problema computacional que é objeto da aplicação [Spiekermann 2010, Quinn and Bederson 2011]. Trabalhadores podem apresentar diferentes características em termos de, por exemplo, localização geográfica, habilidades e motivação para executar tarefas. Considerando a motivação dos trabalhadores, os sistemas atuais podem ser amplamente divididos em mercados de trabalho online e pensamento distribuído[1]. Em mercados de trabalho online, os trabalhadores possuem uma motivação financeira, e um dos principais exemplos é a plataforma Amazon Mechanical Turk[2]. Por outro lado, em plataformas de pensamento distribuído, os trabalhadores executam tarefas como um trabalho voluntário. Um dos principais exemplos deste tipo de plataforma é o projeto Galaxy Zoo[3].

Das diversas aplicações para as quais computação por humanos tem sido utilizada, um tipo de aplicação recorrente é identificar um item máximo em um conjunto de candidatos. Isso ocorre por exemplo quando se deseja escolher a fotografia mais criativa em um conjunto de diversas fotografias [Venetis et al. 2012] ou quando se deseja obter a melhor tradução de uma frase em um conjunto de traduções candidatas [Sun et al. 2011]. Nesses casos, se realizadas todas as possíveis comparações entre os itens do conjunto, e se o resultado de cada comparação reflete o senso comum sobre os itens comparados, então o item máximo do conjunto é aquele selecionado como máximo no maior número de comparações. Um fator observado nesses contextos é que um ser humano não consegue (ou apresenta maior dificuldade para) comparar as diversas opções candidatas de uma só vez e identificar a melhor delas [Sweller et al. 1998]. Essa dificuldade agrava-se na

[1]Também chamado pensamento voluntário (*Volunteer Thinking*)

[2]`mturk.com`

[3]`galaxyzoo.org`

proporção em que cresce o número de opções a serem comparadas [Venetis et al. 2012].

Uma abordagem utilizada para resolver esse problema é combinar o conjunto de opções candidatas em pares e instruir trabalhadores para que escolham o item máximo em cada combinação [Sun et al. 2011, Venetis et al. 2012, Ajtai et al. 2009]. A partir da agregação destas comparações entre pares de opções, é possível identificar o item máximo. No caso mais simples, é possível comparar todas as opções em pares, e o item máximo é aquele que vencer mais comparações. Porém, é possível também encontrar o item máximo com outras estratégias que necessitam de um número de combinações menor do que a combinação de todas as opções com todas as outras. A otimização no número de comparações é desejável para otimizar o uso dos trabalhadores disponíveis. Entretanto, como os trabalhadores podem errar ao efetuar uma comparação, qualquer estratégia de otimização precisa também garantir que conseguirá lidar com tais erros. Neste contexto, mostra-se necessário avaliar em que proporção esses erros impactam na acurácia da escolha do item máximo.

Neste trabalho, tratamos do problema da escolha do item máximo no contexto de computação por humanos. Analisamos o desempenho de três algoritmos estado-da-arte de seleção do item máximo: *2-Max* [Ajtai et al. 2009], *tournament selection* [Sun et al. 2011] e *tournament max* [Venetis et al. 2012]. Propomos então uma extensão desses algoritmos a fim de utilizá-los em tarefas em que a eliminação múltipla de itens candidatos pode ser aplicada. Em um estudo analítico que considera diferentes tamanhos de conjuntos de itens candidatos, mostramos que essa extensão permite reduzir o número de tarefas necessárias para obtenção do item máximo. Nosso estudo analítico é ampliado com um estudo de caso, no qual avaliamos a acurácia e a convergência dos trabalhadores quando se utiliza a estratégia de eliminação múltipla proposta neste artigo.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 apresentamos os trabalhos relacionados destacando os algoritmos *2-Max*, *tournament selection* e *tournament max*. Na Seção 3 apresentamos um estudo analítico no qual comparamos os algoritmos considerando diversos cenários de número de opções candidatas. Em seguida, na Seção 4 apresentamos um estudo experimental com uma avaliação da variação da acurácia dos trabalhadores quando se utiliza a estratégia de eliminação múltipla. Finalmente, na Seção 5 apresentamos as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O problema da obtenção de um item máximo em um conjunto de candidatos por meio de comparações entre pares de itens é recorrente na computação. Em razão disso, diversas abordagens têm sido propostas para solucionar esse problema em diferentes contextos, tais como: sistemas distribuídos compostos por máquinas [Borgstrom and Kosaraju 1993], pesquisa comportamental [Ajtai et al. 2009] e eliminação de resultados de baixa qualidade gerados por seres humanos [Sun et al. 2011, Venetis et al. 2012].

Em sistemas distribuídos compostos por máquinas, Borgstrom e Kosaraju [Borgstrom and Kosaraju 1993] apresentam um estudo analítico do problema que surge quando se deseja ordenar os itens de um conjunto S onde a comparação entre dois itens é imprecisa. O principal foco do estudo é modelar os limites de erros nas respostas para se obter um resultado. O estudo define o mínimo de tarefas necessárias

para se obter o item máximo em um conjunto S, que é dado por $O(\log |S|)$ quando a proporção de tarefas com respostas incorretas (e) não supera 50%, i.e. $e < \frac{1}{2}$. O estudo pressupõe que sempre há uma diferença entre os itens candidatos e essa diferença é detectada pelos processadores ao definir o item máximo. Entretanto, neste trabalho tratamos do problema de obtenção do item máximo em ambiente de computação em que os processadores são seres humanos. Seres humanos apresentam maior dificuldade para comparar opções que apresentam pequena variação entre elas [Sweller et al. 1998]. Focamos na análise de algoritmos para encontrar o item máximo no contexto onde existem comparações com diferentes níveis de dificuldade e em que os seres humanos estão sujeitos a diferentes situações de erro.

Estudos em ciência comportamental também têm sido dedicados às estratégias que permitem obter o item máximo em cenários em que a tarefa de definir qual o item máximo entre dois candidatos é realizada por seres humanos. Ajtai et al. [Ajtai et al. 2009] propõem o algoritmo *2-Max*. A abordagem implementada por esse algoritmo é realizar a busca pelo item máximo partindo de subconjuntos de itens candidatos escolhidos aleatoriamente. Por essa abordagem, obtém-se o item máximo por subconjunto e verifica-se se esse item se mantém quando comparado a todos os demais itens.

Mais próximo do contexto de computação por humanos, têm sido propostas abordagens que utilizam comparação entre pares com o propósito de descobrir o melhor item em um conjunto de itens candidatos gerados por seres humanos. Nesses casos, simplesmente eliminar os resultados menos frequentes não é suficiente, pois nem sempre um item gerado pela maioria dos trabalhadores é um item máximo [Sun et al. 2011, Venetis et al. 2012]. Sun et al. [Sun et al. 2011] apresentam o algoritmo *tournament selection*. Esse algoritmo é baseado na ideia de algoritmos genéticos em que o item mais apto sobressai diante dos menos aptos. No caso, seres humanos definem o item mais apto em comparações de dois itens. O algoritmo realiza r torneios (ou gerações). A cada torneio obtém-se os itens mais aptos selecionados em comparações de n pares escolhidos de forma aleatória, e apenas esses itens participarão das comparações do próximo torneio. O estudo não apresenta um método de definição e/ou otimização dos parâmetros r e n; há apenas a restrição de $n < |S|$.

Venetis et al. [Venetis et al. 2012] também apresentam uma solução baseada em torneios que chamamos neste trabalho de *tournament max*. Nessa solução, o primeiro torneio consiste em combates entre todos os candidatos. Os candidatos que venceram pelo menos um combate compõem um conjunto de itens máximos que duelarão entre si no próximo torneio. O algoritmo implementa esse processo iterativo que termina quando há apenas um item máximo. O algoritmo parametriza a quantidade de itens que serão comparados em cada tarefa, por tratarmos de comparações entre pares, neste artigo assumimos que o valor desse parâmetro é 2.

Neste trabalho analisamos o desempenho dos algoritmos *2-Max*, *tournament selection* e *tournament max* quando se varia o tamanho do conjunto de itens candidatos ($|S|$). Além disso, nós propomos uma extensão desses algoritmos que permite utilizar a estratégia de eliminação múltipla para aumentar a eficiência dos algoritmos *2-Max* e *tournament max* em termos do número de comparações necessárias para se obter o item máximo.

3. Acelerando a obtenção do item máximo

Nesta seção definimos e avaliamos uma estratégia de eliminação múltipla que tem por objetivo acelerar a obtenção do item máximo em um conjunto de itens candidatos. A Tabela 1 apresenta um resumo da notação utilizada ao longo desta seção. Uma das principais métricas de avaliação é o número de comparações necessárias para se obter o item máximo s do conjunto de itens candidatos S ($s \in S$). No contexto de computação por humanos, essa métrica é um indicador da quantidade de trabalho que precisará ser realizado e da quantidade de trabalhadores que serão necessários para se obter o resultado.

Tabela 1. Descrição das variáveis utilizadas

Símbolo	Descrição
S	Conjunto de itens candidatos
s	Item máximo, $s \in S$
r	Número de torneios
(a, b)	Tarefa que compara o item a e o item b, $a \in S$ e $b \in S$.
w	Tamanho máximo de um subconjunto de itens definido como $\lceil\sqrt{\lvert S\rvert}\rceil$

3.1. Definindo eliminação múltipla

Os trabalhos discutidos na seção anterior consideram que dado um conjunto de requisitos que define o que é um item máximo e um par de itens a ser comparado (a, b), um trabalhador é capaz de identificar qual desses itens está em maior conformidade com os requisitos. Entretanto, humanos apresentam dificuldades para identificar pequenas variações entre itens. Tarefas que requerem tomar uma decisão baseada na análise desse tipo de variação tendem a aumentar a fadiga do trabalhador [Eugene Wu 2011], o que tem impacto negativo no tempo necessário para se obter esse resultado e na acurácia do resultado obtido. Isso tende a se agravar quando diversas tarefas com essa característica são escalonadas seguidamente para um mesmo trabalhador.

Consideramos que em aplicações de Computação por Humanos é aceitável obter, como resultado de uma comparação entre dois itens a e b, tanto a quanto b, caso ambos atendam os requisitos para serem um item máximo e sejam indistinguíveis para um ser humano. Entretanto, na situação oposta, se esses itens são indistinguíveis, mas incompatíveis com os requisitos para serem um item máximo, propomos que é adequado eliminar ambos os itens. Isso significa que o resultado de uma comparação pode ser nulo. Com essa eliminação múltipla, previne-se a comparação de um desses itens com outros itens que foram escolhidos em outras comparações. Isso tende a aumentar o número de itens eliminados a cada iteração do algoritmo *2-Max* e reduzir o número de itens que serão comparados entre si em novos torneios no algoritmo *tournament max*. Se ao término do algoritmo nenhum item máximo foi encontrado, todos os itens são ditos insatisfatórios.

3.2. Modelando a eficiência da eliminação múltipla

Nosso propósito é analisar a quantidade de tarefas que precisam ser geradas para se obter o item máximo em um conjunto de itens candidatos S, quando se varia: (i) o número

de itens candidatos ($|S|$), (ii) os algoritmos de escolha do item máximo na versão original e (iii) os algoritmos de escolha do item máximo adaptados para permitir eliminação múltipla.

Nós modelamos o número de tarefas geradas pelos algoritmos *tournament max* e *2-Max* considerando o melhor caso m e o pior caso p de quando a eliminação múltipla é utilizada. O melhor caso é aquele em que, para todo conjunto S, sempre há um item $s \in S$ que atende todos os requisitos para ser um item máximo, enquanto os demais itens desse conjunto não atendem esses requisitos. No que se refere ao pior caso, é importante ressaltar que existe um pior caso extremo em que a opção múltipla não é utilizada pelos trabalhadores. Nesse caso, os algoritmos com eliminação múltipla terão desempenho igual aos algoritmos sem essa opção. Entretanto, neste trabalho consideramos o pior caso como aquele em que a eliminação múltipla é utilizada pelos trabalhadores pelo menos uma única vez a cada iteração do algoritmo. Como mostraremos na Seção 4.2, essa abordagem mostra-se adequada dado que a opção de eliminação múltipla é utilizada por pelo menos um trabalhador a cada conjunto de soluções. Nos próximos parágrafos analisamos esses casos para cada algoritmo.

Tournament Max (TM). Este algoritmo utiliza uma série de torneios para escolher o melhor candidato. O primeiro torneio consiste em confrontar todas as opções candidatas em S e implica na geração de $\binom{|S|}{2}$ tarefas. Os itens escolhidos nessas comparações são mantidos em um novo conjunto de candidatos S; os itens que não são escolhidos em nenhuma comparação deixam de fazer parte do novo conjunto de candidatos S, sendo consequentemente eliminados. Os itens que permanecerem em S são então combinados entre si em comparações no próximo torneio. O algoritmo implementa esse processo iterativo, que sempre remove de S os itens que não vencerem pelo menos uma comparação. O algoritmo termina quando o conjunto de itens candidatos possui apenas o item máximo, i.e., $|S| = 1$.

Na versão original do algoritmo, sempre se remove apenas um item de S a cada iteração, tanto no pior quanto no melhor caso. Em razão disso, o custo do algoritmo varia apenas em função do tamanho do conjunto de itens candidatos ($|S|$). Dessa forma, o custo é calculado reduzindo-se um item candidato a cada iteração, até que não haja mais torneios. Isso ocorre quando não há mais itens a serem combinados, mais formalmente $\binom{|S|-i}{2} = 0$, tal que i é a quantidade de itens candidatos já eliminados do conjunto S. Essa condição é satisfeita quando há apenas um item no conjunto S, i.e., $i = |S| - 1$. O custo desse algoritmo na versão original denominado por t^o é calculado pela Equação 1.

$$t^o = \sum_{i=0}^{|S|-1} \binom{|S| - i}{2} \tag{1}$$

No melhor caso, a versão do algoritmo TM que utiliza eliminação múltipla (TM2) obtém o item máximo com apenas uma combinação de todos os itens do conjunto S. Nessa combinação, o item máximo é escolhido como o melhor em todas as comparações das quais participa (i.e., $|S| - 1$ comparações) e os demais itens são definidos como incompatíveis com os requisitos quando comparados entre si. Neste caso após o primeiro torneio, o novo conjunto de candidatos S terá apenas um candidato, o item máximo. O

custo em termos do número de tarefas (comparações) necessárias é definido como t_m^a e pode ser calculado pela Equação 2. O ganho decorrente do uso da estratégia de eliminação múltipla é calculado como a diferença entre o número de tarefas geradas com a versão original e com a versão adaptada para o uso da opção de eliminação múltipla, i.e., $t^o - t_m^a$.

$$t_m^a = \binom{|S|}{2} \tag{2}$$

Considerando-se o uso da eliminação múltipla no pior caso do algoritmo, remove-se apenas 2 itens a cada iteração, exatamente os itens selecionados pela eliminação múltipla, dado que neste caso a opção de eliminação múltipla é utilizada apenas em uma comparação. Dessa forma, o número de tarefas geradas no pior caso é calculado reduzindo-se dois candidatos a cada torneio até que não haja mais torneios, i.e., $\binom{|S|-2\times i}{2} = 0$, onde i é a quantidade de torneios realizados. Essa condição é satisfeita quando é atingida a quantidade de torneios necessários para que todos os itens não máximos sejam removidos de S com a utilização da eliminação múltipla, que é expresso por $i = \lfloor \frac{|S|}{2} \rfloor$. Portanto o número de tarefas geradas utilizando a opção de eliminação múltipla no pior caso t_p^a é calculado pela Equação 3. Nesse caso, o ganho com o uso da estratégia de eliminação múltipla é calculado como $t^o - t_p^a$.

$$t_p^a = \sum_{i=0}^{\lfloor \frac{|S|}{2} \rfloor} \binom{|S| - 2 \times i}{2} \tag{3}$$

2-Max. O algoritmo 2-Max otimiza a busca pelo item máximo subdividindo o conjunto S em subconjuntos de itens escolhidos de forma aleatória. O tamanho dos subconjuntos é constante e definido como $w = \lceil \sqrt{|S|} \rceil$. Esse algoritmo realiza uma série de torneios e a cada torneio combina-se dois a dois todos os itens do subconjunto previamente escolhido. Cada combinação representa uma tarefa a ser executada por um trabalhador, então, neste caso o custo é de $\binom{w}{2}$ tarefas. Com as respostas para essas tarefas, obtém-se o item máximo $s \in S$ usando voto majoritário. Em seguida, combina-se s com todos os demais itens em S. Cada combinação é uma tarefa de comparação a ser executada por um trabalhador, o que resulta em $|S|-1$ tarefas. Para cada tarefa executada nessa etapa o candidato que não vencer s é eliminado do conjunto original S. Os torneios terminam quando o conjunto de candidatos S se reduz a um subconjunto ($|S| \leq w$). Após isso, combinam-se os itens remanescentes em S, gerando as últimas $\binom{w}{2}$ tarefas. O item máximo é aquele que obtiver o maior número de vitórias nessas últimas tarefas.

No pior caso, o algoritmo 2-Max elimina apenas um candidato por torneio na etapa em que se compara s com os demais candidatos em S. Isso gera $|S| - i$ tarefas, onde i é o número do torneio atual. Como os torneios terminam quando $|S| \leq w$, temos que o algoritmo 2-Max, no pior caso, termina quando $i > |S| - w$. O custo do 2-Max no pior caso é definido como d_p^o e calculado pela Equação 4. Já no melhor caso, após se obter s em S todos os candidatos são eliminados de S logo no primeiro torneio, pois não vencem s na etapa que se compara s com todos os candidatos em S. Dessa forma o custo no melhor caso denotado por d_m^o é de apenas um torneio, calculado pela Equação 5.

$$d_p^o = \sum_{i=1}^{|S|-w} \left(\binom{w}{2} + (|S| - i) \right) + \binom{w}{2} \tag{4}$$

$$d_m^o = \binom{w}{2} + |S| - 1 \tag{5}$$

Para utilizar a opção de múltipla eliminação no algoritmo 2-Max, propomos uma versão modificada (2-Max2). Nessa versão, após a etapa de combinação dos itens do subconjunto w, verifica-se se algum item não foi escolhido como item máximo e foi marcado pela opção de eliminação múltipla. Caso isso ocorra, esse item é removido do conjunto de candidatos S. Com isso, o algoritmo 2-Max2 permite a eliminação de 2 itens candidatos em uma única comparação. Isso permite que menos itens candidatos sejam promovidos acelerando a obtenção do item máximo.

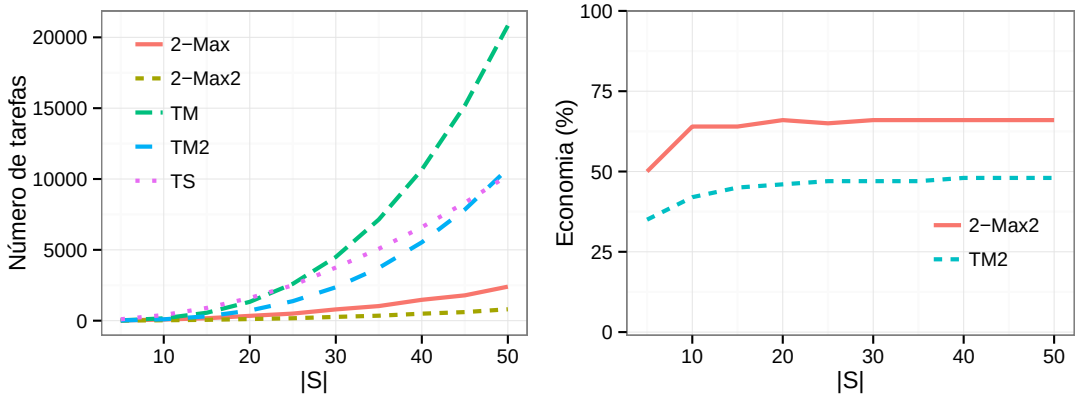
No pior caso, o algoritmo 2-Max2 gera a eliminação de 2 itens candidatos em que a opção de eliminação múltipla é utilizada apenas uma vez. Isso permite que menos itens candidatos sejam promovidos para a próxima etapa do torneio, em que s é comparado com todos os candidatos em $|S| - 2$. Dessa forma com a diminuição de dois candidatos mais a diminuição de 1 candidato por torneio oriunda do 2-Max original, temos a cada torneio $|S| - 3 \times i$ candidatos. Como os torneios terminam com $|S| \leq w$, temos que o 2-Max2 termina quando $i > \frac{|S|-w}{3}$, onde i é o número do torneio. Então, no pior caso, o custo do 2-Max2 pode ser calculado pela Equação 6. No melhor caso, na etapa de combinação dos itens do subconjunto de tamanho w, apenas um item é escolhido como máximo e todos os outros itens são marcados pela eliminação múltipla, então todos os w itens menos o item máximo s são removidos de S, i.e., $w - 1$. Em seguida s é comparado com todos os itens de $|S| - 1 - (w - 1)$. O custo do 2-Max2 no melhor caso é então calculado pela Equação 7. O ganho com o uso da estratégia de eliminação múltipla no melhor caso é calculado como $d_m^o - d_m^a$ e o ganho no pior caso é calculado como $d_p^o - d_p^a$.

$$d_p^a = \sum_{i=1}^{\frac{|S|-w}{3}} \left(\binom{w}{2} + (|S| - 3 \times i) \right) + \binom{w}{2} \tag{6}$$

$$d_m^a = \binom{w}{2} + (|S| - 1 - (w - 1)) \tag{7}$$

Visando analisar o ganho gerado pela opção de eliminação múltipla no pior e no melhor caso, analisamos o número de tarefas geradas e a economia obtida pela estratégia considerando conjuntos de diferentes tamanhos (Figuras 1 e 2). A Figura 1 apresenta os resultados do pior caso e a Figura 2 apresenta os resultados do melhor caso. Essas figuras mostram o número de tarefas geradas e a economia obtida quando se utiliza os algoritmos *2-Max* e TM e as versões adaptadas para permitir eliminação múltipla, 2-Max2 e TM2, respectivamente. Apenas para comparação, essas figuras também apresentam os número de tarefas geradas pelo *tournament selection* (TS). Como não há uma definição dos valores dos parâmetros r e n no algoritmo *tournament selection*, esses valores foram calculados de forma proporcional aos valores utilizados por Sun et al. [Sun et al. 2011] em relação ao tamanho do conjunto de itens candidatos.

No pior caso (Figura 1) os resultados mostram que, para $|S| \leq 25$, o algoritmo menos eficiente é o TS, no qual a estratégia de eliminação múltipla não gera ganho. Para $|S| > 25$ o algoritmo TM firma-se como o algoritmo menos eficiente em termos do número de tarefas e o algoritmo 2-Max2 é o mais eficiente. A Figura 1(b) mostra o ganho gerado pela versão adaptada em comparação à versão original. O algoritmo no qual se obtém maior economia é o algoritmo 2-Max, seguido pelo algoritmo TM. Desta forma o algoritmo 2-Max2 firma-se como o melhor tanto em termos de número de tarefas, quanto no ganho gerado em relação à versão original.



(a) Número de tarefas geradas no pior caso. (b) Percentual de economia em número de tarefas comparado à versão original no pior caso.

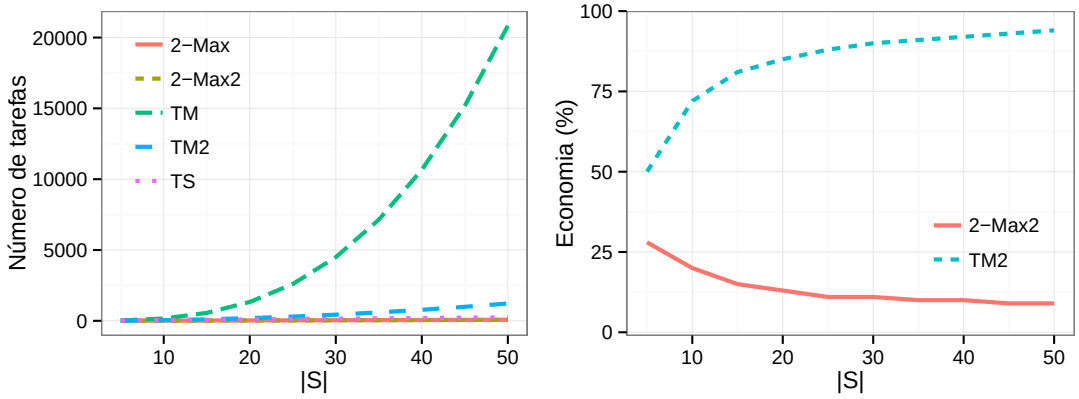
Figura 1. Economia em termos do número de tarefas gerada pela opção de eliminação múltipla no pior caso.

No melhor caso (Figura 2) os resultados mostram que, para $|S| > 10$, o algoritmo menos eficiente é o TM e o mais eficiente é o 2-Max2. A Figura 2(b) mostra o ganho gerado pela versão adaptada em comparação à versão original. O algoritmo adaptado com a opção de eliminação múltipla que se obtém maior economia é o TM, seguido pelo 2-Max.

4. Analisando a acurácia em comparações com eliminação múltipla

A partir dos resultados na seção anterior, identificamos a eliminação múltipla como estratégia promissora para otimizar o número de comparações na busca pelo item máximo em Computação por Humanos. Contudo, a aplicabilidade desta estratégia ainda depende de seu impacto na acurácia dos resultados da computação. Caso, por exemplo, trabalhadores julguem erroneamente que a eliminação múltipla deve ser aplicada com muita frequência, o efeito desta estratégia na aplicação pode ser negativo.

Nesta seção conduzimos um estudo de caso com uma aplicação de Computação por Humanos no qual investigamos o impacto da estratégia de eliminação múltipla na acurácia dos resultados da escolha do item máximo. Primeiro apresentamos o sistema de computação por humanos desenvolvido para conduzir o estudo. Em seguida, discutimos a acurácia dos resultados da escolha do item máximo quando se utiliza a opção de eliminação múltipla.



(a) Número de tarefas geradas no melhor caso. (b) Percentual de economia em número de tarefas comparado à versão original no melhor caso.

Figura 2. Economia em termos do número de tarefas gerada pela opção de eliminação múltipla no melhor caso.

4.1. Sistema de Computação por Humanos

Para avaliar a acurácia do item máximo obtido com comparações entre pares com exclusão múltipla, foi implementada uma aplicação de Computação por Humanos em que trabalhadores voluntários executam a avaliação dos itens. Essa aplicação foi implementada utilizando o PyBossa[4]. PyBossa é um middleware de código aberto para o desenvolvimento de aplicações de Computação por Humanos que lida com aspectos como identidade dos trabalhadores, fila de tarefas a serem executadas e escalonamento das tarefas para os trabalhadores. O PyBossa é escrito na linguagem de programação Python e é inspirado no sistema de middleware de Computação por Humanos *BOINC Open System for Skill Aggregation* (BOSSA) [Korpela 2012] que, por sua vez, é inspirado no sistema de middleware de computação voluntária *The Berkeley Open Infrastructure for Network Computing* (BOINC).

A aplicação desenvolvida consiste em identificar o item máximo em 5 itens candidatos. Os itens candidatos consistem em diferentes formas de reconhecimento das linhas e colunas de uma tabela existente em uma página digitalizada de um livro. Esses itens foram gerados com diferentes configurações de um algoritmo de visão computacional. Em alguns casos os itens são completamente diferentes entre si e em outros casos são mais semelhantes. Em uma comparação (a, b), o critério de escolha do reconhecimento máximo deve considerar as linhas em vermelho de cada reconhecimento que idealmente devem estar posicionadas entre as fronteiras que representam a separação entre linhas e colunas na tabela original. Geralmente, em um reconhecimento não máximo, as linhas vermelhas não estão posicionadas corretamente, ou estão faltando.

Essa aplicação é composta por diversas tarefas que são executadas em paralelo por trabalhadores diferentes. Cada tarefa é definida da seguinte forma: são apresentados os critérios que definem um reconhecimento máximo e dois reconhecimentos candidatos a serem avaliados. O trabalhador deve, então, fornecer como resultado da tarefa o reconhecimento que melhor atende os requisitos para ser um reconhecimento máximo

[4] `www.pybossa.com`

ou nenhum dos reconhecimentos candidatos se ambos os reconhecimentos não atendem os requisitos para ser definido como um reconhecimento máximo (eliminação múltipla). A Figura 3 apresenta um exemplo de um reconhecimento que pode ser considerado máximo (Figura 3(a)) e de um reconhecimento que não pode ser considerado máximo (Figura 3(b)). Esses exemplos são apresentados no tutorial que auxilia o trabalhador a executar as tarefas.

3	Salvador	BA	2.211.539	Centros Regionais		UF
4	Belo Horizonte	MG	2.091.448			
5	Fortaleza	CE	1.965.513	1	São Luis	MA
6	Brasília	DF	1.821.946	2	Maceió	AL
7	Curitiba	PR	1.476.253	3	Natal	RN
8	Recife	PE	1.346.045	4	Teresina	PI
9	Porto Alegre	RS	1.288.879	5	Campo Grande	MS
10	Belém	PA	1.144.312	6	João Pessoa	PB
11	Goiânia	GO	1.004.098	7	São José dos Campos	SP
12	Campinas	SP	908.906	8	Ribeirão Preto	SP
13	São Luís	MA	780.833	9	Cuiabá	MT
14	Maceió	AL	723.230	10	Aracaju	SE
15	Natal	RN	656.037	11	Londrina	PR

(a) Exemplo de reconhecimento que atende aos requisitos para ser um reconhecimento máximo

3	Salvador	BA	2.211.539	Centros Regionais		UF
4	Belo Horizonte	MG	2.091.448			
5	Fortaleza	CE	1.965.513	1	São Luis	MA
6	Brasília	DF	1.821.946	2	Maceió	AL
7	Curitiba	PR	1.476.253	3	Natal	RN
8	Recife	PE	1.346.045	4	Teresina	PI
9	Porto Alegre	RS	1.288.879	5	Campo Grande	MS
10	Belém	PA	1.144.312	6	João Pessoa	PB
11	Goiânia	GO	1.004.098	7	São José dos Campos	SP
12	Campinas	SP	908.906	8	Ribeirão Preto	SP
13	São Luís	MA	780.833	9	Cuiabá	MT
14	Maceió	AL	723.230	10	Aracaju	SE
15	Natal	RN	656.037	11	Londrina	PR

(b) Exemplo de reconhecimento que não atende aos requisitos para ser um reconhecimento máximo

Figura 3. Exemplo de reconhecimento que atende (a) e que não atende (b) os requisitos para ser um reconhecimento máximo.

Os trabalhadores que executaram as tarefas são alunos do curso de Ciência da Computação da Universidade Federal de Campina Grande convocados por e-mail a contribuir. Cada trabalhador, ao acessar o sistema pela primeira vez, é exposto a um tutorial que descreve a aplicação e exemplifica resultados desejáveis de avaliações de itens. No total, 108 trabalhadores executaram 2.397 tarefas. Essas tarefas foram geradas a partir de 52 tabelas diferentes e 5 itens candidatos por tabela, o que resulta em uma combinação de 10 pares por tabela e 520 tarefas diferentes. Em nosso experimento, o ambiente foi configurado de modo que cada comparação é executada por no mínimo 2 trabalhadores diferentes e um mesmo trabalhador não avalia mais de uma vez um mesmo par de itens candidatos. A Figura 4 apresenta a distribuição do número de tarefas executadas pelos trabalhadores.

4.2. Convergência e acurácia

A Figura 5 apresenta o percentual de trabalhadores que utilizaram a opção de eliminação múltipla em cada uma das 520 comparações de dois itens candidatos que foram executadas pelos trabalhadores. Nessa figura, as comparações encontram-se ordenadas de forma

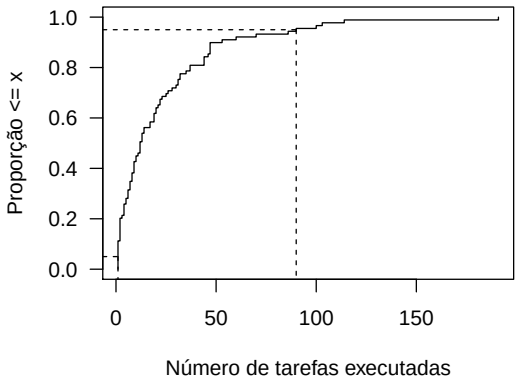


Figura 4. Função de distribuição acumulada do número de tarefas executadas pelos trabalhadores.

decrescente pelo percentual de uso da eliminação múltipla. Em 73 dessas comparações entre dois itens candidatos, 100% dos trabalhadores que executaram essas comparações optaram pela eliminação das duas imagens candidatas. Isso indica que quando a diferença entre as alternativas não é significativa e ambos os candidatos não atendem aos requisitos para serem um item máximo, os trabalhadores convergem na escolha da eliminação múltipla. De outro modo, quando há diferença entre os itens candidatos, os trabalhadores escolhem algum item resultando que 0% dos trabalhadores utilizam a opção de eliminação múltipla nessas comparações. Em nosso experimento, isso ocorre em 167 comparações de duas imagens candidatas. Nos demais casos, pelo menos um trabalhador divergiu dos demais. Nesses casos, o voto majoritário é usado para definir o item máximo entre os dois itens candidatos ou a eliminação de ambos.

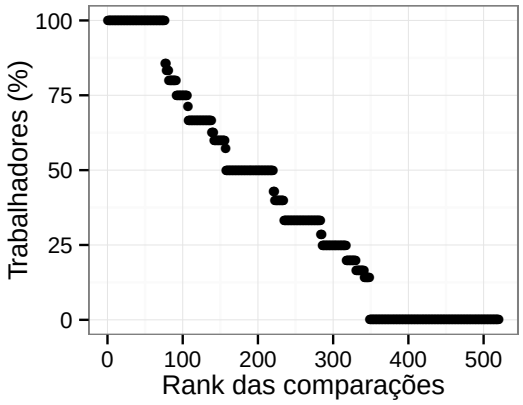


Figura 5. Proporção de uso da opção de eliminação múltipla em cada comparação de 2 itens candidatos.

Para definir a acurácia das respostas geradas pelos trabalhadores utilizamos as respostas corretas definidas por um especialista. O especialista a quem recorremos é o autor do algoritmo de visão computacional usado para gerar as tarefas, que por sua

vez não participou do desenvolvimento deste artigo. Nossa análise é conduzida de duas formas: (i) se em uma dada comparação os trabalhadores escolhem a mesma opção que um especialista escolhe e (ii) se no conjunto de 5 imagens, a imagem escolhida como o item máximo é a mesma escolhida pelo especialista. O resultado obtido é que, em média, os trabalhadores acertam 87.73% das comparações que executam, com um erro de $\pm 6.14\%$, para um nível de confiança de 95%. Ao processar o voto majoritário para se obter o item candidato que não foi eliminado e que foi escolhido como o item máximo o maior número vezes nas comparações com os demais itens candidatos, obteve-se que esse item máximo obtido em cada conjunto de 5 itens candidatos é o mesmo escolhido pelo especialista.

5. Conclusões e trabalhos futuros

Neste trabalho conduzimos um estudo de eficiência em aplicações que visam obter um item máximo em um conjunto de itens candidatos em Sistema de Computação por Humanos. Avaliamos dois algoritmos no estado-da-arte que permitem selecionar um único candidato a cada comparação (*2-Max* e *tournament max*) e propusemos uma estratégia de eliminação múltipla que tem como propósito permitir que esses algoritmos obtenham o item máximo em menos comparações. Nosso método é baseado em um estudo analítico e em um estudo experimental.

Nosso estudo analítico mostra que a versão adaptada dos algoritmos 2-Max e tournament max, que implementa a eliminação múltipla, permite reduzir o número de tarefas geradas por ambos os algoritmos em sua versão original. Entretanto, com o algoritmo 2-Max adaptado obtém-se o maior ganho em termos do número de tarefas gerada e da economia de tarefas em relação à versão original do algoritmo. Nossos resultados experimentais mostram que os trabalhadores convergem para a escolha do item máximo em conjuntos de itens candidatos com diferentes características. Isso indica que a estratégia de otimização proposta não tem efeito negativo na acurácia dos resultados gerados.

Este trabalho pode ser estendido em diversas perspectivas, como: escalonamento de tarefas, tolerância a falhas e qualidade dos resultados. Estratégias de escalonamento podem ser propostas com o objetivo de aumentar a eficiência em termos, por exemplo, do tempo de resposta das tarefas. No que se refere às estratégias de tolerância a falhas, nosso estudo pode ser estendido para identificar se a probabilidade de erro varia em aplicações que se referem a outros tipos de itens, como: fotos, frases e vídeos. Por fim, com relação qualidade dos resultados, pode-se haver um compromisso entre a velocidade em que o item máximo é encontrado e o intervalo de confiança da precisão dos resultados.

Finalmente, os resultados obtidos neste trabalho serão utilizados no contexto do projeto Memória Estatística (www.memoria.org.br). Esse projeto utilizará Computação por Humanos para selecionar e transcrever para arquivos editáveis, publicações históricas relevantes para a história estatística do Brasil presentes na Biblioteca do Ministério da Fazenda.

Agradecimentos. Os autores agradecem ao IPEA pelas imagens utilizadas no experimento, a Guilherme Gadelha pelo algoritmo de reconhecimento das tabelas e pela atuação como especialista na avaliação dos resultados. Lesandro Ponciano é bolsista CAPES/Brasil e Francisco Brasileiro é pesquisador do CNPq/Brasil.

Referências

Ahn, L. (2005). *Human computation*. PhD thesis, Carnegie Mellon University. UMI Order Number: AAI3205378.

Ajtai, M., Feldman, V., Hassidim, A., and Nelson, J. (2009). Sorting and selection with imprecise comparisons. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 37–48, Berlin, Heidelberg. Springer-Verlag.

Borgstrom, R. S. and Kosaraju, S. R. (1993). Comparison-based search in the presence of errors. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 130–136, New York, NY, USA. ACM.

Dustdar, S. and Truong, H.-L. (2012). Virtualizing software and humans for elastic processes in multiple cloudsâ a service management perspective. *International Journal of Next-Generation Computing*, 3(2).

Eickhoff, C. and de Vries, A. (2011). How Crowdsourcable is Your Task? In Lease, M., Carvalho, V., and Yilmaz, E., editors, *Proceedings of the Workshop on Crowdsourcing for Search and Data Mining (CSDM) at the Fourth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 11–14, Hong Kong, China.

Eugene Wu, David R. Karger, S. M. R. C. M. (2011). Platform considerations in human computation. In *CHI 2011 Workshop on Crowdsourcing and Human Computation*, New York, NY, USA. ACM.

Korpela, E. J. (2012). Seti@home, boinc, and volunteer distributed computing. *Annual Review of Earth and Planetary Sciences*, 40(1):69–87.

Little, G. (2011). *Programming with Human Computation*. PhD thesis, Massachusetts Institute of Technology.

Quinn, A. J. and Bederson, B. B. (2009). A taxonomy of distributed human computation. Technical report.

Quinn, A. J. and Bederson, B. B. (2011). Human computation: a survey and taxonomy of a growing field. In *Proceedings of the 2011 annual conference on Human factors in computing system*, CHI '11, pages 1403–1412, Vancourver/BC/Canada. ACM.

Schall, D., Truong, H.-L., and Dustdar, S. (2008). Unifying human and software services in web-scale collaborations. *Internet Computing, IEEE*, 12(3):62 –68.

Spiekermann, K. (2010). Judgement aggregation and distributed thinking. *AI Society.*, 25(4):401–412.

Sun, Y.-A., Dance, C. R., Roy, S., and Little, G. (2011). How to assure the quality of human computation tasks when majority voting fails? In *Proceedings of the Workshop on Computational Social Science and the Wisdom of Crowds (NIPS)*.

Sweller, J., Merrienboer, J. J. G. V., and Paas, F. G. W. C. (1998). Cognitive architecture and instructional design. *Educational Psychology Review*, 10:251–296.

Venetis, P., Garcia-Molina, H., Huang, K., and Polyzotis, N. (2012). Max algorithms in crowdsourcing environments. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 989–998, New York, NY, USA. ACM.

Análise e Contra-Ataque à Poluição e *Whitewashing* em Sistemas P2P de Vídeo ao Vivo.

Rafael B. de Almeida[1], José Augusto M. Nacif[2], Ana Paula C. da Silva[3], Alex B. Vieira[1]

[1]DCC Universidade Federal de Juiz de Fora – Juiz de Fora – MG
[2]Universidade Federal de Viçosa – MG [3]Universidade Federal de Minas Gerais – MG

rafael.barra@ice.ufjf.br, jnacif@ufv.br
ana.coutosilva@dcc.ufmg.br, alex.borges@ufjf.edu.br

Abstract. *In this paper, we analyze the content pollution attack and whitewashing in a P2P live streaming system. Moreover, we propose and implement a distributed reputation system to fight these attacks. Our analytical results show that pollution attacks are harmful, even in a system with a small number of malicious peers. In this case, the peers in the P2P live streaming system may experience up to 400% network overhead. The proposed reputation system quickly identifies and blocks polluters. Our PlanetLab experiments show that, during a content pollution and whitewashing attack, the new reputation mechanism drops the system network overhead to 20% of the streaming rate and the chunk miss rate to values lower than 3%.*

Resumo. *Este artigo analisa o impacto causado por ataques de poluição e whitewashing a sistemas P2P de vídeo ao vivo. Um mecanismo simples e descentralizado de reputação é proposto e implementado em um ambiente de rede real. Os modelos analíticos criados mostram que ataques de poluição são prejudiciais, mesmo em um sistema com poucos poluidores. Nesse caso, um sistema P2P de transmissão ao vivo pode sofrer uma sobrecarga de até 400% em relação à mídia original. O novo mecanismo de reputação bloqueia ataques de poluição e whitewashing rapidamente. No caso de ataque combinado de poluição e whitewashing, o mecanismo de reputação diminui a sobrecarga no sistema para 20% e a perda de chunks para menos de 3%.*

1. Introdução

Nos últimos anos, aplicações de vídeo ao vivo em arquiteturas P2P tem atraído a atenção tanto da academia quanto da indústria. Diversos sistemas comerciais populares, como SopCast[1] e PPLive[2], sustentam milhões de usuários registrados e, a cada dia, esse número cresce. De fato, eventos globais, como a posse do presidente Obama, foram transmitidos com sucesso pela Internet com a ajuda de arquiteturas P2P [3].

Apesar de diversos estudos recentes, os sistemas de transmissão ao vivo em P2P são suscetíveis a comportamentos maliciosos. De fato, a maioria dos sistemas populares tem suas mensagens (dados ou controle) transmitidas sem nenhuma proteção ou criptografia [Hei et al. 2007, Vieira et al. 2013]. Assim, durante as trocas de dados,

[1]www.sopcast.com
[2]www.pplive.com
[3]*"CNN: Inauguration P2P Stream a Success, Despite Backlash"*, The NY Times, fev. de 2009.

participantes (*peers*) maliciosos podem tirar vantagem das falhas existentes para que, de alguma forma, causem dano ao sistema P2P e a seus participantes.

Durante um ataque de poluição, *peers* maliciosos (poluidores) injetam ou forjam dados falsos no sistema P2P. Os demais participantes podem requisitar dados aos poluidores e assim, assistir um trecho de mídia falso. Além disso, os poluidores podem tentar burlar mecanismos de defesa praticando *whitewashing*, ou seja, trocando suas identidades constantemente [Feldman et al. 2006, Kudtarkar and Umamaheswari 2009]. Dada a facilidade de se obter uma nova identidade, *whitewashing* é um desafio, principalmente sob as fortes restrições temporais existentes nas aplicações de vídeo ao vivo. Mais ainda, mecanismos de defesa existentes não conseguem lidar com ataques de *whitewashing* [de Almeida et al. 2012].

Assim, no presente artigo propõe-se um modelo para avaliar os danos causados por ataques de poluição. Também é criado um sistema de reputação distribuído para que os *peers* identifiquem e isolem os poluidores, mesmo quando estes praticam *whitewashing*. Nessa linha, o modelo analítico desenvolvido mostra que poluição é, de fato, um problema. Mesmo em sistemas com um número baixo de poluidores, a sobrecarga na banda de rede, imposta pelas retransmissões de dados, chega a 400%.

Da mesma forma, os resultados experimentais conduzidos no PlanetLab mostram que identificar os dados poluídos e pedir retransmissão é ineficiente. Nesse caso, como os *peers* pedem retransmissão, a sobrecarga média na banda de rede chega a 230%. Nesse cenário, os *peers* também apresentam uma alta taxa de perda no tempo de execução, o que indica que os usuários não assistem um vídeo com qualidade adequada.

O sistema de reputação proposto apresenta valores de sobrecarga abaixo de 5% quando os poluidores não realizam *whitewashing*. Quando há ataque de *whitewashing*, a sobrecarga aumenta para cerca de 20%. Em ambos os casos, a taxa de perda de tempo de execução é baixa, com valores por volta de 3% em momentos de pico. Finalmente, os resultados mostram que os *peers* com problemas temporários, identificados como poluidores, tem oportunidade de se redimir e voltar a contribuir com o sistema P2P.

2. Trabalhos Relacionados

Ataques de poluição de dados são, de fato, um problema em sistemas de transmissão ao vivo em P2P. Caso o sistema P2P não adote medidas, mesmo ataques simples levam a uma alta sobrecarga dos participantes do sistema [Dhungel et al. 2007, Vieira et al. 2008, Haizhou et al. 2011, de Almeida et al. 2012]. Nesse sentido, várias propostas foram criadas para combater esses ataques. Inicialmente, tais propostas baseiam-se na identificação do conteúdo poluído e no pedido de retransmissão desse dado [Dhungel et al. 2007, Haridasan and Renesse 2008].

Nessa linha, [Hu and Zhao 2010] propõem um esquema que detecta ataques de poluição em sistemas P2P de vídeo ao vivo. Eles verificam o conteúdo trocado entre os participantes e tentam detectar poluição o mais rápido possível. Para identificar os poluidores, é proposto um gerenciamento de confiança e assim, o isolamento do poluidor depende do consenso da rede P2P. Embora esse esquema auxilie na redução de sobrecarga no sistema, a convergência do consenso da rede sobre a reputação de um *peer* é demorado e assim, poluidores podem causar problemas por um longo período.

Os mecanismos de reputação propostos, geralmente, usam 2 componentes para avaliar um *peer*. Cada *peer* usa sua experiência individual e o consenso da rede [Vieira et al. 2008, Seibert et al. 2010]. Além da demora na convergência do testemunho da rede P2P, tal sistema de reputação pode sofrer com conluio e falsos testemunhos [Vieira et al. 2009, So and Reeves 2012]. O tratamento pode ser custoso, com necessidade de mecanismos de autenticação centralizado/distribuído [So and Reeves 2012].

Em alguns casos, os *peers* recém-chegados ao sistema são marcados como suspeitos [Seibert et al. 2010]: uma tentativa de reduzir os recursos disponíveis para estes *peers*. Com menos recursos disponíveis, ao se juntar ao sistema, é esperado que a troca de identidade (*whitewashing*) seja desencorajada. Uma das medidas mais comuns para tratar *whitewashing* é a utilização de uma entidade centralizada responsável por identificar, unicamente, participantes recém chegados ao sistema [Oualha and Roudier 2009]. Porém, a desvantagem dessa abordagem é a centralização de informações em um sistema de natureza distribuída, o que pode afetar a escalabilidade [Feldman et al. 2006].

O mecanismo de reputação proposto neste trabalho bane poluidores e combate a prática de *whitewashing*. O sistema proposto é simples e dispensa a necessidade de recursos centralizados. Além disso, o mecanismo proposto se diferencia dos descritos anteriormente por não prejudicar nenhum *peer* recém-chegado ao sistema. Um *peer* novato não precisa ser marcado como suspeito e ter seus recursos limitados. Finalmente, os mecanismos propostos nesse trabalho permitem a reabilitação de participantes que tenham sido classificados como atacantes, por causa de problemas temporários.

3. Impacto do Ataque de Poluição a Sistemas P2P de Transmissão ao Vivo

Os sistemas de transmissão ao vivo em P2P mais populares são baseados em malha com pedidos explícitos por dados (mesh-pull) [Hei et al. 2008]. Esses sistemas apresentam um total de m participantes que colaboram entre si para realizar a transmissão do conteúdo. Um *peer* especial (*servidor*) codifica o vídeo e inicia a transmissão. Os dados a serem transmitidos são particionados (*chunks*) e identificados de forma única.

Cada *peer* p_i possui uma lista com n_i parceiros. Além desta lista, p_i mantém um *buffer* B_i para armazenar *chunks* de vídeo antes de serem executados/compartilhados. O tamanho do *buffer* é delimitado por cada *peer*. Posições disponíveis no *buffer* são inicializadas como vazias. Os participantes do sistema mapeiam seus respectivos *buffers*, criando um mapa de chunks, para posterior sinalização de dados disponíveis e desejados.

O *buffer* B_i é implementado com o mecanismo de janela deslizante. Por simplificação, neste trabalho, considera-se que $B_i[s]$ armazena o dado menos recente, necessário para a execução do vídeo, enquanto $B_i[0]$ armazena o *chunk* que será iminentemente consumido pela aplicação. Periodicamente, os *peers* trocam entre si mapas de *chunks*. Assim, p_i possui o conhecimento dos *chunks* disponíveis nos *buffers* dos seus parceiros e novas requisições de *chunks* podem ser agendadas.

Existem duas políticas para requisição de *chunks* que se destacam: a política dependente da disponibilidade do *chunk* na rede, conhecida como *rarest first* (RF); a política dependente do tempo de exibição do *chunk*, conhecida como *earliest deadline first*, EDF. No primeiro caso (RF), busca-se disseminar rapidamente o *chunk* mais recentemente gerado pelo servidor. No segundo caso (EDF), busca-se uma visualização

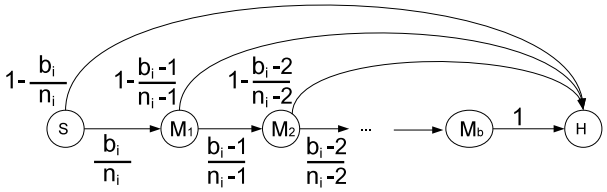


Figura 1. Requisição de um *chunk* até o sucesso.

contínua do vídeo. Neste artigo, é utilizada a política RF, dado que os *peers* tentam reproduzir o conteúdo mais recentemente gerado, preservando a característica de vídeo ao vivo. Consequentemente, a latência[4] de execução do conteúdo é baixa.

No cenário de um ataque de poluição, o sistema P2P possui b *peers* maliciosos, chamados de poluidores. Cada *peer* não poluidor, denominado *peer bom* possui b_i parceiros poluidores, com $0 \leq b_i < n_i$. Ao receber um *chunk* poluído, o *peer bom* p_i deve descartá-lo e pedi-lo novamente a um outro parceiro. Os *peers* possuem uma maneira eficiente para determinar a integridade de um *chunk* (e.g. [Haridasan and Renesse 2008]). O *chunk* danificado será requisitado por p_i até que o mesmo possa ser consumido. Sem perda de generalidade, assume-se que os poluidores estão uniformemente distribuídos entre as parcerias de todos os *peers*. Mais ainda, cada *peer* possui recursos suficientes para servir os seus parceiros. Finalmente, considera-se que os *chunks* de vídeo estão distribuídos uniformemente entre os *peers* do sistema.

Assim, a Figura 1 modela o processo de obtenção de um determinado *chunk* por um determinado peer p_i. A partir do estado inicial S, p_i escolhe um dos seus parceiros com a finalidade de requisitar dados. A escolha de um parceiro é uniforme, considerando que todos os seus parceiros possuem recursos e *chunks* equivalentes. Caso p_i escolha um *parceiro bom*, o *chunk* é obtido com sucesso (*data hit*), e assim, o processo de busca pelo *chunk* se encerra (estado H). Caso contrário, se o peer p_i escolhe um poluidor, o *chunk* obtido deverá ser descartado. O processo de escolha para requisição do *chunk* deverá ser repetido. Nesse caso, p_i não requisita o mesmo *chunk* para parceiros a quem ele já requisitou. Esta dinâmica se repete até que o dado seja obtido sem poluição.

Dado que p_i possui b_i parceiros poluidores, entre n_i parceiros, a probabilidade de se obter, na primeira tentativa, o *chunk* não poluído é igual a $1 - (b_i/n_i)$. Caso p_i receba um dado poluído, este remove o poluidor da sua lista de parceiros candidatos e repete o processo de requisição. A probabilidade de sucesso, na segunda tentativa é igual a $1 - \left({}^{b_i - 1}/_{n_i - 1}\right)$. No pior caso, p_i irá pedir o *chunk* h a todos os parceiros poluidores, antes de conseguir escolher um *peer bom*. A probabilidade de obter o *chunk* não poluído, após p_i requisitar o mesmo a todos os poluidores, é igual a $1 - \left({}^{b_i - b_i}/_{n_i - b_i}\right)$. Essa probabilidade é igual a 1, dado que todos os parceiros poluidores são descartados da lista de candidatos[5].

As retransmissões realizadas enquanto um *peer* p_i não obtém um *chunk* sem poluição impõe ao sistema P2P uma sobrecarga importante. A sobrecarga l, é definida como o número de *chunks* poluídos recebidos por p_i, até que p_i consiga um *chunk* bom

[4]Tempo médio decorrido entre a geração e a execução de um *chunk*.

[5]Como $0 \leq b_i < n_i$, p_i garantidamente consegue um *chunk* não poluído.

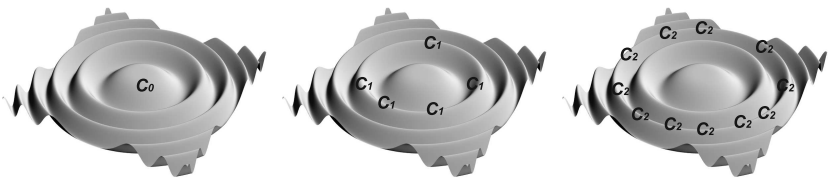
(estado H - Fig.1). Tal sobrecarga *média*, $E[l]$ pode ser calculada como segue:

$$\begin{aligned} E[l] &= \left[1-\left(\frac{b_i}{n_i}\right)\right]*1 \\ &+ \left[1-\left(\frac{b_i-1}{n_i-1}\right)\right]*\frac{b_i}{n_i}*2 \\ &\cdots \\ &+ \left[1-\left(\frac{b_i-b_i}{n_i-n_i}\right)\right]*\frac{b_i}{n_i}*\frac{b_i-1}{n_i-1}*.....*\frac{b_i-(b_i-1)}{n_i-(b_i-1)}*(b_i+1) \\ E[l] &= 1-\left(\frac{b_i}{n_i}\right)+\sum_{s=2}^{b_i+1}1-\left(\frac{b_i-(s-1)}{n_i-(s-1)}\right)*s*\prod_{j=0}^{s-2}\frac{b_i-j}{n_i-j} \end{aligned} \quad (1)$$

Até o momento, o modelo proposto considera que os *chunks* de vídeo estão distribuídos homogeneamente entre os *peers* do sistema P2P. No entanto, o impacto desse ataque é fortemente influenciado pela estratégia de seleção de *chunks* que um *peer* adota. Por exemplo, ao requisitar um *chunk* mais raro, p_i terá menos chance de encontrá-lo entre os seus parceiros. Como poluidores podem anunciar um mapa de *chunks* falso, p_i é forçado a escolher um parceiro que, na realidade, não possui o *chunk* raro verdadeiro.

Mais precisamente, em um sistema sem poluidores, o *chunk* mais raro é aquele criado mais recentemente pelo servidor. O servidor anuncia seu mapa de *chunks* e somente seus parceiros tem a possibilidade de conseguir o *chunk* da posição 0 do *buffer*. Quando o servidor cria um novo *chunk*, este desloca uma posição do *buffer*. O *chunk* que estava previamente na posição 0, vai para posição 1. Neste momento, alguns parceiros do servidor podem possuir este *chunk*, e então, este não é mais considerado o mais raro.

A Figura 2 apresenta o processo de difusão de *chunk* raro, sem a presença de poluidores. No instante inicial de observação, mostrado na Figura 2(a), somente o servidor tem o *chunk*. No próximo instante, Figura 2(b), alguns dos parceiros do servidor recebem esse *chunk*. Finalmente, a Figura 2(c) mostra quando esse *chunk* alcança o terceiro salto a partir do servidor. Nesse momento, o *chunk* está difundido entre diversos participantes do sistema, e *peers* que estão 4 saltos distantes do servidor, podem requisitá-lo aos seus parceiros. Até que o *chunk* alcance pelo menos um dos seus parceiros, p_i não poderá recebê-lo. No entanto, se o sistema P2P está sob ataque de poluição, poluidores podem anunciar dados falsos e assim, p_i poderá requisitá-lo. Até que o sistema alcance um estado em que um *chunk* específico esteja difundido, os *peers* estarão sob um severo ataque de poluição. Notadamente, o período que um *peer* fica sob ataque intenso dos poluidores é proporcional à distância média entre o servidor e os *peers* do sistema.



(a) (b) (c)

Figura 2. Difusão de *chunk* raro em sistemas sem ataque de poluição. (a) Servidor produzindo um *chunk*. (b) Alguns parceiros dos servidores recebendo o *chunk*. (c) *Peers* de terceiro salto recebendo *chunk*.

Em um cenário mais realista, somente uma porção dos parceiros bons de um *peer* podem responder por uma requisição. Nesse sentido, o modelo proposto terá o número

de participantes bons alterado para um número menor, definido por $w = y * (n_i - b_i)$; onde y é a proporção de parceiros que podem servir o *chunk*. Assim, a Eq. 2 apresenta a nova sobrecarga média do sistema, $E[l_n]$. Esta sobrecarga aumenta com o aumento do raio médio ρ da rede, dado que o modelo considera a política de requisição *rarest first*.

$$E[l_n] = \rho + 1 - \left(\frac{b_i}{w}\right) + \sum_{s=2}^{b_i+1} 1 - \left(\frac{b_i - (s-1)}{w - (s-1)}\right) * s * \prod_{j=0}^{s-2} \frac{b_i - j}{w - j}, \quad (2)$$

O modelo de sobrecarga proposto foi analisado usando a ferramenta de verificação de modelo probabilístico PRISM [Kwiatkowska et al. 2009]. Sessões SopCast de vídeo ao vivo foram coletadas para parametrizar o modelo proposto. Todos os parâmetros usados foram coletados durante transmissões de eventos reais ao vivo [Vieira et al. 2012]. A Tabela 1 apresenta os valores médios utilizados na análise. Por exemplo, foram avaliadas situações nas quais cada *peer* apresentava cerca de 50 parceiros e a distância média da rede é pequena (menor que 2 saltos em média). Tal cenário se enquadra bem na maioria das redes P2P dos canais da aplicação popular SopCast.

Parâmetro	Descrição	Valor
m	Total de *peers* no sistema	1000
n_i	Número médio de parceiros	$50 \sim 100$
b_i	Número médio de parceiros poluidores	$1 \sim 10$
y	Proporção de parceiros do tipo *bom peer*	$0{,}5 \sim 1$
ρ	Distância média servidor/*peer*	1,677

Tabela 1. Parâmetros usados na avaliação do modelo.

No cenário no qual os participantes têm 100 parceiros e 1 poluidor entre eles, a sobrecarga média é alta. Mesmo que todos os parceiros possam servir a uma requisição ($y = 1$), foi encontrado cerca de 167% de sobrecarga de dados devido a retransmissões impostas pela poluição. Quando o número de poluidores é aumentado para 10, a sobrecarga média aumenta para 177%. O valor da sobrecarga aumenta se o número de parceiros que um *peer* possui diminui. Quando são considerados 50 parceiros para cada *peer*, a sobrecarga aumenta para cerca de 170%. Se somente 50% desses parceiros podem atender às requisições realizadas, a sobrecarga média aumenta para 330%.

Em resumo, os resultados mostram que mesmo em um cenário no qual os *peers* tem um grande número de parceiros e que quase todos podem fornecer *chunks* sem poluição, os ataques de poluição impactam negativamente no funcionamento do sistema. Os participantes gastam até 3 vezes mais a largura de banda necessária para a obtenção de um *chunk*. Desta forma, medidas devem ser realizadas para amenizar o dano causado por ataques de poluição em um sistema P2P de vídeo ao vivo.

4. Mecanismo de Defesa Baseado em Reputação

Neste artigo, propõe-se um modelo de reputação local em que cada *peer* monitora a troca de dados com seus parceiros. Dessa forma, os participantes do sistema são capazes de associar um valor de reputação a cada um de seus parceiros. O objetivo é permitir que cada *peer* identifique e isole parceiros que disseminem conteúdo poluído.

Em uma abordagem de reputação distribuída tradicional, cada *peer* associa uma nota a cada parceiro. Essa nota é computada utilizando a experiência individual entre o *peer* e seu respectivo parceiro, e o depoimento da rede sobre este

parceiro [Vieira et al. 2008]. Entretanto, ainda não está claro se o uso do depoimento da rede resulta em um custo-benefício satisfatório para aplicações P2P de vídeo ao vivo.

Um *peer* em um sistema de transmissão ao vivo em P2P, tipicamente, apresenta muitas interações com seus parceiros, em curtos intervalos de tempo. Assim, o depoimento da rede sobre um determinado *peer* pode convergir muito lentamente, se comparado à taxa de interações entre 2 *peers*. Mais ainda, caso um determinado p_i queira calcular a reputação de p_j e não tenha muitos parceiros que também sejam parceiros de p_j, o depoimento da rede em relação a p_j pode não ser confiável.

Para evitar esses possíveis problemas, neste artigo é proposto um mecanismo de reputação descentralizado, baseado somente na experiência individual que cada *peer* tem em relação a seus parceiros. Este mecanismo será chamado *mecanismo de reputação simples*. Por esse novo mecanismo, cada *peer* p_i periodicamente calcula a reputação de cada parceiro p_j ($R_i[p_j]$). Mais precisamente, de acordo com a Equação 3, durante cada intervalo de tempo, p_i requisita *r chunks* a p_j. O parceiro p_j pode prover *n* respostas ruins para p_i (onde $0 \leq n \leq r$). Uma resposta é definida como *ruim* quando p_i é forçado a pedir o *chunk* novamente a outro parceiro. A razão n/r representa a qualidade da experiência de p_i em relação a p_j. Se essa razão n/r tem valor acima de um limite T_i^{max}, p_i diminui a reputação local de p_j. Caso contrário, o valor da reputação local de p_j é aumentada.

$$R_i[p_j] = \begin{cases} max(0, R_i[p_j] - \alpha_{p_i} * (1 + n/r)^{y_i}) & \text{Se } n/r > T_i^{max} \\ min(1, R_i[p_j] + \alpha_{g_i} * (1 - n/r)) & \text{Caso contrário} \end{cases} \quad (3)$$

Os parâmetros α_{p_i} e α_{g_i} são, respectivamente, fatores de penalidade e gratificação. Com o objetivo de rapidamente identificar e penalizar *peers* poluidores, considera-se $\alpha_{p_i} \geq \alpha_{g_i}$. Todos os *peers* recém-chegados no sistema recebem um valor inicial de reputação. Cada *peer* p_i tem um limite de reputação mínima R_i^{min}, com $0 \leq R_i^{min} \leq 1$. Caso $R_i[p_j]$ seja menor que R_i^{min}, p_i remove p_j de sua lista de parceiros.

Nesse esquema de reputação, uma vez que p_i considera um parceiro p_j como poluidor, p_j não terá mais oportunidade de trocar dados com p_i. No modelo de reputação clássico, o depoimento da rede poderia ajudar p_j a se redimir e voltar a trocar dados com seu parceiro p_i. Nesse sentido, para permitir reabilitação, é proposto um mecanismo que dinamicamente altera o limite mínimo de reputação R_i^{min}. O princípio para alterar R_i^{min} é fazer com que cada *peer* p_i reaja às condições da rede, percebidas através de suas medições locais. Por exemplo, caso p_i infira que a rede está sob ataque, este aumenta o valor de R_i^{min}, penalizando mais rapidamente os prováveis poluidores. Caso contrário, este diminui o valor de R_i^{min} para permitir a reabilitação das parcerias punidas.

Para mudança do valor de R_i^{min}, dois estados são considerados: (1) *calmaria* e; (2) *tempestade*. Na visão de p_i, o sistema está em *calmaria* se percebe um nível de poluição abaixo de um limite pré-definido. Caso contrário, na visão de p_i, o sistema se encontra no estado de *tempestade*. A cada intervalo de tempo, p_i verifica o estado do sistema e atualiza R_i^{min} de acordo com a Equação 4. Se o sistema está em *calmaria*, p_i diminui seu limite local R_i^{min} no fator de γ_{g_i}; caso contrário, R_i^{min} é aumentado no fator γ_{p_i}. Para que o sistema identifique rapidamente participantes poluidores, $\gamma_{p_i} > \gamma_{g_i}$. Os limites RT_i^{min} e RT_i^{max} são estabelecidos de tal maneira que $0 \leq RT_i^{min} \leq R_i^{min} \leq RT_i^{max} \leq 1$.

$$R_i^{min} = \begin{cases} max(RT_i^{max}, R_i^{min} + \gamma_{p_i}) & \text{Se o sistema está no } \textit{estado de tempestade} \\ min(RT_i^{min}, R_i^{min} - \gamma_{g_i}) & \text{Se o sistema está no } \textit{estado de calmaria} \end{cases} \quad (4)$$

O estado do sistema é definido na perspectiva local de cada *peer*, baseado somente nas experiências obtidas em relação a seus parceiros: caso p_i receba uma quantidade de dados poluídos de seus parceiros, a sua visão local é de que o sistema está sofrendo ataque de poluição. Além disso, as mudanças em R_i^{min}, de acordo com a Eq. 4, são realizadas independentemente das mudanças na reputação local de cada parceiro, definida na Eq. 3

Os poluidores podem trocar suas identidades frequentemente em conjunto com o ataque de poluição. Tais trocas de identidade, conhecida como *whitewashing*, tem por objetivo enganar o sistema de reputação. Ataques combinados podem causar grandes danos ao sistema P2P, e assim, propõe-se um *mecanismo de reputação modificado*.

Nesse sentido, propõe-se um *mecanismo de reputação modificado* no qual os participantes recém-chegados ao sistema recebem um baixo valor de reputação inicial. O valor estabelecido para o valor inicial é um valor próximo ao valor limite, para que troca de *chunks* aconteça ($R_i[p_j] \tilde{=} R_i^{min}$). Para qualquer tentativa de poluição, $R_i[p_j]$ ficará abaixo de R_i^{min} e então p_j será removido de sua lista de parceiros de p_i. Vale ressaltar que apesar da diminuição no valor de reputação na visão de p_i, *peers* recém-chegados podem trocar dados com os demais participantes do sistema. Para evitar punições de *peers bons* que tiveram problemas momentâneos (falso positivos), é proposto que os *peers* do sistema mantenham um pequeno histórico de suas parcerias. Assim, tão logo um *peer bom*, p_j, que saiu do sistema volte, este poderá ser pontuado com o seu valor de reputação anterior.

5. Metodologia e Ambiente de Experimentação

Os mecanismos de reputação propostos (*simples* e *modificado*) foram avaliados em um ambiente real, configurado no PlanetLab. Uma aplicação P2P de vídeo ao vivo baseada em malha e com pedidos explícitos por dados foi implementada. Nessa aplicação, foram incorporados os mecanismos de combate a ataques de poluição e *whitewashing*.

O servidor foi instalado em uma máquina dedicada na rede do campus, transmitindo 30 minutos de vídeo a uma taxa de 120 kbps. Foram utilizados 133 nós PlanetLab como *peers* do sistema P2P de transmissão de vídeo ao vivo. Desses *peers*, 120 são classificados como *peers bons* e 13 como poluidores (10% do total). Não foram impostas quaisquer restrições adicionais, tanto para o servidor quanto para os *peers*. Dessa forma, processamento e largura de banda são restringidas somente pela capacidade de cada máquina e pelas conexões reais existentes entre elas.

Durante os experimentos, todos os *peers* permanecem ativos até o fim da transmissão. Cada *peer* se conecta no máximo a 18 parceiros. Esse número foi definido após observação de comportamento de clientes no SopCast. Se um de seus parceiros falha ou deixa o sistema, um *peer* deve requisitar novos parceiros candidatos ao *bootstrap* do sistema. Além disso, a duração das parcerias foi definido como o comportamento encontrado no SopCast [Vieira et al. 2012].

Cada *chunk* enviado pelo poluidor tem em seu cabeçalho uma marcação para indicar que seu conteúdo é inválido (código *hash* dos dados da mídia). Ressalta-se porém que, qualquer mecanismo existente de verificação de dados (e.g., assinatura *hash* em cadeia [Dhungel et al. 2007, Haridasan and Renesse 2008]) pode ser usado para identificar poluição. A saber, a latência adicional para assinar os dados de uma mídia ao vivo em P2P é inferior a 2s no pior caso. Uma assinatura completa dos *chunks*,

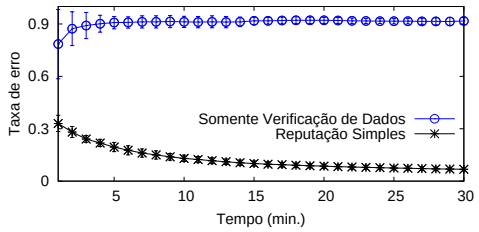


Figura 3. Taxa de erro de chunk.

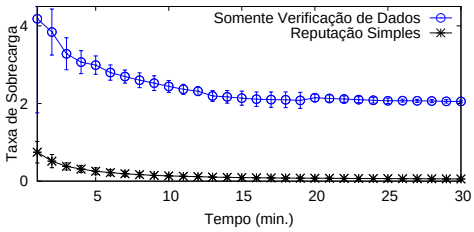


Figura 4. Sobrecarga no sistema.

pode gerar uma sobrecarga de até 30%. Porém, outras técnicas (como assinatura encadeada) apresentam uma sobrecarga menor, por volta de 5% em comparação ao fluxo de dados original [Dhungel et al. 2007, Haridasan and Renesse 2008]. Nos resultados apresentados foram desconsideradas as sobrecargas introduzidas por essas técnicas de verificação de dados. Portanto, os resultados focam na sobrecarga e atraso impostos somente pela retransmissão de *chunks* poluídos e dependem somente do tipo de mecanismo de defesa adotado.

Nos experimentos conduzidos, foram considerados 2 diferentes cenários. No primeiro, os poluidores atacam o sistema durante todo o experimento. No segundo, os poluidores atacam o sistema, porém saem e entram no sistema a cada 3 minutos aproximadamente, assumindo assim, uma nova identidade (ataque de *whitewashing*). Em ambos os cenários, os *peers* poluidores anunciam um mapa de *chunks* completo, forjando possuir todos os dados possíveis.

Não foi considerado *churn* (entrada e saída do sistema) dos *peers*. A dinâmica dos *peers* pode impactar na taxa de erro e no atraso percebido pelo usuário, porém, *churn* não afeta a eficiência dos mecanismos de reputação simplificado/modificado. Tais mecanismos não dependem da convergência da rede sobre o testemunho de um participante. Nos experimentos conduzidos, foi avaliada a dinâmica sobre o ponto de vista da troca de identidade (entrada e saída) somente dos pares maliciosos, o que de fato, pode impactar o desempenho dos sistemas de reputação.

6. Resultados Experimentais e Discussão

Nesta seção são apresentados os resultados obtidos através de experimentos realizados no PlanetLab. As métricas apresentadas são valores médios da realização de 5 experimentos, com 30 minutos de duração cada. A eficiência do sistema P2P sob ataque foi avaliado a partir de 3 métricas: a) taxa de *chunks* poluídos recebidos ou *taxa de erro*; b) a sobrecarga no sistema e; c) taxa de *chunks* perdidos ou taxa de perda. Essas métricas permitem capturar o dano causado ao sistema e também inferir a qualidade de experiência dos usuários.

A Fig. 3 apresenta a taxa de erros ao se requisitar *chunks*. A taxa de erros se refere a primeira requisição por um determinado *chunk* que um *peer* faz. Caso o *chunk* esteja poluído, o *peer* tem um erro, caso contrário, ele tem um acerto. Quanto maior a taxa de erro, maior o domínio de poluidores. Nessa figura são apresentadas a taxa de erros média em um sistema P2P utilizando um mecanismo de verificação de dados e o mecanismo de reputação. Nesse cenário, os poluidores não realizam *whitewashing*.

Nota-se que a taxa de erro de *chunks* é superior a 90% em um sistema sem

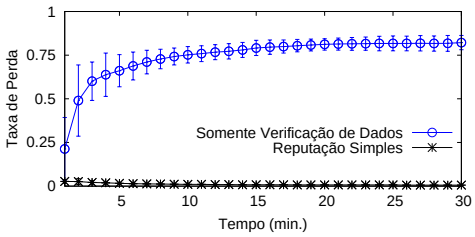


Figura 5. Taxa de perda.

reputação. Mesmo que seja verificada a integridade de um *chunk*, praticamente todos novos pedidos de dados geram uma requisição para um poluidor. Como consequência, a latência e a sobrecarga aumentam. O *mecanismo de reputação simples* proposto diminui a taxa de erro média para menos de 6%. Isso ocorre pois, ao se identificar um poluidor, o *peer* o isola rapidamente de sua lista de contatos.

A sobrecarga de retransmissão imposta por *chunks* poluídos é apresentada na Fig. 4. Observa-se que checar os dados e pedir retransmissão gera uma grande sobrecarga ao sistema P2P. Nesse caso, um *peer* apresenta em média 230% de sobrecarga por retransmissões, com pico de mais de 400%. Em outras palavras, os *peers* devem ter mais de 3 vezes a largura de banda necessária do que necessitariam em um sistema sem poluidores. O *mecanismo de reputação simples* apresenta uma sobrecarga desconsiderável. Em média, incluindo os picos de ataque, a sobrecarga é inferior a 5%.

Finalmente, a Fig. 5 mostra a taxa de perdas no tempo de execução. Uma taxa de perdas alta evidencia que o usuário está assistindo um vídeo ruim, sem a totalidade de seus quadros. Com uma janela de interesse de 20s, a taxa de perdas apresentada pela abordagem *verificar e pedir retransmissão* é elevada, acima de 75% dos *chunks* requisitados. Nesse caso, como os poluidores não são isolados do sistema, *peers* realizam requisições sucessivas a eles. Assim, não se consegue dados de um bom parceiro dentro da janela de interesse. Como o *mecanismo de reputação simples* isola rapidamente os poluidores, a taxa de perda cai para um valor desprezível, abaixo de $0,7\%$.

A Fig. 6 apresenta a taxa de erro em um cenário no qual poluidores também realizam ataques de *whitewashing*. Os resultados mostram que, mesmo com o *mecanismo de reputação simples*, *whitewashing* causa uma alta taxa de erro, alcançando quase 70%. Porém, as simples modificações realizadas reduzem a taxa de erro a 19%. Essa alta taxa de erro se deve ao fato das constantes mudanças de identidade dos poluidores.

A sobrecarga também apresenta uma piora em relação ao cenário no qual os poluidores não fazem *whitewashing*. A Fig. 7 mostra que, utilizando o *mecanismo*

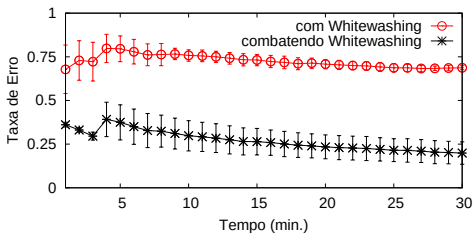


Figura 6. Taxa de erro de *chunks*.

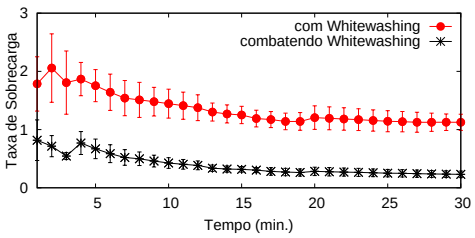


Figura 7. Sobrecarga no sistema.

de reputação simples, os *peers* experimentam 112% de sobrecarga. Ao se adequar o *mecanismo de reputação* para lidar com *whitewashing*, a sobrecarga se reduz a 20%. Novamente, esse alto valor ocorre por conta das mudanças de identidade dos poluidores.

Finalmente, de acordo com a Fig. 8, a taxa de perdas pode ser reduzida de 40%, quando há somente a reputação simples, para valores inferiores a 3%, na versão modificada para lidar com *whitewashing*. Nesse caso, mesmo sob um forte ataque, os *peers* conseguem assistir um vídeo com alta qualidade.

Apesar dos valores altos para sobrecarga e taxa de erro, os resultados alcançados com os mecanismos de reputação propostos não podem ser subestimados. Primeiro, os esquemas de reputação propostos são simples e com implementação de baixo custo. Segundo, não são necessários custosos mecanismos de identificação de *peers*, comumente usados para evitar o ataque de *whitewashing*.

Uma das características importantes dos esquemas de reputação propostos é a recuperação de *peers bons* e que possam ter sido considerados poluidores, devido ao funcionamento incorreto da rede (falsos positivos). Para avaliar a robustez dos mecanismos propostos, um *peer* no sistema foi configurado para enviar 10% de seus *chunks*, propositalmente, como corrompidos[6], simulando assim, problemas de rede.

Foram considerados 3 valores para o limiar de calmaria (*lcalm*=$\{0{,}1;\ 0{,}2;\ 0{,}3\}$). Para $lcalm = 0{,}1$; cerca de 30% dos *peers* identificam o falso poluidor. Porém, 83% deles conseguem redimir o falso poluidor em 110 segundos. Para o caso $lcalm = 0{,}2$; 29% dos *peers* identificam o falso poluidor, sendo que 73% destes recuperam o falso positivo em 97 segundos. Por último, para o caso em que $lcalm = 0{,}3$; cerca de 27% dos *peers* identificam o falso poluidor. Nesse caso, 51% dos *peers* voltam a trocar informações com o falso poluidor em, aproximadamente, 72 segundos.

Para valores menores do parâmetro *lcalm*, uma maior percentagem de *peers* identifica os falsos poluidores, bem como os recuperam. Com valores mais baixos para esse parâmetro, os *peers* do sistema ficam mais sensíveis aos ataques de poluição. Eles constantemente vão para um estado de *tempestade* tentando se defender. Com esse comportamento, os *peers* conseguem remover rapidamente os poluidores reais, porém, eles aumentam sua taxa de falso positivo na identificação de poluidores. Mais ainda, como os *peers* podem ficar períodos maiores em estado de tempestade, o valor da reputação mínima para ser considerado um bom *peer* cresce, e assim, os falsos poluidores demoram mais tempo a serem perdoados por outros *peers*.

[6]Uniformemente distribuído entre os envios de dados.

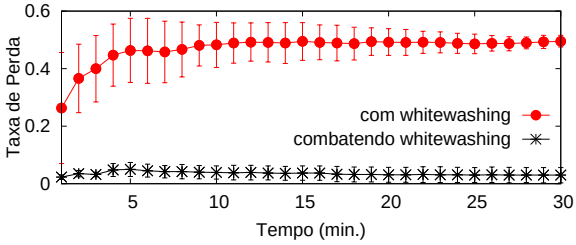


Figura 8. Taxa de Perda.

A Fig. 9 mostra o total de *chunks* recebidos pelo falso poluidor durante a experimentação. Após 10 minutos iniciais, o falso poluidor recebe entre 2900 a 3000 *chunks* a cada 30 segundos. Este valor é a taxa esperada para os demais *peers* do sistema. Assim, mesmo o falso poluidor tenha sido considerado como um poluidor de fato, ele não terá problemas ao exibir o vídeo. Este resultado está intimamente ligado à recuperação que o mecanismo implementado possibilita aos *peers* com problemas temporários.

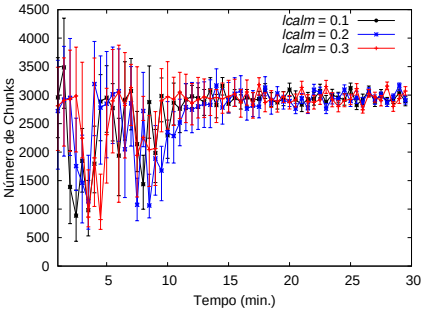


Figura 9. Número de chunks recebidos pelo falso poluidor.

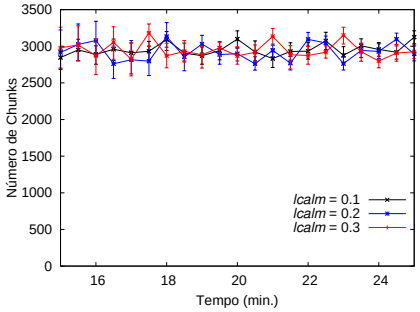


Figura 10. Número de chunks recebidos pelo falso poluidor.

A Fig. 10 mostra o número de *chunks* recebidos pelo falso poluidor, no intervalo entre 15 minutos e 25 minutos. Este período representa um intervalo estável da rede, sem o distúrbio da inicialização do ataque de poluição. Nesse intervalo, o falso poluidor recebe praticamente um fluxo de vídeo completo, 100% de *chunks*.

Para analisar a qualidade de experiência dos usuários, ao se utilizar os mecanismos de reputação propostos, considera-se a métrica atraso de exibição do vídeo. Define-se como atraso o intervalo de tempo entre criação de *chunks* pelo servidor e exibição dos mesmos pelos *peers*. A Fig. 11 apresenta o atraso médio durante uma transmissão ao vivo no sistema P2P, no qual os poluidores não realizam *whitewashing*. O atraso médio, ao se adotar somente a verificação dos dados, é superior a 2 minutos. Além do grande atraso observado, esse ambiente apresenta uma alta taxa de erros, alta taxa de perdas e grande sobrecarga. Pode-se afirmar que a qualidade de experiência dos usuários é ruim. Em contrapartida, de acordo com a Fig. 12, a adoção de um *mecanismo de reputação simples* reduz o atraso médio a apenas 6 segundos. Os usuários, além da baixa latência, apresentam baixa perda e sobrecarga imposta pelo ataque de poluição.

O atraso médio em um sistema sob ataque de poluição e *whitewashing* dos poluidores, em geral, é elevado. Ao se utilizar o *mecanismo de reputação simples*, o atraso médio é de quase 1 minuto. Com a modificação realizada, o atraso médio é reduzido a cerca de 13 segundos, valor aceitável para aplicações desta natureza.

7. Conclusões

Neste trabalho, é analisado o impacto de ataques de poluição de conteúdo em sistemas P2P de transmissão de vídeo ao vivo. Também é verificado o impactos de mudanças de identificação dos atacantes (conhecido como ataque de *whitewashing*). Nessa linha, os danos causados ao sistema P2P verificados sob duas perspectivas. Inicialmente, por meio de um modelo analítico proposto. Tal modelo captura a sobrecarga imposta à banda de rede dos participantes de um sistema de vídeo ao vivo em P2P, do tipo baseados em

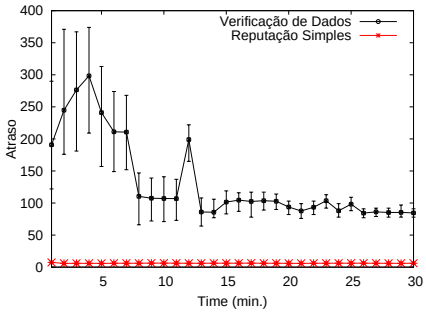


Figura 11. Atraso Médio Durante Ataque de Poluição.

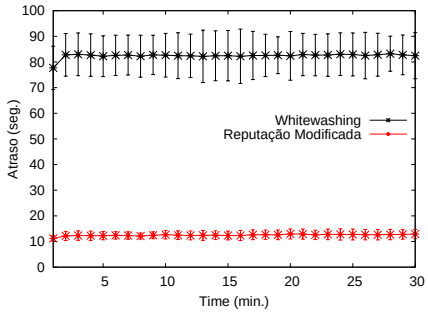


Figura 12. Atraso Médio Durante Ataque de Poluição.

malha com pedidos explícitos por dados. Em segundo lugar, por meio de experimentos conduzidos em um ambiente realista, criado no PlanetLab.

O modelo proposto evidencia que ataques de poluição são danosos a sistema P2P de transmissão ao vivo, mesmo com um número pequeno de atacantes. Os *peers* de tal sistema recebem um grande número de *chunks* poluídos e, ao realizar novas requisições, descartando o *chunk* poluído, são penalizados sobrecarregando suas respectivas larguras de banda. Por exemplo, os *peers* devem ter mais de 3 vezes a largura de banda necessária do que deveriam ter em um sistema sem poluidores.

Os resultados experimentais mostram que o sistema de reputação implementado é efetivo sob ataques de poluição. Nesse caso, a sobrecarga imposta aos *peers* do sistema cai para valores abaixo de 5%. As taxas de perda e de dados poluídos também podem ser negligenciadas. Em um cenário no qual os poluidores realizam *whitewashing*, a sobrecarga observada é de 20%. As perdas no tempo de visualização são inferiores a 3% dos *chunks*.

Apesar dos valores altos para sobrecarga e taxa de erro no cenário com *whitewashing*, os resultados alcançados com os mecanismos de reputação propostos não podem ser subestimados. Os esquemas de reputação propostos são fáceis e baratos de implementar. Mais ainda, não são necessários mecanismos de identificação centralizados.

Como trabalhos futuros, espera-se realizar verificações em novos cenários, com um número maior de participantes. Também deverão ser abordados cenários com comportamento dos participantes mais realistas, incluindo *churn*.

Referências

de Almeida, R. B., Silva, A. P. C., and Vieira, A. B. (2012). Análise do Impacto de Ataques de Poluição Combinado com Whitewashing em Sistemas P2P de Live Streaming. In *Proc. of Workshop de Testes e Tolerância a Falhas (WTF)*.

Dhungel, P., Hei, X., Ross, K., and Saxena, N. (2007). The Pollution Attack in P2P Live Video Streaming: Measurement Results and Defenses. In *Proc. of ACM workshop on Peer-to-peer streaming and IP-TV*.

Feldman, M., Papadimitriou, C., Chuang, J., and Stoica, I. (2006). Free-riding and Whitewashing in Peer-to-Peer Systems. *IEEE JSAC*, 24(5):1010–1019.

Haizhou, W., Xingshu, C., and Wenxian, W. (2011). A Measurement Study of Polluting a Large-Scale P2P IPTV System. In *College of Computer Science, Sichuan University, Chengdu 610065, P. R. China*.

Haridasan, M. and Renesse, R. V. (2008). SecureStream: An Intrusion-Tolerant Protocol for Live-Streaming Dissemination. *Elsevier Computer Communications*, 31(3):563–575.

Hei, X., Liang, C., Liang, J., Liu, Y., and Ross, K. (2007). A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia*, 9(8):1672–1687.

Hei, X., Liu, Y., and Ross, K. W. (2008). IPTV Over P2P Streaming Networks: The Mesh-Pull Approach. *IEEE Communications Magazine*, 46(2):86–92.

Hu, B. and Zhao, H. (2010). Joint Pollution Detection and Attacker Identification in Peer-to-Peer Live Streaming. In *Proc. of IEEE ICASSP*.

Kudtarkar, A. and Umamaheswari, S. (2009). Avoiding White Washing in P2P Networks. In *Proc. of IEEE COMSNETS*.

Kwiatkowska, M., Norman, G., and Parker, D. (2009). PRISM: Probabilistic Model Checking for Performance and Reliability Analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45.

Oualha, N. and Roudier, Y. (2009). A Game Theoretical Approach in Securing P2P Storage against Whitewashers. In *Proc. of 18th IEEE WETICE*.

Seibert, J., Sun, X., Nita-Rotaru, C., and Rao, S. (2010). Towards Securing Data Delivery in Peer-to-Peer Streaming. In *Proc. of IEEE COMSNETS*.

So, J. and Reeves, D. (2012). AntiLiar: Defending Against Cheating Attacks in Mesh Based Streaming. In *Proc. of IEEE International Conference on Peer-to-Peer Computing*.

Vieira, A., da Silva, A. P. C., Henrique, F., Goncalves, G., and Gomes., P. (2013). Sopcast p2p live streaming: Live session traces and analysis. In *Proc. of The ACM Multimedia Systems Conference ACM MMSys 2013*.

Vieira, A. B., Campos, S., and Almeida, J. (2008). Fighting Pollution in P2P Live Streaming Systems. In *Proc. of IEEE International Conference on Multimedia and Expo*.

Vieira, A. B., Campos, S., and Almeida, J. (2009). Fighting Attacks in P2P Live Streaming. Simpler is Better. In *Proc. of IEEE INFOCOM Workshops*.

Vieira, A. B., Gomes, P. C., Nacif, J., Mantini, R., Almeida, J., and Campos, S. (2012). Characterizing SopCast Client Behavior. *Elsevier Computer Communications*, 35(8):1004 – 1016.

Em Busca do Vizinho Perfeito: Um Modelo para Estratégias de Gerência de Vizinhança em Sistemas Descentralizados de Distribuição de Conteúdo

Matheus B. Lehmann, Rodolfo S. Antunes, Marinho P. Barcellos

[1]Instituto de Informática - Universidade Federal do Rio Grande do Sul
Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brasil

{mblehmann,rsantunes,marinho}@inf.ufrgs.br

Resumo. *Estudos prévios e experiência adquirida com o funcionamento de sistemas de distribuição de conteúdo em larga escala permitiram o amadurecimento da pesquisa nessa área. Apesar dos algoritmos de troca de dados terem sido amplamente investigados, pouca atenção foi dedicada às características topológicas resultantes das conexões entre pares. Nesse contexto, existem três classes de algoritmos para o gerenciamento de conexões: inativa, preemptiva e pró-ativa. Até o momento não há estudos que apresentem uma comparação detalhada dessas classes de algoritmos em um cenário comum. O presente trabalho apresenta um modelo analítico baseado em grafos evolutivos que captura com maior precisão o impacto das estratégias de gerência de conexões na robustez e no desempenho de sistemas descentralizados de distribuição de conteúdo. Os resultados obtidos mostram que o uso de mecanismos preemptivos e pró-ativos beneficiam a topologia da rede tanto na robustez do sistema, devido à randomização das conexões entre os pares, quanto no potencial de desempenho, ao centralizar a fonte dos dados na topologia em relação aos demais pares.*

Abstract. *Previous work and the popularity of large scale content distribution systems helped evolve this research area to a mature state. Various studies investigate the algorithms used for data exchange, however little attention has been given to the topological characteristics of the overlay network resulting from connections among peers. Three classes of connection management algorithms are known: inactive, preemptive and pro-active. So far there is no study which presents a detailed comparison among these three strategies in a common scenario. This paper presents an analytical model based on evolving graphs, which captures with more precision the impact of connection management strategies in the robustness and performance of decentralized content dissemination systems. Results show that preemptive and pro-active mechanisms help, firstly, in the formation of a robust topology, due to the randomization of connections among peers, and secondly, in the performance improvement, due to greater centralization of the content source in the overlay.*

1. Introdução

Atualmente a maior parte das soluções descentralizadas de larga escala para disseminação de conteúdo (i.e. compartilhamento de arquivos, *streaming* de vídeo) são sistemas cujo princípio fundamental para a eficiência é o *swarming*, tais como SopCast[1] e BitTorrent[2]. Essas redes são caracterizadas por serem não-estruturadas, ou seja, não há regras de

[1]http://www.sopcast.org
[2]http://www.bittorrent.com

formação da topologia [Zhang et al. 2007]. Além disso, tais redes agrupam pares interessados em um mesmo conteúdo, facilitando sua organização e gerência. Entretanto, como consequência dessas características, as topologias resultantes são em princípio arbitrárias.

Sabe-se que a topologia da rede formada pelas conexões entre pares influencia diretamente no desempenho geral de aplicações descentralizadas. Dessa forma, essas aplicações utilizam estratégias que visam organizar os pares em topologias com boas propriedades. Dentre todas propriedades, as mais desejadas são: robustez, para suportar uma grande população transiente sem particionar a rede, e desempenho, a fim de minimizar o tempo necessário para disseminação dos dados a todos pares.

A literatura descreve três classes de estratégias para gerência de conexões em redes descentralizadas baseadas em *swarming*: inativa, preemptiva e pró-ativa. A primeira estratégia, **inativa**, é a mais simples: uma conexão estabelecida com um vizinho é mantida até que um dos pares saia do sistema [Cohen 2003]. Na segunda, denominada **preemptiva**, os pares podem dar preferência a novas conexões requisitadas em detrimento de uma já estabelecida [Al-Hamra et al. 2009]. A terceira e última estratégia é o procedimento contrário, ou seja, **pró-ativamente** os pares desconectam algum vizinho periodicamente, possibilitando o estabelecimento de novas conexões [Lehmann et al. 2012a, Lehmann et al. 2012b].

Estudos da literatura avaliam cada uma das estratégias individualmente e focam principalmente em métricas de desempenho dos pares. Nenhum trabalho, além disso, apresenta uma comparação de todas as estratégias em cenários comuns. Por fim, poucos estudos apresentam uma análise detalhada de métricas relacionadas com as características topológicas da rede. O presente trabalho, portanto, visa apresentar uma comparação detalhada das três estratégias para gerenciamento de conexões, focando em sua influência nas características topológicas da rede. Para cumprir tal objetivo, foi desenvolvido um modelo de grafos evolutivos que permite representar o comportamento da topologia ao utilizar cada uma das estratégias. Em seguida, o modelo é avaliado através de simulações, permitindo o estudo de diferentes configurações de redes e a qualidade das topologias resultantes segundo métricas de robustez e desempenho.

As principais contribuições do trabalho são: (*i*) um estudo abrangente do impacto das estratégias de gerenciamento de conexões sobre as características topológicas de redes descentralizadas baseadas em *swarming*; (*ii*) o desenvolvimento de um modelo que captura com maior precisão a evolução das topologias, considerando aspectos como conectividade parcial dos pares e diferentes formas de gerência da vizinhança.

A Seção 2 do artigo discute as estratégias de gerência de vizinhança estudadas, assim como mecanismos que as implementam. A Seção 3 define os modelos que descrevem as topologias resultantes segundo as três estratégias de gerência de conexões. Na Seção 4, a metodologia de avaliação é descrita, explicitando cenários e métricas considerados. Os resultados da avaliação comparativa das estratégias são apresentados na Seção 5, juntamente com as principais conclusões obtidas. Por fim, a Seção 6 apresenta as considerações finais e direções para trabalhos futuros.

2. Contexto e Trabalhos Relacionados

Redes descentralizadas baseadas em *swarming* tem como característica principal agrupar usuários com interesses comuns. Isso é realizado geralmente através de algum tipo de ponto de encontro, como por exemplo *trackers* ou *Distribute Hash Tables* (DHT). Como

o tamanho dessas redes pode variar entre poucos pares até milhares deles, é computacionalmente inviável manter uma conexão com todos os pares presentes no sistema. Deste modo, o par seleciona um subconjunto (sua vizinhança) para se conectar e interagir.

A vizinhança é construindo escolhendo aleatoriamente pares com os quais conectar dentre uma lista de pares recebida do ponto de encontro. Adicionalmente, aceita-se qualquer solicitação de conexão de pares remotos. Esse procedimento é repetido até que o número máximo de conexões permitidas pela aplicação seja atingido. Uma vez estabelecida a vizinhança, o par utiliza mecanismos para gerenciar suas conexões com objetivos que variam desde melhorar seu desempenho até aumentar justiça no sistema. A seguir as três estratégias para gerência de vizinhança são apresentadas, assim como trabalhos relacionados a cada uma.

2.1. Estratégia Inativa

A gerência inativa de vizinhança é a estratégia mais simples dentre todas. Conexões estabelecidas com vizinhos são mantidas durante toda a sessão do par e não há qualquer tipo de avaliação em relação a sua qualidade ou tentativa de substituição dos vizinhos atuais. As únicas exceções são quando ocorre a desconexão de um vizinho (i.e. terminou a sessão e saiu da rede) ou quando existe alguma condição desfavorável aos pares. No caso do protocolo BitTorrent, se dois pares possuem a cópia completa do conteúdo, eles não são úteis um para o outro e, portanto, sua conexão é terminada. Já em aplicações de vídeo sob demanda, dois pares que estão em sessões diferentes (i.e. estão assistindo partes distintas do vídeo) não possuem interesses mútuos e, portanto, não há possibilidades para troca de dados úteis.

A intuição por trás dessa estratégia é que a escolha aleatória de vizinhos tende a gerar um grafo randômico [Cohen 2003]. Entretanto, o que se observa na realidade é uma potencial clusterização de pares de acordo com seu tempo de chegada, deteriorando a robustez da rede [Al-Hamra et al. 2009,Lehmann et al. 2012b]. Em um extremo negativo, a rede pode inclusive tornar-se particionada. Por esse motivo, estratégias alternativas para gerência da vizinhança foram propostas, as quais tem a finalidade de aumentar a robustez e desempenho da rede.

2.2. Estratégia Preemptiva

A estratégia preemptiva propõe uma modificação na forma como vizinhos são gerenciados em relação à inativa. A diferença entre as duas estratégias ocorre quando um par recebe um pedido de conexão e sua vizinhança já está completa. Ao invés de simplesmente rejeitar a conexão (estratégia inativa), na estratégia preemptiva, o par desconecta um vizinho aleatório com a finalidade de abrir uma vaga para a nova conexão requisitada.

Essa estratégia tem como principal motivação introduzir aleatoriedade nas conexões formadas entre pares [Al-Hamra et al. 2009], o que nem sempre é possível com a estratégia inativa. Essa aleatoriedade ajuda todos os pares a convergirem para seu tamanho máximo de vizinhança e a prevenir clusterizações prejudiciais. Enquanto a estratégia inativa tende a privilegiar conexões de pares cujos tempos de chegada são próximos, a estratégia preemptiva espalha uniformemente as conexões entre os pares, independentemente de seus tempos de chegada. Além de aumentar a robustez da topologia, essa estratégia tem o potencial de aumentar a diversidade de peças a que um par tem acesso, causando um efeito positivo no desempenho.

Outro mecanismo que apresenta uma estratégia preemptiva é o protocolo *Push-to-Pull* [Locher et al. 2007]. Esse protocolo leva em consideração a latência dos pares ao

decidir se solicitações de conexão serão aceitas. Essa técnica juntamente com o emprego de uma DHT para a organização da topologia garante eficiência e robustez na disseminação de *streams* de vídeo.

2.3. Estratégia Pró-ativa

A última estratégia utilizada para melhorar a gerência da vizinhança é a desconectar para conectar ou pró-ativa. A principal diferença dessa estratégia em relação à inativa é a adição de uma nova condição para desconexão de vizinhos. Periodicamente, o par avalia seus vizinhos e desconecta aqueles que forem classificados como pouco úteis de acordo com um conjunto de métricas pré-definidas. Dessa maneira, o par abre vagas na vizinhança, permitindo que novas conexões sejam estabelecidas.

Alguns mecanismos implementam essa estratégia, tais como *Optimistic Disconnect* (OD), *Lifetime-based Peer Selection* (LIPS) e *Deprived Peer* (DP). O OD, estudado inicialmente em [Dhungel et al. 2011], classifica a vizinhança do par segundo seu nível de colaboração [Lehmann et al. 2012a]. Essas informações são utilizadas para definir se os vizinhos são interessantes, esnobes, pares-carona ou possíveis corruptores de conteúdo. Ao identificar e desconectar vizinhos inúteis, novas conexões podem ser estabelecidas, possivelmente melhorando a qualidade da vizinhança.

O LIPS [Moraes and Duarte 2010] é um mecanismo que monitora a vizinhança baseada no tempo de chegada dos pares. Este assume que pares com tempo de chegada similares estão assistindo à mesma parte do vídeo (i.e. estão na mesma sessão) e, portanto, possuem mais dados em comum para trocar. Os resultados apontam que o LIPS seleciona os melhores vizinhos em relação à eficiência das conexões, garantindo alta continuidade de reprodução do vídeo.

O último mecanismo, DP [Picconi and Massoulie 2008], tem uma proposta de organização do ponto de vista global. A ideia é equilibrar a capacidade média de upload da vizinhança de todos os pares da rede. Com isso, permite-se organizar os pares em uma topologia de malha de forma que todos tenham taxas de *streaming* próximas à ótima e pequeno atraso de difusão. Esse mecanismo e suas variantes também foram analisados quanto às condições necessárias para atingir a estabilidade do sistema [Menasché et al. 2012]. O estudo conclui que a estratégia de selecionar os pares mais necessitados é a que apresenta maior ganho em relação à escalabilidade do sistema.

3. Modelo de Grafos Evolutivos

Essa seção descreve, primeiramente, os principais conceitos relacionados com grafos evolutivos [Ferreira 2002] e como estes são utilizados para modelar a gerência de conexões em sistemas distribuídos. Em seguida, é apresentado o modelo desenvolvido para capturar o comportamento das topologias ao utilizar cada uma das estratégias descritas.

Um modelo de grafos evolutivos segue uma abordagem de grafos dinâmicos que, oposta aos grafos estáticos, permite capturar variações da topologia ao longo do tempo, tais como adição ou remoção de vértices ou arestas. Seja $G = \langle V, A \rangle$ um grafo não-direcionado, em que V é seu conjunto de vértices e A, o conjunto de arestas. Na representação de um modelo evolutivo, a topologia da rede evolui em intervalos discretos de tempo $t = 1, 2, ..., n$. A cada instante t, o estado atual é representado por $G_t = \langle V_t, A_t \rangle$. Duas funções são necessárias para descrever a evolução da topologia a cada instante de tempo t: uma para o crescimento de vértices (f_v) e outra para o de arestas (f_a). Dessa forma, o modelo de grafo evolutivo pode ser completamente descrito por $\langle f_v, f_a \rangle$.

A criação e evolução da topologia de uma rede descentralizada para distribuição de conteúdo pode ser modelada pelos seguintes eventos. Cada par (representado por um vértice) ingressa na rede em um tempo t_v, sendo adicionado ao conjunto de vértices V_t. Periodicamente, a partir da entrada, cada par estabelece conexões (arestas) com outros pares já presentes na rede. Estes dois eventos são descritos, respectivamente, pelas funções de crescimento f_v para os pares e f_a para as conexões.

A função f_v especifica a evolução do conjunto de pares ativos na rede ao longo do tempo, definindo a cada instante t quantos pares entram no sistema. Deste modo, $|V_{t+1}| = |V_t| + f_v(t)$. Já a função f_a descreve quais conexões são estabelecidas pelos pares durante a evolução da topologia. Essa função seleciona um conjunto de arestas que são adicionadas ao grafo G a cada instante t. Desta forma, $A_{t+1} = A_t \cup f_a(G_t, V_t, t)$. Note que o objetivo do modelo é descrever como a topologia está organizada após a entrada de todos os pares na rede, não contemplando, a princípio, possíveis saídas ou desconexões de pares.

Além dessas duas funções, duas definições extras são necessárias para descrever um modelo mais preciso. Primeiro, os pares do sistema possuem uma visão limitada da rede, que é definida por: $N(v)$, o número de vizinhos (ou o grau do vértice) atual do vértice v; e o número máximo de vizinhos permitidos na rede, ou seja, o grau máximo dos pares, representado por m. Segundo, alguns pares na Internet são incapazes de receber conexões (por exemplo, se estiverem atrás de NAT). Para representar tal comportamento, define-se a função $C(v)$, a qual especifica se v pode receber uma conexão iniciada por um par remoto ou não.

3.1. Estratégia Inativa

Para modelar a aleatoriedade inerente ao estabelecimento de conexões, a cada instante t é definido um conjunto de possíveis arestas criado a partir de algumas condições. Uma vez gerado esse conjunto, k conexões são escolhidas e adicionadas ao grafo G. Três condições definem as possíveis conexões: as vizinhanças dos pares não podem estar completas, os pares devem estar presentes na rede, e o par que receberá a conexão deve ser conectável. Portanto, seja P o conjunto de possíveis arestas a serem adicionadas, inicialmente $P = \emptyset$. Após a entrada de cada novo vértice v na rede, P é atualizado para $P = P \cup \{(v, u) \mid u \in V_t \wedge N(u) < m \wedge C(u) = 1\}$.

Após a entrada do par na rede, periodicamente a cada r intervalos de tempo, o par obtém uma nova visão dos pares presentes no sistema, expandindo o conjunto de conexões possíveis. Quando esse evento ocorre ao vértice w, se $N(w) < m$, então são adicionadas novas arestas ao conjunto de conexões possíveis seguindo as condições previamente descritas. Portanto, $P = P \cup \{(v, u) \mid u \in V_t \wedge N(u) < m \wedge C(u) = 1 \wedge (v, u) \notin A_t\}$.

Após adicionar uma nova aresta (v, w) ao grafo G, o modelo verifica se algum vértice atingiu o tamanho máximo da sua vizinhança. Caso sim, por exemplo $N(v) = m$, todas suas entradas no conjunto de conexões possíveis são removidas. Tem-se, portanto, que $P = P \backslash (\{(v, u) \mid u \in V_t\} \cup \{(u, v) \mid u \in V_t\})$. O mesmo é feito, em seguida, para o vértice u se $N(w) = m$.

3.2. Estratégia Preemptiva

A intuição por trás da estratégia preemptiva é que novas conexões são melhores que antigas do ponto de vista global da rede. Dessa forma, quando um par recebe uma nova

conexão, ele a aceita mesmo que sua vizinhança esteja completa. Entretanto, para satisfazer a restrição do tamanho da vizinhança, uma conexão atual é terminada.

A estratégia preemptiva modifica o modelo, relaxando a restrição da vizinhança máxima dos pares na construção do conjunto de possíveis conexões P. Além disso, é necessário criar uma nova função para manter a condição de número máximo de vizinhos. Portanto, ao adicionar um novo vértice v, o conjunto de possíveis conexões é agora atualizado como $P = P \cup \{(v, u) \mid u \in V_t \wedge C(u) = 1\}$.

Após o evento de adição de arestas, verifica-se o tamanho da vizinhança dos pares e conexões são removidas até que $\forall v \in V_t \mid N(v) \leq m$. Nessa etapa, $\forall v \in V_t \mid N(v) > m$, são escolhidos $n = N(v) - m$ pares aleatórios para serem desconectados. A única restrição na desconexão é não remover um vizinho conectado no instante t atual.

3.3. Estratégia Pró-ativa

A ideia por trás da estratégia pró-ativa é substituir uma conexão existente por uma nova a ser estabelecida. Diferentemente da estratégia anterior, cabe ao próprio par a decisão de quando e quais conexões a serem removidas. Portanto, cria-se uma nova função $Q(t)$, que periodicamente remove arestas de pares cuja vizinhança está completa, permitindo que estes criem novas conexões.

A função $Q(t)$ especifica quais conexões são encerradas no instante t. $Q(t)$ é definida como $\sum_{v \in V_t} B_{v,t}$, onde $B_{v,t}$ é um conjunto (possivelmente vazio) para cada vértice v na rede, contendo as conexões encerradas por v no instante t. O número de conexões contidas em $B_{v,t}$ e o critério utilizado para sua seleção são arbitrários e dependem da implementação da estratégia pró-ativa. Por exemplo, no caso mais simples, $B_{v,t}$ possui cardinalidade 1 e a conexão terminada é escolhida aleatoriamente. Por fim, a nova função $Q(t)$ altera o conjunto de conexões A_t, o qual é atualizado como $A_t = (A_t \backslash Q(t)) \cup f_a(G_t, V_t, t)$.

Após a remoção das arestas do grafo G, a estratégia pró-ativa deve estabelecer novas conexões para aqueles pares que desconectaram vizinhos. A etapa de reestabelecimento das conexões é realizada sem qualquer modificação através da função de adicionar arestas f_a.

4. Metodologia

Esta seção apresenta a metodologia utilizada na avaliação das estratégias de gerência de conexões. O foco é quantificar a qualidade da topologia resultante quando cada uma das estratégias é utilizada. Para isso, duas questões principais devem ser respondidas:

- Q1. Qual a resiliência e robustez da topologia em relação a possíveis distúrbios?
- Q2. Qual o potencial de eficiência e desempenho causado pela topologia?

A primeira questão objetiva analisar a capacidade do sistema de organizar sua topologia, mantendo sua robustez. Em outras palavras, se a rede evolui ao longo do tempo sem particionar-se e minimizando o impacto de distúrbios como *churn*.

A segunda questão, por sua vez, é um indicativo da qualidade da disseminação do conteúdo na rede. Como o modelo desconsidera questões práticas como a capacidade dos pares, o foco é avaliar se a estrutura da topologia é adequada para a distribuição eficiente de dados. Por exemplo, uma possibilidade é a rede ter uma topologia com distância mínima dos pares do sistema à fonte do conteúdo.

Para realizar a avaliação foi desenvolvido um simulador baseado em eventos[3], o qual implementa o modelo descrito na seção anterior. As questões apresentadas são usadas para definir: (*i*) os cenários a serem considerados para uma avaliação relevante (Seção 4.1); (*ii*) o conjunto de métricas usado para respondê-las (Seção 4.2).

4.1. Cenários

A avaliação tem como objetivo estudar o comportamento de cada estratégia sob diferentes condições do sistema. Para isso, são definidos dois conjuntos de cenários, que variam dois dos parâmetros com maior impacto na organização topológica: a taxa de entrada dos pares e a porcentagem de pares que são capazes de aceitar conexões. Em todos cenários avaliados considera-se uma rede composta por 400 pares, os quais cada um possui no máximo 80 vizinhos.

O primeiro conjunto de cenários avalia como diferentes padrões de chegada dos pares à rede impactam na sua formação topológica. Para isso, foram consideradas três taxas de chegada dos pares à rede: (*i*) exponencial decrescente (*flash crowd*), quando há alta demanda de recursos e baixa oferta destes; (*ii*) constante, representando um cenário em que não existe sobrecarga na rede; e (*iii*) exponencial crescente, onde a taxa de entrada dos pares é inicialmente pequena e cresce ao longo do tempo. Em todos os casos avaliados, a alcançabilidade da rede é completa.

O segundo conjunto de cenários visa analisar o efeito da restrição de alcançabilidade dos pares na topologia resultante da rede. Na Internet muitos pares não podem receber conexões, por exemplo devido a NAT. Dessa forma, para avaliar cenários mais realistas, avalia-se o comportamento dos pares quando a alcançabilidade da rede varia desde 30% até 100%. Nesta avaliação, considera-se a entrada de pares em *flash crowd*.

4.2. Métricas

Tomando como base as perguntas propostas, as seguintes métricas foram definidas para respondê-las. Em relação à robustez da rede (Q1), as métricas utilizadas são: conectividade de vértice média à fonte e coeficiente de clusterização. Já para quantificar o potencial de desempenho do sistema (Q2), a topologia é avaliada quanto ao grau médio dos vértices e a proximidade (em inglês, *closeness*) da fonte do conteúdo. A seguir, cada uma das métricas é detalhada:

Conectividade de vértice média à fonte (κ): o número médio de vértices que devem ser removidos para que pelo menos um par não possua um caminho ligando-o à fonte. Na prática, representa a resiliência da topologia em situações de *churn*.

Coeficiente de Clusterização (C): representa a probabilidade de dois vértices conectados a um terceiro vértice em comum estarem conectados entre si. Valores próximos a 1 indicam que os vértices estão altamente conectados entre si, enquanto valores próximos a 0 indicam que os vértices possuem poucos vizinhos em comum. Esta métrica exprime a robustez do sistema, indicando o número de vizinhos em comum ou redundantes, assim como sua variabilidade.

Grau médio dos vértices (d): o número médio de vizinhos dos pares da rede. Quanto maior esse valor, maior o número de conexões na rede e, portanto, as possibilidades de interação entre pares do sistema. Essa métrica influencia diretamente no potencial de desempenho da rede.

[3]Disponível em http://www.inf.ufrgs.br/~mblehmann/sbrc2013

Proximidade da fonte (P): uma métrica de centralidade de vértices. Ela indica a distância média de um vértice para todos os outros do grafo. No caso de disseminação de conteúdo, essa métrica pode ser vista como uma medida do tempo médio para o conteúdo ser disseminado da fonte para toda a rede.

5. Resultados

Esta seção apresenta os resultados dos cenários definidos na Seção 4.1. Foram realizadas 30 execuções de cada configuração com o objetivo de obter validade estatística dos resultados. Os dados de entrada e saída usados na geração dos resultados estão disponíveis publicamente[4], permitindo a outros pesquisadores refazer os experimentos.

5.1. Taxa de Entrada dos pares

Esta seção apresenta o impacto das três estratégias de gerência de conexões sob diferentes taxas de entrada dos pares. A análise é iniciada com o cenário de taxa de entrada exponencial decrescente (*flash crowd*) dos pares, por causar sobrecarga ao sistema, o que pode impactar a organização dos pares. Em seguida é avaliado os cenários no qual os pares entram em uma taxa constante, evitando a situação de alta demanda por recursos. Por fim, o último cenário é com taxa de entrada exponencial crescente de pares, que pode ser vista como um sistema no qual a demanda inicial é pequena, mas sofre um aumento repentino devido a popularização do conteúdo.

5.1.1. Taxa de Entrada Exponencial Decrescente (*Flash Crowd*)

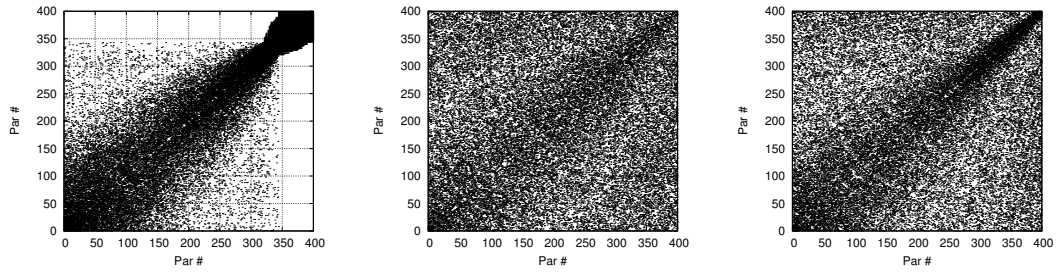
A Figura 1 apresenta um conjunto de matrizes de conexões, uma para cada estratégias utilizadas. A matriz de conexão é uma fotografia das conexões da rede em um determinado momento. Cada ponto (x, y) na matriz representa uma conexão entre os pares x e y. Por fim, os pares estão ordenados pelo seu tempo de chegada, sendo que a fonte original do conteúdo é o par $x = y = 0$.

A análise das matrizes de conexão mostra que a estratégia inativa (Figura 1(a)) tende a agrupar pares que tenham tempos de chegada próximos. Tal tendência se torna mais acentuada nos últimos pares que entram no sistema, pois estes possuem poucas oportunidades de conexão. No cenário de *flash crowd* o efeito do agrupamento é amenizado se comparado aos demais padrões de entra de pares, pois a entrada massiva inicial de pares garante uma boa distribuição das conexões. No caso das estratégias preemptiva (Figura 1(b)) e pró-ativa (Figura 1(c)), observa-se uma redução considerável na intensidade do agrupamento dos pares, sendo a primeira a que mais se aproxima visualmente de uma distribuição uniforme.

Essa análise preliminar ajuda na compreensão do impacto de cada estratégia na topologia da rede. Mas para possuir um entendimento completo sobre elas, é necessário avaliar objetivamente a qualidade das topologias. A Tabela 1 apresenta os intervalos de confiança (para um grau de 95%) dos resultados obtidos para as métricas elencadas na Seção 4.2. Para fins de comparação, apresenta-se também os valores das métricas para um grafo randômico com a mesma configuração.

Os resultados mostram que a estratégia inativa gera uma topologia altamente clusterizada, como é observado através do valor do grau de clusterização duas vezes maior

[4]Endereço para acesso: http://www.inf.ufrgs.br/~mblehmann/sbrc2013



(a) Estratégia Inativa (b) Estratégia Preemptiva (c) Estratégia Pró-ativa

Figura 1. Matrizes de conexões das topologias no cenário de taxa de entrada exponencial decrescente dos pares

Métrica	Inativa	Preemptiva	Pró-Ativa	Randômico
κ	(64,16; 64,22)	(76,37; 76,58)	(76,65; 76,86)	(80,0; 80,0)
C	(0,3859; 0,3867)	(0,1915; 0,1916)	(0,1987; 0,1988)	(0,1963; 0,1964)
d	(78,07; 78,08)	(79,42; 79,43)	(78,92; 78,93)	(80,0; 80,0)
P	(1,915; 1,925)	(1,801; 1,802)	(1,801; 1,802)	(1,796; 1,796)

Tabela 1. Métricas das topologias no cenário de taxa de entrada exponencial decrescente dos pares (intervalo de confiança de 95%)

que o de um grafo randômico, indicando pouca robustez. Entretanto, a resiliência da rede ao *churn* não é comprometida: é necessário que, em média, 64 pares sejam desconectados para que a rede seja particionada. As estratégias preemptiva e pró-ativa, por sua vez, causam maior dispersão das conexões entre pares, reduzindo o coeficiente de clusterização para valores próximos àqueles obtidos pelo modelo randômico. Tal redução se reflete na conectividade dos pares à fonte, a qual tem um ganho de 18,75% em comparação com a estratégia inativa.

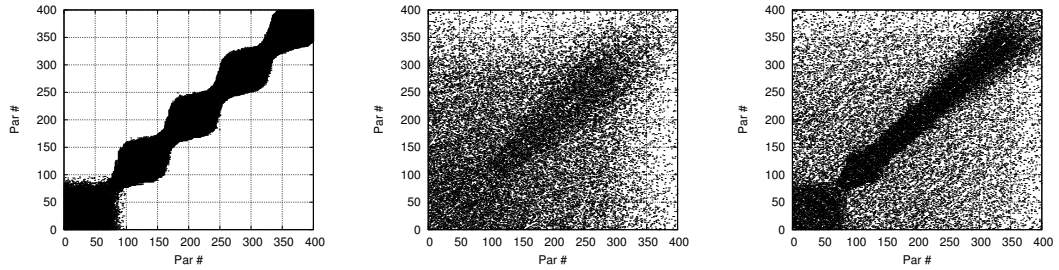
Respondendo a Q1, os resultados mostram que uma topologia gerada pela estratégia inativa é suficientemente robusta em um cenário de *flash crowd*. O agrupamento de pares observado não é um problema pois afeta apenas uma pequena porção dos pares (os últimos a entrarem na rede). As estratégias preemptiva e pró-ativa, por sua vez, geram topologias mais robustas que a inativa em situações de *flash crowd*.

Em relação ao potencial de desempenho da rede, observa-se que todas as três estratégias são capazes de saturar a vizinhança dos pares. Ou seja, nenhum par é prejudicado por possuir menos possibilidades de interações. Considerando a localização da fonte em relação aos demais pares, verifica-se que, em média, a fonte está a 1,92 vizinhos de distância do restante da rede ao adotar a estratégia inativa, apenas 7% pior em relação às demais estratégias. Logo, as estratégias preemptiva e pró-ativa centralizam melhor a fonte, melhorando a disseminação de dados na rede.

Respondendo a Q2, as estratégias preemptiva e pró-ativa apresentam o melhor potencial de desempenho sob *flash crowd*. Tal fato se justifica pela diversificação da vizinhança dos pares, permitindo que a fonte esteja em uma posição mais central na rede. A estratégia inativa não reorganiza a topologia, fazendo com que os pares que chegam por último fiquem mais distantes da fonte em comparação aos primeiros. Do ponto de vista global da rede, entretanto, isso representa um resultado apenas 7% pior em relação às outras estratégias.

5.1.2. Taxa de Entrada Constante

As matrizes de conexões para o cenário em que a entrada de pares é constante (Figura 2) mostram uma mudança drástica na organização da topologia. A estratégia inativa apresenta maior tendência de clusterização: são observados vários grupos formados de acordo com o período em que os pares entraram no sistema. Esta formação topológica decorre da menor taxa de entrada de pares, a qual resulta em menos opções para construir a vizinhança. As estratégias preemptiva e pró-ativa, por sua vez, mantêm a tendência de desfazer os padrões de clusterização da topologia, distribuindo melhor as conexões entre os pares. Para comparar as topologias de forma objetiva, suas métricas foram avaliadas, sendo resumidas na Tabela 2.



(a) Estratégia Inativa (b) Estratégia Preemptiva (c) Estratégia Pró-ativa

Figura 2. Matrizes de conexões das topologias no cenário de taxa de entrada constante dos pares

Métrica	Inativa	Preemptiva	Pró-Ativa	Randômico
κ	(21,88; 22,08)	(63,47; 63,56)	(71,36; 71,50)	(80,0; 80,0)
C	(0,8250; 0,8254)	(0,2049; 0,2050)	(0,2388; 0,2390)	(0,1963; 0,1964)
d	(78,48; 78,49)	(67,35; 67,38)	(73,67; 73,69)	(80,0; 80,0)
P	(4,135; 4,173)	(1,804; 1,805)	(1,800; 1,801)	(1,796; 1,796)

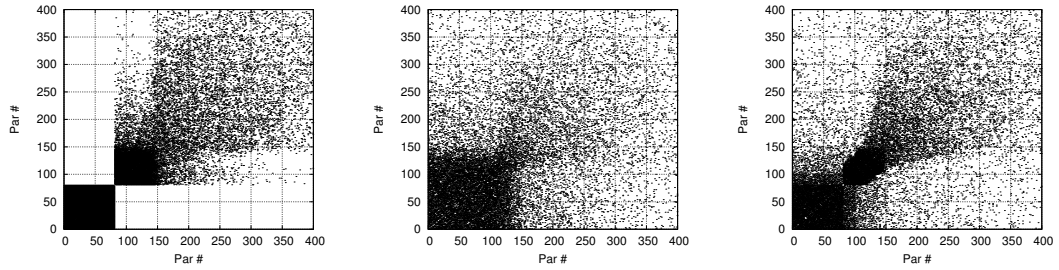
Tabela 2. Métricas no cenário de taxa de entrada constante dos pares (intervalo de confiança de 95%)

O coeficiente de clusterização da estratégia inativa apresenta valores quatro vezes maiores que os do grafo randômico, resultado que indica agrupamento excessivo dos pares. Tal clusterização é prejudicial, pois dificulta ou mesmo impede o acesso de novos pares à fonte (ou uma cópia do conteúdo). Também observa-se que a robustez da topologia é prejudicada pela estratégia inativa: a saída de apenas 5,5% dos pares do sistema pode resultar em um particionamento da rede. As estratégias preemptiva e pró-ativa, por sua vez, reduzem significativamente o grau de clusterização da rede. Essa diversificação das conexões novamente reflete no aumento da robustez da topologia, a qual atinge ganho de aproximadamente 188% e 224%, respectivamente, ao usar as estratégias preemptiva e pró-ativa.

A melhor organização topológica dos pares também se reflete no desempenho da rede. Mesmo embora a estratégia inativa sature a vizinhança de seus pares, estes ficam localizados em média a mais de quatro vizinhos de distância da fonte, prejudicando a disseminação do conteúdo. As estratégias preemptiva e pró-ativa apresentam uma pequena redução na vizinhança média dos pares. Entretanto, elas centralizam a fonte na topologia, aumentando o potencial de disseminação dos dados quando estas estratégias são usadas.

5.1.3. Taxa de Entrada Exponencial Crescente

O último cenário que avalia o impacto da entrada de pares na topologia considera uma taxa de entrada exponencial crescente. A Figura 3(a) ilustra a matriz de conexões da estratégia inativa para este cenário. Observa-se que a baixa taxa de entrada no começo da disseminação faz com que os pares se agrupem em torno da fonte. Como resultado, um clique se forma na topologia, isolando a fonte do restante da rede. Os pares que entram em seguida não conseguem estabelecer conexões com nenhum par pertencente ao clique, criando um segundo componente do grafo. Ambas estratégias preemptiva (Figura 3(b)) e pró-ativa (Figura 3(c)) são capazes de evitar o particionamento da rede, permitindo que todos os pares possuam um caminho à fonte do conteúdo.



(a) Estratégia Inativa (b) Estratégia Preemptiva (c) Estratégia Pró-ativa

Figura 3. Matrizes de conexões das topologias no cenário de taxa de entrada exponencial crescente dos pares

Os valores médios das métricas das topologias, apresentados na Tabela 3, confirmam os resultados observados na Figura 3. A estratégia inativa apresenta o maior grau de clusterização em função do clique gerado pelos primeiros pares que entram no sistema. Este coeficiente de clusterização não é ainda maior devido à posterior entrada massiva de pares, a qual distribui as conexões de modo similar a uma situação de *flash crowd*. Além disso, observa-se que a conectividade de vértice à fonte é 0, ou seja, a rede está particionada, de modo que um subconjunto dos pares não tem acesso aos dados/*stream*. As estratégias preemptiva e pró-ativa, por sua vez, apresentam bons resultados em relação à robustez, sendo o principal evitar o particionamento da rede. Além disso, essas estratégias acrescentam um nível de resiliência ao *churn*, sendo necessário que pelo menos 34 pares abandonem o sistema para que um particionamento possa ocorrer.

Métrica	Inativa	Preemptiva	Pró-Ativa	Randômico
κ	(0,0, 0,0)	(34,13; 34,22)	(38,51; 38,56)	(80,0; 80,0)
C	(0,6750; 0,6754)	(0,2986; 0,2991)	(0,3722; 0,3727)	(0,1963; 0,1964)
d	(48,00; 48,01)	(41,70; 41,72)	(45,88; 45,89)	(80,0; 80,0)
P	(242,42; 242,42)	(1,836; 1,839)	(1,863; 1,865)	(1,796; 1,796)

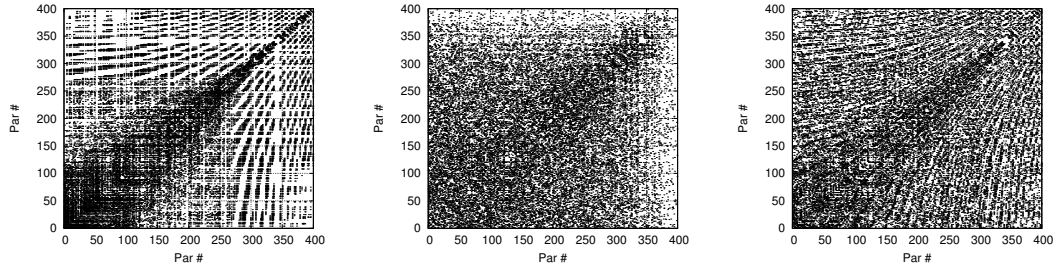
Tabela 3. Resultados das métricas da topologia no cenário de taxa de entrada exponencial crescente dos pares (intervalo de confiança de 95%)

A estratégia inativa apresenta a maior vizinhança média dos pares. Entretanto, apenas 19,75% da rede possui acesso aos dados. Dessa forma, o desempenho do sistema pode ser considerado péssimo, pois o conteúdo não é devidamente disseminado aos interessados, mesmo que a rede possua a capacidade de upload necessária. As estratégias preemptiva e pró-ativa apresentam pequena redução na vizinhança média dos pares, o

que pode comprometer as possibilidades de interação, mas não prejudicam a organização da rede. Os resultados mostram que essas estratégias centralizam a fonte mesmo sob condições adversas.

5.2. Alcançabilidade dos Pares

Um fator que afeta a robustez e organização das topologias é a restrição da alcançabilidade dos pares, fato comum na Internet. Nesse conjunto de cenários avaliados, um subconjunto dos pares não é capaz de receber conexões. Como resultado, as possibilidades de estabelecimento de conexões entre pares são reduzidas. A título de exemplo, a Figura 4 apresenta as matrizes de conexões no cenário de alcançabilidade 50%.

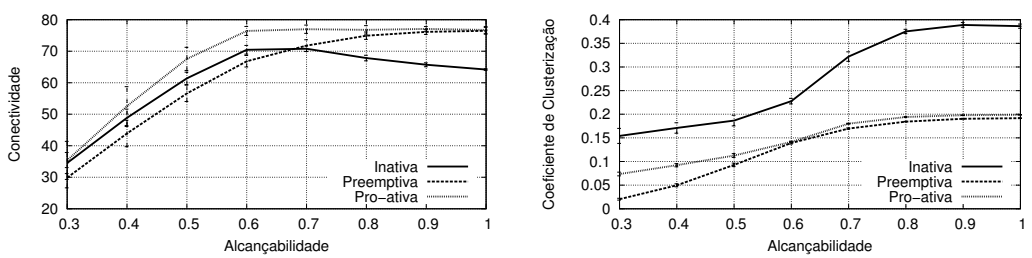


(a) Estratégia Inativa (b) Estratégia Preemptiva (c) Estratégia Pró-ativa

Figura 4. Matriz de conexões das topologias no cenário de 50% de alcançabilidade da rede

Conforme esperado, a densidade das matrizes é menor independentemente da estratégia utilizada. Nesse cenário, a estratégia inativa distribui melhor as conexões entre os pares em comparação com o cenário de alcançabilidade total. Tal fato ocorre porque os primeiros pares não saturam suas vizinhanças, pois não há possibilidades de conexões suficientes. Seguindo a tendência apresentada nos cenários anteriores, as estratégias preemptiva e pró-ativa desfazem qualquer padrão de conexões, distribuindo uniformemente as conexões da rede.

Ao analisar a robustez da rede nesse cenário, ilustrada pela Figura 5, a estratégia pró-ativa apresenta o melhor resultado. A conectividade da topologia usando essa estratégia é a maior considerando todas condições de alcançabilidade, obtendo, em média, 5% a mais de resiliência ao *churn*. Isso é explicado pela forma como estratégia pró-ativa organiza seus pares, distribuindo melhor as conexões e apresentando o comportamento mais próximo a um grafo randômico.

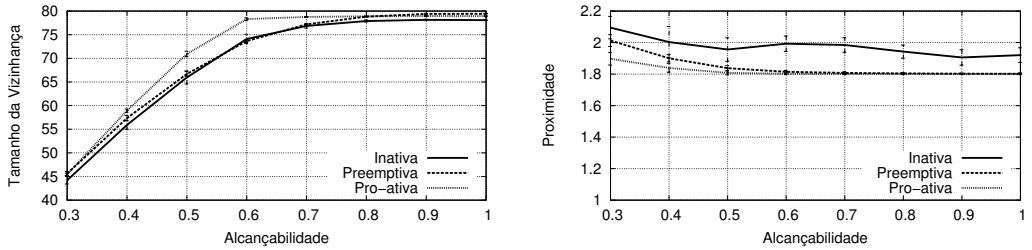


(a) Conectividade média à fonte (b) Coeficiente de Clusterização

Figura 5. Resultados das métricas de robustez nos cenários de alcançabilidade da rede

A estratégia preemptiva apresenta a clusterização dos pares semelhante à pró-ativa em alguns cenários. Especialmente, com alcançabilidade de 70% ou mais, essa estratégia apresenta resistência à saída de pares equiparável à pró-ativa. Entretanto, com 50% ou menos de pares que aceitam conexões, a rede torna-se muito dispersa, causando um efeito negativo na robustez, visto pelos piores resultados obtidos nesses cenários. Por fim, nos casos em que menos de 60% dos pares são alcançáveis, a estratégia inativa apresenta boa robustez em comparação à estratégia pró-ativa. Isso é consequência do agrupamento natural dos usuários, que fica em torno de 0,20, valor esperado em um grafo randômico. A medida que mais pares da rede podem aceitar conexões, há um fenômeno maior de clusterização, afetando negativamente a resiliência da rede.

A Figura 6 apresenta as métricas relativas ao potencial de desempenho da rede. Novamente a estratégia pró-ativa tem os melhores resultados, apresentando as maiores vizinhanças médias entre todas estratégias. Esta mesma estratégia também possui a menor proximidade da fonte à rede. A estratégia que destoa das demais é a inativa, que apresenta, em média, proximidade da fonte 8,5% pior em relação à pró-ativa. Isso evidencia a incapacidade da rede de se reorganizar frente à alterações na topologia quando a estratégia inativa é utilizada.



(a) Grau médio dos vértices (b) Proximidade da fonte

Figura 6. Resultados das métricas de potencial de desempenho nos cenários de alcançabilidade da rede

6. Conclusão

Redes descentralizadas de larga escala baseadas em *swarming* tem seu desempenho influenciado pela forma como os pares se conectam. Além disso, uma boa topologia garante que a rede seja robusta para suportar populações transientes de forma que todos os pares tenham acesso ao conteúdo. Para atingir tal objetivo, os pares utilizam algoritmos de gerência de conexões para manter sua vizinhança. Foram identificadas três estratégias principais para gerência dos vizinhos: inativa, preemptiva e pró-ativa. O presente trabalho foi o primeiro a comparar todas estratégias em cenários comuns, além de avaliá-las detalhadamente considerando métricas de robustez e desempenho.

Os resultados obtidos a partir de um conjunto de cenários apresentam diferentes conclusões. Considerando a taxa de entrada dos pares, o cenário *flash crowd* não mostrou-se estressante para a rede considerando sua organização topológica. Nesse cenário, todas estratégias obtiveram bons resultados. Ao analisar diferentes taxas de entrada, verificou-se um padrão de agrupamento excessivo dos pares, prejudicial à robustez da rede e ao seu potencial de desempenho. Enquanto a estratégia inativa é incapaz de organizar a topologia de modo satisfatório nessas configurações, as estratégias preemptiva e pró-ativa conseguem melhorar sua robustez e potencial de desempenho.

Já em relação à alcançabilidade da rede, conforme esperado, a topologia se torna menos robusta quando menos pares podem receber conexões. Tal ocorre devido ao fato de que, nesse cenário mais adverso, menos conexões são possíveis, limitando a interação entre os pares. Neste cenário, a estratégia pró-ativa obteve novamente os melhores resultados, organizando boas topologias mesmo em condições muito restritivas. A estratégia preemptiva, por sua vez, apresentou bom potencial de desempenho, mas pouca resiliência à saída de pares do sistema quando a alcançabilidade da rede é baixa. Finalmente, a estratégia inativa apresenta queda da robustez quando a porcentagem de pares alcançáveis é alta, devido à alta clusterização dos pares.

Para obter entendimento maior do impacto das estratégias na topologia resultante, pretende-se implantar essas estratégias em aplicações reais tanto de distribuição de dados quanto de *streaming*. Isso permitiria que as estratégias fossem estudadas em condições reais de execução, aumentando a confiabilidade dos resultados obtidos.

Referências

Al-Hamra, A., Liogkas, N., Legout, A., and Barakat, C. (2009). Swarming overlay construction strategies. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1 –6.

Cohen, B. (2003). Incentives build robustness in BitTorrent.

Dhungel, P., Hei, X., Wu, D., and Ross, K. (2011). A measurement study of attacks on bittorrent seeds. In *2011 IEEE International Conference on Communications (ICC)*.

Ferreira, A. (2002). On models and algorithms for dynamic communication networks: The case for evolving graphs. In *4e rencontres francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL'2002)*, pages 155–161. INRIA Press.

Lehmann, M., Müller, L., Antunes, R. S., and Barcellos, M. P. (2012a). Desvendando o impacto da desconexão otimista no bittorrent. In *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Lehmann, M. B., Muller, L. F., Antunes, R. S., and Barcellos, M. P. (2012b). Disconnecting to connect: Understanding optimistic disconnection in bittorrent. In *2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*, pages 138 –148.

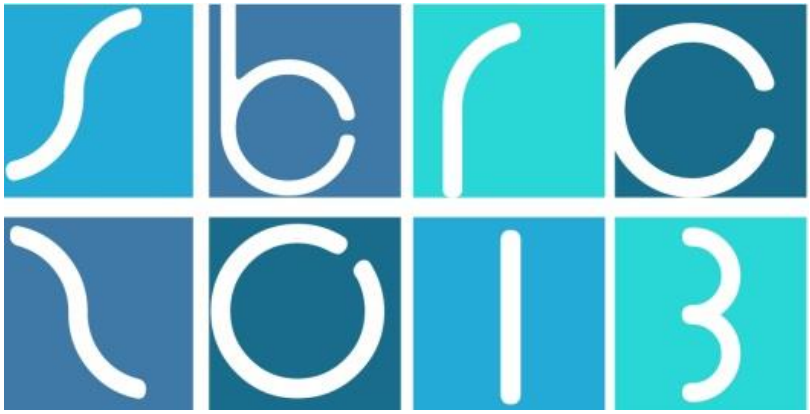
Locher, T., Meier, R., Schmid, S., and Wattenhofer, R. (2007). Push-to-pull peer-to-peer live streaming.

Menasché, D. S., Rocha, A. A., de Souza e Silva, E., Leão, R. M. M., and Towsley, D. (2012). Stability of peer-to-peer swarming systems. In *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Moraes, I. M. and Duarte, O. C. M. B. (2010). Seleção de parceiros em sistemas par-a-par de vídeo sob demanda. In *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Picconi, F. and Massoulie, L. (2008). Is there a future for mesh-based live video streaming? In *2008 Eighth International Conference on Peer-to-Peer Computing*.

Zhang, H., Neglia, G., Towsley, D., and Lo Presti, G. (2007). On unstructured file sharing networks. In *26th IEEE International Conference on Computer Communications (INFOCOM)*, pages 2189–2197.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 8

Redes e Protocolos para RFID

Um mecanismo para garantia de QoS na Internet das Coisas com RFID

Rafael Perazzo Barbosa Mota[1], **Daniel Macêdo Batista**[1]

[1]Departamento de Ciência da Computação – Universidade de São Paulo (USP)
Rua do Matão 1010 – 05508-090 – São Paulo – SP – Brasil

{perazzo,batista}@ime.usp.br

Abstract. *Despite advances in the development of new mechanisms for the Internet of Things (IoT), there are few studies aimed at ensuring QoS requirements and evaluated in scenarios with a large number of nodes. Mechanisms proposed to ensure IoT QoS requirements would improve the performance in environments sensitive to packet loss, and in scenarios of tracking and localization, especially in terms of scalability and network overhead. In this paper we present an algorithm that reduces the amount of messages in IoT scenarios. Simulations confirm the effectiveness of the algorithm. For example, in one scenario, the packet loss rate which was between 10% and 42% was reduced to less than 3%.*

Resumo. *Apesar dos avanços no desenvolvimento de novos mecanismos para a Internet das Coisas (IoT), existem poucos trabalhos voltados para a garantia de requisitos de QoS e que sejam avaliados em cenários com muitos nós. A proposta de mecanismos para garantir requisitos de QoS na IoT melhoraria o desempenho de ambientes sensíveis à perda de pacotes, e em cenários de rastreamento e localização, principalmente em termos de escalabilidade e sobrecarga da rede. Neste artigo é apresentado um mecanismo para garantia de QoS que reduz a quantidade de mensagens na rede em cenários da IoT. Experimentos de simulação confirmam a eficácia do algoritmo. Por exemplo, em um cenário, a taxa de perda de pacotes que era de 10% a 42% foi reduzida para menos de 3%.*

1. Introdução

Atualmente, a maior parte das interações na Internet é realizada entre seres humanos [Miorandi et al. 2012]. No entanto, em um futuro próximo, qualquer "coisa" (*thing*) poderá ser endereçada na grande rede. A Internet, então, tornar-se-á a Internet das coisas (*Internet of things - IoT*). As comunicações serão concebidas não apenas entre humanos mas também entre humanos e coisas e entre coisas sem a interação com seres humanos. Este novo paradigma vem rapidamente adquirindo espaço principalmente no moderno cenário das comunicações sem fio. Em resumo, a Internet das Coisas consiste na presença difusa de uma variedade de coisas ou objetos ao nosso redor, como, por exemplo etiquetas RFID - *Radio Frequency IDentification* (identificação por radiofrequência), telefones celulares inteligentes, redes de sensores sem fio - RSSF, entre outros, que se comunicam a fim de trocar muitas mensagens, além das poucas trocadas por simples sensores [Atzori et al. 2010].

Segundo previsão do NIC (*US National Intelligence Council*), até o ano de 2025, os nós da Internet poderão estar em todas as coisas e permitirão inúmeras

oportunidades para o desenvolvimento tanto econômico como tecnológico mundial [Evdokimov et al. 2011]. Nesta "nova" internet haverá um sem-número de objetos heterogêneos [Liu e Zhou 2012]. Dessa forma, conforme [Miorandi et al. 2012, Nef et al. 2012, Atzori et al. 2010], os diferentes tipos de objetos envolvidos tornam a IoT um paradigma diferente das atuais RSSF. Enquanto os protocolos e os nós em uma RSSF são voltados para cenários geralmente específicos para observação de fenômenos ambientais [Nef et al. 2012], na IoT espera-se expandir este cenário permitindo também aplicações onde os objetos possuam alguma conectividade sem necessariamente precisar lidar com fenômenos ambientais. Neste artigo são analisados cenários que oferecem serviços de localização e rastreamento de objetos, ou seja, cenários caracterizados pela sensibilidade à alta perda de pacotes.

O objetivo deste artigo é propor um mecanismo de QoS para cenários de IoT cujos nós sejam etiquetas RFID e que sejam sensíveis à perda de pacotes. O mecanismo proposto teve seu desempenho avaliado por meio de experimentos simulados no simulador ns-2[1], e os resultados confirmam a eficácia do mecanismo. Por exemplo, em um cenário, a perda de pacotes que era de 10% a 42% foi reduzida para menos de 3% quando o mecanismo foi utilizado. A escolha da tecnologia RFID deve-se ao fato da mesma ser uma das tecnologias chave da IoT assim como a mais adequada para aplicações de rastreamento e localização, devido sua própria característica implícita de identificação.

As contribuições deste artigo são:

- Proposta de um mecanismo para garantia de QoS em cenários IoT sensíveis a perda de pacotes e sua análise de desempenho.
- Desenvolvimento de uma extensão para simular cenários com etiquetas e leitores RFID no ns-2;

Este artigo difere-se dos encontrados na literatura porque realiza experimentos simulados de cenários IoT reais, com o ns-2, e quantidade de etiquetas variáveis, o que possibilita uma análise mais aprofundada do mecanismo de QoS proposto.

O artigo está organizado da seguinte forma: a Seção 2 aborda a contextualização da tecnologia RFID. A Seção 3 descreve os trabalhos relacionados. A Seção 4 detalha o mecanismo de QoS proposto, enquanto os cenários IoT propostos estão explicados na Seção 5. A análise de desempenho, a extensão RFID para ns-2 e os resultados obtidos nas simulações dos cenários, são expostos e discutidos na Seção 6. Finalmente, na Seção 7 são apresentados os trabalhos futuros e as conclusões da pesquisa.

2. Contextualização

A tecnologia RFID refere-se a um sistema de identificação sem fio que permite a comunicação entre etiquetas e leitores, por intermédio de ondas de radiofrequência. A principal vantagem da RFID é que os objetos identificados não precisam estar muito próximos dos leitores, ocasionando uma fácil automação do processo de leitura. A quantidade de etiquetas depende do contexto da aplicação. Elas podem ser empregadas em diversas situações como controle de acesso, ingressos eletrônicos, rastreamento animal, localização e rastreamento, passaportes, entre outras [Finkenzeller et al. 2010].

[1]http://www.isi.edu/nsnam/ns/

De acordo com [Finkenzeller et al. 2010], um sistema de RFID é formado por três componentes básicos:

- *Transponders* ou etiquetas RFID: localizadas nos objetos a serem identificados, armazenando o código de identificação;
- *Transceivers* ou leitores RFID: responsável pelas leituras/escritas nas etiquetas e
- *Middleware* ou aplicação: responsável pelo processamento da informação obtida pelo leitor.

Em geral, sistemas RFID passivos são baseados no fato de que "o leitor fala primeiro", conforme definido no padrão em [EPCglobal 2008]. Assim, o leitor tem a função de enviar requisições para etiquetas ao seu alcance que respondem com seus respectivos identificadores. Estes são recebidos pelo leitor que envia para a aplicação, onde os dados serão processados. Pode-se perceber que a aplicação exerce papel fundamental, pois esta utilizará os dados da forma que lhe for conveniente e pode criar, com uma única tecnologia, um ambiente de Internet das coisas. A título de exemplo, imaginemos um cenário em que diferentes tipos de objetos são etiquetados com etiquetas RFID, leitores são posicionados em pontos estratégicos e existe uma aplicação que em tempo real consegue disponibilizar a localização dos objetos etiquetados para um público específico. A partir deste ponto é possível visualizar uma gama de possibilidades de aplicações e cenários de IoT, adotando-se apenas a tecnologia RFID [Miorandi et al. 2012]. Em resumo, o funcionamento do sistema segue os passos a seguir: (i) O leitor faz a solicitação que é enviada a partir de sua antena de rádio usando um canal denominado de leitor-etiqueta; (ii) A etiqueta recebe a requisição também através de sua antena, prepara um pacote de resposta com seu identificador e (iii) envia de volta pelo canal etiqueta-leitor ; (iv) O leitor recebe o identificador e (v) repassa para a aplicação. Este procedimento constitui o protocolo básico de comunicação da tecnologia RFID.

É importante observar que o paradigma da IoT está fortemente relacionado à efetiva integração RSSF e RFID, pois enquanto no sistema de RFID, geralmente, os nós são passivos e respondem apenas quando solicitados pelos leitores; nas RSSFs, os nós são, na maioria, autônomos e podem, por iniciativa própria, estabelecer uma comunicação em um ambiente IoT [Miorandi et al. 2012]. Este aspecto confirma a expressiva importância dessas duas tecnologias ao tratarmos de Internet das Coisas. Entretanto, o foco deste artigo será na melhoria de cenários de RFID. A integração entre RFID e RSSF é um dos trabalhos futuros conforme apresentado na Seção 7.

3. Trabalhos relacionados

Em [Welbourne et al. 2009] é relatado que na Universidade de Washington foram utilizadas etiquetas RFID para identificar pessoas (voluntárias) e objetos para implementar a Internet das Coisas, em um projeto chamado de "*RFID Ecosystem*", que tem o objetivo de oferecer os serviços de localização e rastreamento das coisas. Vários leitores foram posicionados para abranger os sete andares dos oito mil metros quadrados de um prédio do campus. O projeto contou com quarenta e quatro leitores, cada um equipado com quatro antenas, distribuídos no local. Os leitores enviavam os dados coletados das etiquetas para um servidor central. O mecanismo de funcionamento do RFID Ecosystem baseia-se em consultas periódicas do *software* aos leitores, que buscam a detecção de novas etiquetas e geram um evento denominado TRE - *Tag-reader event*, ou seja, um evento

leitura-etiqueta por antena, por segundo. A partir dos experimentos, os resultados obtidos foram os dados relativos ao número de requisições/respostas realizadas, erros de leitura e confiabilidade das informações fornecidas pelo leitor às aplicações. Por conseguinte [Welbourne et al. 2009] sinalizaram a importância do desenvolvimento de mecanismos que lidem com a sobrecarga da rede, já que este foi um problema identificado, apesar de terem sido utilizadas apenas 324 etiquetas com 44 leitores distribuídos por toda a área de cobertura. A partir deste fato pode-se observar a fragilidade do modelo experimentado, pois foram utilizados muitos leitores, poucas etiquetas e mesmo assim muitas mensagens sobrecarregaram a rede, já que as consultas eram realizadas a cada segundo, gerando na maioria das vezes pacotes e comunicações desnecessárias.

Com relação à Qualidade de Serviço, [Nef et al. 2012] sinaliza a necessidade de se definir tipos de serviços na IoT, para que seja possível a determinação de parâmetros de QoS. O artigo utiliza as RSSF como uma das tecnologias chave, e a partir de suas características genéricas, compara os variados protocolos de acesso ao meio *(MAC - Media access control)* para expor os mecanismos já existentes para prover qualidade dos serviços. No entanto, uma RSSF, como parte de uma IoT, pode necessitar de parâmetros diferentes, já que outros tipos de dispositivos podem também estar inseridos na rede. Finalmente, a partir de uma classificação proposta para as mais diversas aplicações de IoT, propõe-se uma topologia IoT que melhor adequa-se à utilização de RSSF integradas nos cenários. Para cada tipo diferente de aplicação, são descritos os modelos de serviços das mesmas, para que seja possível a identificação e proposta dos parâmetros de QoS.

Segundo [Duan et al. 2011], cenários IoT, orientados a aplicação, consistem de uma integração de várias tecnologias, conforme já sinalizam outros autores, sendo necessária assim uma criteriosa investigação sobre arquiteturas de QoS, para possibilitar na prática, a oferta de serviços com qualidade. Posteriormente, requisitos de QoS são resumidos através da análise das características de aplicações para controle, consultas, monitoramento em tempo real e monitoramento genérico. Cada modelo de aplicação pode requerer parâmetros diferentes de QoS. Dessa forma os autores propõem a organização da IoT em três camadas: Aplicação, Rede e Percepção. Todas com capacidade de prover QoS. A arquitetura proposta visa ajudar pesquisadores da área, para tentar facilitar na busca de solução de problemas relacionados a QoS na IoT.

Diversas propostas de novos mecanismos anti-colisão são discutidos e analisados em [Wu et al. 2013, Han et al. 2012, Leonardo e Victor 2012, Zhong et al. 2012]. No entanto os objetivos das mesmas consiste em diminuir o tempo de singularização em uma única leitura, não sendo consideradas aplicações para a Internet das Coisas. Dessa forma, o mecanismo proposto neste artigo pode ser implementado em quaisquer protocolos previamente propostos na literatura.

Em termos de simulação de cenários de funcionamento de RFID, existem algumas soluções propostas na literatura. Podemos citar o *Rifidi*[2], ns-2[3] , ns-3[4] e OMNet++ [Pal 2012]. O *Rifidi* é um emulador específico para a tecnologia RFID que permite a montagem de cenários com leitores e etiquetas, em diversos tipos de funcionamento. Apesar de utilizado na literatura, o mesmo não dispõe de recursos necessários para a investigação

[2]http://www.transcends.co/

[3]http://www.isi.edu/nsnam/ns/

[4]http://www.nsnam.org/

de análise de desempenho, pois se trata de um *software* direcionado apenas para testes de cenários antes da compra dos equipamentos. O ns-3 é a nova geração do consagrado predecessor ns-2. Lançado em julho de 2008, foi completamente remodelado e reescrito em linguagem C++. Embora possua uma série de melhorias em relação ao anterior, ainda é discretamente aproveitado pela comunidade acadêmica relacionada às redes de computadores, além disso, tem limitada documentação e poucos novos módulos disponíveis. O OMNet++, assim como o ns-2 e ns-3, é também um simulador baseado em eventos discretos, tendo como finalidade preencher as lacunas deixadas entre aplicações livres como o ns e programas comerciais de alto custo. Finalmente, o popular ns-2 continua a ser o simulador mais utilizado, inclusive em publicações atuais, para as diferentes situações, amplamente aceito como ferramenta para validação e testes de arquiteturas, aplicações, tecnologias, protocolos, entre outros. Além disso, o mesmo possui implementações de tecnologias sem fio, como WiMax e RSSF, que poderão possibilitar trabalhos futuros para análises de cenários heterogêneos, característica esta presente na IoT. Por este motivo o ns-2 foi o escolhido para a realização dos experimentos deste trabalho. A fim de facilitar a realização das simulações, e também para auxiliar a comunidade que realiza pesquisas em RFID, nós implementamos e disponibilizamos uma extensão para o ns-2 suportar experimentos com leitores e etiquetas RFID. Nem o ns-3 nem o OMNet++ possuem módulos prontos que permitam simulações de cenários com dispositivos RFID.

4. Mecanismo proposto

Conforme [Welbourne et al. 2009], o excesso de pacotes trocados entre leitores e etiquetas RFID afeta significativamente a escalabilidade da rede e também as garantias de requisitos de QoS das aplicações. O nosso mecanismo visa reduzir essa quantidade de pacotes e pode ser aplicado a quaisquer mecanismos anti-colisão propostos na literatura.

O mecanismo proposto tem como objetivo garantir a qualidade dos serviços oferecidos em cenários de localização e rastreamento, reduzindo as taxas de perdas de pacotes. Uma característica comum pode ser observada nestas situações: As etiquetas não precisam responder a todas requisições do leitor. O fato é justificado pois as etiquetas apenas precisam responder requisições vindas de leitores diferentes da requisição anterior, ou seja, o nó não precisa informar a localização, se a mesma não foi alterada. O mecanismo proposto está detalhado no Algoritmo 1. A Figura 1 ilustra a máquina de estados das etiquetas. A implementa

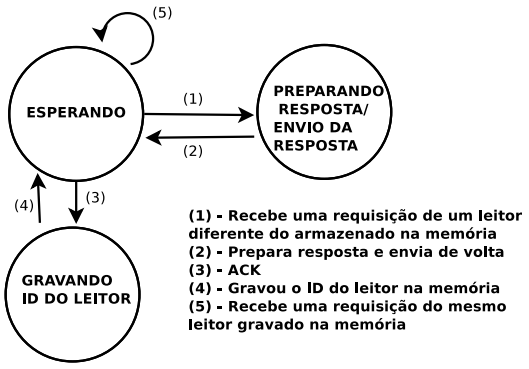


Figura 1. Máquina de estados das etiquetas para o mecanismo proposto

Algoritmo 1: Mecanismo implementado nas etiquetas para garantia de QoS em aplicações da IoT para rastreamento e localização

Entrada: Pacote de solicitação do leitor
Saída: Pacote contendo o ID da etiqueta, que é enviado após um tempo aleatório, conforme o Algoritmo 2 (Subseção 6.1), ou $\emptyset$

```
1  A cada novo pacote de requisição recebido pela etiqueta:
2  se (Primeira requisição recebida) então
3  |  Enviar o pacote de resposta com seu identificador
4  |  se (Receber pacote com ACK do leitor) então
5  |  |  Entrar em modo silencioso até receber uma requisição de um leitor
   |  |  diferente
6  |  fim
7  senão
8  |  se (A origem da requisição for a mesma já armazenada na etiqueta)
   |  então
9  |  |  Não responde a requisição
10 |  senão
11 |  |  Enviar pacote de resposta
12 |  |  se (Receber pacote com ACK do leitor) então
13 |  |  |  Gravar o identificador do leitor
14 |  |  |  Entrar em modo silencioso até receber uma requisição de um
   |  |  |  leitor diferente
15 |  |  fim
16 |  fim
17 fim
```

Observa-se que a etiqueta possui três estados: (i) "Esperando": A etiqueta está esperando uma requisição ou recebeu uma requisição do mesmo leitor que possui armazenado em sua memória. Caso a etiqueta receba uma requisição de um leitor diferente a mesma passa para o estado de (ii)"preparação e envio de resposta", e volta ao estado esperando. Caso o leitor receba a resposta corretamente, o mesmo envia um ACK de confirmação para etiqueta, que no estado "esperando" passa para o estado (iii) "gravando ID do leitor" onde é armazenado o ID do novo leitor, e posteriormente volta-se ao estado esperando. O Algoritmo 1, para implementação nas etiquetas realiza exatamente os passos descritos pelo diagrama de estados da Figura 1, ou seja, a etiqueta verifica a origem da requisição (linhas 2 e 7) e responde apenas se a origem for diferente do ID já armazenado na memória (linhas 3 e 11) . Ao responder, a etiqueta grava a nova ID do leitor apenas ao receber um ACK de confirmação (linhas 12 e 13). Como pode-se observar, a saída do Algoritmo 1 depende da execução do Algoritmo 2 que define o instante em que o pacote com o ID da etiqueta, caso seja gerado, será enviado. O Algoritmo 2 será apresentado na Subseção 6.1.

5. Cenários e experimentos

A fim de avaliar o desempenho do mecanismo proposto na Seção 4, dois cenários de IoT com dispositivos RFID foram simulados. A lista abaixo descreve os dois cenários.

- i) Um cenário que simula uma sala de aula, com turmas de 30 a 430 alunos variando de 50 em 50 que: Permanecem na sala durante a primeira aula (cinquenta minutos); Saem para o intervalo (20 minutos); Voltam para a segunda aula (cinquenta minutos); Saem do prédio onde se localiza a sala de aula. Foram utilizados três leitores, um na sala de aula, outro no pátio do intervalo e o último na saída do prédio. O cenário está ilustrado na Figura 2a;
- ii) Um cenário com cinco leitores, representando parte de uma feira de exposições, e quantidade de nós entre 50 e 1050, variando de 100 em 100, com posições e movimentos aleatórios. A Figura 2b ilustra o cenário.

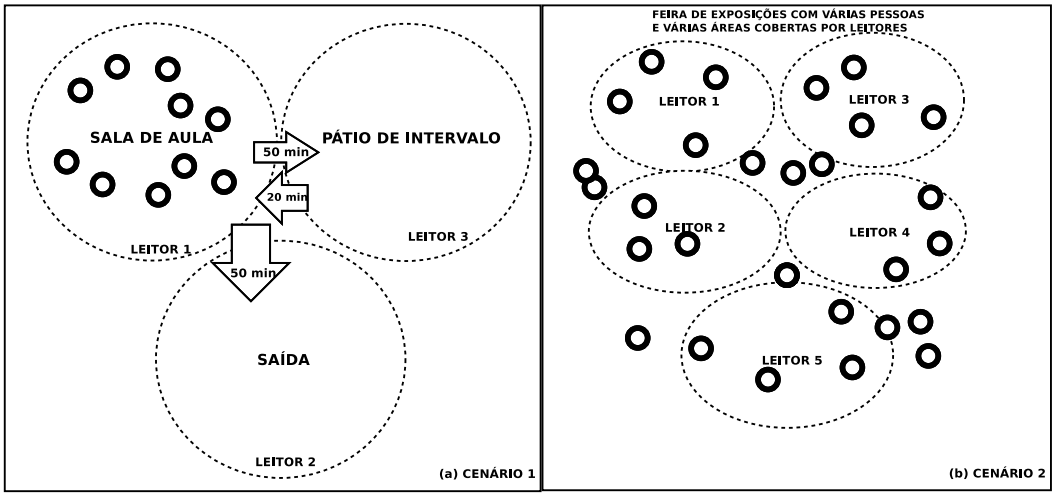


Figura 2. Cenários modelados

Os seguintes parâmetros foram levados em consideração:

- O tempo do experimento para o cenário (i) foi de cento e dez minutos, com leituras enviadas a cada três segundos. Para o cenário (ii) simulou-se trinta minutos, com solicitações a cada cinco segundos.
- O mecanismo RTS/CTS - *Request to Send / Clear to Send* - (Requisição para enviar / Livre para enviar) e o protocolo de roteamento foram desabilitados por não fazerem parte do padrão de comunicação RFID [Finkenzeller et al. 2010]. Estes parâmetros foram ajustados a partir do protocolo de acesso ao meio (*Medium Access Control - MAC* 802.11), simulando assim o MAC do sistema RFID.
- A potência de transmissão - *Power transmission - Pt* foi diminuída a fim de alcançar valores próximos dos reais, de um a dez metros, com os parâmetros *Pt_ 0.28 (mW) e RXThresh_ (potência mínima para que os pacotes sejam recebidos pelos receptores) 2.12249e-07 (W)* [Chen et al. 2007], calculados a partir de uma a ferramenta que encontra-se em `ns-2.35/indep-utils/propagation/threshold.cc` na árvore de diretório descompactada do ns-2 versão 2.35. Necessitou-se modificar os parâmetros de potência, pois o protocolo 802.11 possui a potência do sinal muito maior do que um leitor ou etiqueta RFID, possibilitando desta forma a simulação correta da potência do sinal.
- As taxas de transmissão dos canais leitor-etiqueta e etiqueta-leitor foram definidas como 9Kbps e 128Kbps [Chen et al. 2007], respeitando valores reais.

Cada cenário foi executado vinte vezes para cada quantidade de nós. O percentual de perda de pacotes para cada simulação foi calculado através da média aritmética entre as vinte medições, conforme a Equação 1. A Equação 2 mostra a fórmula de cálculo do tráfego gerado (quantidade de Kbytes transferidos) pelas respostas das etiquetas, no sentido etiqueta-leitor. As simulações foram realizadas em um servidor equipado com processador Intel Core i7-2700K 3.5Ghz, 16GB de memória RAM e 1TB de espaço em disco rodando o sistema operacional Debian GNU/Linux versão 6.0.

$$Tp = \frac{\sum_{i=1}^{20}(\frac{\sum_{j=1}^{ql}(\frac{\sum_{k=1}^{s}(\frac{d_k}{d_k + r_k})}{s})}{ql})}{20} \quad (1)$$

$$Qt = \frac{\sum_{i=1}^{20}(\frac{(d_i + r_i) * 8}{1000})}{20} \quad (2)$$

onde:

- Tp = Percentual de taxa de perda de pacotes (0-1);
- Qt = Quantidade, em *KBytes*, de tráfego gerado pelas etiquetas aos leitores;
- d_i = Quantidade de pacotes descartados simulação *i*;
- r_i = Quantidade de pacotes recebidos na simulação *i*;
- i = Número da simulação;
- ql = Quantidade de leitores;
- s = Quantidade de requisições;
- d_k = Quantidade de pacotes descartados na requisição *k*;
- r_k= Quantidade de pacotes recebidos na requisição *k* ;
- 8 - Tamanho em *bytes* do pacote RFID.

6. Análise de desempenho

A análise de desempenho dos cenários com e sem a implementação do mecanismo proposto foi realizada através de simulações com o *ns-2*, seguindo a Equação 1 e a Equação 2 para a obtenção dos valores que serão mostrados em gráficos nesta seção.

6.1. Extensão RFID para ns-2

O *software* escolhido, ns-2, não possui pacotes ou módulos próprios ou de terceiros que modelem os componentes de um sistema de RFID, como etiquetas e leitores e seu protocolo de comunicação. Na literatura também não existem publicações que ofereçam tal

funcionalidade. Por isso foi implementado uma extensão para o ns-2 que modela o funcionamento de componentes RFID. Os parâmetros da camada de enlace foram configurados a partir do padrão 802.11, ajustando-os para adequarem-se aos valores reais do padrão RFID.

A extensão desenvolvida considera as seguintes características existentes em sistemas RFID:

- As etiquetas não iniciam uma comunicação. Apenas respondem a uma requisição de um leitor, informando seu identificador.
- Os leitores enviam pacotes de difusão, solicitando que todas as etiquetas em seu alcance retornem seus identificadores. Estes, quando recebidos, são enviados a uma aplicação central;
- A aplicação central realiza todo processamento relacionado aos identificadores das etiquetas, e suas respectivas localizações a partir da informação recebida pelo leitor;
- O mecanismo anti-colisão das etiquetas é baseado em um algoritmo probabilístico (Algoritmo 2) em que a etiqueta espera um tempo aleatório entre zero e dois segundos, antes de responder ao leitor, minimizando, mas não eliminando, a possibilidade de colisão de pacotes na chegada ao leitor. Inseri esta frase: Este tipo de solução foi escolhido para generalizar as soluções probabilísticas propostas na comunidade científica.

Algoritmo 2: Algoritmo anti-colisão probabilístico implementado nas etiquetas

Entrada: Pacote de solicitação do leitor
Saída: Envio do pacote contendo o identificador da etiqueta
1 Desempacota a requisição recebida do leitor;
2 Sorteia número aleatório entre 0 e 2 segundos (t), baseado em uma distribuição uniforme;
3 Prepara o pacote de resposta;
4 Agenda o envio do pacote para *t* segundos.

A extensão[5] pode ser usada para modelar diversos cenários, com vários leitores e várias etiquetas espalhadas por toda uma área predefinida. Além disso, pode-se fazer com que as etiquetas movimentem-se, como se estivessem representando uma pessoa em movimento, ou um objeto pertencente a uma pessoa em movimento. Características como as apresentadas na seção 5 devem ser configuradas no arquivo *tcl* de modelagem do cenário, pois tratam-se de características da camada de acesso ao meio. Já os tipos de etiquetas suportadas seguem o padrão definido em [EPCglobal 2008].

As seguintes modificações foram realizadas no ns-2, versão 2.35:

- Criação dos agentes *RfidReader* (arquivos rfidReader.cc e rfidReader.h) e *RfidTag* (arquivos rfidTag.cc e rfidTag.h);

[5]A extensão desenvolvida está melhor descrita e validada no sítio eletrônico `http://www.ime.usp.br/~perazzo/rfid_module.php`

- Criação do tipo de pacote *RfidPacket* (rfidPacket.cc e rfidPacket.h), com os campos EPC (identificador da etiqueta) e ID (identificador do leitor que fez a solicitação);
- Inclusão do novo pacote *RfidPacket* no arquivo *common/packet.h*;
- Inclusão dos agentes *RfidReader* e *RfidTag* no arquivo *tcl/lib/ns-default.tcl* que é responsável pela disponibilização do acesso aos cabeçalhos do novo pacote;
- Alteração do arquivo *Makefile.in*, para inclusão dos agentes *rfidReader.o, rfidTag.o e rfidPacket.o*;

A Figura 3 ilustra o diagrama de classes criadas, com seus atributos e métodos. A classe *RfidReader* possui os atributos *id_* que representa o identificador do leitor, *tagEPC_* que armazena o código EPC recebido de uma etiqueta e *singularization_* que configura se o leitor requer ou não algum mecanismo anti-colisão, como o Algoritmo 2. A entidade etiqueta possui os atributos *id_* que armazena o identificador do leitor que enviou a requisição e *tagEPC_* que representa o próprio identificador da etiqueta. Já um pacote RFID é formado pelos seguintes campos:

- id_: Identificador do leitor que enviou a requisição;
- tagEPC_: Identificador da etiqueta que enviou a resposta;
- tipo_: Direção do fluxo: 0 para etiqueta-leitor e 1 para leitor-etiqueta;
- singularization_: 0 para nenhum mecanismo e 1 para mecanismo probabilístico;
- service_: 0 sem mecanismo de QoS e 1 com mecanismo de QoS.

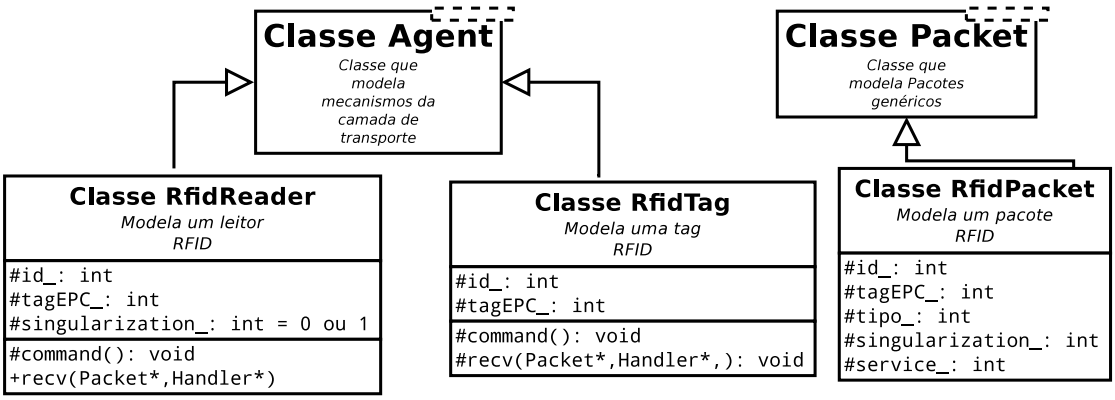


Figura 3. Diagrama de classes criadas

6.2. Resultados e discussão

As Figuras 4 e 5 apresentam gráficos que plotam o percentual de perda de pacotes em função da quantidade de nós para os cenários modelados. Duas curvas são apresentadas em cada gráfico. Uma delas, identificada como "Cenário 1 sem mecanismo" apresenta os resultados sem a implementação do mecanismo proposto no Algoritmo 1. A outra curva, identificada como "Cenário 1 com mecanismo" apresenta os resultados com a implementação do mecanismo proposto no Algoritmo 1. O Algoritmo 2 foi utilizado para ambas as curvas, ou seja, as etiquetas sempre esperam um tempo aleatório antes de responder, a fim de reduzir o número de colisões.

Observa-se no Cenário 1 (Figura 4) que o percentual de perdas de pacotes permanece constante quando a turma varia entre 30 e 180 alunos, mantendo uma taxa de perdas abaixo de 5%. Este fato é justificado pela característica do leitor que consegue lidar bem

com estas quantidades de nós. Percebe-se também que após os 230 alunos, a taxa de perda cresce significativamente, caracterizando que o leitor está no limite de sua capacidade de leituras, gerando perdas que podem inviabilizar a implantação do cenário. Ainda na Figura 4 pode-se observar que o mecanismo de QoS proposto levou os índices de perdas para um valor constante baixo, independente da quantidade de etiquetas, viabilizando a implementação do cenário sem que haja a necessidade de aquisição de novos leitores. Comparando o percentual de perdas no cenário com 430 nós observa-se que houve uma diminuição de aproximadamente 95% quando o mecanismo de QoS foi implementado.

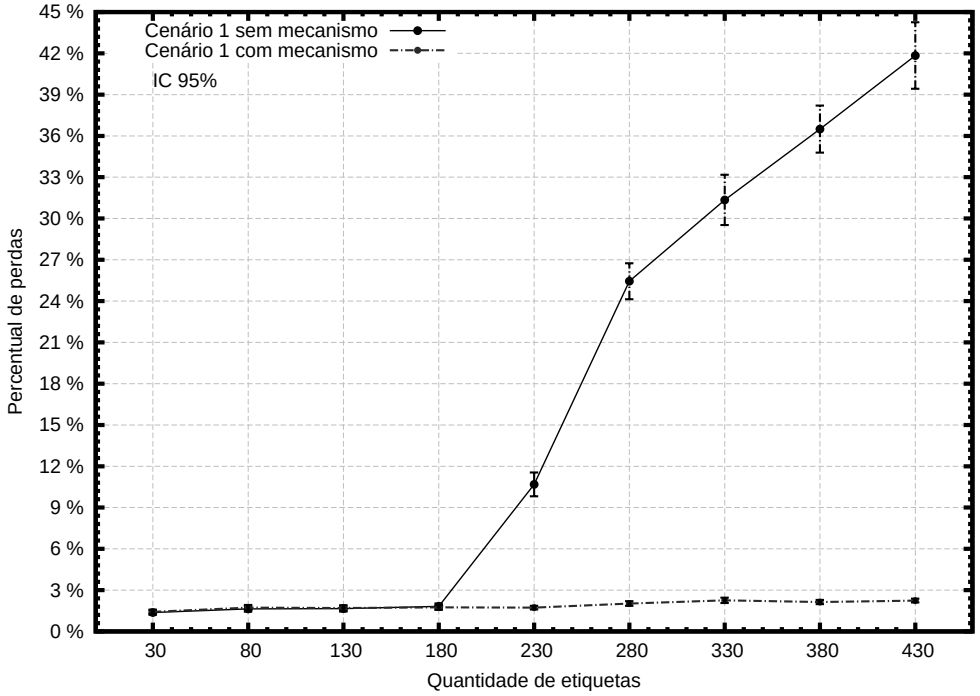


Figura 4. Percentual de perdas x Quantidade de etiquetas para o Cenário 1

O Cenário 2, que simula uma feira de exposições, onde diversas pessoas movimentam-se aleatoriamente por vários locais diferentes, revela pequenos percentuais de perdas até os 450 nós, conforme Figura 5. Quando o mecanismo proposto não foi utilizado, as perdas chegaram a quase 18% quando haviam até 1050 etiquetas. O mecanismo proposto mostrou-se bastante eficiente, pois assim como no Cenário 1, a curva permanece praticamente com valor constante pequeno. Comparando os resultados quando haviam 1050 etiquetas observa-se que a utilização do mecanismo levou a uma redução de 81% na porcentagem de perdas de pacotes.

Para cada um dos cenários, percebe-se que o a utilização do algoritmo anti-colisão não resolve o problema de falta de escalabilidade dos ambientes, ou seja, se aumentarmos muito a quantidade de nós, a taxa de perdas elevar-se-á significativamente. Este fato é resolvido com o mecanismo de QoS proposto.

As Figuras 6 e 7 apresentam a quantidade de Kbytes transferidos na rede nos Cenários 1 e 2 respectivamente. Como era de se esperar, a implementação do mecanismo de QoS reduz significativamente a quantidade de bytes na rede, o que trouxe como consequência a redução na taxa de perdas. Por exemplo, no Cenário 1, com 430 nós, a

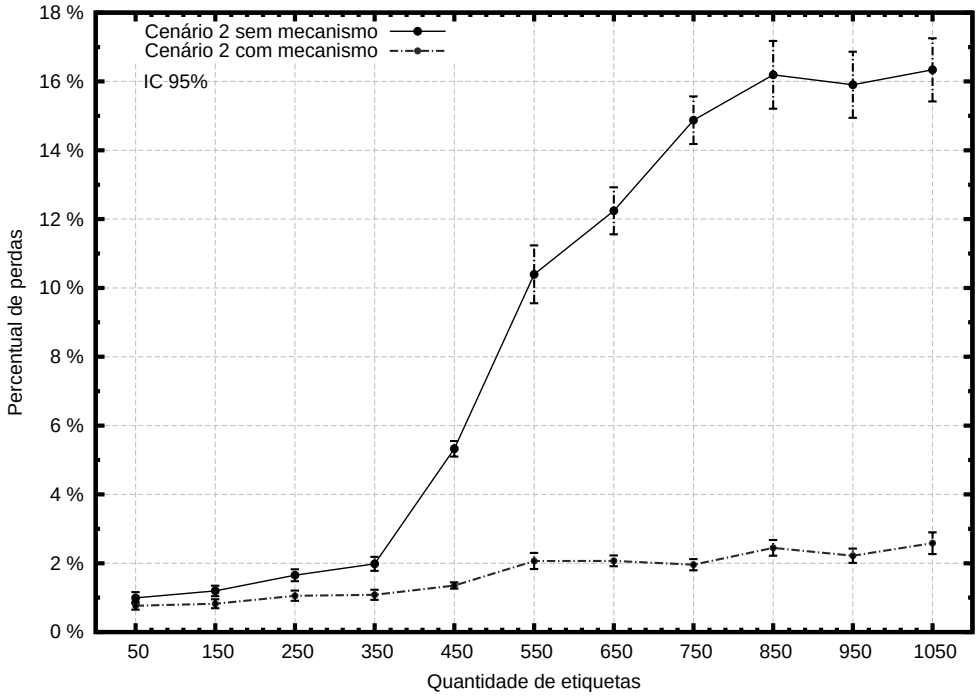


Figura 5. Percentual de perdas x Quantidade de etiquetas para o Cenário 2

quantidade de bytes foi reduzida em 84%. No cenário 2, a redução com a quantidade de 1050 etiquetas, foi de 63%.

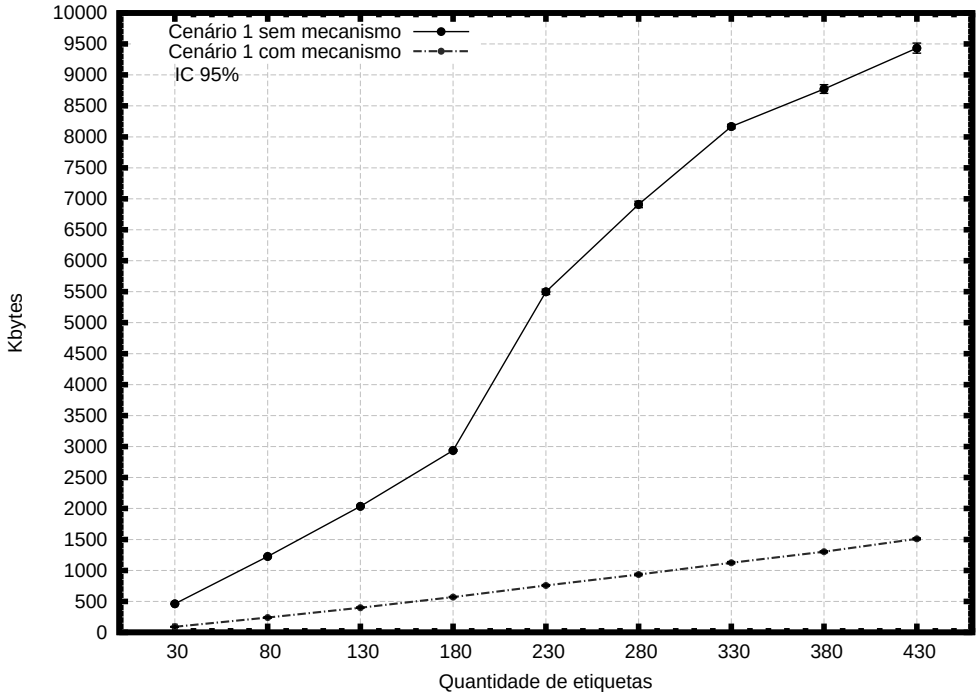


Figura 6. Quantidade média de Kbytes transferidos no Cenário 1

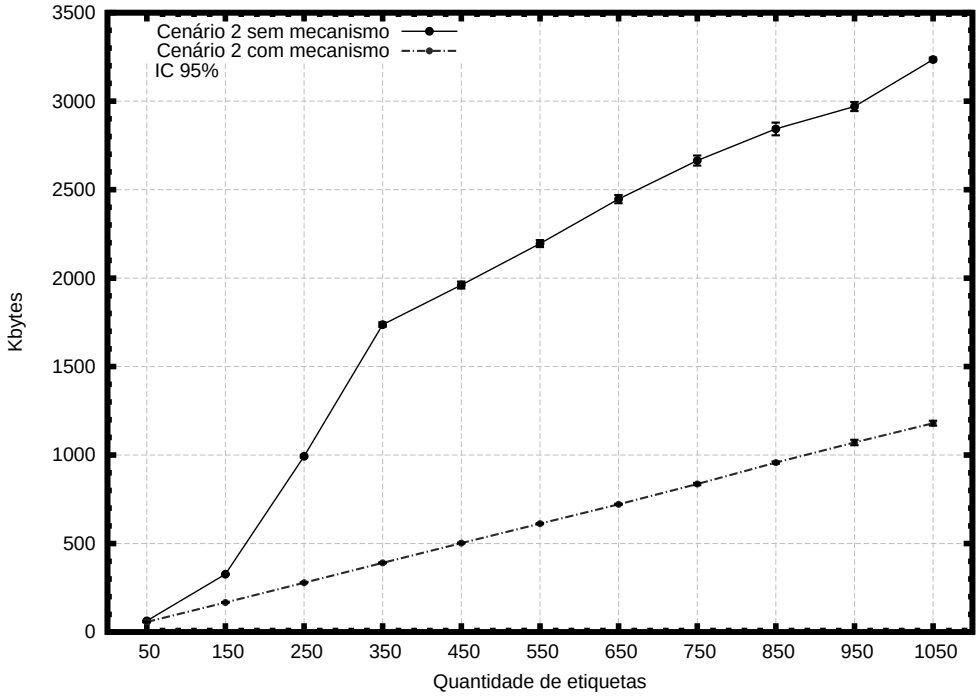


Figura 7. Quantidade média de Kbytes transferidos no Cenário 2

7. Conclusões

Este artigo analisou o desempenho de cenários de IoT com RFID, propôs e validou um mecanismo de QoS para cenários sensíveis à perda de pacotes e apresentou uma extensão à aplicação *ns* que modela a tecnologia RFID para utilização na IoT. Os resultados mostraram que o mecanismo implementado para a IoT consegue diminuir a taxa de perdas, assim como a quantidade de Kbytes transferidos na rede.

Dada a crescente utilização de RFID na prática, como por exemplo a utilização em passaportes e uniformes escolares[6] no Brasil, pode-se observar a importância da pesquisa realizada, visto que em um futuro não tão distante, as aplicações IoT, cada vez mais, farão parte da rotina das pessoas.

Como trabalhos futuros propomos ampliar o escopo da extensão desenvolvida, para que seja possível a realização de novos experimentos para simulação de novos mecanismos, assim como a integração com outras tecnologias.

Referências

Atzori, L., Iera, A., e Morabito, G. (2010). The Internet of Things: A Survey. *Computer Networks*, 54(15):2787–2805.

Chen, Q., Schmidt-Eisenlohr, F., Jiang, D., Torrent-Moreno, M., Delgrossi, L., e Hartenstein, H. (2007). Overhaul of IEEE 802.11 Modeling and Simulation in ns-2. In *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, MSWiM '07, páginas 159–168.

[6]http://www.dpf.gov.br/servicos/passaporte/passaporte-eletronico/
http://www.congressorfid.com.br/entrevista-edilson/

Duan, R., Chen, X., e Xing, T. (2011). A QoS Architecture for IOT. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, páginas 717 –720.

EPCglobal, I. (2008). EPCTM Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 1.2.0.

Evdokimov, S., Fabian, B., Günther, O., Ivantysynova, L., e Ziekow, H. (2011). *RFID and the Internet of Things: Technology, Applications, and Security Challenges*. Now Publishers Inc.

Finkenzeller, K. et al. (2010). *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-field Communication*. Wiley.

Han, H., Park, J., e Lee, T.-J. (2012). RFID Anti-Collision Protocol for Monitoring System of Tags in Motion. In *Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on*, páginas 318–321.

Leonardo, D. e Victor, M. (2012). Adding Randomness to the EPC Class1 Gen2 Standard for RFID Networks. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, páginas 609–614.

Liu, Y. e Zhou, G. (2012). Key Technologies and Applications of Internet of Things. In *Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference on*, páginas 197 –200.

Miorandi, D., Sicari, S., Pellegrini, F. D., e Chlamtac, I. (2012). Internet of Things: Vision, Applications and Research Challenges. *Ad Hoc Networks*, 10(7):1497 – 1516.

Nef, M., Perlepes, L., Karagiorgou, S., Stamoulis, G., e Kikiras, P. (2012). Enabling QoS in the Internet of Things. In *CTRQ 2012, The Fifth International Conference on Communication Theory, Reliability, and Quality of Service*, páginas 33–38.

Pal, D. (2012). A Comparative Analysis of Modern Day Network Simulators. In Wyld, D. C., Zizka, J., e Nagamalai, D., editors, *Advances in Computer Science, Engineering & Applications*, volume 167, páginas 489–498. Springer Berlin / Heidelberg.

Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Balazinska, M., e Borriello, G. (2009). Building the Internet of Things Using RFID: The RFID Ecosystem Experience. *Internet Computing, IEEE*, 13(3):48 –55.

Wu, H., Zeng, Y., Feng, J., e Gu, Y. (2013). Binary Tree Slotted ALOHA for Passive RFID Tag Anticollision. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):19–31.

Zhong, W., Chen, J., Wu, L., e Pan, M. (2012). The Application of ALOHA Algorithm to Anticollision of RFID Tags. In *Measurement, Information and Control (MIC), 2012 International Conference on*, volume 2, páginas 717–720.

AMAS: Anonimato e Autenticação Mútua em Sistemas RFID com Protocolos Anticolisão baseados em Árvore

Bruno Gentilini D'Ambrosio, Paulo André da S. Gonçalves

Centro de Informática (CIn)- Universidade Federal de Pernambuco (UFPE)
50.740-560 – Recife – PE – Brasil

{bgda,pasg}@cin.ufpe.br

Abstract. *The most recent tag-reader mutual authentication schemes for RFID systems based on passive tags, SAMA and SEAS, are able to maintain anonymity of the tags. However, a minimum requirement is that the tag's real ID must never be transmitted in clear text during any message exchange with the reader(s). Moreover, the real ID is commonly used and transmitted in plain text during the execution of tree-based anticollision protocols. This execution precedes the authentication process, so that the anonymity of the tags can not be guaranteed. This paper proposes a scheme to be used with tree-based anticollision protocols that allows tag-reader mutual authentication and preserves the anonymity of the tags. The proposed scheme, namely AMAS (Anonymous Mutual Authentication Scheme), is designed with focus on systems that use passive tags, which have limited computing capabilities. The proposal introduces the use of random and temporary IDs since the execution of the tree-based anticollision protocol, not allowing an attacker to correlate these IDs with the real IDs.*

Resumo. *Os esquemas mais atuais de autenticação mútua etiqueta-leitor para sistemas RFID baseados em etiquetas passivas, o SEAS e o SAMA, são capazes de manter o anonimato das etiquetas. Mas para isso, um requisito mínimo necessário é nunca transmitir em claro o ID real delas durante qualquer troca de mensagem com o(s) leitor(es). Por outro lado, o ID real é comumente utilizado e transmitido em claro durante a execução de protocolos anticolisão baseados em árvore. Essa execução precede o processo de autenticação, fazendo com que o anonimato das etiquetas não possa ser garantido. Este artigo propõe um esquema para ser utilizado com protocolos anticolisão baseados em árvore que permite a autenticação mútua etiqueta-leitor e preserva o anonimato das etiquetas. O esquema proposto, denominado AMAS (Anonymous Mutual Authentication Scheme), é projetado com foco em sistemas com etiquetas passivas, as quais possuem recursos computacionais limitados. A proposta introduz o uso de IDs aleatórios e temporários desde a execução do protocolo anticolisão baseado em árvore, não permitindo a um atacante correlacionar tais IDs com os IDs reais.*

1. Introdução

Em sistemas RFID (*Radio Frequency IDentification*), o controle de acesso das etiquetas ao meio é arbitrado pelo leitor através do uso de um protocolo anticolisão [Klair et al. 2010]. Nos protocolos anticolisão baseados em árvore, o leitor envia uma requisição às etiquetas. Cada etiqueta que satisfaz à requisição responde com seu identificador único (ID).

Quando duas ou mais etiquetas respondem ao mesmo tempo, ocorre uma colisão. O leitor faz, então, uma série de requisições, dividindo recursivamente essas etiquetas em subgrupos até que apenas uma etiqueta responda. O processo descrito pode ser representado por uma árvore. A raiz representa a população de etiquetas a ser identificada. Os nós intermediários representam subgrupos de etiquetas que colidiram ao responderem a uma mesma requisição do leitor. Cada folha representa a resposta de uma única etiqueta a uma requisição do leitor, permitindo sua identificação pelo ID, ou ainda, sua seleção para qualquer outra comunicação exclusiva com essa etiqueta e prevista pela aplicação, como por exemplo, um processo de autenticação mútua etiqueta-leitor [Klair et al. 2010].

Prover a autenticação mútua etiqueta-leitor é um dos maiores desafios em sistemas RFID baseados em etiquetas passivas devido à impossibilidade de se utilizar primitivas criptográficas como aquelas baseadas na função SHA-256 e no AES (*Advanced Encryption Standard*) [Feldhofer and Rechberger 2006]. Isso ocorre devido às limitações de recursos computacionais dessas etiquetas, as quais possuem no máximo 4.000 portas lógicas dedicadas aos mecanismos de segurança [Myneni et al. 2011, Misra et al. 2009]. Além disso, por limitações de tempo de processamento e de consumo de energia, a quantidade máxima de ciclos de relógio utilizada por tais mecanismos está limitada em 220 [Myneni et al. 2011, Misra et al. 2009].

Diversos esquemas buscam prover autenticação em sistemas RFID com etiquetas passivas [Peris-Lopez et al. 2006, Misra et al. 2009, Myneni et al. 2011]. Dentre eles, o SAMA (*Serverless Anonymous Mutual Authentication*) [Myneni et al. 2011] e o SEAS (*Secure and Efficient Anonymity Scheme*) [Misra et al. 2009] são os mais atuais para prover autenticação mútua etiqueta-leitor. Esses dois esquemas são capazes de manter o anonimato das etiquetas. Mas para isso, um requisito mínimo necessário é nunca transmitir em claro o ID real dessas etiquetas durante qualquer troca de mensagem com o(s) leitor(es). Por outro lado, o ID real das etiquetas é comumente utilizado e transmitido em claro durante a execução de protocolos anticolisão baseados em árvore. Essa execução precede o processo de autenticação, fazendo com que o anonimato das etiquetas não possa ser garantido.

Este artigo propõe um mecanismo para ser utilizado com protocolos anticolisão baseados em árvore que permite a autenticação mútua etiqueta-leitor e preserva o anonimato das etiquetas. O esquema proposto, denominado AMAS (*Anonymous Mutual Authentication Scheme*), é projetado com foco em sistemas que utilizam etiquetas passivas. Assim sendo, as limitações de recursos computacionais dessas etiquetas são levadas em consideração. A proposta introduz o uso de IDs aleatórios e temporários desde a execução do protocolo anticolisão baseado em árvore, não permitindo a um atacante correlacionar tais IDs com os IDs reais. Adicionalmente, este artigo provê uma análise da segurança do esquema proposto e avalia o seu custo em termos de quantidade de portas lógicas e de ciclos de relógio.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve o modelo de sistema e o modelo de ameaça considerados para o desenvolvimento deste trabalho. A Seção 4 apresenta o esquema proposto de anonimato e autenticação mútua. Em seguida, a Seção 5 apresenta uma análise da segurança do esquema proposto e avalia o seu custo em termos de quantidade de portas lógicas e ciclos de relógio. Por fim, a Seção 6 apresenta a conclusão do trabalho.

2. Trabalhos Relacionados

Existem diversas propostas de mecanismos que lidam com requisitos de autenticação e anonimato em sistemas RFID. Essas propostas podem ser divididas em três classes distintas [Misra et al. 2009]: as baseadas em funções *hash* [Dimitriou 2005, Lee et al. 2005, Weis et al. 2004, Yang et al. 2005]; as baseadas em algoritmos criptográficos [Dimitriou 2006, Dominikus et al. 2005, Feldhofer et al. 2004, Feldhofer and Rechberger 2006] e; as baseadas em operações lógicas e bit a bit [Peris-Lopez et al. 2006, Misra et al. 2009, Myneni et al. 2011].

As propostas da primeira classe se baseiam em funções *hash* como: SHA-1, SHA-256, MD4 e MD5. Um problema das propostas dessa classe é a utilização de um número significativo de portas lógicas e de ciclos de relógio. Em particular, a função MD4 é a menos custosa entre as citadas e requer 7.350 portas lógicas e 456 ciclos de relógio para que possa ser implementada [Feldhofer and Rechberger 2006]. As propostas da segunda classe se baseiam em algoritmos criptográficos como o AES, o qual é capaz de prover um nível adequado de segurança ao sistema. As propostas dessa classe também consomem uma quantidade significativa de postas lógicas e ciclos de relógio. Em [Feldhofer and Rechberger 2006], por exemplo, a implementação mais otimizada do AES requer 3.400 portas lógicas e 1.032 ciclos de relógio.

Como dito anteriormente, as etiquetas passivas possuem no máximo 4.000 portas lógicas para uso dedicado dos mecanismos de segurança e a quantidade máxima de ciclos de relógio utilizada por tais mecanismos está limitada em 220 [Myneni et al. 2011, Misra et al. 2009]. Por causa disso, surgiram propostas baseadas em operações lógicas e bit a bit [Peris-Lopez et al. 2006, Misra et al. 2009, Myneni et al. 2011]. Essas propostas buscam prover anonimato e autenticação utilizando mecanismos menos custosos em termos de portas lógicas e ciclos de relógio para viabilidade de uso em etiquetas passivas.

O M^2AP (*Minimalist Mutual-Authentication Protocol*) [Peris-Lopez et al. 2006] é um protocolo de autenticação mútua etiqueta-leitor que busca preservar o anonimato das etiquetas. Nesse protocolo, a etiqueta possui dois *IDs*: o primeiro é o ID real e o segundo é conhecido como IDS (*index-pseudonym*). Este último é um pseudônimo que funciona como um índice e permite ao leitor encontrar numa tabela no banco de dados onde estão armazenadas as informações da etiqueta correspondente.

O M^2AP é realizado em duas fases: autenticação e atualização. Na fase de autenticação, o leitor e a etiqueta trocam duas mensagens. Essas mensagens são construídas de forma pré-estabelecida a partir da realização de operações bit a bit entre o IDS, o ID real da etiqueta, quatro chaves de segurança pré-compartilhadas e dois números aleatórios gerados pelo leitor. Na fase de atualização, a etiqueta e o leitor atualizam de forma sincronizada o IDS e as quatro chaves de segurança utilizadas na fase anterior. Isso objetiva manter o anonimato das etiquetas e impedir que um atacante identifique as chaves de segurança que serão utilizadas na próxima rodada de autenticação.

Em [Barasz 2007] é mostrado um ataque contra o M^2AP. Esse ataque permite a uma entidade maliciosa ter acesso às chaves de segurança utilizadas e ao ID real da etiqueta. Para isso, basta que o atacante consiga capturar poucas rodadas de trocas de mensagens entre uma etiqueta-alvo e o leitor durante a execução do M^2AP. O acesso a essas informações permite a um atacante clonar essa etiqueta.

Em [Misra et al. 2009] é apresentado um esquema de autenticação mútua denominado SEAS (*Secure and Efficient Anonymity Scheme*). Nesse esquema, cada etiqueta possui, além do ID real, um segredo exclusivo que é pré-compartilhado com um servidor conectado ao leitor. Cada leitor também possui um segredo pré-compartilhado com o servidor. Para construir as mensagens trocadas durante o processo de autenticação mútua, o SEAS faz operações bit a bit utilizando três números aleatórios, uma função de deslocamento circular à esquerda e os segredos pré-compartilhados. Essas mensagens são repassadas ao servidor, o qual as utiliza para autenticar tanto o leitor quanto a etiqueta. A proposta do SEAS assume que a etiqueta já tenha passado pelo processo anticolisão para iniciar o processo de autenticação e que seu ID real nunca seja transmitido em claro a fim de manter o anonimato da mesma.

Em [Myneni et al. 2011] é proposto o SAMA (*Serverless Anonymous Mutual Authentication*). O SAMA se propõe a oferecer um esquema de autenticação mútua etiqueta-leitor sem a necessidade de uso de um servidor conectado ao leitor. A vantagem alegada é a não necessidade de haver conexões persistentes entre o leitor e o servidor para a realização do processo de autenticação. Os contras da eliminação do servidor incluem: 1) o aumento do custo e do consumo de energia do leitor, já que o mesmo precisa armazenar toda a base de informações da aplicação e; 2) a necessidade de um mecanismo que sincronize a base de dados de todos os leitores do sistema.

O processo de autenticação do SAMA utiliza dois componentes: o NLFSR (*Non-Linear Feedback Shift Register*)[Dubrova et al. 2008] e uma função de perturbação. O NLFSR é um registrador que realiza deslocamentos e que possui uma função de transição composta por uma quantidade pré-definida de portas lógicas do tipo XOR e AND. Essa função de transição faz com que cada NLFSR modifique a sua entrada de uma maneira única. Maiores detalhes sobre o NLFSR serão apresentados na Seção 4. A função de perturbação é um mecanismo auxiliar que visa manter o anonimato das mensagens transmitidas durante o processo de autenticação. No processo de autenticação da etiqueta perante o leitor, a etiqueta utiliza o NLFSR para gerar uma espécie de assinatura que é transmitida para o leitor. Essa assinatura muda toda vez que o processo de autenticação é executado uma vez que números pseudoaleatórios compõem a entrada para o NLFSR. A assinatura é utilizada pelo leitor para identificar e autenticar a etiqueta sem a necessidade de trasmissão do ID real. A autenticação do leitor perante a etiqueta é feita de forma análoga, passando o leitor a utilizar o NLFSR para gerar a assinatura a ser autenticada pela etiqueta. O SAMA não leva em consideração como é realizado processo anticolisão, mas requer o sigilo do ID real para manutenção do anonimato das etiquetas.

Este artigo se diferencia dos trabalhos relacionados por propor um esquema de autenticação mútua etiqueta-leitor para ser utilizado em conjunto com protocolos anticolisão baseados em árvore e ao mesmo tempo preservar o anonimato das etiquetas. O esquema proposto, denominado AMAS (*Anonymous Mutual Authentication Scheme*), reutiliza mecanismos empregados no processo de autenticação para que as etiquetas gerem IDs aleatórios e temporários a serem utilizados durante a execução do protocolo anticolisão baseado em árvore. Essa abordagem busca garantir o anonimato das etiquetas desde o processo de anticolisão, não permitindo a um atacante correlacionar os IDs aleatórios e temporários com os IDs reais. O AMAS também é projetado com foco em sistemas que utilizam etiquetas passivas.

3. Modelo do Sistema e Modelo de Ameaça

Esta seção detalha o modelo do sistema RFID adotado neste trabalho e o modelo de ameaça contra o esquema de autenticação e anonimato proposto.

3.1. Modelo do Sistema

O sistema RFID analisado nesse trabalho é composto por três componentes: um servidor S, leitores e etiquetas. As etiquetas são passivas e possuem memória necessária para armazenar as informações utilizadas pelo esquema proposto. Adicionalmente, as etiquetas possuem um gerador de números pseudoaleatórios. Toda comunicação entre o leitor e o servidor é feita através de uma conexão segura e inviolável. A comunicação entre leitor e etiquetas é realizada através de radiofrequência (RF) e pode ser facilmente capturada. O leitor utiliza um protocolo anticolisão baseado em árvore para controle de acesso das etiquetas ao meio de comunicação.

Cada etiqueta T_i possui espaço de memória reservado para um identificador aleatório e temporário (IDT_i) de n *bits*. Esse identificador substitui seu identificador real (IDR_i) de n *bits* durante a execução do protocolo anticolisão baseado em árvore. Para a geração do IDT_i e a realização do processo de autenticação, cada etiqueta possui uma chave atualizável (K_i) e um $NLFSR_i$ único, sendo ambos de n *bits*.

3.2. Modelo de Ameaça

Um modelo de ameaça tem como objetivo definir o que uma entidade maliciosa pode fazer para tentar quebrar a segurança de um sistema. Neste artigo, é assumido que uma entidade maliciosa pode ameaçar o sistema RFID das seguintes formas:

1. Capturando quaisquer mensagens trocadas entre o leitor e as etiquetas durante a execução do protocolo de anticolisão;
2. Capturando quaisquer mensagens entre as etiquetas e o leitor durante o processo de autenticação;
3. Realizando ataques de *replay*;
4. Utilizando dados capturados para tentar obter informações sigilosas;
5. Realizando ataques de dessincronização.

Com a *ameaça 1*, o atacante tem acesso aos identificadores que estejam sendo utilizados pelas etiquetas durante a execução do protocolo anticolisão baseado em árvore. Essa ameaça também permite a um atacante que ele tente se passar por um leitor e realize o processo de anticolisão de etiquetas. Isso significa que o atacante pode obter os identificadores aleatórios e temporários utilizados pelas etiquetas a qualquer momento. A *ameaça 2* permite a um atacante tentar obter alguma informação relevante que o auxilie em algum outro tipo de ataque. As informações capturadas nas *ameaças 1* e *2* podem ser utilizadas para que um atacante tente rastrear ou mesmo clonar uma ou mais etiquetas do sistema.

A *ameaça 3* permite a uma entidade maliciosa reutilizar as mensagens capturadas nas *ameaças 1* e *2* para tentar se passar por uma etiqueta autêntica do sistema. A *ameaça 4* permite a um atacante utilizar as mensagens capturadas nas *ameaças 1* e *2* para tentar obter o identificador real, a chave atualizável ou o $NLFSR_i$ das etiquetas (vide Seção 3.1). Na *ameaça 5*, a entidade maliciosa pode bloquear a recepção de mensagens trocadas entre qualquer etiqueta e o leitor. Isso pode impedir a etiqueta de atualizar sua chave (K_i).

Existem ameaças inerentes às redes sem fio que não são tratadas neste artigo, como por exemplo: comprometer um dispositivo sem fio fisicamente; ataques de homem no meio (*man-in-the-middle*) e; outros tipos de negação de serviço (DoS - *Denial of Service*) além da dessincronização. Proteger um sistema de forma efetiva contra essas ameaças ainda é um desafio atualmente.

4. AMAS

.O AMAS (*Anonymous Mutual Authentication Scheme*) é um esquema para ser utilizado com protocolos anticolisão baseados em árvore. Ele é realizado em duas etapas: a primeira é responsável pela geração do IDT_i e a segunda é reponsável pela autenticação mútua etiqueta-leitor. No AMAS, cada etiqueta possui um $NLFSR_i$ único pré-compartilhado com o servidor S. Esse NLFSR é utilizado no processo de geração do IDT_i para garantir que apenas um leitor autorizado consiga identificar a etiqueta. O NLFSR também é utilizado no processo de autenticação mútua etiqueta-leitor, garantindo que tal processo só possa ser feito por etiquetas e leitores autorizados. Todas as operações realizadas durante a execução do AMAS utilizam números de n bits, onde n é um múltiplo de 8.

O esquema de autenticação mútua etiqueta-leitor do AMAS é baseado no do SAMA e suas diferenças são discutidas nesta seção. A seguir, são apresentados: uma descrição detalhada do funcionamento do NLFSR, o processo de geração do IDT_i, o processo de autenticação mútua etiqueta-leitor e, por fim,as diferenças entre o AMAS, o SEAS e o SAMA.

4.1. NLFSR

O NLFSR [Dubrova et al. 2008] é um registrador de x *bits* que realiza y rodadas de deslocamento à direita, onde y é o tamanho da saída do NLFSR. Cada NLFSR possui uma função de transição única composta por uma quantidade pré-definida de portas lógicas do tipo XOR ou AND. Cada uma das portas lógicas pode receber como entrada a valor do *bit* que se encontra em uma das posições do registrador ou mesmo a saída de outra porta lógica. Antes de cada rodada de deslocamento, o NLFSR substitui o valor de cada *bit* do registrador pelo resultado de um XOR entre o valor atual do *bit* e o resultado da função de transição. A cada rodada de deslocamento, o *bit* mais a direita do registrador é concatenado aos *bits* de saída do NLFSR. É importante ressaltar que quaisquer modificações realizadas na função de transição como, por exemplo, a alteração no número de portas lógicas, faz com que a saída do NLFSR seja completamente modificada. A Figura 1 apresenta um exemplo de *NLFSR* de 32 *bits* que possui uma função de transição composta por sete portas lógicas, sendo quatro do tipo XOR e três do tipo AND.

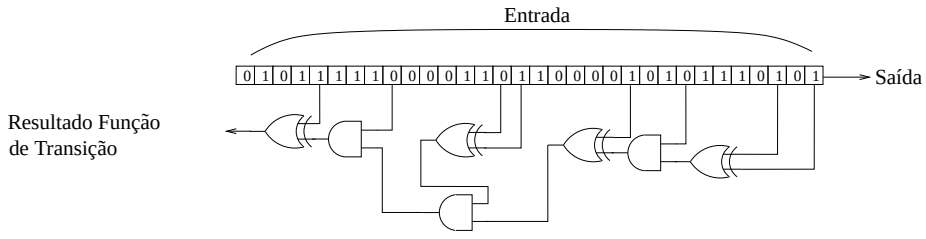


Figura 1. Exemplo de NLFSR de 32 bits com sete portas lógicas.

4.2. Geração do IDT_i

A Figura 2 apresenta o processo de geração do IDT_i realizado por cada uma das etiquetas a serem identificadas. O leitor R_j precisa que as etiquetas gerem os seus respectivos IDT_i antes do início da execução do protocolo anticolisão. Para isso, ele gera um número aleatório r_j^1 e o transmite em *broadcast* para as etiquetas. Cada uma das etiquetas recebe r_j^1 e o utiliza como entrada para o seu respectivo $NLFSR_i$, passando por uma sequência de n deslocamentos. A saída desse processo é um número r_j^1. Em seguida, cada etiqueta gera um número aleatório r_{ij}^2 e faz um XOR entre r_j^1, r_{ij}^2, o seu identificador real IDR_i e sua chave K_i, gerando o número p_{ij}^1. Esse número é fornecido como entrada para o $NLFSR_i$ que, após os n deslocamentos, gera o IDT_i.

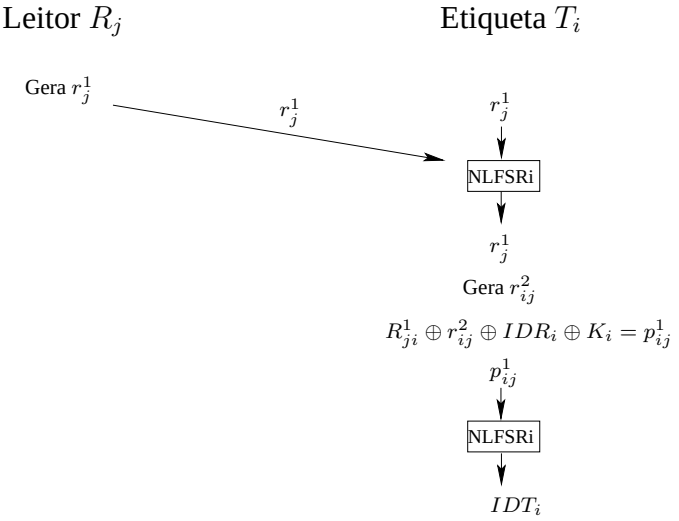


Figura 2. Geração do IDT_i por uma etiqueta T_i.

Após o fim da primeira etapa do AMAS, o leitor executa o protocolo anticolisão. É importante observar que existe a possibilidade de que duas etiquetas gerem o mesmo IDT_i. No entanto, dado que até 500 etiquetas com IDs de 32 *bits* estejam sendo identificadas ao mesmo tempo, a probabilidade de que pelo menos duas gerem dois IDT_i iguais é de aproximadamente $2,91 \times 10^{-5}$ conforme o paradoxo do aniversário. O procedimento a ser adotado pelo protocolo anticolisão para tratar esse problema está fora do escopo deste artigo.

4.3. Autenticação da Etiqueta perante o Leitor

A Figura 3 apresenta o processo de autenticação de uma etiqueta T_i perante o leitor R_j. O processo se inicia assim que a mesma é selecionada durante a execução do protocolo anticolisão baseado em árvore. O leitor R_j armazena o IDT_i da etiqueta e inicia a verificação da autenticidade da mesma. Para isso, o leitor requisita o r_{ij}^2 gerado pela etiqueta durante o processo de geração de seu IDT_i. Após o recebimento de r_{ij}^2, o leitor envia IDT_i, r_j^1 e r_{ji}^2 para o servidor S. O servidor busca no seu banco de dados uma terna (IDR_i, K_i, $NLFSR_i$) que consiga gerar o IDT_i quando combinada com r_j^1 e r_{ij}^2. Em outras palavras, o servidor verifica um a um os dados das etiquetas presentes no banco de dados até que a terna correta seja encontrada. Quando isso acontece, ele envia a terna para o leitor que, por sua vez, autentica a etiqueta. Caso nenhuma terna correspondente seja encontrada, a etiqueta é considerada falsa e a comunicação com essa etiqueta é encerrada.

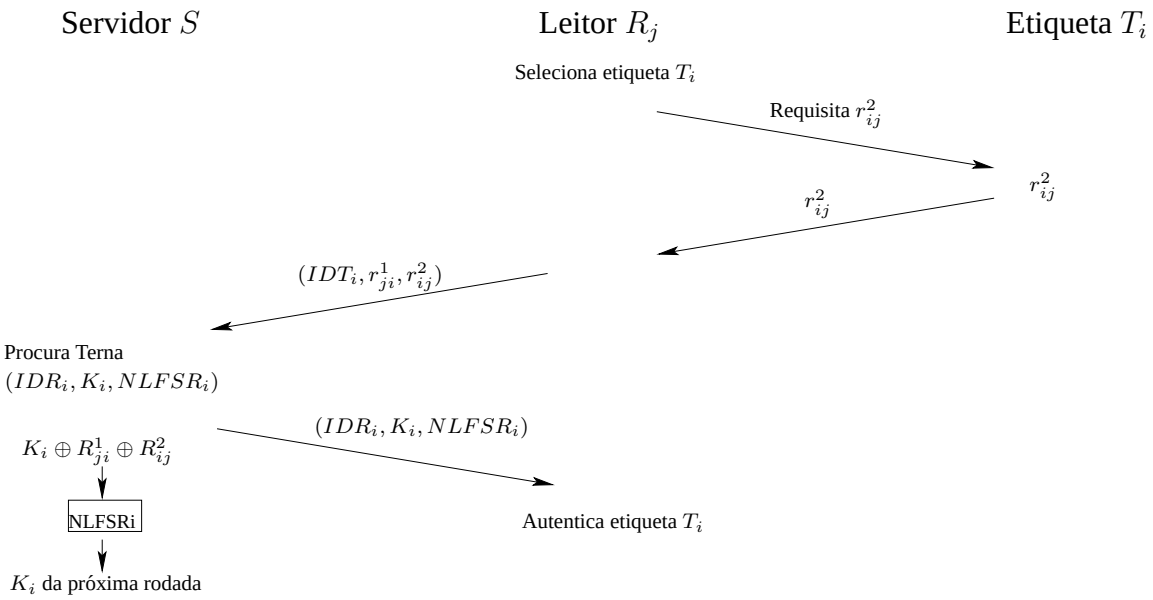


Figura 3. Autenticação bem sucedida de uma etiqueta T_i perante o leitor R_j.

Para garantir uma maior segurança do mecanismo de geração do IDT_i, a chave K_i é atualizada pelo servidor após a autenticação da etiqueta. Para isso, é feito um XOR entre K_i, r_j^1 e R_{ij}^2. Este último é o valor de r_{ij}^2 após ele ser modificado pelo $NLFSR_i$. O resultado do XOR é fornecido como entrada para o $NLFSR_i$, o qual gera como saída o valor de K_i que será utilizado na próxima geração do IDT_i. O servidor guarda o valor antigo de K_i para evitar que ataques de dessincronização possam afetar o sistema RFID.

4.4. Autenticação do Leitor perante à Etiqueta

A Figura 4 apresenta o processo de autenticação do leitor R_j por uma etiqueta T_i. Para isso, o leitor fornece r_{ij}^2 como entrada para o $NLFSR_i$, o qual gera a saída R_{ij}^2. Em seguida, o leitor gera um novo número aleatório r_{ji}^3 e faz um XOR desse número com R_{ji}^2, gerando x_{ji}^1. Este último número é fornecido como entrada ao $NLFSR_i$, o qual gera a saída m_{ji}^1. Os valores de m_{ji}^1 e r_{ji}^3 são concatenados, gerando a mensagem M_{ji}^1. Essa mensagem é enviada para a etiqueta que verifica a sua validade utilizando r_{ij}^2, r_{ji}^3 e o $NLFSR_i$. Caso M_{ji}^1 seja válida, o processo de autenticação do leitor é finalizado.

Após autenticar o leitor, a chave K_i é atualizada na etiqueta da mesma forma que foi feita no servidor S. Em seguida, T_i faz um XOR entre R_{ij}^2, r_j^1 e $r_j^3 i$ e fornece o resultado dessa operação como entrada para o $NLFSR_i$, gerando como saída o número m_{ij}^2. Este último número é enviado para o leitor e serve como confirmação para o servidor de que o processo foi concluído corretamente. Após o recebimento, o servidor utiliza R_{ij}^2, r_j^1 e $r_j^3 i$ para verificar o valor de m_{ij}^2, verificando se foi realmente a etiqueta que enviou o número.

4.5. Diferenças entre o AMAS, o SEAS e o SAMA

O AMAS e o SEAS são esquemas diferentes. O AMAS utiliza uma chave de segurança atualizável para cada etiqueta e baseia sua segurança no NLFSR. Já o SEAS utiliza uma chave de segurança fixa para cada etiqueta e baseia sua segurança apenas em operações

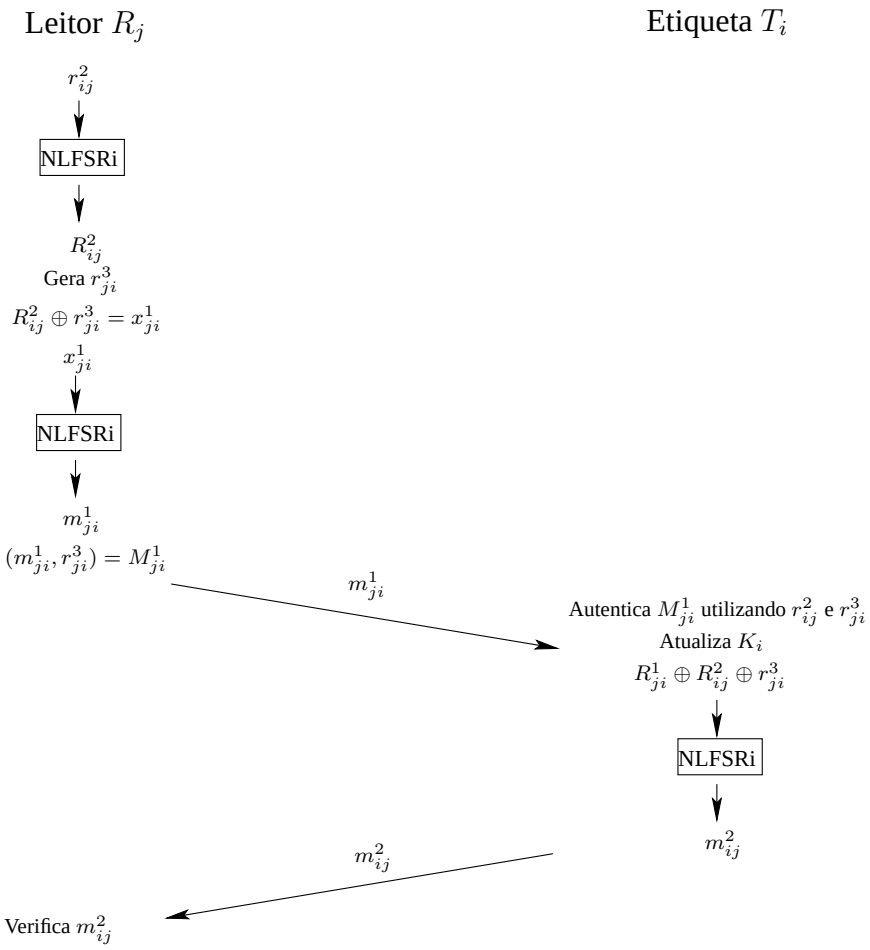


Figura 4. Autenticação bem sucedida de um leitor R_j perante uma etiqueta T_i.

XOR e em uma função linear de deslocamento à esquerda. O SEAS não leva em conta como o processo anticolisão é feito nem as vulnerabilidades desse processo.

O esquema de autenticação do AMAS é uma versão modificada daquele usado pelo SAMA [Myneni et al. 2011]. Primeiramente, o esquema utilizado pelo SAMA se difere pela não utilização de um servidor no processo de autenticação, fazendo com que cada leitor tenha que armazenar os dados de todas as etiquetas do sistema para realizar tal processo. No AMAS, o servidor é utilizado para armazenar e buscar todas as informações relativas às etiquetas. Isso reduz significativamente a quantidade de memória que cada leitor precisa ter disponível para realizar o processo de autenticação.

A segunda diferença entre o AMAS e o SAMA ocorre no processo de autenticação da etiqueta perante o leitor. No AMAS, esse processo inclui a utilização do IDT_i, do ID real da etiqueta (IDR_i) e da chave atualizável (K_i). Essa modificação permite que o IDT_i seja utilizado tanto pelo processo anticolisão baseado em árvore como na autenticação da etiqueta perante o leitor, recebendo a segurança adicional fornecida pela chave atualizável (K_i).

Outra diferença entre o AMAS e o SAMA é a não utilização da função de perturbação pelo AMAS, sendo substituída, quando necessário, pelo $NLFSR_i$. Isso é feito porque a função de perturbação pode ser invertida facilmente. Assim sendo,

a utilização dessa função não traria nenhum ganho para a segurança do esquema de autenticação do AMAS e resultaria em um consumo desnecessário de recursos para sua implementação e execução. Assim como o SEAS, o SAMA não leva em conta como o processo anticolisão é realizado nem as fraquezas desse processo.

5. Análise da Segurança e do Custo do AMAS

Essa seção apresenta, inicialmente, uma criptoanálise do NLFSR e uma análise da segurança do AMAS perante as ameaças definidas na Seção 3. Em seguida, é apresentada uma avaliação de seu custo em termos de quantidade de portas lógicas e ciclos de relógio, comparando-os com os quantitativos obtidos pelo SEAS, SAMA, SHA-1, MD4 e AES.

5.1. Análise da Segurança

5.1.1. Criptoanálise do NLFSR

O $NLFSR$ é o dispositivo de segurança básico utilizado pelo AMAS. Assim sendo, um atacante precisa primeiramente identificar o $NLFSR_i$ da etiqueta T_i que ele pretende atacar para conseguir burlar o AMAS. No entanto, de acordo com o corolário apresentado e provado em [Myneni et al. 2011], a probabilidade de um atacante conseguir identificar um $NLFSR$ de 32 bits com 9 portas lógicas é igual a $\frac{1}{6{,}36X10^{25}}$, sendo um valor desprezível. A partir desse valor podemos afirmar que um atacante não consegue identificar que uma entrada x foi utilizada pelo $NLFSR$ para gerar uma saída x'.

5.1.2. Rastreamento

Uma etiqueta T_i não é rastreável se um atacante não conseguir o seguinte:

- **caso 1 -** correlacionar seus identificadores aleatórios e temporários, gerados em rodadas distintas, como pertencentes à etiqueta T_i.
- **caso 2 -** relacionar seu identificador aleatório e temporário atual com o seu identificador real;

Em ambos os casos, um atacante poderia tentar rastrear uma etiqueta T_i utilizando dois métodos. No primeiro, o atacante apenas escuta mensagens trocadas entre a etiqueta T_i e o leitor. No segundo, o atacante tenta se passar por um leitor legítimo e iniciar o processo anticolisão posteriormente ao envio por ele de um r_j^1 para que a etiqueta gere seu IDT_i.

O primeiro método não funciona no **caso 1** pelo seguinte: um atacante precisa identificar o $NLFSR_i$ e a chave atualizável (K_i) da etiqueta T_i que foram utilizados no processo de geração de dois ou mais identificadores aleatórios e temporários. Como apresentado na Seção 5.1.1, a probabilidade dele identificar o $NLFSR_i$ é desprezível. Além disso, a chave K_i é mantida em sigilo durante todo o processo e é atualizada a cada rodada de autenticação.

O segundo método não funciona no **caso 1** pelo seguinte: as etiquetas utilizam um número aleatório r_{ij}^2 para gerar o IDT_i. Toda vez que o processo anticolisão for rodado, a etiqueta gerará um novo número aleatório r_{ij}^2 e, consequentemente, um novo IDT_i. Para conseguir identificar que dois ou mais identificadores IDT_i foram gerados por uma

mesma etiqueta T_i, um atacante precisaria obter o seu $NLFSR_i$ e sua chave atualizável (K_i), os quais ele não tem acesso.

Ambos os métodos não funcionam para o **caso 2**. Isso ocorre porque o atacante precisa obter o $NLFSR_i$ e a chave atualizável (K_i) da etiqueta T_i, os quais ele não tem acesso, para conseguir relacionar o IDT_i atualmente utilizado pela etiqueta com o seu identificador real.

5.1.3. Clonagem

Para que um atacante consiga clonar corretamente uma etiqueta, ele precisa conseguir burlar o mecanismo de autenticação do AMAS. Para isso, a etiqueta falsa T_i' teria que conseguir gerar corretamente o IDT_i a partir de um r_j^1 enviado pelo leitor. A única forma de gerar IDT_i corretamente é utilizando o $NLFSR_i$ e a chave atualizável K_i da etiqueta T_i. Assim sendo, T_i' não consegue burlar esse mecanismo, pois ela não consegue identificar qual é o $NLFSR_i$ utilizado por T_i e não tem acesso à chave K_i. É importante observar que se um atacante conseguir acessar fisicamente uma etiqueta, ele terá acesso a todos os segredos contidos dentro da mesma e poderá cloná-la com facilidade. No entanto, essa ameaça não foi considerada no modelo de ataque pelo motivo já citado na Seção 3.

5.1.4. Ataque de Replay

Mesmo que um atacante consiga capturar diversas mensagens de autenticação transmitidas entre um leitor e uma etiqueta, ele não conseguirá reutilizá-las posteriormente em um ataque de *replay*. Isso ocorre devido à utilização do IDT_i e dos números r_j^1, r_{ij}^2 e r_{ji}^3, os quais são gerados aleatoriamente a cada rodada da autenticação.

5.1.5. Acesso a Informações Sigilosas

As únicas informações sigilosas mantidas pelo AMAS são a chave K_i, o ID real IDR_i e o $NLFSR_i$. No entanto, o $NLFSR_i$ nunca é transmitido e a chave K_i e o IDR_i não são transmitidos sem terem sido combinados com outras variáveis e modificados pelo $NLFSR_i$. Portanto, um atacante não consegue obtê-los somente capturando mensagens trocadas entre a etiqueta e o leitor. Mesmo que ele conseguisse adivinhar a chave K_i de uma determinada rodada por força bruta, a mesma já não teria mais valor na rodada seguinte devido à atualização da chave no processo de autenticação.

5.1.6. Ataques de Dessincronização

O único parâmetro atualizado pelo AMAS a cada rodada é a chave K_i. Ele consegue garantir que um ataque de dessincronização simples não seja capaz de afetar a atualização sincronizada desse parâmetro pela etiqueta e pelo servidor. Isso ocorre porque o servidor utiliza o número m_{ij}^2 como garantia de que uma etiqueta T_i tenha conseguido atualizar K_i corretamente. Se o atacante conseguir evitar que o processo seja concluído com algum ataque de dessincronização mais elaborado, isso não afetará a geração do IDT_i e o

processo de autenticação mútua etiqueta-leitor. Isso ocorre porque o servidor armazena para cada etiqueta do sistema, tanto a K_i atualizada quanto o valor de K_i utilizado na rodada anterior do processo autenticação. Dessa forma, ele garante que o leitor consegue autenticar a etiqueta em uma certa rodada mesmo que a chave K_i não seja atualizada corretamente na rodada anterior. A aleatoriedade de IDT_i também não é afetada por conta da utilização de um novo r_{ij}^2 a cada vez que ele é gerado.

5.2. Avaliação de Custos

Esta seção avalia os custos do AMAS em termos de quantidade de portas lógicas e ciclos de relógio, comparando-os com os custos do SEAS, do SAMA, da SHA-1, da MD4 e do AES. A comparação com a SHA-1 e a MD4 foi realizada por elas serem as funções *hash* que possuem o menor custo computacional em termos de portas lógicas e ciclos de relógio. O AMAS foi implementado como uma máquina de estados combinacional na linguagem de *hardware System Verilog*. Essa implementação foi utilizada para calcular seus custos através do programa de síntese de *hardware ALTERA Quartus II*. A implementação utilizou n igual a 32, ou seja, todos os parâmetros e mecanismos utilizados foram de 32 bits. Isso foi feito para que fosse possível comparar o AMAS com os trabalhos relacionados, os quais também utilizam 32 bits.

O SAMA também foi implementado no ambiente utilizado para a implementação do AMAS. Os custos obtidos para o SAMA através dessa implementação foram semelhantes aos obtidos em [Myneni et al. 2011]. A versão da implementação do SAMA no *ALTERA Quartus II* utiliza 1.420 portas lógicas e 70 ciclos de relógio, ao passo que, os autores do SAMA informam que o esquema utiliza 1.393 portas lógicas e 70 ciclos de relógio. A pequena diferença se deve ao fato dos resultados serem dependentes da forma como o algoritmo é codificado. Para fins de comparação, este artigo adota os custos do SAMA obtidos no ambiente *ALTERA Quartus II*. Os custos da SHA-1, da MD4 e do AES foram obtidos em [Feldhofer and Rechberger 2006] enquanto os custos do SEAS foram obtido em [Misra et al. 2009]. Esses custos devem ser vistos como uma aproximação para comparação com o SAMA e o AMAS. A implementação no mesmo ambiente do AMAS não traria potencialmente mudanças significativas a ponto de alterar as conclusões deste trabalho.

A Figura 5(a) apresenta uma comparação do número necessário de portas lógicas para a implementação dos esquemas estudados. Note que a quantidade de portas lógicas do AMAS (1214 portas lógicas) é a menor de todas. A diferença de portas lógicas entre o AMAS e o SAMA ocorre por dois motivos. O primeiro é a não utilização pelo AMAS da função de perturbação do SAMA, a qual utiliza aproximadamente 300 portas lógicas em sua implementação. O segundo é fato de que os mecanismos de geração do IDT_i e de atualização da chave K_i do AMAS reutilizam o mesmo $NLFSR$ do esquema de autenticação para prover IDs para o processo anticolisão. Com a reutilização, esses mecanismos contribuem na quantidade adicional de aproximadamente 100 portas lógicas para serem implementados.

A Figura 5(b) apresenta o resultado de ciclos de relógio para os esquemas estudados. O custo do AMAS (150 ciclos) ficou abaixo do limite de 220 ciclos para etiquetas passivas embora tenha sido mais elevado do que o cilco de relógio do SAMA (70 ciclos) e do SEAS (44 ciclos). A necessidade de se utilizar mais ciclo de relógio no AMAS em

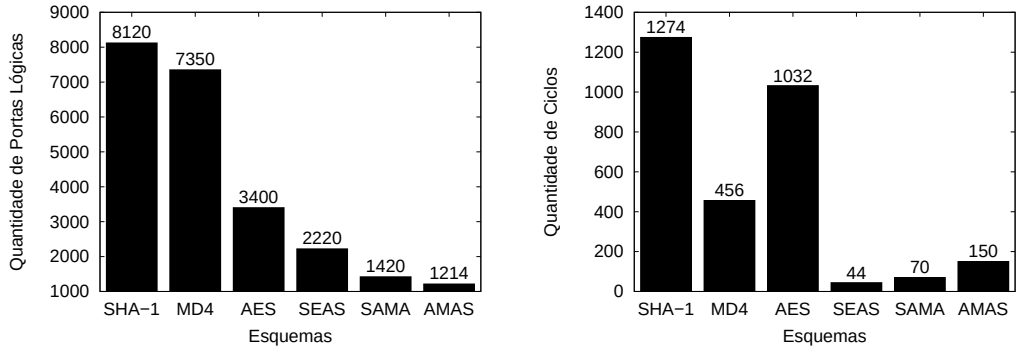


Figura 5. Comparação entre os diversos esquemas.

relação ao SAMA se deve ao mecanismo de geração do IDT_i e ao uso, por mais vezes, do $NLFSR$, gerando o consumo de 80 ciclos de relógio a mais do que o SAMA. No entanto, quando comparado com os mecanismos tradicionais de segurança como o AES (1032 ciclos), o AMAS possui um custo significativamente menor.

6. Considerações Finais

Prover autenticação mútua é um dos maiores desafios de sistemas RFID que utilizam etiquetas passivas. Isso ocorre devido às limitações de recursos computacionais e memória inerentes a esse tipo de etiqueta. O SAMA e o SEAS são as propostas mais atuais para prover autenticação mútua etiqueta-leitor em sistemas RFID baseados em etiquetas passivas. Esses dois esquemas são capazes de manter o anonimato das etiquetas. Mas para isso, um requisito mínimo necessário é nunca transmitir em claro o ID real dessas etiquetas durante qualquer troca de mensagem com o(s) leitor(es). Por outro lado, o ID real das etiquetas é comumente utilizado e transmitido em claro durante a execução de protocolos anticolisão baseados em árvore. Essa execução precede o processo de autenticação, fazendo com que o anonimato das etiquetas não possa ser garantido.

Este artigo propôs um esquema de autenticação mútua etiqueta-leitor para ser utilizado em conjunto com protocolos anticolisão baseados em árvore e ao mesmo tempo preservar o anonimato das etiquetas. Essa é uma das principais contribuições deste artigo. O esquema proposto, denominado AMAS, reutilizou mecanismos empregados no processo de autenticação para que as etiquetas gerassem IDs aleatórios e temporários a serem utilizados durante a execução do protocolo anticolisão baseado em árvore. Essa abordagem buscou garantir o anonimato das etiquetas desde o processo de anticolisão, não permitindo a um atacante correlacionar os IDs aleatórios e temporários com os IDs reais.

Este artigo também apresentou uma análise da segurança do AMAS, demonstrando que o esquema consegue defender o sistema de todos os ataques presentes no modelo de ameaça adotado. Adicionalmente, os custos do AMAS, em termos de quantidade de portas lógicas e ciclos de relógio, foram avaliados. Os resultados demonstraram que o AMAS atende aos requisitos necessários para que ele possa ser utilizado em sistemas RFID baseados em etiquetas passivas.

Referências

Barasz, M. (2007). Passive Attack Against the M2AP Mutual Authentication Protocol for RFID Tags. In *Proceedings of the First Int'l Workshop RFID Technology (EURASIP)*.

Dimitriou, T. (2005). A Lightweight RFID Protocol to Protect Against Traceability and Cloning Attacks. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*, pages 59–66.

Dimitriou, T. (2006). A Secure and Efficient RFID Protocol that could make Big Brother (partially) Obsolete. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pages 269–275.

Dominikus, S., Oswald, E., and Feldhofer, M. (2005). Symmetric Authentication for RFID Systems in Practice. In *Proceedings of the Ecrypt Workshop on RFID and Lightweight Cryptography*, pages 14–15.

Dubrova, E., Teslenko, M., and Tenhunen, H. (2008). On Analysis and Synthesis of (n, k)-Non-linear Feedback Shift Registers. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, pages 1286–1291.

Feldhofer, M., Dominikus, S., and Wolkerstorfer, J. (2004). Strong Authentication for RFID Systems Using the AES Algorithm. *Lecture Notes In Computer Science*, pages 357–370.

Feldhofer, M. and Rechberger, C. (2006). A Case Against Currently Used Hash Functions in RFID Protocols. *Lecture Notes In Computer Science*, 4278:372–381.

Klair, D., Chin, K.-W., and Raad, R. (2010). A Survey and Tutorial of RFID Anti-Collision Protocols. *IEEE Communications Surveys Tutorials*, 12(3):400–421.

Lee, S. et al. (2005). Efficient Authentication for Low-Cost RFID Systems. In *Proceedings of the International Conference on Computational Science and its Applications (ICCSA)*, pages 619–627.

Misra, S. et al. (2009). SEAS: A Secure and Efficient Anonymity Scheme for Low-Cost RFID tags. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 1–6.

Myneni, S., Misra, S., and Xue, G. (2011). SAMA: Serverless Anonymous Mutual Authentication for Low-Cost RFID Tags. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 1–5.

Peris-Lopez, P. et al. (2006). M2AP: A Minimalist Mutual-Authentication Protocol for low-cost RFID tags. In *Proceedings of the Third Int'l Conf. Ubiquitous Intelligence and Computing (UIC)*.

Weis, S. et al. (2004). Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. *Lecture Notes In Computer Science*, 2802:201–212.

Yang, J. et al. (2005). Mutual Authentication for Low-Cost RFID Systems. In *Proceedings of the Ecrypt Workshop on RFID and Lightweight Cryptography*, pages 17–24.

Um Estimador Acurado para o Protocolo DFSA em Sistemas RFID

Júlio D. de Andrade, Paulo André da S. Gonçalves

Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
50.740-560 – Recife – PE – Brasil

{jda, pasg}@cin.ufpe.br

Abstract. *DFSA (Dynamic Framed Slotted ALOHA) has been widely used as an anticollision protocol for RFID systems. Under this protocol, the size of each frame succeeding the first frame is dynamically adjusted based on an estimate of the number of competing tags in the previous frame. The estimator's accuracy plays an important role on the performance of DFSA. Among the main existing estimators for DFSA there are Vogt and Eom-Lee. The latter is more recent and more accurate for a wide range of tag populations. This paper proposes two modified versions of the Vogt's estimator: the IV-I and the IV-II. Performance evaluations show that IV-II has better accuracy than or equivalent to Eom-Lee. When better, IV-II can be up to 2.61 times more accurate than Eom-Lee, which reflects in up to 69 less time slots to finish the tag identification process.*

Resumo. *O DFSA (Dynamic Framed Slotted ALOHA) vem sendo amplamente utilizado como protocolo anticolisão em sistemas RFID. Sob tal protocolo, o tamanho de cada quadro subsequente ao quadro inicial é ajustado dinamicamente com base na população estimada de etiquetas que competiram por slots no quadro anterior. A acurácia do estimador usado tem um papel primordial no desempenho do DFSA. Dentre os principais estimadores para o DFSA encontram-se o Vogt e o Eom-Lee. Este último é mais recente e mais acurado para uma ampla faixa de quantidade de etiquetas. Este artigo propõe duas versões modificadas do estimador Vogt: o IV-I e o IV-II. Os resultados de avaliação de desempenho mostram que o estimador IV-II proposto possui acurácia equivalente ou superior à acurácia do estimador Eom-Lee. Quando melhor, o IV-II pode ser até 2,61 vezes mais acurado do que o Eom-Lee, refletindo em uma economia de até 69 slots no processo de identificação de etiquetas.*

1. Introdução

Um dos maiores desafios em sistemas RFID (*Radio Frequency IDentification*) baseados em etiquetas passivas é a resolução de colisões causadas por transmissões simultâneas de etiquetas. As etiquetas passivas são limitadas em recursos computacionais e de memória, não sendo capazes de detectar colisões de sinais nem capazes de escutar transmissões de outras etiquetas. Por causa disso, o acesso ao meio de comunicação deve ser arbitrado pelo leitor e isso é feito através do uso de um protocolo anticolisão [Klair et al. 2010].

O DFSA (*Dynamic Framed Slotted ALOHA*) é um protocolo anticolisão popular para sistemas RFID [Klair et al. 2010]. Nesse protocolo, o leitor organiza o tempo em um ou mais quadros (*frames*), onde cada quadro é subdividido em *slots* de tempo. As

etiquetas são requisitadas a transmitir em um *slot* a cada quadro até serem identificadas pelo leitor. O tamanho de cada quadro subsequente ao quadro inicial é ajustado dinamicamente com base na estimativa da população de etiquetas competindo por *slots* no quadro precedente [Andrade and Gonçalves 2011]. Assim sendo, o DFSA requer a utilização de um estimador.

A acurácia do estimador empregado tem um papel primordial no desempenho do DFSA. Em [Eom and Lee 2010] é apresentado um estimador para o DFSA. Tal estimador, doravante denominado Eom-Lee, é comparado com os principais estimadores na literatura: o *Lower Bound* [Vogt 2002], o Schoute [Schoute 1983], uma versão do Vogt [Vogt 2002], o Chen [Chen 2009] e o C-Ratio [Cha and Kim 2005]. As avaliações de desempenho mostram que o Eom-Lee possui a melhor acurácia, exceto quando o tamanho do quadro é próximo do tamanho da população de etiquetas. Nessa situação, a versão do Vogt alcança resultados similares e o Schoute leva ligeira vantagem apesar dele estar entre os piores estimadores. Além disso, os resultados mostram que o Eom-Lee permite concluir o processo de identificação de etiquetas com uma quantidade similar ou menor de *slots* do que as outras melhores propostas dependendo do tamanho da população de etiquetas no intervalo $[100, 1.000]$ e ao se considerar um quadro incial de 64 *slots*.

Esse artigo propõe dois estimadores de tamanho de quadro para o DFSA: o IV-I (*Improved Vogt* - I) e o IV-II (*Improved Vogt* - II). Ambos são baseados no Vogt e diferem entre si e dele na estratégia de estimação quando todos os *slots* em um quadro estão em colisão. As avaliações de desempenho dos estimadores propostos são realizadas com uma população de 100 a 1.000 etiquetas tanto para um quadro inicial de 64 *slots* quanto para um quadro inicial de 128 *slots*. Os resultados mostram que o IV-II é capaz de produzir uma acurácia equivalente ou superior à acurácia do estimador Eom-Lee. Os resultados também mostram que o IV-II permite concluir o processo de identificação de etiquetas utilizando uma quantidade equivalente ou menor de *slots* do que o Eom-Lee. Quando melhor, o IV-II pode ser até 2,61 vezes mais acurado do que o Eom-Lee, refletindo em uma economia de até 69 *slots* no processo de identificação de etiquetas.

O restante deste artigo está organizado da seguinte forma: a Seção 2 detalha os principais trabalhos relacionados no contexto deste artigo. A Seção 3 apresenta uma análise de uma função utilizada pelo estimador Vogt. As Seções 4 e 5 apresentam, respectivamente, os estimadores IV-I e IV-II propostos com seus resultados de desempenho. Por fim, a Seção 6 conclui este trabalho.

2. Trabalhos Relacionados

Esta seção detalha o funcionamento de 3 estimadores: o Vogt [Vogt 2002], o Eom-Lee [Eom and Lee 2010] e uma versão do Vogt utilizada em avaliações de desempenho em [Eom and Lee 2010]. No texto que segue, os quadros de interesse no DFSA são denominados *quadro finalizado* e *quadro posterior*. O primeiro se refere ao quadro que acabou de ser finalizado, permitindo a obtenção dos quantitativos s_v, s_s e s_c que representam, respectivamente, a quantidade *slots* vazios, a quantidade de *slots* bem sucedidos e a quantidade de *slots* em colisão.

O *quadro posterior* é aquele subsequente ao *quadro finalizado*. A avaliação da necessidade de um *quadro posterior* bem como o cálculo de seu tamanho são feitos com base nos quantitativos s_v, s_s e s_c observados no *quadro finalizado*. Se houver ao menos

1 *slot* em colisão, há necessidade de se invocar o estimador e gerar um novo quadro. Se não houver ao menos 1 *slot* em colisão, o *quadro finalizado* é o último no processo de identificação. O tamanho do *quadro posterior* é representado por $\hat{f}$. Para todos os estimadores descritos neste artigo, $\hat{n}$ representa a estimativa do número de etiquetas que competiram por *slots* no *quadro finalizado*.

2.1. O Estimador Vogt

O estimador Vogt [Vogt 2002] usa conceitos de probabilidade para estimar a quantidade de etiquetas que competiram em um *quadro finalizado* de tamanho L dado $1 \leq s_c < L$. Assumindo uma distribuição binomial, a quantidade esperada de *slots* contendo transmissões de r etiquetas em um quadro de tamanho L é dada por:

$$a_r^{L,n} = L \binom{n}{r} \left(\frac{1}{L}\right)^r \left(1 - \frac{1}{L}\right)^{n-r} \quad , \tag{1}$$

onde n representa a população de etiquetas que competem por *slots* no quadro.

Com base na Eq. (1) e nos quantitativos s_v, s_s e s_c, Vogt define a função ϵ conforme representada pela Eq. (2).

$$\epsilon(L, s_v, s_s, s_c, n) = \left\| \begin{pmatrix} a_0^{L,n} \\ a_1^{L,n} \\ a_{\geq 2}^{L,n} \end{pmatrix} - \begin{pmatrix} s_v \\ s_s \\ s_c \end{pmatrix} \right\| \quad , \tag{2}$$

onde $\|\bullet\|$ representa a norma Euclidiana. A estimativa da quantidade de etiquetas que competiram por *slots* em um quadro de tamanho L é dada por:

$$\hat{n} = \underset{n \geq 1}{\operatorname{argmin}} \quad \epsilon(L, s_v, s_s, s_c, n) \ , \tag{3}$$

significando que $\hat{n}$ é igual ao n que minimiza o valor da função ϵ. Note que Vogt propõe estimar o número de etiquetas buscando a minimização da distância entre o vetor $< s_v, s_s, s_c >$ e o vetor contendo os valores esperados para s_v, s_s e s_c. Essa proposta se baseia na desigualdade de Chebyshev, a qual afirma que o resultado de um experimento envolvendo uma variável aleatória X é, provavelmente, próximo do valor esperado de X [Vogt 2002].

O tamanho do *quadro posterior* é definido com base no número estimado $\hat{n}$ de etiquetas. Os tamanhos possíveis são apresentados na Tabela 1. Caso $\hat{n} \in [51, 56]$ tanto $\hat{f} = 64$ quanto $\hat{f} = 128$ são escolhas adequadas.

Tabela 1. Tamanhos de quadro em função de $\hat{n}$.

$\hat{f}$	$\hat{n} \in [x, y]$	$\hat{f}$	$\hat{n} \in [x, y]$
16	$[1, 9]$	128	$[51, 129]$
32	$[10, 27]$	256	$[112, \infty]$
64	$[17, 56]$		

A Eq. (3) não é utilizada quando o número de *slots* em colisão é igual ao tamanho do quadro. Para esse caso, Vogt considera a seguinte aproximação para o valor de $\hat{n}$:

$$\hat{n} \; = \; 2 * s_c \; , \tag{4}$$

ou seja, é considera a estimativa dada pelo estimador conhecido como *Lower Bound* [Vogt 2002]. Esse estimador adota o limite inferior de duas etiquetas por *slot* em colisão, sendo uma aproximação grosseira. Ao utilizar o *Lower Bound*, o estimador Vogt calcula o tamanho do próximo quadro como sendo $\hat{f} \; = \min\left(2 * s_c \; , \; 256\right)$.

2.2. O Estimador Vogt (Eom-Lee)

Em [Eom and Lee 2010] é apresentada uma avaliação de desempenho comparativa entre diversos estimadores, incluindo uma versão modificada do estimador Vogt original. Essa versão, doravante denominada *Vogt (Eom-Lee)*, elimina duas restrições: a do tamanho dos quadros ser obrigatoriamente uma potência de 2 e a desse tamanho estar confinado em um intervalo pré-definido de quantidade de *slots* com limite superior igual a 256. Quando $1 \leq s_c < L$, a estimativa $\hat{n}$ continua sendo dada pela Eq. (3). Contudo, o tamanho do *quadro posterior* passa a ser dado por $\hat{f} = \hat{n} - s_s$.

Quando todos os *slots* do quadro estão em colisão, o *Lower Bound* continua sendo utilizado como aproximação. Nesse caso, o tamanho do *quadro posterior* e a estimativa $\hat{n}$ são simplesmente iguais ao dobro do tamanho do *quadro finalizado*.

2.3. O Estimador Eom-Lee

O estimador Eom-Lee [Eom and Lee 2010] utiliza um algoritmo iterativo tanto para estimar a quantidade de etiquetas competindo por *slots* em um quadro quanto o tamanho $\hat{f}$ do quadro subsequente. O quadro a ser analisado para se estimar o tamanho do próximo quadro possui tamanho L. O valor de L é assumido ser igual ao número estimado de etiquetas que competiram no quadro multiplicado por um fator β a ser determinado. Logo, o valor de L pode ser representado por:

$$L = \beta \cdot \hat{n} \; . \tag{5}$$

O número de etiquetas competindo em um *slot* em colisão é assumido ser igual a γ. Considerando o tamanho do próximo quadro igual ao *backlog*[1], o valor de $\hat{f}$ pode ser calculado como:

$$\hat{f} = \hat{n} - s_s = \gamma \cdot s_c \; . \tag{6}$$

Note que o problema para obter a estimativa $\hat{n}$ está na determinação do valor de γ ou de β. A relação entre ambos é explicitada pela Eq. (7) e é determinada considerando-se o seguinte: a probabilidade de r etiquetas, dentre o universo total de etiquetas, transmitirem em um mesmo *slot* pode ser aproximada por uma distribuição binomial e; a equação de cômputo de tal probabilidade pode ser aproximada por uma distribuição de Poisson com média n/L para L suficientemente grande.

$$\gamma = \frac{1 - e^{-\frac{1}{\beta}}}{\beta(1 - (1 + \frac{1}{\beta})e^{-\frac{1}{\beta}})} \; . \tag{7}$$

[1]Quantidade de etiquetas que ainda precisam ser identificadas.

Encontrar uma solução fechada para se determinar os valores de γ e β a partir da Eq. (7) é um desafio. Para contornar o problema, o estimador Eom-Lee utiliza um algoritmo iterativo. Sejam γ_k e β_k, respectivamente, aproximações para o valor de γ e de β na k-ésima iteração do algoritmo. Tais aproximações são obtidas de acordo com as seguintes equações:

$$\beta_k = \frac{L}{\gamma_{k-1} \cdot s_c + s_s}\,, \tag{8}$$

$$\gamma_k = \frac{1 - e^{-\frac{1}{\beta_k}}}{\beta_k(1 - (1 + \frac{1}{\beta_k})e^{-\frac{1}{\beta_k}})}\,. \tag{9}$$

No primeiro passo do algoritmo iterativo considera-se $\beta_1 = \infty$ e $\gamma_1 = 2$ e em cada passo k seguinte determina-se uma nova aproximação para β e γ com o auxílio das Eqs. (8) e (9), respectivamente. Quando $|\gamma_{k^*-1} - \gamma_{k^*}|$ for menor que um limiar pré-definido $\epsilon_{threshold}$, o processo iterativo é interrompido. Os valores γ_{k^*-1} e γ_{k^*} representam, respectivamente, a aproximação anterior e atual para o valor de γ. A partir de então, o tamanho $\hat{f}$ do próximo quadro e a quantidade estimada $\hat{n}$ de etiquetas são obtidos, respectivamente, pelas Eqs. (10) e (11), onde β_{k^*} é a aproximação mais recente para o valor de β.

$$\hat{f} = \gamma_{k^*} \cdot s_c\,. \tag{10}$$

$$\hat{n} = \frac{\hat{f}}{\beta_{k^*}}\,. \tag{11}$$

3. Análise da Função ϵ de Vogt

Esta seção apresenta, graficamente, o comportamento da função ϵ de Vogt definida pela Eq. (2) para alguns casos de estudo. Esses casos se diferenciam em relação ao quantitativo de *slots* em colisão, vazios e bem sucedidos. Esses quantitativos e o tamanho L de quadro utilizados para cada um dos casos são apresentados na Tabela 2.

Tabela 2. Casos para análise da função ϵ de Vogt.

Caso #	L	s_v	s_s	s_c		Caso #	L	s_v	s_s	s_c
1	64	0	32	32		7	64	63	0	1
2	64	32	0	32		8	64	0	0	64
3	64	14	40	10		9	128	0	0	128
4	64	44	14	6		10	256	0	0	256
5	64	4	4	56		11	512	0	0	512
6	64	0	63	1						

O *caso 1* não possui *slots* vazios e há uma quantidade idêntica de *slots* em colisão e bem sucedidos. O *caso 2* possui uma quantidade igual de *slots* vazios e em colisão, não havendo *slots* bem sucedidos. Os *casos 3* a *5* consideram todos os quantitativos dos diferentes tipos de *slot* maiores que zero ao mesmo tempo em que consideram, respectivamente, uma ocorrência significativamente maior de *slots* bem sucedidos, vazios e em

colisão. No *caso 6*, a quantidade de *slots* bem sucedidos é próxima do tamanho do quadro e não há *slots* vazios. No *caso 7*, a quantidade de *slots* vazios é próxima do tamanho do quadro, não havendo *slots* bem sucedidos. Os *casos 8* a *11* possuem todos os *slots* em colisão e se diferenciam apenas pelo tamanho do quadro.

A Figura 1 mostra como o valor da função ϵ varia em função de n para os casos definidos na Tabela 2. O mínimo da função ϵ pode ser encontrado utilizando-se um método numérico apropriado. Em particular, o mínimo da função ϵ para os *casos 1* a *7* aparece nas respectivas figuras e ocorre para n dentro do intervalo $[1,\ 1000]$.

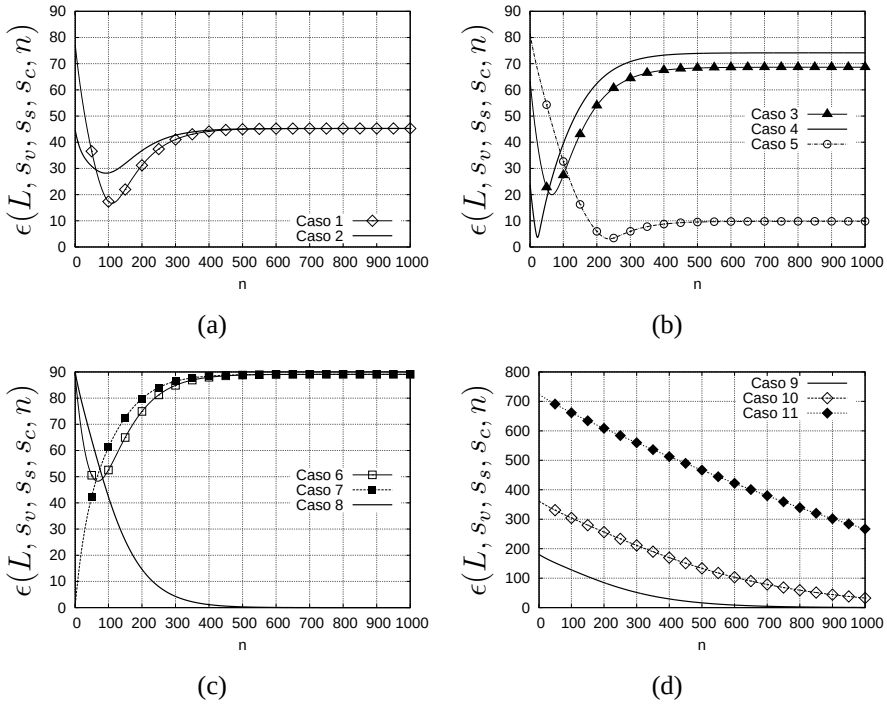


Figura 1. Comportamento da função ϵ de Vogt para os casos estudados.

Em particular aos *casos 8* a *11*, a função ϵ descresce exponencialmente à medida que n aumenta. O mínimo da função ϵ não está representado nas respectivas figuras para os casos considerados. Quando todos os *slots* estão em colisão, o mínimo da função ϵ assume o valor zero e tal mínimo ocorre quando n tende a infinito. Por limitações de espaço, uma demonstração matemática pode ser consultada em [Andrade 2012]. Pela razão exposta, Vogt utiliza a aproximação do *Lower Bound* para estimar a população de etiquetas que competiram em um quadro com todos os *slots* em colisão.

4. O Estimador IV-I Proposto

A seção anterior mostrou que a aplicação da Eq. (2) não permite encontrar, através de seu mínimo, um valor finito para a estimativa da população de etiquetas quando todos os *slots* do quadro estão em colisão. Isso ocorre porque o mínimo da função ϵ só é alcançado quando n tende a infinito. Enquanto Vogt contorna o problema utilizando a aproximação do *Lower Bound*, os estimadores propostos neste artigo o contornam utilizando outras estratégias.

Algoritmo 4.1 Estimador IV-I proposto. Entradas δ, L, s_s, s_v e s_c.

1: **Se** $(L \neq s_c)$ **então**
2: $\hat{n} \leftarrow$ Resultado da Eq. (3);
3: **Senão**
4: $n \leftarrow 2 \times L$;
5: $a_0 \leftarrow L \times \left(1 - \frac{1}{L}\right)^n$;
6: $a_1 \leftarrow n \times \left(1 - \frac{1}{L}\right)^{n-1}$;
7: $\epsilon \leftarrow \sqrt{2} \times \sqrt{a_0^2 + a_1^2 + (a_0 \times a_1)}$;
8: $\epsilon \leftarrow$ **Truncar**$(\delta \times \epsilon)$;
9: $\epsilon_{anterior} \leftarrow \epsilon + 1$;
10: **Enquanto** $(\epsilon < \epsilon_{anterior})$ **faça**
11: $n \leftarrow n + 1$;
12: $\epsilon_{anterior} \leftarrow \epsilon$;
13: $a_0 \leftarrow L \times \left(1 - \frac{1}{L}\right)^n$;
14: $a_1 \leftarrow n \times \left(1 - \frac{1}{L}\right)^{n-1}$;
15: $\epsilon \leftarrow \sqrt{2} \times \sqrt{a_0^2 + a_1^2 + (a_0 \times a_1)}$;
16: $\epsilon \leftarrow$ **Truncar**$(\delta \times \epsilon)$;
17: **fim Enquanto**
18: $\hat{n} \leftarrow n - 1$;
19: **fim Senão**
20: **Retorne**$(\hat{n})$;

O primeiro estimador proposto neste artigo é denominado IV-I (*Improved Vogt - I*) e seu pseudocódigo para estimar a população de etiquetas competindo em um quadro de interesse é apresentado pelo Algoritmo 4.1. Os parâmetros de entrada são o tamanho do quadro (L) e um fator multiplicativo (δ). O algoritmo é chamado sempre que houver ao menos um *slot* em colisão no *quadro finalizado*, ou seja, a codição $1 \leq s_c < L$ é garantida antes de se chamar o algoritmo. A estratégia do IV-I é percorrer a exponencial decrescente gerada a partir da Eq. (2) quando $L = s_c$ enquanto uma determinada condição seja satisfeita. O valor estimado $\hat{n}$ retornado é igual ao valor de n que torna verdadeira a condição de parada decrementado de uma unidade.

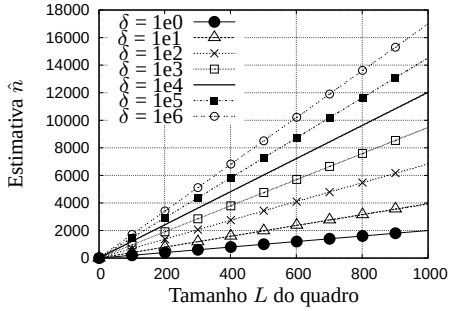
O IV-I computa $\hat{n}$ exatamente como no Vogt caso todos os *slots* não estejam em colisão. Caso contrário, o algoritmo inicia atribuindo o valor da aproximação fornecida pelo *Lower Bound* a n (*linha 4*). Em seguida, o valor da Eq. (2) é calculado (*linhas 5-7*). O valor de ϵ é, então, atualizado para a parte inteira do resultado do produto do fator multiplicativo (δ) por ϵ e um valor maior que ϵ é atribuido a $\epsilon_{anterior}$ (*linhas 8-9*). Ao entrar no *loop* (*linhas 10-17*), o valor de n é incrementado de uma unidade enquanto a condição de parada não for satisfeita. O valor de n decrementado de uma unidade é atibuído a $\hat{n}$, o qual é retornado como resultado final (*linhas 18-20*).

O tamanho $\hat{f}$ do próximo quadro no estimador IV-I é dado por:

$$\hat{f} = \max\left(2, \hat{n} - s_s\right), \tag{12}$$

ou seja, não pode haver nenhum quadro com menos de 2 *slots*.

A Figura 2 mostra como o número estimado de etiquetas $\hat{n}$ varia em função do tamanho L do quadro para alguns valores de δ ao se utilizar o estimador IV-I quando $L = s_c$. Os valores de L considerados estão no intervalo $[1, 1000]$. O impacto do aumento de δ é aumentar o valor da estimativa $\hat{n}$ para um mesmo valor de L tal que $L > 1$.



(a)

Figura 2. Estimativa $\hat{n}$ do IV-I em função de L e δ quando $L = s_c$.

4.1. Acurácia e Impacto

Esta seção avalia a acurácia do estimador de IV-I, comparando-a com a acurácia dos outros estimadores apresentados na Seção 2. A acurácia é estudada através da métrica *erro absoluto médio de estimação*. Essa métrica é definida como sendo o somatório do módulo da diferença entre o tamanho de cada quadro estimado e o *backlog* real, sendo tal resultado dividido pela quantidade total de quadros estimados ao longo do processo de identificação de etiquetas. Esta seção também estuda o impacto da acurácia dos melhores estimadores, mostrando a diferença de *slots* gerados no processo de identificação.

Os resultados apresentados são médias de 2.000 simulações para cada quantidade simulada de etiquetas que varia de 100 a 1.000 em passos de 100. Foram feitas avaliações para um quadro inicial de 64 e 128 *slots*. Um intervalo de 99% de confiança é adotado, sendo o mesmo representado em tabelas ou por barras de erros nos gráficos. Em particular ao Eom-Lee, adota-se o parâmetro $\epsilon_{threshold}$ igual a 0,001 conforme definido pelos autores.

A Tabela 3 apresenta, detalhadamente, a acurária dos estimadores estudados para um quadro inicial de 64 e 128 *slots*. Tanto o intervalo de confiança quanto todos os valores numéricos obtidos são apresentados. Nessa tabela, as melhores acurácias na comparação entre o Vogt (Eom-Lee) e o Eom-Lee estão destacadas com um asterístico **(*)**. O processo de marcação da melhor acurácia entre os dois estimadores, por linha da tabela, é o seguinte: primeiro, marca-se a melhor acurácia média por linha e, em seguida, marca-se a acurácia do outro estimador se ela se sobrepor com a melhor acurácia considerando-se os intervalos de confiança.

Com base na Tabela 3, o texto seguinte compara exclusivamente o Vogt (Eom-Lee) com o Eom-Lee. Quando o tamanho do quadro incial é igual a 64 *slots*, o seguinte pode ser observado: o Vogt (Eom-Lee) é o mais acurado na faixa de 100 a 400 etiquetas, embora o Eom-Lee tenha acurácia equivalente para 200 etiquetas e muito próxima para 100 e 300 etiquetas e; na faixa de 500 a 1.000 etiquetas, o Eom-Lee é mais acurado. Quando o tamanho do quadro incial é igual a 128 *slots*, o seguinte pode ser notado: o Vogt (Eom-Lee) é o mais acurado na faixa de 100 a 900 etiquetas; o Eom-Lee é o mais acurado para 1.000 etiquetas e; as maiores diferenças de acurácia entre os dois estimadores só ocorrem para 800 e 1.000 etiquetas.

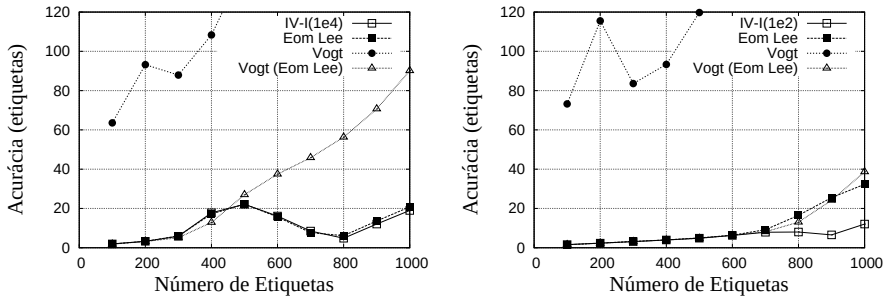
Tabela 3. Acurácia em nº de etiquetas dos Estimadores Estudados.

Quadro Inicial de 64 *slots*				
Etiquetas	Vogt	Vogt (Eom-Lee)	Eom-Lee	IV-I ($\delta = 1e4$)
100	63,59 ± 0,78	(*) **1,82 ± 0,04**	1,92 ± 0,04	1,95 ± 0,05
200	93,22 ± 0,89	(*) **3,17 ± 0,08**	(*) **3,25 ± 0,08**	**3,28 ± 0,09**
300	87,92 ± 0,86	(*) **5,07 ± 0,18**	6,02 ± 0,43	5,84 ± 0,40
400	108,36 ± 0,78	(*) **12,91 ± 0,56**	17,21 ± 0,89	17,78 ± 0,89
500	143,08 ± 0,87	26,98 ± 0,53	(*) **22,21 ± 0,33**	**22,11 ± 0,37**
600	185,33± 0,75	37,46 ± 0,42	(*) **15,68 ± 0,12**	16,15 ± 0,12
700	233,12 ± 0,68	45,84 ± 0,48	(*) **7,69 ± 0,16**	8,49 ± 0,13
800	286,68 ± 0,71	56,30 ± 0,75	(*) 6,16 ± 0,08	**4,85 ± 0,08**
900	351,87 ± 0,72	70,68 ± 1,13	(*) 13,60 ± 0,09	**12,07 ± 0,09**
1000	425,24 ± 0,70	90,15 ± 1,19	(*) 20,80 ± 0,13	**18,97 ± 0,13**
Quadro Inicial de 128 *slots*				
Etiquetas	Vogt	Vogt (Eom-Lee)	Eom-Lee	IV-I ($\delta = 1e2$)
100	73,23 ± 0,86	(*) **1,54 ± 0,03**	(*) **1,53 ± 0,03**	1,62 ± 0,04
200	115,52 ± 0,81	(*) **2,26 ± 0,04**	2,38 ± 0,04	**2,35 ± 0,05**
300	83,61 ± 0,53	(*) **3,11 ± 0,07**	(*) **3,19 ± 0,06**	**3,12 ± 0,06**
400	93,34 ± 0,71	(*) **3,92 ± 0,09**	(*) **4,00 ± 0,09**	**3,97 ± 0,09**
500	119,75 ± 0,51	(*) **4,76 ± 0,12**	5,04 ± 0,13	**4,87 ± 0,13**
600	155,97 ± 0,66	(*) **6,35 ± 0,21**	(*) **6,37 ± 0,20**	**6,27 ± 0,19**
700	203,71 ± 0,64	(*) **8,14 ± 0,35**	9,19 ± 0,59	**7,90 ± 0,25**
800	260,02 ± 0,72	(*) 13,04 ± 0,73	16,52 ± 1,14	**8,00 ± 0,17**
900	328,42 ± 0,72	(*) 24,15 ± 1,16	(*) 25,54 ± 1,28	**6,55 ± 0,21**
1000	404,84 ± 0,73	38,66 ± 1,19	(*) 32,32 ± 0,84	**12,07 ± 0,15**

Adicionalmente, a Tabela 3 destaca em negrito as melhores acurácias obtidas para cada quantidade de etiquetas e tamanho de quadro incial estudados. O processo de marcação das melhores acurácias, por linha da tabela, é o seguinte: primeiramente, enfatiza-se a melhor acurácia média por linha e, em seguida, ressaltam-se as acurácias equivalentes, isto é, aquelas que na mesma linha possuem alguma sobreposição com a melhor acurácia considerando-se os intervalos de confiança.

O texto seguinte faz uma comparação entre todos os estimadores estudados considerando os resultados na Tabela 3. Para um quadro inicial de 64 *slots*, nota-se o seguinte: o Vogt (Eom-Lee) é o mais acurado de todos na faixa de 100 a 400 etiquetas, ainda que as acurácias dos três melhores estimadores sejam muito próximas para 100 e 300 etiquetas e equivalentes para 200 etiquetas; o Eom-Lee é o mais acurado na faixa de 500 a 700 etiquetas, embora o IV-I possua acurácia equivalente para 500 etiquetas e próxima para 600 e 700 etiquetas; o IV-I é o mais acurado na faixa de 800 a 1.000 etiquetas, embora o Eom-Lee possua acurácias próximas. Quando o quadro inicial é de 128 *slots*, o IV-I é o mais acurado na faixa de 200 a 1.000 etiquetas e; para 100 etiquetas a acurácia do IV-I é muito próxima daquela obtida pelo Vogt (Eom-Lee) e pelo Eom-Lee, os quais são os melhores para 100 etiquetas.

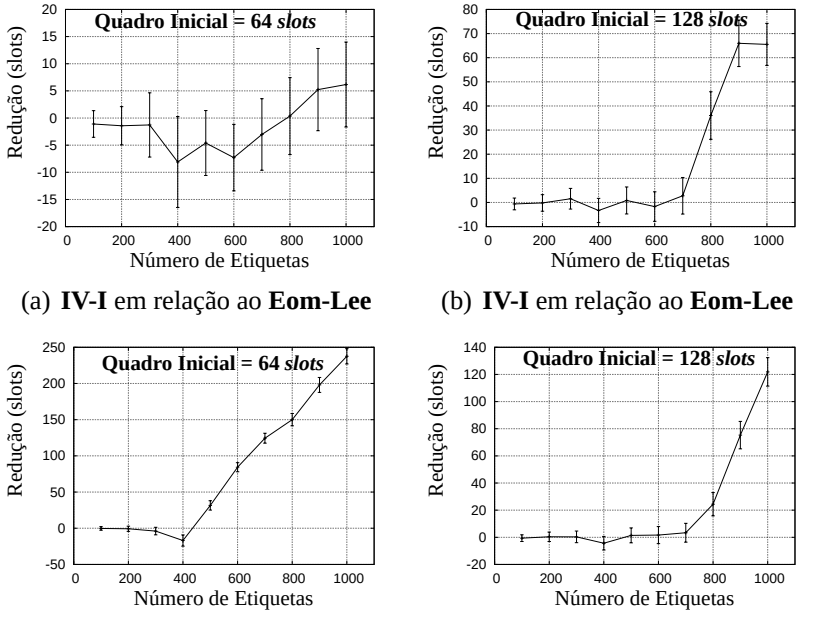
As Figuras 3(a) e 3(b) apresentam, graficamente, a acurária dos estimadores estudados para um tamanho de quadro inicial igual a 64 e 128 *slots*, respectivamente. Por questões de legibilidade, os intervalos de confiança não são apresentados, mas podem



(a) Quadro inicial = 64 *slots* (b) Quadro inicial = 128 *slots*

Figura 3. Acurácia dos Estimadores.

ser obtidos na Tabela 3. A Figura 3(a) reforça o seguinte para um quadro inicial de 64 *slots*: os estimadores IV-I, Vogt (Eom-Lee) e Eom-Lee possuem acurácias próximas no intervalo de 100 a 500 etiquetas e; acima de 500 etiquetas, as acurácias do IV-I e do Eom-Lee são as melhores e estão próximas entre si. A Figura 3(b) reforça o seguinte para um quadro inicial de 128 *slots*: o IV-I, o Vogt (Eom-Lee) e o Eom-Lee possuem acurácias próximas para até 700 etiquetas e; acima desse valor, o IV-I é o mais acurado entre todos os estimadores, alcançando uma acurácia 2,67 vezes maior do que a acurácia do Eom-Lee para 1.000 etiquetas.



(a) **IV-I** em relação ao **Eom-Lee** (b) **IV-I** em relação ao **Eom-Lee**

(c) **IV-I** em relação ao **Vogt (Eom-Lee)** (d) **IV-I** em relação ao **Vogt (Eom-Lee)**

Figura 4. Redução de *slots*: IV-I em relação ao Eom-Lee e ao Vogt (Eom-Lee).

As Figuras 4(a) e 4(b) apresentam a redução, em número de *slots*, obtida pelo IV-I em relação ao Eom-Lee para um quadro inicial com 64 e 128 *slots*, respectivamente. Dada a incerteza, a Figura 4(a) sugere que, exceto para 600 etiquetas, o IV-I e o Eom-Lee são equivalentes em termos de quantidade de *slots* gerados quando o quadro inicial é de 64 *slots*. Já a Figura 4(b) sugere que para um quadro inicial de 128 *slots*, o Eom-Lee e o IV-I são equivalentes em termos de quantidade de *slots* gerados até uma população de

700 etiquetas. Acima desse valor, o IV-I leva vantagem, chegando a usar 66 *slots* a menos do que o Eom-Lee para concluir o processo de identificação de 900 etiquetas.

As Figuras 4(c) e 4(d) apresentam a redução, em número de *slots*, do IV-I em relação ao Vogt (Eom-Lee) para um quadro inicial igual a 64 e 128 *slots*, respectivamente. Dada a incerteza, a Figura 4(c) mostra que o IV-I só é inferior ao Vogt (Eom-Lee) quando a população de etiquetas gira em torno 400 e um quadro inicial de 64 *slots* é utilizado. A redução máxima observada do IV-I em relação ao Vogt (Eom-Lee) é de 238 *slots* para 1.000 etiquetas. A Figura 4(d) sugere que o IV-I e o Vogt (Eom-Lee) são equivalentes, em termos de quantidade de *slots* gerados, até uma população de 700 etiquetas. Acima desse valor, o IV-I é melhor, gerando até 121 *slots* a menos do que o Vogt (Eom-Lee) para concluir o processo de identificação de 1.000 etiquetas.

5. O Estimador IV-II Proposto

O estimador IV-II (*Improved Vogt* - II) propõe uma outra estratégia para estimar a população de etiquetas quando todos os *slots* do quadro estão em colisão. Nesse caso, a estimativa $\hat{n}$ é definida por uma função linear do tamanho do quadro. Essa função é obtida a partir de aproximações para as curvas apresentadas na Figura 2 de acordo com o valor do fator multiplicativo δ.

Algoritmo 5.1 Estimador IV-II proposto. Entradas δ, L, s_s, s_v e s_c.

1: **Se** $(L \neq s_c)$ **então**
2: $\hat{n} \leftarrow$ Resultado da Eq. (3);
3: **Senão**
4: **Comute para** (δ)
5: $1e0 : n \leftarrow 2,001001000 * (L-1) + 2$; **interrompa**;
6: $1e1 : n \leftarrow 3,947947950 * (L-1) + 2$; **interrompa**;
7: $1e2 : n \leftarrow 6,851851850 * (L-1) + 2$; **interrompa**;
8: $1e3 : n \leftarrow 9,497497500 * (L-1) + 2$; **interrompa**;
9: $1e4 : n \leftarrow 12,047047047 * (L-1) + 2$; **interrompa**;
10: $1e5 : n \leftarrow 14,518518500 * (L-1) + 2$; **interrompa**;
11: $1e6 : n \leftarrow 17,011011000 * (L-1) + 2$; **interrompa**;
12: **fim Comute**
13: $\hat{n} \leftarrow \lceil n \rceil$;
14: **fim Senão**
15: **Retorne** $(\hat{n})$;

O pseudocódigo para estimar a população de etiquetas competindo em um quadro de interesse utilizando-se o IV-II é apresentado pelo Algoritmo 5.1. Os parâmetros de entrada são o tamanho do quadro (L) e a referência (δ) para a reta a ser utilizada nos cálculos. O algoritmo é chamado sempre que houver ao menos um *slot* em colisão no textitquadro finalizado, ou seja, a codição $1 \leq s_c < L$ é garantida antes de se chamar o algoritmo. O algoritmo computa $\hat{n}$ exatamente como no Vogt para L diferente de s_c. Se o quadro possuir todos os *slots* em colisão, o algoritmo obtém n a partir da equação da reta definida pela referência δ (*linhas 4* a *12*). O algoritmo retorna (*linha 15*) o valor de $\hat{n}$ computado anteriormente (*linha 13*) como sendo igual ao valor de n arredondado para o menor inteiro maior ou igual a n. O tamanho $\hat{f}$ do próximo quadro no estimador IV-II também é obtido através da Eq. (12).

5.1. Acurácia e Impacto

Esta seção avalia a acurácia do IV-II e a compara com a acurácia dos demais estimadores estudados. Também é estudado o impacto da acurácia dos melhores estimadores, mos-

trando a diferença na quantidade de *slots* gerados no processo de identificação. A métrica de acurácia usada e os cenários avaliados são os mesmos descritos na Seção 4.1.

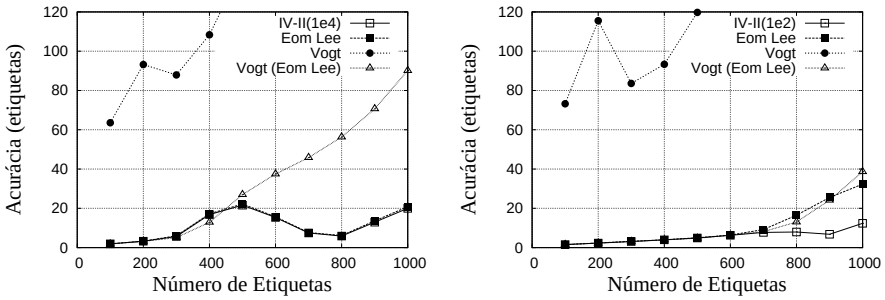
A Tabela 4 destaca em negrito as melhores acurácias obtidas para cada quantidade de etiquetas e tamanho de quadro incial. O processo de marcação das melhores acurácias, por linha da tabela, é o mesmo descrito na Seção 4.1. Para um quadro inicial de 64 *slots*, observa-se o seguinte: o Vogt (Eom-Lee) é o mais acurado na faixa de 100 a 400 etiquetas, ainda que as acurácias dos três melhores estimadores sejam equivalentes para 200 etiquetas e muito próximas para 100 e 300 etiquetas; o IV-II é o mais acurado na faixa de 500 a 1.000 etiquetas, embora o Eom-Lee possua acurácia equivalente para 500 e 700 etiquetas. Quando o quadro inicial é de 128 *slots*, o IV-II é o mais acurado na faixa de 100 a 1.000 etiquetas, embora as acurácias na faixa de 100 a 700 etiquetas sejam próximas às acurácias do Eom-Lee e do Vogt (Eom-Lee).

Tabela 4. Acurácia em nº de etiquetas dos Estimadores Estudados.

Quadro Inicial de 64 *slots*				
Etiquetas	Vogt	Vogt (Eom-Lee)	Eom-Lee	IV-II ($\delta = 1e4$)
100	63,59 ± 0,78	**1,82 ± 0,04**	1,92 ± 0,04	1,91 ± 0,04
200	93,22 ± 0,89	**3,17 ± 0,08**	**3,25 ± 0,08**	**3,25 ± 0,08**
300	87,92 ± 0,86	**5,07 ± 0,18**	6,02 ± 0,43	5,68 ± 0,36
400	108,36 ± 0,78	**12,91 ± 0,56**	17,21 ± 0,89	16,82 ± 0,85
500	143,08 ± 0,87	26,98 ± 0,53	**22,21 ± 0,33**	**21,59 ± 0,34**
600	185,33± 0,75	37,46 ± 0,42	15,68 ± 0,12	**15,37 ± 0,13**
700	233,12 ± 0,68	45,84 ± 0,48	**7,69 ± 0,16**	**7,48 ± 0,12**
800	286,68 ± 0,71	56,30 ± 0,75	6,16 ± 0,08	**5,87 ± 0,09**
900	351,87 ± 0,72	70,68 ± 1,13	13,60 ± 0,09	**12,96 ± 0,10**
1000	425,24 ± 0,70	90,15 ± 1,19	20,80 ± 0,13	**19,98 ± 0,15**
Quadro Inicial de 128 *slots*				
Etiquetas	Vogt	Vogt (Eom-Lee)	Eom-Lee	IV-II ($\delta = 1e2$)
100	73,23 ± 0,86	**1,54 ± 0,03**	**1,53 ± 0,03**	**1,57 ± 0,03**
200	115,52 ± 0,81	**2,26 ± 0,04**	2,38 ± 0,04	**2,31 ± 0,05**
300	83,61 ± 0,53	**3,11 ± 0,07**	**3,19 ± 0,06**	**3,11 ± 0,06**
400	93,34 ± 0,71	**3,92 ± 0,09**	**4,00 ± 0,09**	**3,96 ± 0,09**
500	119,75 ± 0,51	**4,76 ± 0,12**	5,04 ± 0,13	**4,91 ± 0,13**
600	155,97 ± 0,66	**6,35 ± 0,21**	**6,37 ± 0,20**	**6,28 ± 0,20**
700	203,71 ± 0,64	**8,14 ± 0,35**	9,19 ± 0,59	**7,75 ± 0,25**
800	260,02 ± 0,72	13,04 ± 0,73	16,52 ± 1,14	**7,96 ± 0,17**
900	328,42 ± 0,72	24,15 ± 1,16	25,54 ± 1,28	**6,80 ± 0,21**
1000	404,84 ± 0,73	38,66 ± 1,19	32,32 ± 0,84	**12,36 ± 0,16**

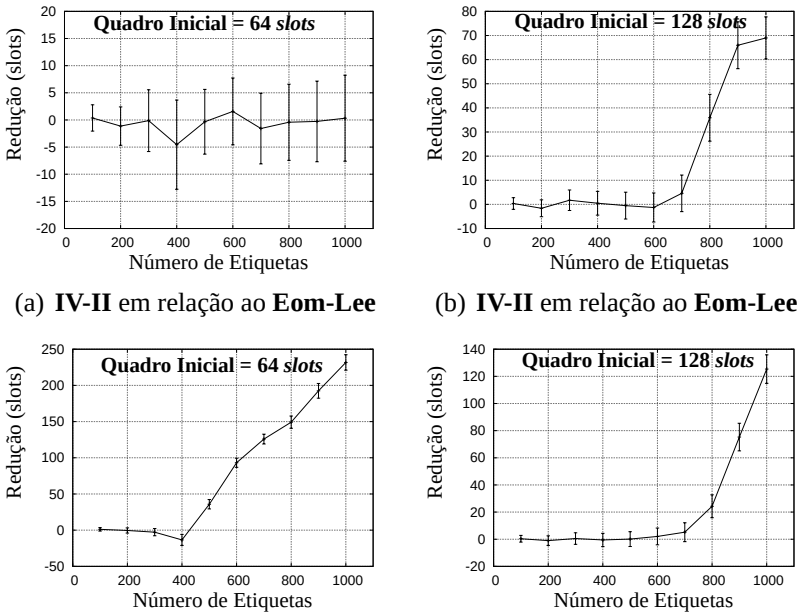
As Figuras 5(a) e 5(b) apresentam, graficamente, a acurária dos estimadores estudados para um tamanho de quadro inicial de 64 e 128 *slots*, respectivamente. Por questões de legibilidade, os intervalos de confiança são apresentados apenas na Tabela 4. Quando o quadro inicial é de 64 *slots*, o seguinte é reforçado: o IV-II, o Vogt (Eom-Lee) e o Eom-Lee possuem acurácias próximas na faixa de 100 a 500 etiquetas e; o IV-II e o Eom-Lee são os dois mais acurados para quantitativos de etiquetas acima de 500, mas as acurácias são próximas. Para um quadro inicial de 128 *slots*, a Figura 5(b) reforça que os três melhores estimadores (IV-II, Vogt (Eom-Lee) e Eom-Lee) possuem acurácias próximas até

700 etiquetas. Acima desse valor, o IV-II é o mais acurado de todos. Sua acurácia chega a ser 2,61 vezes maior do que a acurácia do Eom-Lee para 1.000 etiquetas.



(a) Quadro inicial = 64 *slots* (b) Quadro inicial = 128 *slots*

Figura 5. Acurácia dos Estimadores.



(a) **IV-II** em relação ao **Eom-Lee** (b) **IV-II** em relação ao **Eom-Lee**

(c) **IV-II** em relação ao **Vogt (Eom-Lee)** (d) **IV-II** em relação ao **Vogt (Eom-Lee)**

Figura 6. Redução de *slots*: IV-II em relação ao Eom-Lee e ao Vogt (Eom-Lee).

A Figura 6 apresenta a redução, em número de *slots*, obtida pelo IV-II em relação ao Eom-Lee e ao Vogt (Eom-Lee). Dada a incerteza, a Figura 6(a) sugere que o IV-II e o Eom-Lee são equivalentes, em termos de quantidade de *slots* gerados, quando o quadro inicial é de 64 *slots*. A Figura 6(b) sugere que para um quadro inicial de 128 *slots*, o Eom-Lee e o IV-II são equivalentes, em termos de quantidade de *slots* gerados, até uma população de 700 etiquetas. Contudo, acima desse valor, o IV-II leva vantagem, chegando a usar 69 *slots* a menos do que o Eom-Lee para concluir o processo de identificação de 1.000 etiquetas.

A Figura 6(c) mostra que o IV-II só produz poucos *slots* a mais do que o Vogt (Eom-Lee) quando a população de etiquetas está em torno 400 e um quadro inicial de 64 *slots* é utilizado. Para os demais quantitativos de etiquetas e com base nas incertezas,

o IV-II é equivalente ou melhor. A redução máxima observada com o uso do IV-II em relação ao Vogt (Eom-Lee) é de 232 *slots* para 1.000 etiquetas. A Figura 6(d) sugere que o IV-II e o Vogt (Eom-Lee) são equivalentes, em termos de quantidade de *slots* gerados, até uma população de 700 etiquetas. Acima disso, o IV-II é melhor, usando até 125 *slots* a menos do que o Vogt (Eom-Lee) para concluir a identificação de 1.000 etiquetas.

6. Conclusões

Este artigo propôs os estimadores IV-I e IV-II para o protocolo DFSA em sistemas RFID. O IV-II se mostrou muito acurado e melhor do que o IV-I no confronto direto com o Vogt, o Eom-Lee e o Vogt (Eom-Lee). Os resultados mostraram que o Eom-Lee e o IV-II são os melhores estimadores para um quadro inicial de 64 *slots* e uma população desconhecida de 100 a 1.000 etiquetas. Nessas condições, eles concluem o processo de identificação de etiquetas com uma quantidade equivalente de *slots*.

Por outro lado, o IV-II é o melhor estimador para um quadro inicial de 128 *slots* e uma população desconhecida de 100 a 1.000 etiquetas. Nessas condições, o IV-II permite usar uma quantidade equivalente ou menor de *slots* do que o Eom-Lee e o Vogt (Eom-Lee). Quando melhor, o IV-II gera até 69 *slots* a menos do que o Eom-Lee e até 125 *slots* a menos do que o Vogt (Eom-Lee). Para ambos os tamanhos de quadro inicial estudados, o IV-II foi significativamente mais acurado do que o Vogt.

É importante ressaltar que o estimador é executado no leitor ou em um servidor associado ao leitor, não havendo limitações de recursos computacionais como nas etiquetas. Mesmo assim, é importante haver um estudo aprofundado sobre os custos e o tempo de execução dos estimadores. Tal estudo será realizado em trabalhos futuros.

Referências

Andrade, J. D. (2012). Um Método de Gerenciamento de Frames para o Protocolo DFSA em Sistemas RFID. Dissertação de Mestrado, Centro de Informática – CIn/UFPE.

Andrade, J. D. and Gonçalves, P. A. S. (2011). Uma Função de Cálculo de Tamanho de Frames para o Protocolo DFSA em Sistemas RFID. In *Proc. of XVI Workshop de Gerência e Operação de Redes (WGRS)*, pp. 61–74, Campo Grande, MS.

Cha, J. R. and Kim, J. H. (2005). Novel Anti-collision Algorithms for Fast Object Identification in RFID Systems. In *Proc. of the 11th International Conf. on Parallel and Distributed Systems*, vol. 2, pp. 63–67.

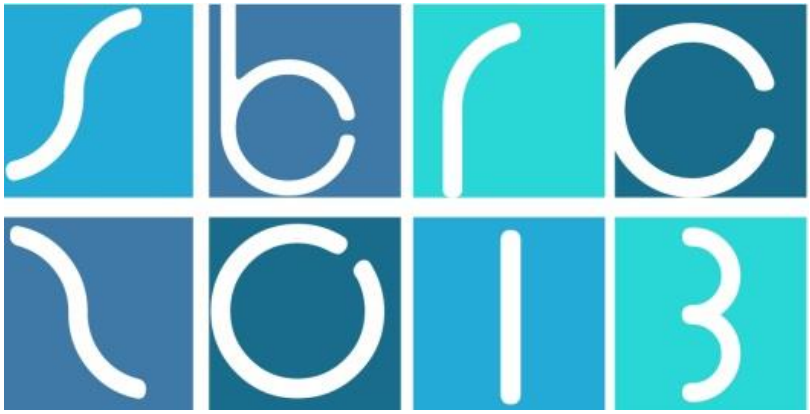
Chen, W.-T. (2009). An Accurate Tag Estimate Method for Improving the Performance of an RFID Anticollision Algorithm Based on Dynamic Frame Length ALOHA. *IEEE Transactions on Automation Science and Engineering*, 6(1):9 –15.

Eom, J.-B. and Lee, T.-J. (2010). Accurate Tag Estimation for Dynamic Framed-slotted ALOHA in RFID Systems. *IEEE Communications Letters*, 14:60–62.

Klair, D., Chin, K.-W., and Raad, R. (2010). A Survey and Tutorial of RFID Anti-Collision Protocols. *IEEE Communications Surveys Tutorials*, 12(3):400 –421.

Schoute, F. C. (1983). Dynamic Frame Length ALOHA. *IEEE Transactions on Communications*, 31:565–568.

Vogt, H. (2002). Efficient Object Identification with Passive RFID Tags. In *Proc. of the First International Conf. on Pervasive Computing*, pp. 98–113, London, UK.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 9

Redes de Sensores sem Fio: Roteamento

Algoritmo Adaptativo Baseado em Energia-Conectividade para Roteamento em Redes de Sensores sem Fio Subaquáticas

Eduardo Chinelate Costa[1], Leonardo Chinelate Costa[1], Luciano Jerez Chaves[2], Alex Borges Vieira[1], Ana Paula Couto da Silva[3*]

[1]Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora – Juiz de Fora – MG

[2]Instituto de Computação
Universidade Estadual de Campinas – Campinas – SP

[3]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais – Belo Horizonte – MG

{eduardo,leonardocosta}@ice.ufjf.br, luciano@lrc.ic.unicamp.br

alex.borges@ufjf.edu.br, ana.coutosilva@dcc.ufmg.br

Resumo. *Neste artigo é proposto um algoritmo adaptativo, em múltiplos saltos, para o roteamento em redes de sensores sem fio subaquáticas. Nesse ambiente, a movimentação dos sensores e economia de energia são pontos desafiadores. Mais ainda, nem sempre é possível ter uma visão completa da rede. Assim, de acordo com o novo algoritmo de roteamento proposto, a escolha do sensor que receberá a informação deve considerar a energia residual dos sensores dentro do raio de alcance, bem como a conectividade deles. Três níveis de decisão são considerados, e políticas que consideram energia ou conectividade ou a combinação de ambos fatores são implementadas. Através de simulação, verifica-se que o algoritmo de roteamento adaptativo proposto aumenta o tempo de vida da rede em até* 75%*, quando nenhuma informação do estado da rede é considerada no processo de roteamento. Quando comparado a algoritmos que consideram somente o fator energia ou conectividade, o ganho da nova abordagem pode chegar a* 57%*.*

Abstract. *In this paper we propose an adaptive routing algorithm for underwater sensor networks that takes into account context information. The choice of the sensor that will receive the data is made considering its residual energy as well as its conectivity. In order to apply the policy that will guide the choice of the next sensor in the routing path, the algorithm is based on three different choice levels. A sensor is chosen by considering either only its residual energy or only its conectivity or the combination of both. Through simulations, we show that applying our adaptive routing algorithm increases the network lifetime by up to* 75% *when compared with the random policy for chosing a sensor. When compared with policies that take into accouting either residual energy or conectivity, our adaptive routing algorithm increases the network lifetime by up to* 57%*.*

*Este trabalho é parcialmente financiado pelas Agências de Fomento CAPES, CNPq e FAPEMIG.

1. Introdução

Sensores fazem parte do nosso dia-a-dia em diversas situações e cenários. Sensores de presença, de monitoramento de temperatura e de umidade, de controle de estoque estão permeados em nosso cotidiano. Na maioria dos casos, estes sensores formam redes nas quais informações são armazenadas, processadas ou simplesmente repassadas entre os sensores que as compõem. Essas redes são denominadas redes de sensores sem fio (RSSFs) e diversos artigos na literatura apresentam aplicações e algoritmos direcionados a esta classe de redes [Akyildiz et al. 2002].

Nos últimos anos, o interesse no monitoramento de regiões subaquáticas tem aumentado expressivamente. Diferentes exemplos corroboram este fenômeno: a questão da segurança de cidades ou até mesmo países, nos casos de desastres naturais, como maremotos e *tsunamis*, bem como vazamento em reservatórios de exploração de petróleo. Usinas hidrelétricas devem monitorar suas barragens, verificando e prevenindo futuros vazamentos, bem como o nível de água para eventuais políticas de racionamento de recursos. Biólogos desejam estudar vidas marinhas existentes ou descobrir novas espécies. Para estes diferentes tipos de monitoramento a melhor estrutura é um sistema distribuído de redes de sensores sem fio subaquáticas.

O monitoramento destes diferentes tipos de regiões através de redes de sensores apresenta diversos desafios. Entre eles, destacam-se a movimentação dos sensores em função das correntes aquáticas e a necessidade de prolongar o tempo de vida da bateria dos mesmos, consequentemente aumentando o tempo de vida da rede como um todo. O tempo de vida da rede é definido como o tempo total em que a rede exerce a atividade de monitoramento continuamente. Dessa forma, contribuições para a melhoria do desempenho destes tipos de redes se concentram, na maioria dos casos, em prover coleta e difusão de informação visando economizar energia da bateria do conjunto de sensores.

Este artigo tem como principal objetivo apresentar um algoritmo adaptativo para o roteamento, em múltiplos saltos, de informações coletadas por sensores em redes subaquáticas. Através de informações locais de cada um dos sensores da rede, o próximo salto no roteamento é definido com base em trêse políticas. Essas políticas levam em consideração a energia residual, a conectividade ou a combinação das duas características de todos os sensores dentro de um raio de cobertura específico, com o principal objetivo de aumentar o tempo de funcionamento da rede. A escolha da política a ser utilizada é baseada na diferença do nível de energia entre os sensores pertencentes à região do raio de cobertura. A utilização do termo adaptativo neste trabalho remete à capacidade do algoritmo em decidir, dinamicamente, pelo uso da política baseada em conectividade, em energia residual, ou na combinação das duas.

Os resultados das simulações realizadas mostram que o algoritmo adaptativo proposto aumenta o tempo de vida da rede em até 75%, quando comparado a um algoritmo em que nenhuma informação do estado da rede é considerada no processo de roteamento. Quando comparado com algoritmos que consideram somente o fator energia ou conectividade isoladamente, o ganho do algoritmo adaptativo proposto pode chegar a 57%. Adicionalmente, o uso do novo algoritmo resulta em uma distribuição mais homogênea da energia residual dos diferentes sensores pertencentes à rede ao final do período de monitoramento. Este resultado mostra que o algoritmo adaptativo, em comparação a algoritmos que não combinam características do contexto da rede, utiliza a energia dos

sensores de maneira mais eficiente. Com os resultados mostrados, o algoritmo adaptativo pode ser utilizado como política de escolha de nós para a implementação de protocolos de roteamento, devido a simplicidade das informações requeridas para o seu funcionamento e da natureza distribuída do mesmo.

2. Trabalhos Relacionados

As restrições impostas pelo meio aquático, como a maior necessidade de economia de energia, principalmente em função do maior consumo nas transmissões acústicas ou a disposição da rede devido ao movimento natural dos sensores no ambiente, fazem com que os protocolos de roteamento desenvolvidos para as RSSFs terrestres não sejam diretamente aplicáveis em RSSFs subaquáticas. Os principais protocolos de roteamento de RSSFs subaquáticas propostos na literatura estão descritos em [Ayaz et al. 2011] e alguns deles são detalhados a seguir. Uma característica comum entre todos os protocolos é a aplicação de políticas simples para escolha dos nós, sendo que a maioria deles usam posicionamento ou energia residual dos sensores como critério de orientação da informação.

O protocolo *Focused Beam Routing* (FBR) [Jornet et al. 2008], estabelece a rota dinamicamente durante a retransmissão dos pacotes de dados. O nó sensor escolhe o próximo nó da rota considerando a energia necessária para transmitir pacotes para este determinado nó. O nó escolhido como receptor deve ser o mais próximo ao emissor, uma vez que se gastará menos energia para o envio de dados a ele. Tal abordagem tenta economizar energia com uma visão local, mas perde a flexibilidade de encontrar regiões com maior número de sensores. Outro protocolo é o *Directional Flooding-Based Routing* (DFR) [Hwang e Kim 2008], onde o nó receptor dos dados é escolhido pelo transmissor considerando sua energia residual e a proximidade deste nó do *sink*.

Alguns protocolos utilizam uma característica específica que é a informação de profundidade dos nós sensores (*Depth-Based Routing* – DBR [Yan et al. 2008]) e de proximidade com o *data sink* (*Sector-Based Routing with Destination Location Prediction* – SBR-DLP [Chirdchoo et al. 2009]). A localização do sensor pode ser utilizada de tal forma que se escolha o mais próximo ao *data sink*. Em outros, como no protocolo *Distributed Underwater Clustering Scheme* (DUCS) [Domingo e Prior 2007], nós sensores podem ser aleatoriamente escolhidos para a formação de *clusters*, onde ocorre a agregação dos dados para posterior retransmissão dos pacotes.

O algoritmo proposto neste artigo se difere dos demais da literatura dado que a escolha do próximo nó é feita de forma adaptativa. Os protocolos existentes utilizam políticas de escolha fixas, baseando-se em somente uma característica dos nós sensores que compõem a rede subaquática. Apesar da abordagem adaptativa proposta neste trabalho ter potencial para ser utilizada em qualquer tipo de RSSFs, ela se compatibiliza fortemente com as redes subaquáticas, devido à simplicidade no processo de escolha dos próximos nós, que é uma característica fundamental neste tipo de ambiente.

3. Formalização do Sistema

A rede RSSF subaquática considerada neste artigo é formada por nós sensores de dois tipos: nós que recolhem as informações do ambiente sendo monitorado e nós que repassam esta informação até o destino final. O destino é um nó especial denominado *sink*, que permanece estático durante todo o tempo de monitoramento.

A informação a ser transmitida é dividida em pacotes de tamanho L, que se propagam no meio aquático, baseando-se em transmissão acústica, com velocidade de $1500m/s$. A rede é formada pelo conjunto de nós sensores $\mathcal{N}$, com cardinalidade $|\mathcal{N}|$. O subconjunto de nós $\mathcal{N}_g \subset \mathcal{N}$ gera informações de sensoriamento a uma taxa de λ e os enviam aos nós sensores vizinhos, ou seja, aos nós dentro dos seus respectivos raios de cobertura; o roteamento é realizado pelos próprios nós sensores em múltiplos saltos até chegar ao nó *sink* N_s. Matematicamente, a rede RSSF é representada por um grafo $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, sendo $\mathcal{N}$ o conjunto de nós sensores e $\mathcal{L}$ o conjunto de ligações que podem ser estabelecidas entre os nós sensores dentro de um raio de vizinhança ρ.

Seja $\mathcal{V}_t(s) \subset \mathcal{N}$ o conjunto de nós sensores dentro do raio de vizinhança de um dado nó sensor s, no instante de tempo t. A lista de nós sensores vizinhos de s, $\mathcal{V}_t(s)$, é dinâmica, com possíveis mudanças a cada movimentação de nós sensores (na Seção 5 é apresentado o modelo de movimentação utilizado neste artigo). A cardinalidade de $\mathcal{V}_t(s)$ é o grau de conectividade do sensor s: quanto maior a quantidade de nós sensores dentro do raio de alcance de s, maior o seu grau de conectividade. Para todo $s \in \mathcal{N}$, denotamos como $e_t(s)$ a energia residual do nó sensor s no tempo t; $e_t(s)$ é a energia máxima que poderá ser utilizada para enviar/receber pacotes. A cada transmissão e recepção de um pacote, um valor constante γ e μ são retirados do montante total de energia $e_t(s)$, respectivamente. A diminuição contínua da energia de um nó resulta na sua inatividade e, consequentemente, no seu posterior desligamento da rede.

Um sensor s pode adotar diferentes estratégias (ou políticas) de escolha do próximo nó sensor a receber o pacote de informação. A mais simples de todas é a aleatória, onde o próximo nó é escolhido uniformemente entre todos os seus vizinhos, desconsiderando informações de contexto do ambiente. Apesar de ser a mais simples, esta política é a mais ineficiente em termos de economia de energia (conforme resultados observados na Seção 5). O principal foco deste artigo é propor uma nova estratégia de escolha do nó que irá receber o pacote a cada salto do roteamento até ao *sink*. A política se adapta às condições atuais da rede, considerando a diferença do nível de energia entre os sensores dentro do raio de alcance do sensor s: a escolha poderá se basear na energia, conectividade ou na combinação destes dois fatores. O objetivo é aumentar o tempo de vida da rede, fazendo com que ela transmita, durante o maior intervalo de tempo possível, informações de monitoração do ambiente aquático em estudo.

4. Algoritmo Adaptativo Baseado em Energia-Conectividade

4.1. Discussão inicial

Nesta seção é apresentada a motivação para o algoritmo proposto neste artigo. Através de resultados obtidos via simulação, foi analisado um cenário (de referência, descrito na Seção 5) em que são implementadas quatro políticas para a escolha do nó sensor que receberá o pacote no próximo salto do roteamento.

A Figura 1 apresenta o tempo total de vida da rede de sensores, ou seja o tempo em que ela permanece em funcionamento, considerando as seguintes políticas:

1. **Aleatória:** Sensores dentro do raio de cobertura de um dado sensor s são escolhidos aleatoriamente, desconsiderando características relacionadas ao contexto da rede ou dos próprios sensores.

2. **Energia:** A probabilidade de escolha de um nó que receberá a informação é proporcional a sua energia residual. Em outras palavras, o sensor com maior energia dentro do raio de cobertura de s terá maior chance de ser escolhido.
3. **Conectividade:** A probabilidade de escolha de um nó que receberá a informação é proporcional a sua conectividade. Em outras palavras, o sensor com maior número de vizinhos dentro do raio de cobertura de s terá maior chance de ser escolhido. A intuição desta política está na possibilidade de explorar sensores com um maior número de nós dentro do raio de cobertura. Um maior espaço amostral de vizinhos, pode, probabilisticamente, aumentar a chance de encontrar nós com maior energia residual.
4. **Energia OU Conectividade:** Nesta política é introduzida aleatoriedade na escolha da característica a ser considerada na escolha do próximo nó. Ou seja, em alguns intervalos de tempo opta-se, probabilisticamente, por direcionar a informação para o sensor de maior energia. Em outros instantes de tempo, opta-se por direcionar a informação para o sensor com maior conectividade. No entanto, esta escolha não é baseada em informações do estado do sistema.

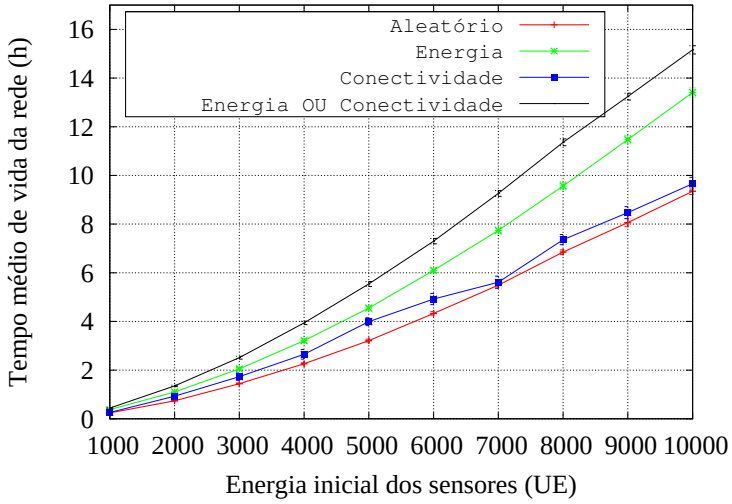


Figura 1. Tempo de vida da rede de sensores para políticas não-adaptativas.

Os resultados apresentados na Figura 1 indicam que considerar alternadamente as características de energia e conectividade aumentam em até 67% o tempo de vida da rede quando comparado ao tempo de vida da política aleatória. Para o caso onde é considerada somente a energia ou somente a conectividade, o ganho alcança 57%. O interessante é que este ganho acontece em uma política onde a alternância entre uma característica e outra não segue nenhuma informação de contexto de todo o sistema ou da visão parcial dentro do raio de cobertura de um sensor. Estes resultados motivam a proposta de um algoritmo que considere energia e conectividade na escolha do nó sensor, bem como a proposta de um mecanismo que adapte esta escolha para condições que podem ser analisadas dentro do raio de cobertura de um determinado sensor s.

4.2. Descrição

Nesta seção é apresentado o algoritmo adaptativo baseado nas características de energia e de conectividade. Os parâmetros utilizados pelo algoritmo são discutidos, bem como o

mecanismo de adaptação proposto.

Seja s o nó sensor que deverá encaminhar a informação a um dos nós em $\mathcal{V}_t(s)$, em um instante de tempo t. Sem perda de generalidade, sejam $k, j \in \mathcal{V}_t(s)$ os sensores que possuem, respectivamente, os valores máximo e mínimo de energia residual, $MAX_e = e_t(k)$ e $MIN_e = e_t(j)$. Seja $\Delta = (MAX_e - MIN_e)/MAX_e$ a variação dos valores de energia dos nós pertencentes ao conjunto $\mathcal{V}_t(s)$. Seja l_h, $h = \{1, 2\}$ valores limites para o parâmetro Δ. O intervalo limitado superiormente por l_1 representa uma pequena variação de energia. O intervalo limitado inferiormente por l_2 representa uma grande variação de energia. A adaptação da política a ser utilizada na escolha de um próximo nó baseia-se na diferença do nível de energia Δ entre os sensores dentro do raio de cobertura de um determinado sensor s:

1. **Pequena Variação -** $0 < \Delta \leq l_1$: Para os casos com pequena variação nos valores de energia residual dos sensores em $\mathcal{V}_t(s)$, a escolha do próximo nó i é proporcional a sua conectividade. Como os valores de energia estão próximos, não é preciso se preocupar com qual nó irá gastar energia na transmissão. Assim, torna-se mais interessante explorar nós com maior conectividade, principalmente porque esta abordagem irá oferecer, no próximo salto da transmissão, mais opções de escolhas, potencializando as chances de encontrar nós com maior energia residual.
2. **Variação Balanceada -** $l_1 < \Delta \leq l_2$: Para os casos intermediários, a escolha do próximo nó i é proporcional a função $f(e_t(i), |\mathcal{V}_t(i)|) = \alpha e_t(i) + (1 - \alpha)|\mathcal{V}_t(i)|$. Nestes casos, onde a diferença de energia não é predominante, combinar as duas características possibilita explorar nós com grande quantidade de energia associado à uma vizinhança maior. Escolher nós com maiores quantidades de energia e de vizinhos possibilita, com maior probabilidade, explorar futuramente uma área de cobertura com nós que possuem uma maior quantidade de energia. O fator α representa o peso dado a cada uma das características consideradas.
3. **Grande Variação -** $l_2 < \Delta \leq 1$: Para os casos com grande variação nos valores de energia residual dos sensores em $\mathcal{V}_t(s)$, a escolha do próximo nó i é proporcional a sua energia residual. Nestes casos, deve-se escolher nós com maior energia, evitando aqueles que poderão ficar com energia abaixo do mínimo necessário para transmitir ou receber informações.

A partir do algoritmo adaptativo proposto, o roteamento em múltiplos saltos considera características locais relacionados ao sensor s, ou seja, a energia residual e/ou a conectividade dos nós pertencentes à $\mathcal{V}_t(s)$. A combinação de diferentes políticas possibilita uma exploração mais ampla dos sensores que possuem maior quantidade de energia residual, evitando o envio de informação para sensores com pouca energia, e, consequentemente, aumentando o tempo de vida útil da rede como um todo. O Algoritmo 1 sumariza o roteamento adaptativo proposto, onde τ é a proporção de nós cujo a energia residual está abaixo do nível necessário para transmitir/receber pacotes de informação.

5. Resultados Numéricos

Nesta seção é investigado o desempenho do algoritmo adaptativo considerando diferentes métricas de interesse. Adicionalmente, será analisado o comportamento do algoritmo quando os valores de seus parâmetros mudam. O desempenho do algoritmo proposto é comparado com o de algoritmos que consideram somente energia ou conectividade, bem

enquanto *sensores em funcionamento* $\geq \tau\%$ *do total de sensores* **faça**
  para todo *sensor* $s \in \mathcal{N}$ **faça**
    se *evento disparado é de transmissão de um pacote* **então**
      /* Escolher o próximo salto*/
      se $0 \leq \Delta \leq l_1$ **então** /*somente conectividade*/
        para todo $i \in V_t(s)$ **faça**
          $prob_i = \frac{|V_t(i)|}{\sum_{\forall_{j;j \in V_t(s)}} V_t(j)}$
        fim
      fim
      se $l_1 < \Delta \leq l_2$ **então** /*energia e conectividade*/
        para todo $i \in V_t(s)$ **faça**
          $f(e_t(i), |\mathcal{V}_t(i)|) = \alpha e_t(i) + (1-\alpha)|V_t(i)|$
          $prob_i = \frac{f(e_t(i),|\mathcal{V}_t(i)|)}{\sum_{\forall_{j;j \in V_t(s)}} f(e_t(j),|\mathcal{V}_t(j)|)}$
        fim
      fim
      se $l_2 < \Delta < 1$ **então** /*somente energia*/
        para todo $i \in V_t(s)$ **faça**
          $prob_i = \frac{e_t(i)}{\sum_{\forall_{j;j \in V_t(s)}} e_t(j)}$
        fim
      fim
      Escolha o sensor i (próximo salto) segundo $prob_i$
    fim
  fim
fim

Algorithm 1: Algoritmo de Roteamento Adaptativo.

como de algoritmos que não consideram nenhuma informação de contexto. Vale ressaltar que a avaliação feita neste artigo não está atrelada a nenhum protocolo de roteamento em particular, mas sim à política de escolha do sensor no próximo salto do roteamento. Desta forma, os resultados apresentados podem ser considerados resultados do melhor caso para os protocolos de roteamento que se baseiam nas políticas avaliadas.

5.1. Implementação do simulador

Os resultados apresentados foram obtidos através de um simulador de eventos discretos implementado especificamente para o sistema descrito na Seção 3. A simulação é organizada em duas fases. Primeiramente, os nós sensores são distribuídos uniformemente em uma superfície quadrada e isopicnal, ou seja, de densidade constante, onde a movimentação do fluxo é praticamente reduzida a duas dimensões. Os nós sensores são projetados para permanecerem em apenas uma dessas superfícies. A área de monitoramento simulada se assemelha àquelas implantadas para o monitoramento de usinas hidrelétricas, regiões costeiras ou campos de petróleo, onde os sensores se encontram a pequenas distâncias uns dos outros. Durante a segunda fase, um subconjunto de sensores recolhem informações que devem ser roteadas até ao *sink*. A simulação continua até que seja alcançada uma porcentagem de $\tau\%$ de sensores com energia residual abaixo da necessária para transmitir/receber informações. Os principais eventos implementados foram o de geração, transmissão e recepção de pacotes, além o evento de movimentação dos sensores.

Os sensores se movimentam segundo o modelo *Random Waypoint* [Broch et al. 1998], projetado para redes *ad-hoc*. Após o posicionamento inicial dos sensores dentro da área de simulação, a movimentação dos nós é feita da seguinte maneira. Cada sensor da rede seleciona aleatoriamente uma posição de destino dentro da região de interesse. O sensor segue em linha reta até o destino, com velocidade constante uniformemente distribuída entre 0 e v_{max}, onde v_{max} é a maior velocidade permitida para todos os nós sensores. A velocidade e a direção de um nó são escolhidas independentemente dos outros nós. Ao alcançar o destino, o nó deixa de se movimentar durante um intervalo de tempo $\mathcal{T}_{pausa}$. Após o tempo de pausa, o nó escolhe, aleatoriamente, um novo destino e uma nova velocidade de deslocamento. Esse processo se repete continuamente até o final da simulação.

Cada sensor s possui um raio de cobertura onde os sensores pertencentes a $\mathcal{V}_t(s)$ estão posicionados. Ao receber ou transmitir um pacote, a energia de um sensor $i \in \mathcal{V}_t(s)$ é consumida. A quantidade total de energia residual $e_t(i)$ é medida em Unidades de Energia (UE). A UE se refere a toda relação de consumo e armazenamento de energia elétrica da RSSF. Cada nó possui uma bateria, com capacidade inicial total de β UE. A cada envio, γ UE são decrescidas da bateria do sensor s. A cada recebimento de pacote, μ UE são decrescidas da bateria do sensor i. Segundo [Akyildiz et al. 2002], a tarefa de transmissão de pacotes é a tarefa que mais consome energia.

Para a análise do desempenho do algoritmo proposto, as seguintes métricas de interesses são calculadas pelo simulador:

1. **Tempo médio de vida da rede**. Denotada por $\mathcal{T}_{\mathcal{G}}$, a variável aleatória tempo de vida da rede é o tempo no qual a rede exerce ininterruptamente suas atividades de monitoramento e transmissão de dados até atingir uma situação em que não é mais

possível realizar as tarefas de transmissão/recebimento de pacotes. Nesse artigo, é calculado o valor médio $E[\mathcal{T}_{\mathcal{G}}]$, considerando as diferentes rodadas de simulação realizadas.

2. **Energia residual média da rede**. Denotada por $\mathcal{E}_{\mathcal{G}}$, a variável aleatória energia residual da rede é a soma total da energia que permanece disponível nos nós sensores após a rede ter interrompido suas atividades de monitoramento. Nesse artigo, é calculado o valor médio $E[\mathcal{E}_{\mathcal{G}}]$, considerando as diferentes rodadas de simulação realizadas.
3. **Jain's Fairness Index para a energia residual dos sensores**. Como o objetivo é prolongar o tempo de vida da rede de sensores, a política ideal é aquela que equilibra o uso da energia dos diversos sensores. Para mensurar o equilíbrio entre valores de energia residual e definindo como mais justa a política onde não existe, ao final da simulação, uma grande variação nos valores de $e(i)$, $\forall_i \in \mathcal{N}$, utiliza-se a métrica *Jain's Fairness Index* [Jain 1991]:

$$\mathcal{J}(e(1), e(2), \ldots, e(|\mathcal{N}|)) = \frac{(\sum_{i=1}^{|\mathcal{N}|} e(i))^2}{|\mathcal{N}| \cdot \sum_{i=1}^{|\mathcal{N}|} e(i)^2}.$$

   No melhor caso, $\mathcal{J} = 1$ e equivale ao caso onde todos os nós sensores possuem o mesmo valor final de energia residual. Em outras palavras, a política de escolha resultou em um equilíbrio na utilização da energia individual dos nós sensores, não sobrecarregando nós específicos. Quanto mais próximo $\mathcal{J}$ está de 1, menor a variação entre os valores da energia residual. No pior caso, $\mathcal{J} = 1/n$.

5.2. Cenário de Referência

Com o objetivo de analisar o desempenho do algoritmo adaptativo, considera-se o seguinte cenário de referência: a rede é composta por 100 sensores, uniformemente posicionados em uma região quadrada de $1000\,m \times 1000\,m$. O total de $|\mathcal{N}_g| = 10$ sensores geram informações de sensoriamento. A velocidade máxima de deslocamento dos nós no interior da área monitorada é $v_{max} = 20\,m/s$ [Yoon et al. 2003, Kumar et al. 2011]. O intervalo entre deslocamentos sucessivos é de $\mathcal{T}_{pausa} = 5s$. O raio de cobertura considerado é $\rho = 100\,m$. A Unidade de Energia inicial total β varia no conjunto $\{1000, 2000, \cdots, 10000\}$. Cada envio consome $\gamma = 5$ UE e cada recebimento consome $\mu = 1$ UE [Sanchez et al. 2011]. Em relação aos parâmetros do algoritmo adaptativo, serão utilizados os valores $\alpha = 0.5$; l_1 e l_2 iguais a 0.3 e 0.6, respectivamente. Nas simulações realizadas, pacotes de controle (sinalização) são desconsiderados.

Os resultados das métricas de interesse foram obtidos através da média de 500 rodadas independentes de simulação, com intervalo de confiança de 95%. Inicialmente o critério de parada da simulação é o descarregamento da bateria de 15% dos nós sensores. Em todas as execuções realizadas, o algoritmo adaptativo convergiu e se mostrou estável.

5.3. Desempenho do Algoritmo Adaptativo

Uma das métricas mais importantes a ser analisada é o tempo médio de vida da rede, $E[\mathcal{T}_{\mathcal{G}}]$. A Figura 2 apresenta os resultados obtidos pelo algoritmo que implementa políticas adaptativas e os resultados das demais políticas de escolha de sensores, descritas na Seção 4.1. No melhor caso, o aumento do tempo médio de vida da rede é de aproximadamente 75% quando comparado com a política aleatória, aproximadamente 26% quando

somente a informação de energia é considerada e aproximadamente 57% quando somente a conectividade é considerada.

Conforme esperado, o aumento do valor de β resulta em um melhor desempenho quando as informações de contexto dos sensores são incluídas na política de escolha do próximo nó sensor. Para pequenos valores de β (≤ 4000), o desempenho do algoritmo adaptativo está próximo ao dos algoritmos que consideram somente energia ou conectividade. O ganho do algoritmo adaptativo é ainda mais expressivo nos casos onde β possui valores mais elevados. Nesses casos, os sensores possuem um intervalo maior de valores de energia que pode ser melhor explorado pelas políticas de escolhas do próximo nó. É possível observar um comportamento exponencial no tempo médio de vida da rede para todos os algoritmos. Esse comportamento é consequência do modelo utilizado nas simulações, onde o número de eventos de transmissão diminui à medida que se aproxima do fim da simulação e os nós vão se desligando da rede por falta de energia.

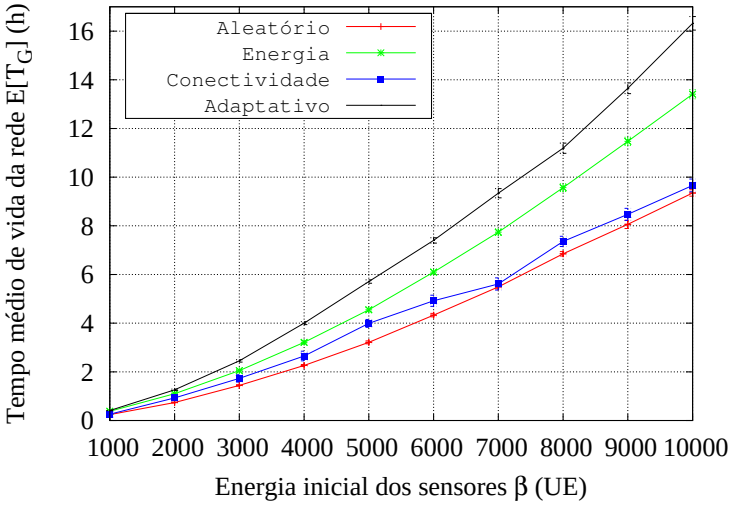


Figura 2. Tempo médio de vida da rede - Comparação entre algoritmos.

A Figura 3 apresenta os resultados da energia média residual para as diferentes políticas de escolha do próximo nó. O algoritmo adaptativo apresenta uma menor energia residual que as abordagens aleatória e conectividade. Isso indica uma melhor utilização da energia se comparado a essas duas abordagens. Para o caso da política que considera somente energia, o desempenho é equivalente ao do algoritmo adaptativo proposto. O comportamento mostrado na Figura 3 é consequência da natureza das políticas que se baseiam em energia, dado que o principal objetivo é equilibrar o uso da energia de todos os nós sensores da rede de monitoramento.

Finalmente, a Tabela 1 apresenta os valores para o *Jain's Fairness Index*, $\mathcal{J}$, considerando diferentes políticas de escolha do próximo nó. Devido à própria definição do fator $\mathcal{J}$, que considera a energia residual entre os diversos sensores como fator de justiça, a política que considera somente a energia residual e o algoritmo adaptativo proposto alcançam os maiores valores de $\mathcal{J}$. Como o objetivo é aumentar o tempo de vida da rede, e este tempo está diretamente ligado ao uso equilibrado de energia dos sensores, esta definição é a que se enquadra melhor no contexto aqui apresentado. No entanto, vale ressaltar que, apesar de possuírem valores bem próximos do fator $\mathcal{J}$, o algoritmo adapta-

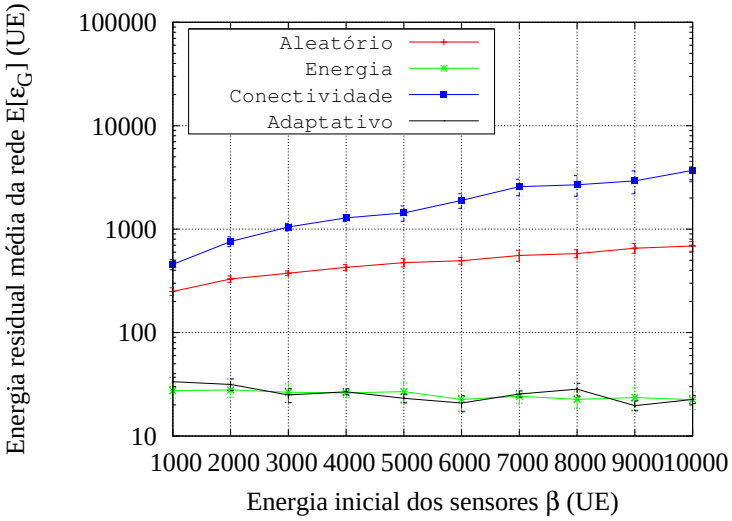


Figura 3. Energia residual média - Comparação entre algoritmos.

Algoritmo	$\mathcal{J}$
Aleatório	0.85
Somente Energia	0.98
Somente Conectividade	0.83
Adaptativo	0.99

Tabela 1. Jain's Fairness Index - Comparação entre algoritmos.

tivo aumenta em até 26% o tempo de vida da rede quando comparado com algoritmos de roteamento que consideram somente energia.

5.4. Análise dos Parâmetros do Algoritmo Adaptativo

Conforme mostrado anteriormente, o algoritmo adaptativo aumenta em até 75% o tempo de vida da rede quando comparado com políticas mais simples que não consideram características da rede. Dessa forma, o uso do algoritmo proposto é atrativo principalmente nos casos onde a substituição de sensores não é uma tarefa trivial. Para finalizar o estudo do algoritmo adaptativo, nesta seção é analisado o comportamento do algoritmo quando os valores dos seus parâmetros são variados. Para os parâmetros não explicitados serão considerados os valores descritos no cenário de referência (Seção 5.2).

Variação dos níveis de energia l_1 e l_2

Sejam os valores l_1 e l_2 que determinam os tamanhos dos intervalos onde serão consideradas somente a conectividade, somente a energia ou a combinação de ambas características na escolha do sensor no próximo salto do roteamento. A Figura 4 mostra os valores do tempo médio de vida da rede para três pares diferentes de valores de l_1 e l_2. Os resultados indicam que o algoritmo adaptativo alcança o melhor desempenho no caso em que é feita a escolha equilibrada da política a ser utilizada na escolha do próximo nó. A diversidade espacial deve ser explorada, diminuindo a sobrecarga em sensores específicos. Em outras palavras, todas as características dos sensores devem ser exploradas de forma equilibrada,

não privilegiando uma determinada característica. Por exemplo, no caso da conectividade, explorar somente esta característica pode levar a situações em que o mesmo sensor tem a maior chance de ser escolhido a cada salto do roteamento, diminuindo muito a sua energia residual. Por consequência, diminuindo o tempo de vida da rede. Para a variação dos parâmetros τ e α, serão considerados os valores de $l_1 = 0.3$ e $l_2 = 0.6$. No entanto, vale ressaltar que o desempenho do algoritmo proposto é superior a todos os casos onde somente uma das características é considerada.

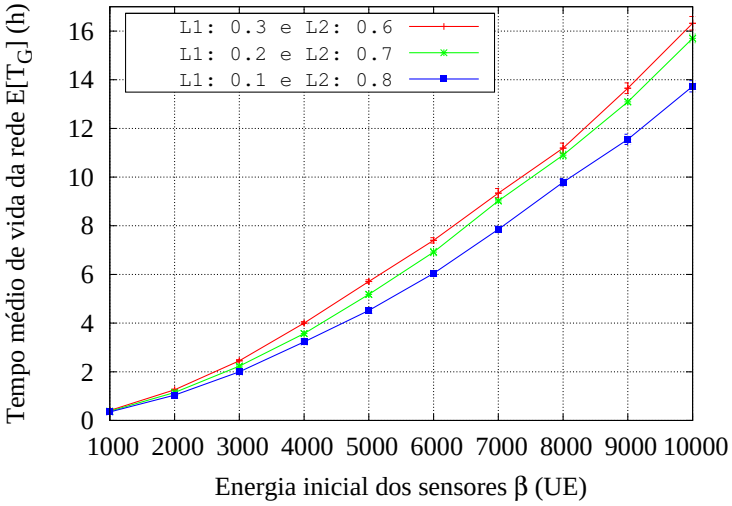


Figura 4. Variação dos parâmetros $l_h, h = \{1, 2\}$.

Variação do critério de parada τ

O critério de parada considerado nas simulações realizadas é a porcentagem de nós sensores que não podem colaborar para o roteamento das informações monitoradas desde a origem até o *sink*, dado que não possuem a quantidade de energia mínima necessária. A Figura 5 mostra os resultados para valores de $\tau = \{5\%, 10\%, 15\%, 20\%\}$. Para todos os casos considerados, os valores do tempo médio de vida permanecem praticamente iguais.

Com a utilização do algoritmo adaptativo, os nós sensores da rede consomem energia uniformemente. Ao se descarregar alguns sensores, a tendência é que vários sensores se descarregam logo a seguir. Assim, a diferença dos tempos entre descarregar 5% ou 20% dos sensores da rede é mínima. Este fenômeno explica a proximidade do tempo médio de vida para os diferentes valores de τ considerados. Vale ressaltar que, para todos os casos apresentados, a rede possui um tempo de vida longo.

Variação do peso α para conectividade e energia

Finalmente, é avaliado o comportamento do novo algoritmo adaptativo quando se varia o valor do peso dado para a energia e para a conectividade quando um nó é escolhido (parâmetro α). No caso do algoritmo adaptativo proposto, α é utilizado para calibrar a função $f(e_t(i), |\mathcal{V}_t(i)|)$ aplicada na escolha do próximo nó dentro do intervalo $[l_1, l_2]$. Os

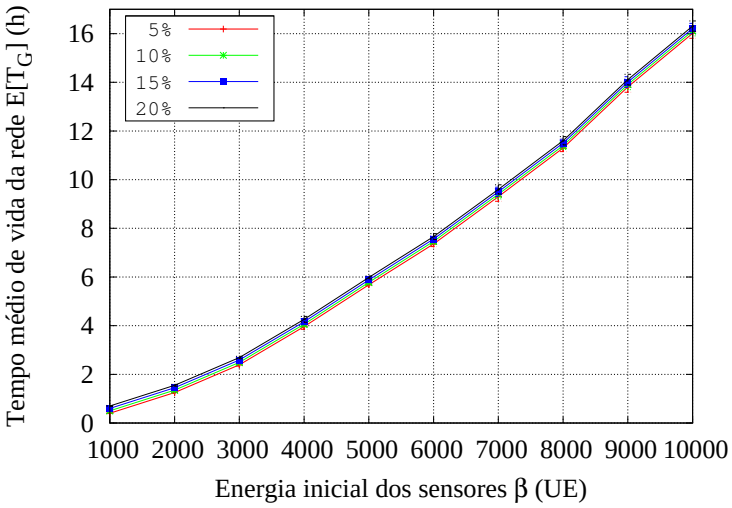


Figura 5. Variação do critério de parada τ.

resultados na Figura 6 mostram que a variação do parâmetro α não causa grande impacto no tempo de vida da rede. Este resultado indica que o fator de impacto mais importante é considerar diferentes políticas de escolha a cada salto do roteamento, e não o peso que deve ser dado quando é utilizada a função $f(e_t(i), |\mathcal{V}_t(i)|)$. Dessa forma, o algoritmo é robusto a variações do parâmetro α.

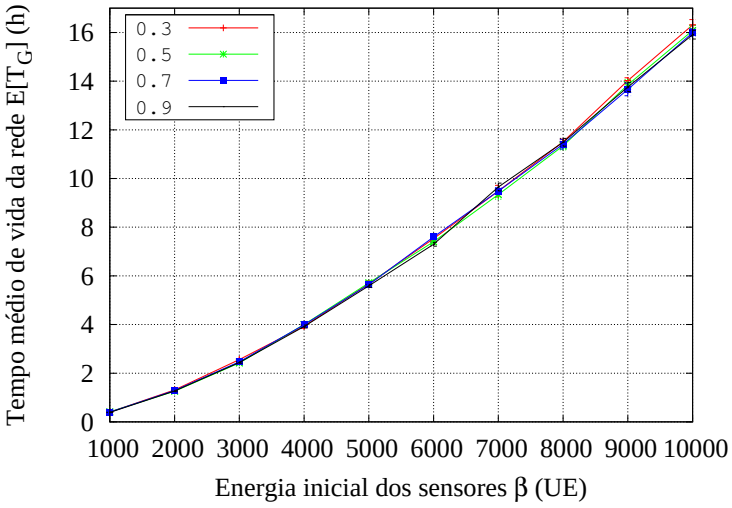


Figura 6. Variação do peso α para conectividade e energia.

6. Conclusões

Neste artigo foi proposto um algoritmo adaptativo para o roteamento de informações em redes de sensores sem fio subaquáticas. Devido a natureza destas redes, a questão da economia de energia deve ser considerada, dado que a troca de baterias dos sensores não é uma tarefa trivial.

O algoritmo adaptativo considera três diferentes níveis e políticas para a escolha do nó do próximo salto de roteamento. Dada a diferença entre os níveis de energia

dos sensores presentes no raio de cobertura de um sensor em particular s, o algoritmo considera somente a energia, a conectividade ou a combinação das duas características para associar a importância do nó, e consequentemente, a maior chance de ser o nó que receberá a informação.

Através de simulação, verificou-se que o algoritmo de roteamento adaptativo proposto aumenta o tempo de vida da rede em até 75%, quando nenhuma informação do estado da rede é considerada no processo de roteamento. Quando comparado a algoritmos que consideram somente o fator energia ou conectividade, o ganho da nova abordagem pode chegar a 57%. Adicionalmente, mostrou-se que o algoritmo é robusto a variação dos valores dos seus parâmetros.

Como trabalhos futuros, será realizada uma análise mais formal da convergência e estabilidade do algoritmo adaptativo proposto.

Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., e Cayirci, E. (2002). Wireless sensor networks: a survey. *Comput. Netw.*, 38:393–422.

Ayaz, M., Baig, I., Abdullah, A., e Faye, I. (2011). A survey on routing techniques in underwater wireless sensor networks. *J. Netw. Comput. Appl.*, 34(6):1908–1927.

Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., e Jetcheva, J. (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of the ACM MobiCom '98*, páginas 85–97.

Chirdchoo, N., Soh, W.-S., e Chua, K. C. (2009). Sector-based routing with destination location prediction for underwater mobile networks. In *Proc. of the IEEE WAINA '09*, páginas 1148–1153.

Domingo, M. C. e Prior, R. (2007). Design and analysis of a gps-free routing protocol for underwater wireless sensor networks in deep water. In *Proc. of the SENSORCOMM '07*, páginas 215–220.

Hwang, D. e Kim, D. (2008). Dfr: Directional flooding-based routing protocol for underwater sensor networks. In *Proc. of the OCEANS '08*, páginas 1–7.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. Wiley Interscience.

Jornet, J. M., Stojanovic, M., e Zorzi, M. (2008). Focused beam routing protocol for underwater acoustic networks. In *Proc. of the ACM WuWNeT '08*, páginas 75–82.

Kumar, S., Sharma, S. C., e Suman, B. (2011). Simulation based performance analysis of routing protocols using random waypoint mobility model in mobile ad hoc network. *Glob. J. of Comput. Sc. and Tech.*, 11(1).

Sanchez, A., Blanc, S., Yuste, P., e Serrano, J. (2011). A low cost and high efficient acoustic modem for underwater sensor networks. In *Proc. of the OCEANS '11*, páginas 1 –10.

Yan, H., Shi, Z. J., e Cui, J.-H. (2008). DBR: Depth-based routing for underwater sensor networks. *Int. Federation for Information Processing*, páginas 72–86.

Yoon, J., Liu, M., e Noble, B. (2003). Random waypoint considered harmful. In *Proc. of the IEEE INFOCOM '03*, páginas 1312–1321.

Controle energeticamente eficiente de múltiplos saltos para Redes de Sensores sem Fio heterogêneas utilizando Lógica Fuzzy

Alexandre M. Silva[1], Christiano Maciel[2]

[1]Instituto de Ciências da Computação – PPGCC - Universidade Federal do Pará (UFPA)
Caixa Postal 479 – 66.075-110 – Belém – PA – Brasil

[2]Instituto de Engenharia Elétrica – PPGEE - Universidade Federal do Pará (UFPA)
Belém, PA - Brasil.

{amelo,christiano}@ufpa.br

Abstract. *This paper, we propose a centralized control to elect more appropriate Cluster Heads, assuming three levels of heterogeneity and multi-hop communication between Cluster Heads. The centralized control uses the k-means algorithm, responsible for the division of clusters and Fuzzy Logic to elect the Cluster Head and selecting the best route of communication between elected. The simulations indicate that the centralized control, the inclusion of three levels of heterogeneity and multi-hop communication to farthest Cluster Heads can increase the period of stability and lifetime in WSN.*

Resumo. *Neste artigo é proposto um controle centralizado para eleger Cluster Heads mais adequados, admitindo três níveis de heterogeneidade e uma comunicação de múltiplos saltos entre Cluster Heads. O controle centralizado utiliza o algoritmo k-means, responsável pela divisão dos clusters e Lógica Fuzzy para eleição do Cluster Head e seleção da melhor rota de comunicação entre os eleitos. As simulações indicam que um controle centralizado, a inserção de três níveis de heterogeneidade e a comunicação com múltiplos saltos para Cluster Heads mais afastados permitem aumentar o período de estabilidade e o tempo de vida útil em RSSF.*

1. Introdução

Representando uma subclasse das redes *ad hoc* sem fio, as Redes de Sensores Sem Fio (RSSF) são consideradas como uma nova geração de sistemas embarcados de tempo real com recursos computacionais, energia e memória limitados. Tendo na restrição de energia um dos principais entraves apresentados devido à capacidade limitada das baterias internas dos nós sensores. Vários fatores são culminantes para o desgaste da bateria dos nós, sendo o módulo de rádio um dos principais consumidores de energia dos nós sensores no processo de transmissão de dados [1].

O consumo de energia pode ser reduzido, admitindo que apenas alguns nós possam enviar dados para a Estação Base (*base station)*. RSSF hierárquicas organizam seus nós em agrupamentos (*clusters*) e elegem um nó líder do grupo, denominado *cluster head* (CH). O CH é responsável por coletar todos os dados dos nós de seu *cluster*, informações provenientes de sensoriamento, podendo agregá-los e posteriormente encaminhá-los à Estação Base (BS) [2].

A estrutura hierárquica pode ser formada por dois tipos de redes, homogêneas ou heterogêneas. Em estruturas heterogêneas, tratado nesta proposta, alguns nós sensores podem apresentar requisitos de hardware diferenciados, como melhor capacidade energética. Estes nós sensores dão a rede um maior período de estabilidade [3, 4]. Trabalhos que consideram a heterogeneidade dos nós podem ser encontrados em [3, 5, 6, 2, 4, 7].

O algoritmo LEACH (*Low Energy Adaptative Clustering Hierarchy*) proposto por [8] é a solução mais popular encontrada na literatura para formação de *clusters* e serve como base para inúmeros trabalhos voltados para clusterização. Como no LEACH, o grande problema destes algoritmos é a utilização de informações locais com base em cálculos de probabilidade para eleição dos líderes dos *clusters*. Esse tipo de seleção pode gerar CHs muito próximos da borda da rede, aumentando a dissipação de energia devido à distância de transmissão dos nós para o CH. Outro grande problema com a forma de eleição utilizada pelo algoritmo LEACH, é a falta de tratamento discriminatório sobre as discrepâncias energéticas dos nós que formam a rede, uma vez que os CHs selecionados devem possuir recursos energéticos suficientes para suportar as cargas de transmissão dos nós associados a ele.

Considerando os entraves relacionados à eleição de CHs com base em informações locais sem considerar critérios de posicionamento, este artigo propõe uma estratégia para eleição do CH ideal em RSSF heterogêneas, utilizando Lógica *Fuzzy* com base em informações centralizadas na BS. Este controle centralizado define o CH com base em informações adquiridas no momento de formação da rede. As informações coletadas são utilizadas para carregar o algoritmo *k-means*, responsável pela divisão dos *clusters*, como também, o sistema *fuzzy*, que se encarrega de selecionar o líder de cada grupo formado pelo algoritmo *k-means*.

O sistema *fuzzy* também é responsável por determinar quais CHs passarão pelo processo de encaminhamento de dados dos CHs mais afastados, nós eleitos que ultrapassam o limiar de comunicação com a BS devem enviar seus dados para os nós mais próximos. Para esta seleção o Sistema *Fuzzy* utiliza como critérios: distância de comunicação e níveis de energia. Os critérios adotados para seleção de CHs são: nível de energia, centralidade e proximidade para a BS. A inserção de três níveis de heterogeneidade, a utilização de informações centralizadas na BS e a comunicação através de múltiplos saltos permitem eleger CHs bem posicionados e com níveis adequados de energia para suportar a carga de transmissão de seu *cluster*, aumentando o período de estabilidade e vida útil da rede.

O restante do trabalho está organizado da seguinte forma. Seção 2 apresenta alguns trabalhos relacionados à eleição de CHs, exemplos de motivação de algoritmos propostos e definições de termos. Seção 3 apresenta uma breve introdução a Lógica *Fuzzy* e algoritmo *k-means*. Os modelos analíticos e a modelagem utilizada na proposta são apresentados na Seção 4. Seção 5 discute os resultados e a simulação para o sistema fuzzy e, finalmente, Seção 6 conclui o artigo.

2. Trabalhos Correlatos

O Algoritmo LEACH (*Low Energy Adaptative Clustering Hierarchy*), proposto por [8], elege o CH com base em informações locais a cada novo ciclo (*round*). Para que o nó seja eleito, o número escolhido deve ser menor que o limiar *T*. Após a eleição, o novo

CH envia mensagens de anúncio para todos os nodos da rede. Os demais nós associados decidem a que CH devem se conectar.

Embora o LEACH apresente uma estrutura hierárquica que permite reduzir o consumo de energia em RSSF, o mesmo não adota nenhum critério de posicionamento no momento de eleição do CH, podendo selecioná-lo próximo a borda da rede. Outro problema do algoritmo é descrito no trabalho apresentado em [7] SEP (*Stable Election Protocol*), comprovando que o LEACH não é eficiente em estruturas heterogêneas, por não considerar a discrepância de energia dos nós que formam a rede. Semelhante ao DEEC (*Distributed energy efficient clustering*) proposto por [9], utiliza informações locais para eleição do CH, entretanto, é capaz de tratar a heterogeneidade da rede. Sendo que a heterogeneidade inserida pelos autores diz respeito apenas à capacidade energética diferenciada de um conjunto de nós de que formam a rede.

Baseado no algoritmo DEEC, os algoritmos E-DEEC (*Enhanced Distributed Energy Efficient Clustering*) proposto por [3] e LEACH-HPR proposto por [6] também consideram a energia residual dos nós no processo de eleição para o CH da rede. O diferencial está na inserção de três tipos de nós com diferentes níveis de energia: nós normais, nós avançados e super nós, permitindo prolongar o período totalmente funcional da rede com a adição dos super nós. Utilizam informações locais para eleição do CH. Para minimizar a dissipação de energia na fase de comunicação entre os CHs mais afastados e a BS, os autores de LEACH-HPR propõem um algoritmo de múltiplos saltos entre os CHs eleitos, semelhante ao ACHTLEACH (*Adaptive Cluster Head Election and Two-hop* LEACH) proposto por [3].

Em [10] os autores propõem um algoritmo que considera uma estratégia híbrida móvel com o objetivo de distribuir o consumo de energia por toda a rede com o CH movimentando-se para um local de maior concentração de energia quando ocorrer um evento e com possibilidade de controlar sua potência de transmissão. Entretanto, este deslocamento pode maximizar o consumo de energia gasto no envio de dados para a BS. A estratégia móvel híbrida do algoritmo BS-CH permite a movimentação da estação base para minimizar a distância para o CH e o consumo na transmissão de dados, desconsiderando a energia consumida, supondo que a estação base é móvel e sem informação se a fonte de alimentação é contínua.

3. Lógica *Fuzzy* e Algoritmo *K-Means*

Proposto por [11], a Lógica *Fuzzy* utiliza métodos com o objetivo de controlar a linguagem vaga e a imprecisão utilizada diretamente pelo homem, por meio de um conjunto de valores representados por variáveis linguísticas. Cada conjunto de valores tem um intervalo diretamente associado a regras semânticas. Ao contrário da Lógica booleana, na lógica *fuzzy* a avaliação de uma determinada proposição pode compreender valores e graus de pertinência que variam no intervalo de (0,1).

O algoritmo *k-means* é uma técnica de agrupamento de dados por k-médias muito popular por sua facilidade de implementação. Normalmente os algoritmos de clusterização são amplamente utilizados em aplicações que necessitem gerar padrões, dividindo os objetos em grupos úteis ou significativos [7]. Para a proposta apresentada neste trabalho o algoritmo k-means é carregado com as coordenadas de todos os nós que foram à rede. Desta forma o algoritmo gera um padrão, dividindo os clusters, estes, formados pelos nós mais próximos.

4. Modelagem

Para a solução proposta neste trabalho, a escolha do *cluster head* se dá a cada *round*. Um *round* termina no final do processo de agregação e envio de dados para a BS. Basicamente o processo é dividido em três etapas: (i) A primeira etapa consiste no início do processo de formação dos *clusters*. A divisão dos *clusters* é feita utilizando o algoritmo *k-means*; (ii) A segunda etapa é dividida em duas fases. A primeira fase consiste na seleção do CH para cada *cluster* formado pelo *k-means*. Os dados de cada *cluster* são carregados no Sistema *Fuzzy*, e este, baseado nos critérios adotados seleciona os CHs mais adequados para cada *cluster*.

Na segunda fase, após o resultado da seleção de líderes, a BS calcula a distância dos CHs que ultrapassaram o limiar de comunicação para todos os *Cluster Heads* eleitos. Este cálculo é utilizado para definir para que líder o CH mais afastado deve transmitir seus dados já agregados. Após a conclusão das duas fases, a estação base envia mensagens em *broadcast* para os nós da rede informando o ID do líder do grupo para que os nós possam enviar dados para seu respectivo CH; (iii) A terceira etapa concerne no processo de agregação dos dados pelo CH. Este processo consiste em comprimir os dados e enviá-los a BS.

No processo de associação dos nós de um determinado *cluster* para seu respectivo CH, os nós que formam o *cluster* recebem a mensagem de anúncio da BS informando a que líder deve enviar pedidos de associação. O líder reserva um *slot* TDMA (*Time Division Multiple Access*) para cada nó associado, para que possam transmitir seus dados. Os líderes responsáveis por propagar para o ponto de coleta os dados já agregados de CHs mais afastados, recebem da BS, juntamente com a mensagem de anúncio, o pedido de reserva de um *slot* TDMA para este processo.

4.1. Modelo de dissipação de energia

O modelo de dissipação de energia adotado é semelhante ao modelo utilizado pelo LEACH. O modelo consiste na energia dissipada na transmissão e recepção de *k-bit* de mensagem em uma distância ***d,*** o radio consome:

$$E_{Tx}(k,d) = E_{elec} * k + \varepsilon_{fs} * k * d^2 \quad para\ d < d_0 \quad E_{Tx}(k,d) = E_{elec} * k + \varepsilon_{amp} * k * d^4 \quad para\ d \geq d_0 \quad (1)$$

$$E_{Rx}(k,d) = E_{elec} * k \quad (2)$$

A energia dissipada na transmissão e recepção do radio é representada por $E_{elec} = 50\frac{nJ}{bit}$. Dois modelos são utilizados, *Free space* e *Multipath*, para que o amplificador de transmissão alcance um nível aceitável, dependendo da distância entre o transmissor e o receptor. Se esta distância não ultrapassar o limiar d_0, o modelo de *Free space* é utilizado. A energia dissipada pelo amplificador de transmissão é dada por $E_{fs} = 10\frac{\frac{pJ}{bit}}{m^2}$, caso o limiar seja ultrapassado o modelo *Multipath* é utilizado $\frac{E_{amp} = 0.0013\frac{pJ}{bit}}{m^4}$. Onde $d_0 = \sqrt{\frac{E_{fs}}{E_{amp}}}$, [12, 5].

O modelo de *Multipath* gera maior dissipação de energia no processo de comunicação. Estratégias energeticamente eficientes devem considerar o

posicionamento dos nós no processo de seleção de CHs para que o modelo de *Free space* seja mantido.

4.2. Critérios para eleição do CH

Os critérios utilizados para carregar o sistema *fuzzy* são: (i) Energia – O nível de energia de cada nó da rede é representado pela variável linguística Bateria e possui os valores linguísticos Baixa, Moderada e Alta. Consideramos que a rede possui características heterogêneas no que diz respeito a energia dos nós. Logo, os super nós e nós avançados representam maior chance para eleição do CH, Figura 1. (ii) Centralidade – A variável centralidade diz respeito ao posicionamento do nó em relação ao centro do *cluster*. E respectivamente os valores linguísticos que representam a variável centralidade são: Perto, Moderado e Longe. Quanto menor o valor de centralidade mais próximo o nó está do centro do cluster, Figuras 1 e 2.

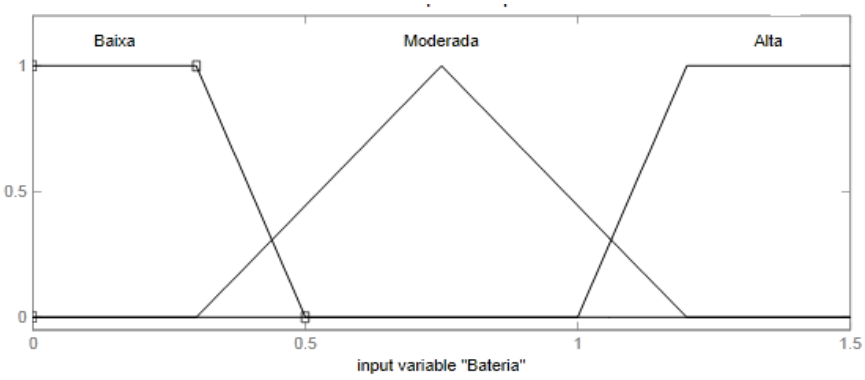


Figura 1. Variável bateria com os valores linguísticos correspondentes aos níveis de energia dos três tipos de nós sensores disponíveis na rede.

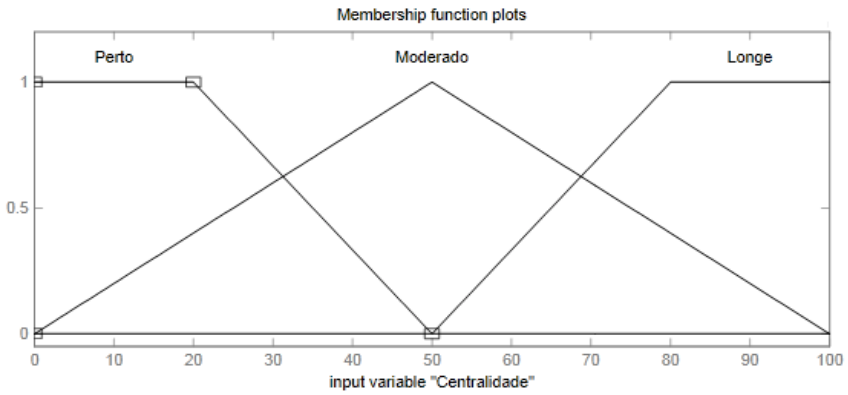


Figura 2. Variável linguística centralidade.

(iii) Distância para BS – A variável é representada no sistema como DistBS. Semelhante a variável centralidade, utiliza os valores linguísticos: Perto, Moderado e Longe. O critério de distância para a BS é utilizado para gerar uma aproximação do CH com a BS, objetivando minimizar o consumo de energia do CH na fase de transmissão de dados, Figura 3.

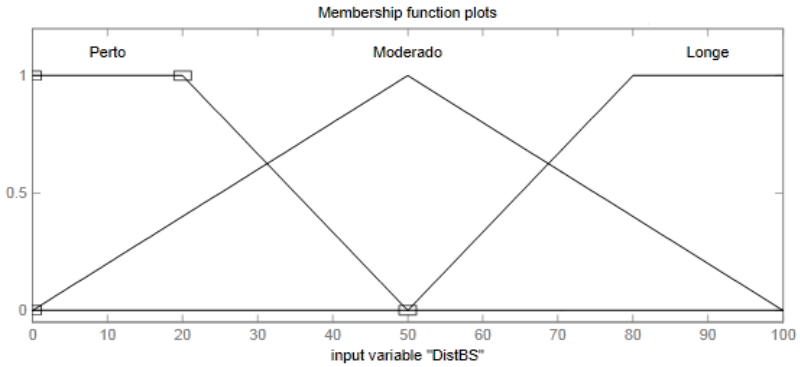


Figura 3. Distância para BS.

As variáveis descritas acima correspondem aos antecedentes do sistema *fuzzy* e dão entrada ao processo de fuzzyficação. Cada valor de entrada é mapeado para um conjunto *fuzzy* de entrada com seu determinado grau de pertinência e os consequentes do sistema, conjuntos *fuzzy* de saída do sistema que vão determinar o CH ideal em um *cluster x,* usam os seguintes valores linguísticos: *muito fraco, fraco, médio, forte e muito forte,* Figura. 4.

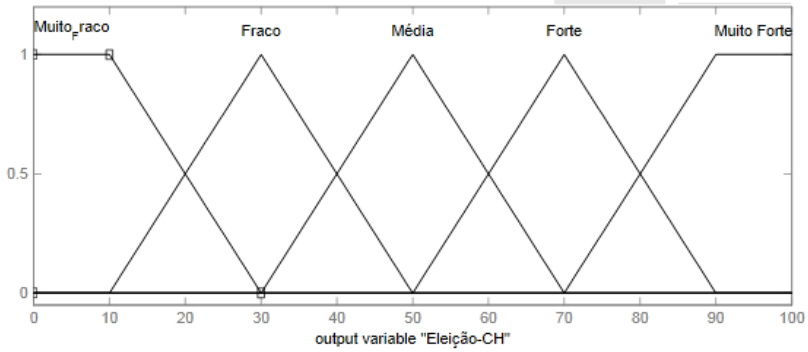


Figura 4. Saída do sistema *fuzzy*.

No processo de mapeamento dos conjuntos *fuzzy* de saída para valores *crisp* (defuzzificação), cada nó sensor apresenta seu respectivo valor de saída do sistema *fuzzy*. Os maiores valores de saída indicam os *cluster heads* selecionados para o *round* atual. A da Base de Regras do sistema é composta por vinte e sete regras 3^3=27, Tabela 1. A melhor condição para a eleição do *cluster head* é dada pela seguinte regra: Se *Centralidade é* PERTO e *Bateria é* ALTA e *DistBS é* PERTO então *Eleição cluster head* ou *Saída é* MUITO FORTE.

4.3. Centralidade dos nós e distância para BS

Assumimos que a BS detém o conhecimento de nível de energia e posicionamento dos nós. Estas informações são enviadas no início de formação da rede e os nós dissipam energia neste processo. Para determinar os valores de centralidade, a BS seleciona cada nó e calcula a distância euclidiana destes nós para o centro dos seus respectivos *clusters*, definidos pelo algoritmo *k-means*, Eq. (3). O nó que apresentar maior centralidade, como CH, permitirá que a dissipação de energia na comunicação ocorra de forma mais homogênea em relação aos demais nós associados.

$$d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (3)$$

Tabela 1. Base de regras do sistema fuzzy

Regras	*Entradas*			*Saída*
	Centralidade	Bateria	DistBS	Eleição
25.	Longe	Alta	Longe	Fraco
23.	Longe	Alta	Moderado	Fraco
9.	Longe	Alta	Perto	Média
7.	Longe	Baixa	Longe	Muito Fraco
22.	Longe	Baixa	Moderado	Muito Fraco
20.	Longe	Baixa	Perto	Muito Fraco
24.	Longe	Moderada	Longe	Muito Fraco
8.	Longe	Moderada	Moderado	Fraco
21.	Longe	Moderada	Perto	Muito Fraco
27.	Moderado	Alta	Longe	Média
16.	Moderado	Alta	Moderado	Forte
6.	Moderado	Alta	Perto	Forte
4.	Moderado	Baixa	Longe	Muito Fraco
15.	Moderado	Baixa	Moderado	Fraco
17.	Moderado	Baixa	Perto	Muito Fraco
19.	Moderado	Moderada	Longe	Fraco
5.	Moderado	Moderada	Moderado	Forte
18.	Moderado	Moderada	Perto	Média
3.	Perto	Alta	é Perto	Muito Forte
10.	Perto	Alta	Longe	Forte
14.	Perto	Alta	Moderado	Muito Forte
1.	Perto	Baixa	Longe	Muito Fraco
26.	Perto	Baixa	Moderado	Muito_Fraco
13.	Perto	Baixa	Perto	Muito Fraco
11.	Perto	Moderada	Longe	Média
2.	Perto	Moderada	Moderado	Forte
12.	Perto	Moderada	Perto	Forte

O mesmo processo ocorre no cálculo da distância de cada nó para a BS. Entretanto, vale resaltar que este critério é muito importante quando a BS não está localizada demasiadamente longe do cluster em questão. Logo, a disposição da rede deve ser considerada para que o critério distância para BS seja válido. Com isso, definimos regras seguras para não gerar CHs próximo da borda da rede, já que a centralidade é mais importante.

4.4. Múltiplos Saltos entre CHs

Esta estratégia é utilizada para *Cluster Heads* que ultrapassam o limiar de comunicação, $d > d_0$, para o ponto de coleta. A aproximação para a BS, apresentada anteriormente, nem sempre se aplicará, uma vez que dependendo da disposição dos nós, *clusters* podem ser formados demasiadamente longe**.** A estratégia de múltiplos saltos é implementada para minimizar a dissipação de energia com comunicação.

Nesta proposta, para determinar que líderes passarão pelo processo de propagação de dados dos CHs mais afastados, o Sistema *Fuzzy* adota critérios de níveis de energia e distância. Primeiramente é utilizado o cálculo (3) para determinar as distâncias de comunicação entre os CHs mais afastados e o restante dos líderes eleitos. Com o resultado do cálculo de distância e os níveis de energia de cada líder, o Sistema *Fuzzy* é carregado. As variáveis linguísticas que representam o nível de Bateria dos líderes e distância de todos os CHs eleitos para os CHs mais afastados são representadas, respectivamente, abaixo, Figuras 5 e 6.

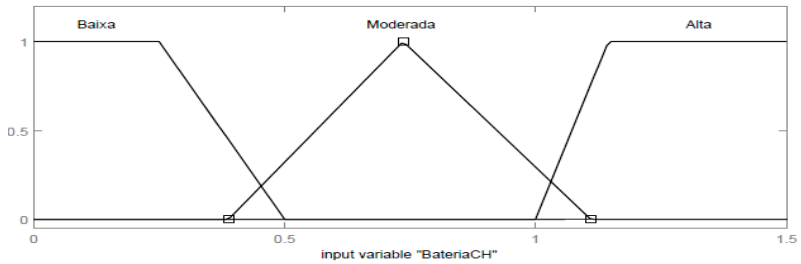


Figura 5. Variável Bateria CH.

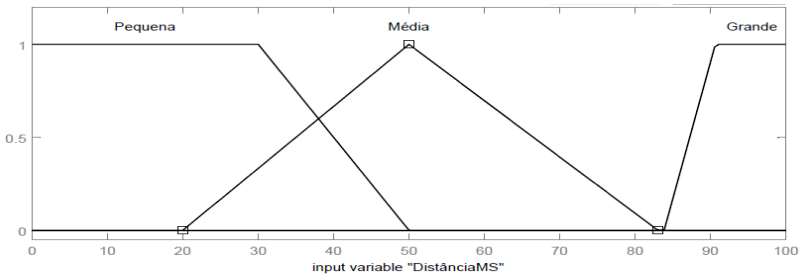


Figura 6. Distância para Múltiplos Saltos.

Ao longo do tempo de simulação é natural que os nós apresentem um declínio de nível de energia. Logo, para evitar que nós líderes com baixo nível de energia passem pelo processo de encaminhamento de dados dos nós mais afastados, o critério energia é utilizado de forma discriminatória a fim de excluir estes nós do processo. Os sensores avançados e super nós tem maior chance de participarem do processo de propagação, uma vez que seus recursos energéticos são aumentados. A variável DistânciaMS é utilizada para selecionar os nós líderes que propagarão os dados, já agregados, dos CHs cuja distância de comunicação para a BS ultrapassa o limiar d_0. A variável possui os valores linguísticos: *Pequena*, *Moderada* e *Grande*, e o universo de discurso varia entre 0 e 100 m. A função de pertinência que representa o conjunto nebuloso *Grande* é definida a partir de 83 m, limiar de comunicação estabelecido com base no quociente dos amplificadores de potência de sinal, $d_0 = \sqrt{E_{fs}/E_{amp}}$.

A base de regras é formada por 9 regras, uma vez que temos dois antecedentes como entrada, 3^2. A melhor condição utilizada na seleção do CH ideal para propagação é dada por: *Se DistanciaMS é Pequena e BateriaCH é Alta Então a Saída é Forte*. Sendo a distância fator agravante, a base de regras exclui os nós que apresentam distância significativa para o CH mais afastado. A Figura 7 mostra o gráfico de superfície do sistema.

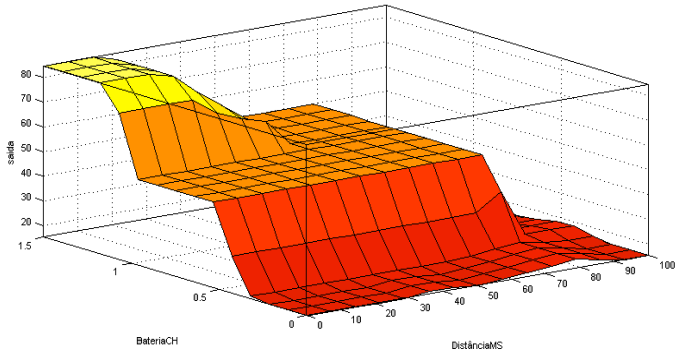


Figura 7. Gráfico de Superfície.

4.5. Modelo de Sistema Fuzzy

Para o modelo de Lógica *Fuzzy*, foram utilizadas funções de pertinência triangulares e trapezoidais, máquina de inferência de *Mamdani* e Defuzzificador Centro Ponderado. No processo de seleção de CH, para cada entrada (x_1, x_2, x_3), a saída do sistema é calculada, como mostra a Eq. (4).

$$y(x_1, x_2, x_3) = \frac{\sum_{l=1}^{27} \mu_{F_l^1}(x_1)\mu_{F_l^2}(x_2)\mu_{F_l^3}(x_3)c_{avg}^l}{\sum_{l=1}^{27} \mu_{F_l^1}(x_1)\mu_{F_l^2}(x_2)\mu_{F_l^3}(x_3)} \quad (4)$$

Para a seleção dos nós líderes que participarão do processo de propagação de dados de CHs mais afastados, a saída é calculada como mostra a Eq. (5). Onde (x_1, x_2) compreendem as entradas *BateriaCH* e *DistânciaMS*.

$$y(x_1, x_2) = \frac{\sum_{l=1}^{9} \mu_{F_l^1}(x_1)\, \mu_{F_l^2}(x_2)\, C_{avg}^l}{\sum_{l=1}^{9} \mu_{F_l^1}(x_1)\, \mu_{F_l^2}(x_2)} \quad (5)$$

4.6. Modelagem da Rede

Para o modelo de rede assumimos que *N* sensores estão distribuídos em uma área N x M. Três tipos de nós sensores, sensores normais, sensores avançados e super sensores, apresentando diferentes níveis de energia inicial, representam a heterogeneidade da rede. O cálculo que determina a quantidade de sensores normais, sensores avançados e super sensores na rede é semelhante ao utilizado por E-DEEC:

$$N.(1 - mf) \quad (6)$$

$$N.mf(1 - mp) \quad (7)$$

$$N.mf.mp \quad (8)$$

Onde mf é a fração do número total de N nós sensores e mp a percentagem para o número total de nós sensores que apresentam *e* mais energia que o nó sensor normal na rede. O cálculo de energia inicial total da rede adotado é o mesmo apresentado em [3].

$$E_{total} = N.(1 - mf).E_0 + N.mf(1 - mp).(2).E_0 + N.mf.mp.E_0(1 + e) = N.E_0(1 + mf(2 + mp.e)) \quad (9)$$

Para a proposta apresentando três níveis de heterogeneidade, a energia total da rede é acrescida, considerando a maior capacidade energética dos sensores avançados e super sensores. Esta diferença é dada pelo fator $1 + mf(2 + mp.e)$.

4.7. Propriedades da Rede

Para o cenário de rede proposto, assumimos algumas propriedades: (i) No que concerne à distribuição de N nós da rede, esta é feita de forma aleatória e não possuem nenhuma mobilidade; (ii) Os nós enviam sua localização para a BS utilizando GPS; (iii) Os nós dissipam energia para o envio de informação; (iv) Todos os nós detêm a mesma capacidade de transmissão e processamento, a heterogeneidade é aplicada a níveis de

energia, já que alguns nós possuem recurso energético aumentado, o que difere dos sensores normais. (v) A BS é fixa e sua localização é pré-definida no algoritmo; (vi) Os nós sempre têm dados para transmitir para o CH.

5. Simulação e Resultados

A simulação é dividida em *rounds*, a cada *round* obtém-se um valor de saída com base nós parâmetros de entrada do sistema e um novo CH é eleito para cada *k cluster*. Os valores são atualizados para entrada do *round* seguinte, onde executa as fases descritas na seção 4. Cada nó é distribuído aleatoriamente em uma área de 100 m^2, onde a classificação do número de nós normais, avançados e super, com seus respectivos níveis de energia, é calculada utilizando as Eq. 5, Eq. 6 e 7, sendo $e = 1$, $mf = 1$ e $mp = 0.6$. O nível de energia inicial para os nós normais é de 0.5J, para os nós avançados e de 1.0J e para super nós 1.5J. A BS é previamente definida com as coordenadas x= 5 e y=95. O modelo de rádio utilizado segue a descrição da Tabela 2.

Tabela 2. Parâmetros do Modelo de Rádio

Parâmetros	*Valores*
E_{elec}	$50\frac{nJ}{bit}$
ε_{fs}	$10\frac{pJ}{\frac{bit}{m^2}}$
E_{amp}	$\frac{0.0013\frac{pJ}{bit}}{m^4}$
E_{DA} (Energia dissipada na agregação de dados)	$5\frac{nJ}{\frac{bit}{sinal}}$
K-bit mensagem (dados)	$4000\ bits$
K-bit mensagem (info)	$100\ bit_s$
d_0 *(Limiar de distância)*	$87m$

Estão distribuídos 100 nós, divididos em *k clusters*. Cada nó envia 4000 Bits de mensagem por *round* para o *cluster head* da rede. A taxa de compressão dos dados é de 5%. A simulação é feita com 5000 rounds. Neste cenário é aplicada a métrica FND (*First Node Dies*) para determinar o período de estabilidade da rede.

Na primeira fase da simulação é obtida a coordenada e nível de energia de cada nó que compõe a rede. A energia dissipada no envio das informações de cada nó é calculada utilizando a Eq. (1). A energia inicial de cada nó é decrementada neste processo. Após o envio das coordenadas, o algoritmo *k-means*, implementado na BS, estipula um padrão com base na coordenada dos nós, calculando o posicionamento de cada nó e dividindo os que formam a rede em *k clusters*. O algoritmo também calcula o centro de cada cluster, informação utilizada posteriormente para o processo de eleição do CH, descrito na subseção 4.2. O número de *clusters* utilizados para simulação é *k=5*.

Cada nó, com seu respectivo valor de centralidade, proximidade para BS e nível de energia, terá um consequente. Um valor com grau de pertinência y , determinando a chance deste nó se tornar *cluster head*. No processo de *defuzzyficação*, o nó que apresentar maior valor de saída *crisp*, será eleito como CH ideal no *round* atual. Após este processo, a fase dois da segunda etapa é iniciada, verificando se algum nó ultrapassa o limiar de comunicação. No caso do limiar ultrapassado, o Sistema Fuzzy determina o nó líder mais adequado para propagar os dados do líder mais afastado, considerando seu nível de energia e distância para o nó que ultrapassa o limiar.

A Figura 8, exibe a divisão dos *clusters,* em uma área de 100 m², e o final da eleição de CHs no *round* 0, sendo representado por '◊' vermelho o *cluster head* eleito pelo sistema e □ representa a BS. Os nós circulados representam os CHs eleitos que ultrapassam o limiar de comunicação com a BS.

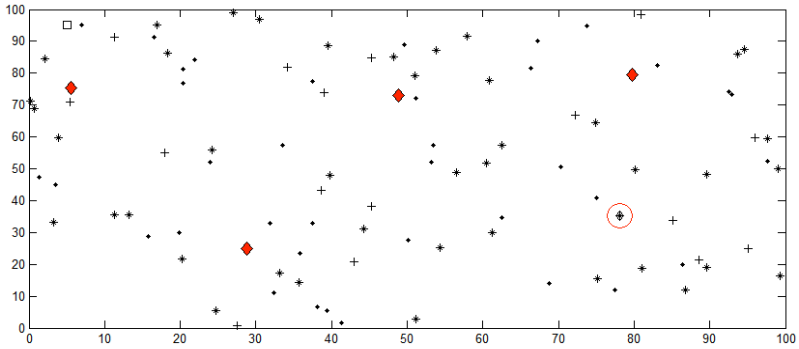


Figura 8. CHs Selecionados pelo Sistema *Fuzzy* no Round 0.

A Tabela 3 exibe o resultado de seleção de líderes para o *round* 0. Os valores de energia dos nós, exibidos na Tabela 3 já apresentam a dissipação de energia gerada na transmissão de informações para o ponto de coleta e dissipação na comunicação gerada no processo de associação ao CH eleito. O CH 5 apresentou maior dissipação de energia devido a quantidade de nós associados.

Tabela 3. Resultados da Seleção de *Cluster Heads*

CH	*Nível de energia*	*Centralidade No Cluster*	*Número de nós p/ cluster*	*Distância para BS*
1	1.4965	7.7574	21	19,6 m < d_0
2	1.4965	2.4817	21	49,0 m < d_0
3	1.4972	12.2433	17	76,3 m < d_0
4	1.4975	12.1966	15	94,5 m > d_0
5	1.4957	4.6183	26	73,9 m < d_0

O CH 4, selecionado pelo Sistema *Fuzzy*, ultrapassa o limiar de comunicação. Em uma transmissão direta para o ponto de coleta, este nó, precisaria utilizar o modelo *Multipath,* gerando maior dissipação de energia no processo de comunicação, uma vez que o expoente de *path loss* seria d^4. O consumo de energia gerado neste tipo de transmissão seria de aproximadamente 2.3935 J, levando o nó à inatividade de forma prematura e quebrando o período de estabilidade da rede. Entretanto, com a estratégia de múltiplos saltos adotada, o CH comunica-se com líder mais adequado, mantendo o

modelo *Free space* de rádio. A Tabela 4 exibe a saída do Sistema *Fuzzy*, no *round* 0, para seleção do nó líder adequado para propagação dos dados do CH mais afastado. O maior valor de saída *crisp* corresponde ao nó líder selecionado, CH 3.

Tabela 4. Saída do Sistema *Fuzzy*

CH	Nível de energia	Distância para o CH > d_0	Saída *Fuzzy*
1	1.4965	83.1566	18.2613
2	1.4965	47.8956	52.9779
3	1.4972	44.4752	57.3361
5	1.4957	50.3652	50.0000

A proposta desde artigo foi comparada com os algoritmos LEACH e E-DEEC. Para avaliação de desempenho é utilizado o final do período de estabilidade da rede e o tempo de vida útil. A escolha dos algoritmos para comparação se dá principalmente pela utilização de informações locais sem considerar critérios de posicionamento, para eleição dos CHs. Além do método de escolha do líder, o algoritmo LEACH, não trata as discrepâncias de energia dos nós que compõem a rede.

Diferente de LEACH, o algoritmo E-DEEC, considera a heterogeneidade dos nós para eleição do CH. Entretanto, utiliza informações locais para eleição do líder e insere três níveis de heterogeneidade, semelhante a proposta apresentada neste artigo.

A Figura 9 exibe a quantidade de nós sensores ativos no tempo de vida útil da rede. Esta medida reflete o número total de nós que ainda não esgotaram sua energia.

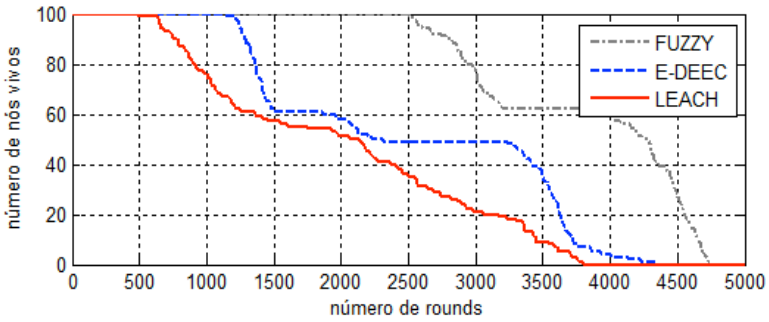


Figura 9. Numero de Sensores Ativos no Período de Simulação.

Os resultados indicam claramente que a inserção de novos níveis de heterogeneidade, a abordagem da lógica *fuzzy* como ferramenta de seleção, a utilização de múltiplos saltos entre os CHs e informações centralizadas na BS, permitem eleger líderes mais eficientes, aumentando o período de estabilidade e o tempo de vida da rede.

A proposta apresenta melhores resultados, quando comparado com os algoritmos LEACH e E-DEEC. O algoritmo LEACH apresentou o menor período de estabilidade, ocorrendo por volta de 500 *rounds*. O algoritmo E-DEEC apresentou melhor desempenho sobre o LEACH, com o período de instabilidade da rede iniciado por volta 1148 *rounds*. Como observado na Figura 10, a proposta mostra um melhor desempenho

sobre os algoritmos comparados, aumentando o período de estabilidade da rede até aproximadamente 2500 *rounds*, quando ocorre a primeira inatividade de um nó por falta de energia.

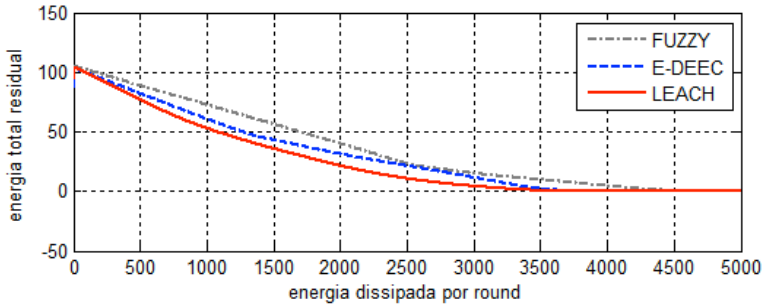


Figura 10. Energia Total Residual de LEACH, E-DEEC e FUZZY.

A Figura 10 exibe a energia dissipada pela rede ao longo dos 5000 rounds de simulação para cada algoritmo comparado. O total de energia para cada rede é de 104.5J. A dissipação de energia apresenta um declínio linear ao longo de 2500 *rounds*, para a proposta apresentada e para o algoritmo E-DEEC por volta de 1500 *rounds* , mudando a partir do momento em que o primeiro nó na rede fica inativo, quebrando o período de estabilidade. Ambos os algoritmos, *Fuzzy* e E-DEEC, permitem tratar as discrepâncias energéticas de cada nó na rede para eleição do CH, enquanto o algoritmo LEACH apresenta maior dissipação de energia por *round*, consequência dos problemas descritos anteriormente sobre a forma de eleição de CH pelo algoritmo e a falta de tratamento discriminatório das discrepâncias energéticas dos nós que formam a rede.

6. Conclusões

Os resultados indicam que a proposta apresentada oferece grandes vantagens, permitindo selecionar os nós mais adequados para líderes do grupo a cada *round* com base nos valores de defuzificação do Sistema *Fuzzy*, como também, a utilização da Lógica *Fuzzy* como ferramenta de decisão para implementação de múltiplos saltos entre CHs, uma vez que minimiza a dissipação de energia dos CHs selecionados mais afastados do ponto de coleta. A inserção de três níveis de heterogeneidade, correspondente aos sensores normais, avançados e super sensores, contribui consideravelmente para o aumento do período de estabilidade da rede, uma vez que esta inserção dá a rede maior recurso energético. Entretanto, os resultados indicam que se esta discrepância não for considerada no momento de seleção do CH ela não influencia de forma considerável no aumento do período estável.

Outra grande vantagem que contribui para os resultados obtidos neste trabalho é a utilização de um controle central na BS. Por não possuir severas limitações de energia, processamento e armazenamento como os nós que formam a rede, a BS apresenta vantagens sobre o processamento local de informações em cada nó, processo este encontrado nos algoritmos tradicionais para eleição de CHs, o envio de atualizações das informações de nível de energia dissipada dos nós em cada *round*. Entretanto, mesmo com esta atualização a proposta ainda apresenta melhorias sobre os outros modelos apresentados. Outra vantagem do controle central na BS está no momento de seleção do CH, por ter o papel de informar a rede sobre os líderes selecionados para cada *cluster*, previamente divididos pelo algoritmo *k-means*.

Este processo difere-se do processo encontrado nos algoritmos que utilizam informações locais para seleção de seu líder, cabendo ao próprio CH eleito enviar mensagens em *broadcast* para a rede, gerando dissipação de energia no momento da propagação. Finalmente, o trabalho apresentado tem a principal contribuição na eleição do CH mais eficiente, considerando sua localização e discrepâncias de níveis de energia, como também, na inclusão de novos níveis de heterogeneidade, permitindo aumentar o período de estabilidade da rede, ou seja, o período que a rede é totalmente funcional, aumentando consideravelmente o tempo de vida útil em RSSF heterogêneas.

Referências

[1] Pottie, G. J., Kaiser, W. J. (2000) "Wireless integrated network sensors (WINS)". Communications of the ACM. 43, 5, 51-58.

[2] Akyildiz, I.F.; Weilian Su; Sankarasubramaniam, Y.; Cayirci, E. (2002) "A survey on sensor networks," *Communications Magazine, IEEE* , vol.40, no.8, pp. 102- 114.

[3] Saini, P., Sharma, A. K., (2010) "E-DEEC- Enhanced Distributed Energy Efficient Clustering scheme for heterogeneous WSN," Parallel Distributed and Grid Computing (PDGC), 1st International Conference on , vol., no., pp.205-210, 28-30.

[4] Quing, L., Zhu, Q., Wang, M. (2006) "Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks". ELSEVIER, Computer Communications 29, pp 2230- 2237.

[5] Mubarak, T.M., Sattar, S. A., Rao, G. A., Sajitha, M. (2011)"Intrusion detection: An energy efficient approach in heterogeneous WSN," Emerging Trends in Electrical and Computer Technology (ICETECT), International Conference on vol., no., pp.1092-1096, 23-24.

[6] Han, L. (2010) "LEACH-HPR: An energy efficient routing algorithm for Heterogeneous WSN," Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on , vol.2, no., pp. 507-511, 29-31.

[7] Smaragdakis, G., I. M., Bestavros A. (2004) "SEP: A Stable Election Protocol for clustered heterogeneous wireless sensor networks", in: Second International Workshop on Sensor and Actor Network Protocols and Applications (SANPA).

[8] Mubarak, T.M., Sattar, S. A., Rao, G. A., Sajitha, M. (2011) "Intrusion detection: An energy efficient approach in heterogeneous WSN," Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on , vol., no., pp.1092-1096, 23-24.

[9] Quing, L., Zhu, Q., Wang, M., (2006) "Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks". ELSEVIER, Computer Communications 29, pp 2230- 2237.

[10] Yan, B., Wu, X., Zhou, X. (2010) "A Improved Base Station Cooperative Mobile Strategy for WSN with Finite Powered Cluster Heads" Wireless Communications Networking and Mobile Computing (WiCOM), 6th International Conference, vol., no., p.1-4, 23-25.

[11]Zadeh, L. A. (1965)"Fuzzy Sets. Information and Control", vol. 8, pp 338- 353.

[12]Saini, P., Sharma, A. K. (2010) "E-DEEC- Enhanced Distributed Energy Efficient Clustering scheme for heterogeneous WSN," Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on , vol., no., pp.205-210, 28-30.

CodeDrip: Protocolo de Disseminação de Dados em Redes de Sensores Sem Fio Utilizando Codificação na Rede

Nildo dos Santos Ribeiro Júnior[1], **Luiz F. M. Vieira**[1], **Marcos A. M. Vieira**[1]

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

{nildo,lfvieira,mmvieira}@dcc.ufmg.br

Abstract. *In this paper, we present CodeDrip, a data dissemination protocol for Wireless Sensor Networks that utilizes Network Coding to improve performance. The dissemination goal is to distribute data from the sink node to all nodes in the network. The Network Coding technique consists of combining packets before transmitting them. The dissemination process becomes more robust to packet loss with Network Coding since the lost packets can be recovered by combining the received packets. We simulate CodeDrip in many wireless scenarios, varying the network density and link quality. We compare CodeDrip with the state-of-the-art Drip. Results show that CodeDrip is faster than Drip to disseminate information and transmits fewer packets.*

Resumo. *Este artigo apresenta o protocolo de disseminação de dados CodeDrip para Redes de Sensores Sem Fio utilizando a técnica de Codificação em Rede a fim de obter um melhor desempenho nesse processo. O objetivo da disseminação é fazer com que dados do nó sorvedouro cheguem a todos os nós sensores da rede. Tendo vários pacotes a serem distribuídos, a técnica de Codificação em Rede consiste em fazer com que alguns pacotes sejam combinados antes de serem enviados. O processo de disseminação se torna mais robusto a perda de pacotes com a técnica de Codificação em Rede pois aqueles pacotes que foram perdidos poderão ser recuperados a partir da combinação de pacotes recebidos. Simulamos o CodeDrip em redes com várias densidades e vários níveis de qualidade do canal de enlace. Comparamos o CodeDrip com o protocolo estado da arte Drip. Os resultados mostram que a disseminação do CodeDrip é mais rápida e transmite menos pacotes do que o Drip.*

1. Introdução

Redes de Sensores Sem Fio (RSSFs) são redes com grande número de microsensores compactos com capacidade de comunicação sem fio, chamados de nós sensores [Ruiz et al. 2004]. O objetivo destas redes é sensoriar o meio e coletar dados. Em várias situações existe a necessidade de se monitorar condições ambientais, tanto em locais internos, como um prédio ou uma fábrica, quanto em locais externos, como uma floresta ou nas ruas da cidade. Instalar fios pelo ambiente é inviável em várias situações. A melhor alternativa é utilizar comunicação sem fio. Essa estrutura formada por nós sensores que se comunicam entre si é uma Rede de Sensores Sem Fio.

Em RSSFs, é importante notificar cada nó sensor. Isso permite que administradores de rede possam reconfigurar, criar consultas, comando e tarefas e reprogramar a

rede. Para isso, é necessário fazer com que várias mensagens cheguem a todos os nós sensores da rede. Esse processo é chamado de disseminação (figura 1). Um protocolo de disseminação de dados eficiente precisa transpor algumas dificuldades impostas pela natureza de uma rede de sensores sem fio. A primeira é que a energia disponível em um nó sensor é limitada por uma bateria, então é preciso economizá-la para aumentar o tempo de atividade dos sensores. A segunda é a limitação de capacidade computacional nos nós sensores, que podem não ser capazes de rodar protocolos de rede sofisticados. Por fim, a comunicação é suscetível à erros de transmissão e perdas de pacotes. Para um protocolo de disseminação de dados em uma RSSF, além de consumir pouca energia dos nós sensores, é desejável que a disseminação aconteça rapidamente.

Nesse artigo, apresentamos um protocolo de disseminação de dados em redes de sensores sem fio que utiliza a técnica de Codificação em Rede para melhorar a sua eficiência. Em vez de simplesmente retransmitir os pacotes de dados que recebem, os nós sensores combinam vários pacotes em um só, e transmitem essa combinação para os seus vizinhos. Com isso, a perda de pacotes de dados é atenuada, uma vez que pacotes perdidos poderão ser obtido através da decodificação da combinação de outros pacotes recebidos. Evitando retransmissões, a disseminação dos dados ocorre em menos tempo.

Protocolos de disseminação de dados para redes de sensores sem fio existentes utilizam diferentes técnicas para diminuir o número de mensagens enviadas por período de tempo, diminuindo o consumo de energia do nó sensor. Essas técnicas comprometem o desempenho do protocolo na questão do tempo de disseminação, podendo ocorrer casos em que ela demora muito mais que o esperado por causa de perdas de pacotes durante o processo. O protocolo de disseminação apresentado nesse artigo, ao utilizar a técnica de Codificação em Rede provê mais eficiência ao processo em termos de número de pacotes enviados e tempo de disseminação, pois a perda de pacotes afeta menos a transmissão dos dados.

A próxima seção desse artigo apresenta os trabalhos relacionados e as ferramentas que foram utilizadas para o desenvolvimento desse trabalho. Na seção 3 falamos sobre o princípio de funcionamento e vantagens do método de Codificação em Rede. O algoritmo do protocolo desenvolvido é explicado na seção 4, o processo de simulação de redes de sensores sem fio é descrito na seção 5 e os resultados obtidos destas simulações são mostrados na seção 6. Finalmente na seção 7 são apresentadas as conclusões do trabalho.

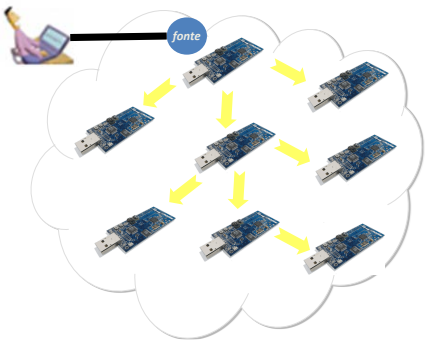


Figura 1. Usuários podem enviar mensagens para todos os nós da rede através do nó fonte utilizando um protocolo de disseminação.

2. Trabalhos Relacionados

Trickle [Levis et al. 2004] é um algoritmo para propagação e manutenção de atualização de código em Redes de Sensores Sem Fio. Nele os nós sensores transmitem periodicamente uma mensagem de aviso para seus vizinhos informando a versão do seu código, mas evitam transmitir se eles receberam recentemente um aviso igual ao seu.

O TrickleTimer é o temporizador utilizado no algoritmo Trickle. Esse temporizador já está implementado como um componente do TinyOS e é utilizado tanto pelo protocolo Drip quanto pelo CodeDrip. Ele incrementa o tempo de espera a cada vez que é disparado. Sendo assim, quanto maior é o tempo decorrido, menor será a frequência que o temporizador disparará um trecho de código a ser executado. A interface TrickleTimer do TinyOS oferece uma função que faz com que o intervalo de espera volte a ser pequeno. Essa função é utilizada quando alguma mensagem nova chega no nó sensor, fazendo com que essa mensagem nova seja retransmitida com mais frequência nesse início, quando a probabilidade de seus vizinhos ainda não a terem recebido é maior.

O Drip [Tolle and Culler 2005] é um protocolo de disseminação de dados para Redes de Sensores Sem Fio implementado como um componente do TinyOS 2.1. Ele provê uma interface de camada de transporte para várias camadas de disseminação de mensagens confiável. O Drip utiliza TrickleTimers para fazer transmissões periódicas do dado que está sendo disseminado para garantir que esse dado eventualmente chegará a todos os nós sensores da rede. Ele é um protocolo estado-da-arte para disseminação de pequenos valores.

O componente implementado no TinyOS para disseminação é o DisseminatorC. Ele implementa o protocolo Drip para fazer a disseminação de um dado na rede. Para cada valor que se queira disseminar na rede, um componente DisseminatorC é instanciado em cada nó sensor. Esse componente é responsável por criar o temporizador que ele precisa, enviar e receber as mensagens para atualizar o dado que o nó sensor possui. Duas interfaces providas por esse componente permitem o nó sensor acessar a mensagem sendo disseminada: DisseminationValue e DisseminationUpdate. A interface DisseminationValue provê um evento que é disparado quando uma nova mensagem chega e o dado que ela contém é novo para esse nó sensor. A interface DisseminationUpdate possui uma função que permite o nó sensor atualizar o seu dado e, assim, começar a disseminação desse dado na rede.

Deluge [Hui and Culler 2004] é outro protocolo de disseminação de dados para RSSFs também implementado no TinyOS. Seu foco é a disseminação de um grande objeto de dados, isto é, que não cabe na memória RAM, de um ou mais nós fonte para o resto da rede. Para isso, esse protocolo representa o objeto de dados como um conjunto de páginas de tamanho fixo, o que fornece uma unidade gerenciável de transferência que permite a multiplexação espacial e suporta atualizações incrementais eficientes. O Deluge também utiliza TrickleTimers para fazer o controle de quando enviar pacotes pela rede.

Como esse protocolo é específico para grandes objetos de dados, as características dele são muito diferentes das do Drip e do CodeDrip, que focam na disseminação de dados pequenos. Por terem objetivos diferentes, o Deluge é um trabalho complementar ao CodeDrip na manutenção de RSSFs.

O trabalho de Ahlswede et al. [Ahlswede et al. 2000] motivou a pesquisa sobre

Codificação em Rede e mostrou que, em geral, a codificação de pacotes poderia resultar em um nível ótimo de aproveitamento da capacidade de transmissão da rede que não poderia ser atingida por outro possível esquema de apenas encaminhamento. Por exemplo, em tráfego multicast, a capacidade de transmissão é definida como a taxa máxima de transmissão em que um nó fonte pode enviar pacotes para todos os membros de um conjunto de receptores. Ela é dada pelo mínimo dos fluxos máximos (s, t) entre o nó fonte s e cada receptor t. Foi mostrado que com Codificação em Rede pode-se atingir uma capacidade de transmissão muito maior que um esquema de somente encaminhamento. Além disso, Li, Yeung and Cai [Li et al. 2003] mostraram que é suficiente que a função de codificação seja linear.

Foi demonstrado que, em ambientes sem fio, a codificação em rede oferece vários benefícios, como diminuição do consumo de energia por meio da redução do número de transmissões, aumento da taxa de transmissão de dados e maior robustez, por causa da possibilidade de um nó receber múltiplas cópias de um único pacote [Deb 2005].

Katti et al. [Katti et al. 2008] demonstraram que o uso de codificação em rede pode melhorar a taxa de transmissão global da rede. Desde então, protocolos têm sido desenvolvidos para cenários específicos. O CodeTorrent [Lee et al. 2006] faz distribuição de conteúdo em VANETs usando codificação em rede.

T. Cui et al. [Cui et al. 2010] consideraram fontes múltiplas com utilitários e codificação em rede em sessões e forneceram um algoritmo para controle de taxa de junção, codificação em rede e agendamento.

Zhang et al. [Zhang and Zhang 2009] fizeram uma formulação para calcular a taxa de transmissão máxima de tráfego unicast que pode ser alcançada com codificação em rede cooperativa. Keller et al. [Keller et al. 2011] investigam experimentalmente o atraso de protocolos baseados na técnica de inundação (flooding) para uma aplicação em detecção de atiradores usando codificação em rede. O DutyCode [Chandanala and Stoleru 2010] combina a ideia de codificação em rede e ciclo de trabalho (duty-cycle) na camada MAC, porém eles não discutem disseminação de dados para redes de sensores sem fio.

A disseminação de dados em RSSFs com codificação em rede também é apresentada em [Wang et al. 2010] porém o foco é bem diferente do nosso trabalho. Eles supõem a camada MAC composta por um protocolo TDMA e focam em um escalonamento de pacotes, determinando quais pacotes a serem combinados e transmitidos dado um cronograma de funcionamento dos nós sensores. Eles provaram que o problema é NP-Difícil e forneceram uma formulação de programação linear. Eles não proveram um protocolo para disseminação e não mencionaram, por exemplo, a questão do uso de temporizadores.

2.1. Ferramentas Utilizadas

Nessa seção apresentamos duas ferramentas utilizadas para o desenvolvimento do nosso trabalho: o TinyOS, um sistema operacional para sensores sem fio e o TOSSIM, um simulador de RSSFs.

O TinyOS é um sistema operacional leve projetado para sensores sem fio de baixa potência. Ele se diferencia da maioria dos outros sistemas operacionais porque o seu projeto foca em operações com baixo consumo de energia. [Levis and Gay 2009]. Ele fa-

cilita o desenvolvimento de aplicações para Redes de Sensores Sem Fio provendo serviços e abstrações como sensoriamento, comunicação, armazenamento e termporizadores. Esse sistema é licenciado sobre a BSD, uma licença de código aberto.

O TinyOS provê um conjunto de componentes de sistema reutilizáveis que podem ser conectados à aplicação. Assim, cada aplicação pode escolher os componentes que necessitar, evitando que serviços não usados sejam compilados e passados para o dispositivo. Então, ao invés de ser um sistema operacional tradicional, o TinyOS permite que um novo sistema específico seja construído durante a compilação, contendo apenas os componentes que interessam à aplicação. Como os recursos computacionais disponíveis em um nó sensor são poucos, esse sitema se torna bastante interessante.

O TinyOS é desenvolvido em nesC, uma linguagem de programação para sistemas de rede incorporados, como nós sensores. [Gay et al. 2003]. Ela é uma linguagem orientada a componentes com um modelo de execução baseado em eventos. O nesC foi desenvolvido tendo em vista superar os desafios que existem na construção de aplicações para Redes de Sensores Sem Fio, sendo que o principal desses desafios é a disponibilidade de recursos computacionais muito limitados, como armazenamento, processamento e energia.

O TOSSIM é um simulador de Redes de Sensores Sem Fio para aplicações desenvolvidas com o TinyOS. Ele consegue simular o comportamento e as interações de milhares de nós sensores em uma rede. [Levis et al. 2003] A arquitetura do TOSSIM é composta pelos seguintes elementos: suporte para compilar os componentes do TinyOS para a infraestrutura de simulação, uma fila de eventos discreta, abstrações de componentes de hardware do TinyOS, mecanismos para modelos de rádio e de conversores analógico/digital, e serviços de comunicação para programas externos interagirem com o simulador.

O TOSSIM permite simular o mesmo código que é executado no nó sensor e usa uma abstração simples porém poderosa para uma rede de sensores sem fio. A rede é representada por um grafo direcionado, onde cada vértice é um nó sensor e cada aresta representa uma possível comunicação com uma probabilidade de erro. Essa abstração permite que sejam feitos testes tanto com perfeita condição de transmissão, quanto com muitas perdas na rede.

3. Codificação em Rede

Codificação em Rede (do inglês *Network Coding* [Ahlswede et al. 2000]) é uma técnica que permite combinar pacotes codificados na rede e provê ganhos para a rede aumentando a vazão desta, diminuindo o consumo de energia, reduzindo o número de mensagens que são transmitidas para disseminar informação, entre outros [Vieira et al. 2007]. O ganho da Codificação em Rede vem da combinação da informação dos pacotes na rede, possibilitando uma disseminação de informação na rede mais eficiente.

Nas transmissões em redes, pacotes podem ser perdidos. A perda de pacotes pacotes exige a retransmissão destes pacotes. Combinando os pacotes de forma inteligente é possível recuperar as informações transmitidas sem ter que retransmitir todos os pacotes perdidos para todos os nós da rede. Utilizaremos um exemplo simples e didático para ilustrar e explicar o funcionamento da Codificação em Rede.

Considere a topologia da figura 2. O nó fonte, no centro da figura, deseja disseminar dois pacotes, chamados de P_1 e P_2. Os nós 1 e 2 estão no raio de alcance do nó fonte e possivelmente podem receber os pacotes. O pacote P_1 é transmitido pelo nó fonte e é recebido apenas pelo nó 1. A seguir o pacote P_2 é transmitido pelo nó fonte e, desta vez, recebido apenas pelo nó 2. Cada nó receptor tem um pacote diferente.

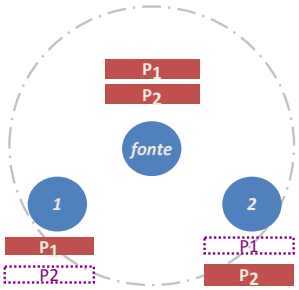


Figura 2. Nó fonte quer disseminar dois pacotes para os dois nós da rede.

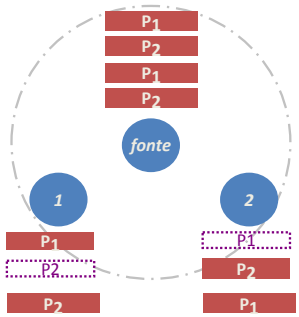


Figura 3. Retransmissão no roteamento tradicional. No total 4 mensagens são transmitidas.

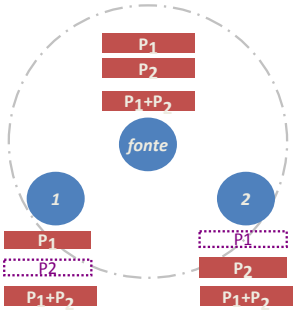


Figura 4. Retransmissão utilizando codificação em Rede. No total 3 mensagens são transmitidas.

No roteamento tradicional, os pacotes perdidos são retransmitidos. No nosso exemplo, o nó fonte terá que retransmitir os pacotes P_1 e P_2 e, se tiver sucesso, o nó 1 irá receber o pacotes P_2 que estava faltando e, de forma similar, o nó 2 irá receber o P_1. No total, considerando que haja sucesso na primeira retransmissão de cada pacotes, serão gastos no total 4 transmissões de pacotes na rede, como mostra a figura 3.

A Codificação em Rede permite que pacotes sejam combinados usando um operador lógico. Podemos combinar os pacotes utilizando o operador XOR (ou exclusivo). Um novo pacotes será criado onde cada bit do novo pacote é o resultado do ou exclusivo de cada bit entre P_1 e P_2. Observe que o tamanho do novo pacote é o mesmo de P_1 e P_2.

O nó fonte, ao invés de retransmitir os pacotes P_1 e P_2, pode retransmitir um pacote que é $P_1 XOR P_2$. O nó 1, ao receber esse pacote é capaz de aprender P_2 aplicando o operador ou exclusivo entre o pacote recebido e P_1, que ele havia recebido anteriormente visto que $P_2 = P_1 XOR (P_1 XOR P_2)$. Da mesma maneira, o nó 2 é capaz de receber P_1 quando receber o novo pacote e aplicar o operador ou exclusivo com o pacote P_2 que ele tinha. $P_1 = P_2 XOR (P_1 XOR P_2)$. Ao final, serão gastos no total 3 transmissões de pacotes na rede, como mostra a figura 4.

O exemplo ilustra bem o ganho de se utilizar Codificação em Rede para uma topologia com um salto. O número de mensagens na rede foi reduzido de 4 para 3. Para uma topologia maior e que são utilizados vários saltos, se faz necessário o desenvolvimento de um protocolo que consiga manter os ganhos introduzidos pela combinação de pacotes na rede.

4. Algoritmo

O CodeDrip surgiu da ideia de aplicar a técnica de Codificação em Rede no processo de disseminação de dados em uma Rede de Sensores Sem Fio e verificar os ganhos que ela traz. Em muitas situações é necessária a disseminação de vários dados na rede, e não apenas um. Nesse caso, essa técnica se torna interessante, pois existirão vários pacotes sendo transmitidos na rede para disseminar diferentes valores, e a combinação de alguns deles tornará o processo mais robusto a perda, uma vez que pacotes perdidos poderão ser recuperados se o nó sensor tiver recebido uma mensagem combinada e uma original.

Assim como o Drip, o CodeDrip utiliza TrickleTimers para transmitir periodicamente mensagens contendo os dados que estão sendo disseminados com o objetivo de que eles cheguem eventualmente a todos os nós sensores da rede. Porém, além das mensagens originais, algumas mensagens combinadas também serão transmitidas. Como o CodeDrip é agnóstico à topologia da rede, não existe custo adicional para adquirir e armazenar informações sobre ela. Por isso a política utilizada para decidir quando transmitir uma mensagem combinada é muito importante e afetará diretamente o desempenho do processo de disseminação.

As mensagens que serão enviadas e recebidas pelos nós sensores tiveram que ser adaptadas para que a técnica de Codificação em Rede pudesse ser aplicada. É necessário indicar no cabeçalho da mensagem qual é a combinação que ela contém para permitir o processo de decodificação. Cada dado a ser disseminado na rede possui um número inteiro de 1 byte como identificador único. Cada mensagem possui um conjunto de identificadores e o conteúdo a ser transmitido. O conjunto de indicadores é o custo adicional necessário na mensagem para que a codificação e decodificação possam funcionar. Uma mensagem original possui apenas um identificador diferente de zero, e seu conteúdo é o dado original a ser disseminado. Uma mensagem codificada possui dois ou mais identificadores diferentes de zero, e seu conteúdo é o resultado da operação XOR entre os dados cujos identificadores estão na lista dessa mensagem. Um número máximo de mensagens a serem combinadas precisa ser definido. Para os nossos testes ele foi definido como duas mensagens.

Cada nó sensor possui um conjunto de mensagens originais e um buffer de mensagens combinadas inicialmente vazios. Ao receber uma mensagem, o nó sensor irá verificar se ela é original ou combinada. Se for uma mensagem original, ele a armazena no seu vetor de mensagens originais e a partir desse momento ele enviará essa mensagem a cada vez que o TrickleTimer correspondente disparar. Se for uma mensagem combinada, o nó sensor verificará se ele consegue decodificar uma nova mensagem a partir das originais que ele já possui. Se isso não for possível, a mensagem será armazenada no buffer de mensagens combinadas até que cheguem mensagens que tornem possível a sua decodificação.

O protocolo utiliza o evento disparado pelos TrickleTimers para fazer o broadcast de uma mensagem. Cada mensagem original armazenada no nó sensor tem um temporizador. Quando este é disparado, o nó sensor enviará uma mensagem contendo o dado correspondente. Porém algumas vezes, em vez de simplesmente enviar esse dado sozinho, o nó sensor o combinará com alguma outra mensagem e enviará essa combinação. A probabilidade de enviar uma mensagem codificada em vez de uma mensagem original irá afetar o desempenho do protocolo.

Tanto no Drip quanto no CodeDrip muitas mensagens redundantes serão recebidas pelos nós sensores. Uma mensagem redundante é aquela que o nó sensor já possui. Um nó da rede receber várias vezes a mesma mensagem é um sinal de que a maioria de seus vizinhos já possuem essa mensagem. Então quanto mais redundância houver no recebimento de determinado dado, menor é a necessidade de enviá-lo em um tempo próximo. Para atrasar o envio desse dado, e consequentemente diminuir o número de mensagens enviadas em um determinado período de tempo, esses protocolos utilizam uma função provida pela interface do TrickleTimer que aumenta o intervalo de espera do temporizador, adiando o envio daquela mensagem. Com a possibilidade de receber mais de uma mensagem com apenas uma transmissão, o CodeDrip suprime o envio de mensagens desnecessárias mais rápido do que o Drip. Por exemplo: ele pode receber uma combinação de duas mensagens redundantes e adiar o temporizador dessas duas mensagens de uma só vez, enquanto o Drip precisaria de receber duas mensagens distintas para fazer a mesma coisa.

5. Simulação

Um conjunto de topologias de redes de sensores sem fio foi gerado para simular a disseminação de dados utilizando o protocolo Drip e o CodeDrip utilizando o TOSSIM. Uma topologia de rede é um grafo direcionado valorado, onde um vértice representa um nó sensor e uma aresta diz que existe comunicação entre os sensores que ela liga. O peso de cada aresta é a probabilidade da transmissão ser feita com sucesso.

O algoritmo que implementamos para gerar uma topologia recebe dois parâmetros: um número n que representa a quantidade de nós sensores na topologia gerada e um número d que representa a distância média de cada nó aos seus vizinhos. O algoritmo sorteia coordenadas x e y para o primeiro nó da rede, e insere o segundo nó a uma distância d do primeiro. Para inserir o restante dos nós da topologia, são sorteadas novas coordenadas x e y para cada um, e essas coordenadas têm que satisfazer duas condições: 1) a distância do novo nó até todos os que já estão na topologia tem que ser maior que $d - 5$; 2) a distância do novo nó até pelo menos dois dos nós que já estão na topologia tem que ser menor que $d + 5$. Se uma das duas condições não for satisfeita, outras coordenadas são sorteadas. Caso contrário, o novo nó é adicionado. A saída do algoritmo será um arquivo de configuração de uma topologia para o TOSSIM. Nela, distância de cada nó a cada um de seus nós vizinhos está no intervalo $[d - 5, d + 5]$. Esse algoritmo garante que a topologia será conexa e que cada nó terá pelo menos dois vizinhos com os quais ele consegue se comunicar. O objetivo disso é que todos os nós da rede possam receber as mensagens que estão sendo disseminadas.

Nós geramos topologias que têm de 10 a 100 nós sensores, variando esse número de dez em dez. Nas simulações que fizemos, consideramos a perda de pacotes sendo uniforme em toda a rede. Isso significa que todo pacote enviado tem a mesma probabilidade de se perder. Fizemos simulações variando o valor dessa probabilidade de 10 a 70 por cento, variando essa probabilidade de dez em dez.

Em cada simulação, considera-se o processo de disseminação iniciado a partir do momento que primeiro nó sensor recebe o primeiro dado a ser disseminado. Conta-se a partir de então toda mensagem que for transmitida na rede, e considera-se esse o tempo de início. A simulação acaba quando todos os nós tiverem recebido todas as mensagens

sendo disseminadas, sendo o momento em que o último nó sensor consegue a última mensagem que o faltava definido como o tempo final. Então são registrados qual foi esse intervalo de tempo e qual o número total de mensagens que foram transmitidas. Em cada simulação três dados diferentes foram disseminados para a rede.

6. Resultados

Utilizando a técnica de Codificação na Rede, o resultado esperado é que processo de disseminação precise enviar menos mensagens para que todos os nós da rede recebam todos os três dados que estão sendo disseminados, e que esse ganho seja maior o quanto pior for a qualidade de transmissão na rede.

Os gráficos apresentados na figura 5 mostram o desempenho dos dois protocolos (Drip e CodeDrip) à medida que a perda de pacotes na rede aumenta em uma topologia com 100 nós sensores. Nos gráficos, cada ponto representa a média dos valores obtidos em 10 simulações, e a barra de erro representa o desvio padrão desse conjunto de dados. Nesses gráficos é possível observar que o CodeDrip é menos afetado pela perda de pacotes durante a disseminação e precisa de transmitir menos mensagens do que o Drip para completar o processo. O tempo gasto para que os dados cheguem a todos os nós sensores também foi, em média, menor no CodeDrip.

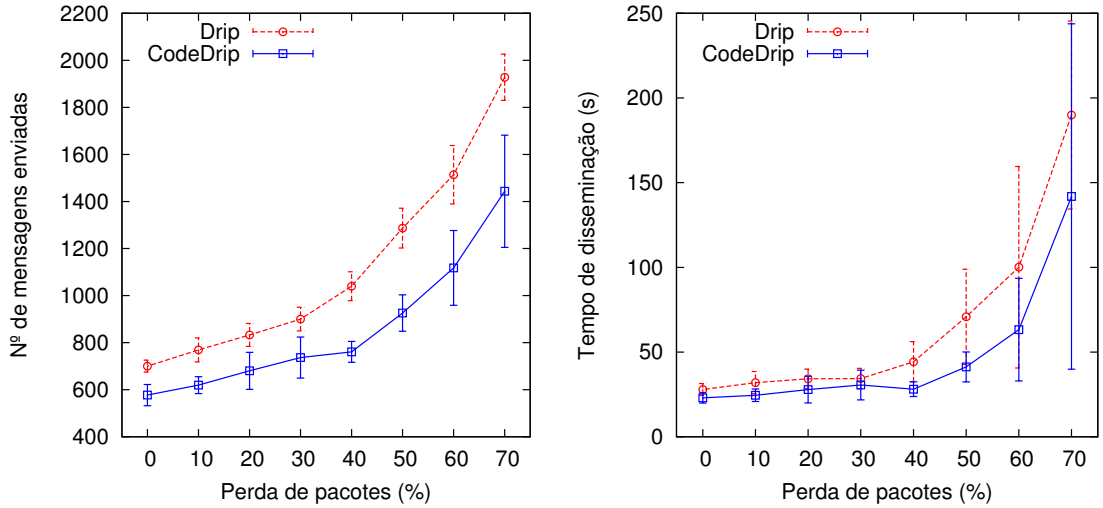


Figura 5. Número de mensagens enviadas e tempo de disseminação para o Drip e o CodeDrip à medida que a perda de pacotes aumenta em uma topologia com 100 nós sensores.

Os gráficos da figura 6 mostram o desempenho dos protocolos quando mantemos fixa uma perda média de 50% dos pacotes transmitidos e aumentamos o número de nós sensores na rede. A adição de nós na rede faz com que o desempenho do CodeDrip em comparação ao Drip melhore cada vez mais. Logo, quanto mais nós a rede tiver, maior será a vantagem em se utilizar o CodeDrip ao invés do Drip para fazer a disseminação de dados.

O gráfico da figura 7 mostra a porcentagem de nós sensores da rede que já receberam todas as mensagens em função do tempo de disseminação em uma rede. Observe que a perda de pacotes causa uma dificuldade para o protocolo Drip logo quando o processo

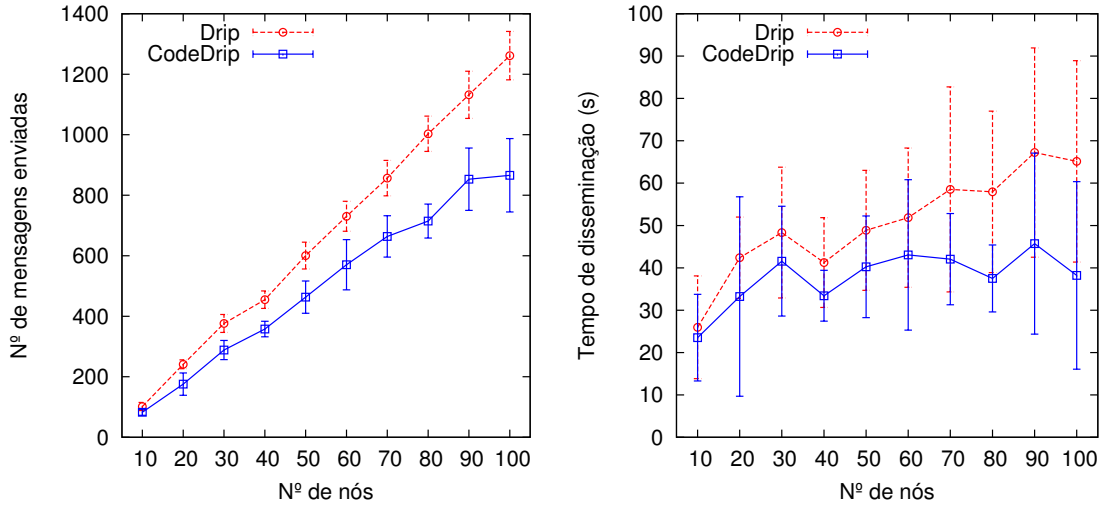


Figura 6. Número de mensagens enviadas e tempo de disseminação para o Drip e o CodeDrip à medida que o número de nós aumenta em uma topologia com perda média de 50% dos pacotes.

de disseminação está começando, pois existem existem poucos nós transmitindo dados e vários deles são perdidos. No CodeDrip mesmo quando a quantidade de nós transmitindo é pouca, os dados fluem pela rede de maneira mais uniforme. Nesse gráfico também podemos observar que o tempo de disseminação se prolonga bastante por causa de apenas um nó sensor que não tenha recebido alguma mensagem, e a combinação de mensagens aumenta a probabilidade desse nó sensor conseguir o dado que lhe falta, terminando a disseminação mais cedo.

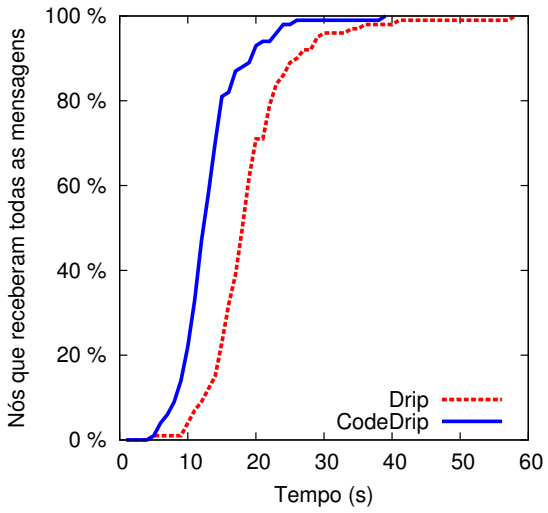


Figura 7. Porcentagem de nós da rede que já receberam todas as mensagens à medida que o tempo passa. Topologia com 100 nós sensores e 60% de chance de perda.

No CodeDrip são implementadas duas políticas de decisão probabilísticas. A primeira é para decidir se a mensagem enviada será original ou combinada. Essa decisão é feita imediatamente antes de enviar a mensagem, ou seja, quando o TrickleTimer corres-

pondente é disparado. A segunda política decide o que fazer quando o nó sensor recebe uma mensagem que ele já possui. Receber com muita frequência uma mesma mensagem indica que seus vizinhos já a possuem. Logo, é razoável que se esse nó sensor não precise ter pressa para enviar essa mensagem, já que provavelmente ela não fará tanta falta e não enviá-la agora prejudicará pouco o processo de disseminação. Então o nó sensor pode decidir suprimir, ou seja, adiar o envio dessa mensagem.

A probabilidade com que essas decisões são tomadas irá afetar diretamente o desempenho do protocolo. Para descobrir qual é a melhor configuração de probabilidades do CodeDrip foram feitas simulações aplicando diferentes valores para as mesmas. Os resultados são mostrado nas figuras 8 e 9.

A figura 8 mostra como o desempenho do CodeDrip é afetado à medida que aumenta-se a probabilidade usada para combinar uma mensagem antes de enviá-la. Quando essa probabilidade passa de 30%, a quantidade de mensagens combinadas enviadas passa a prejudicar o processo de disseminação. Se um nó sensor recebe apenas mensagens combinadas, ele não consegue decodificá-las pois, para isso, é preciso que ele tenha ou receba uma mensagem original. Com essa probabilidade alta, muitos sensores ficarão apenas com combinações armazenadas esperando chegar mensagens originais, que são enviadas com menos frequência. Logo, com base nos gráficos da figura 8, o melhor é que a probabilidade de enviar uma mensagem combinada seja 30%.

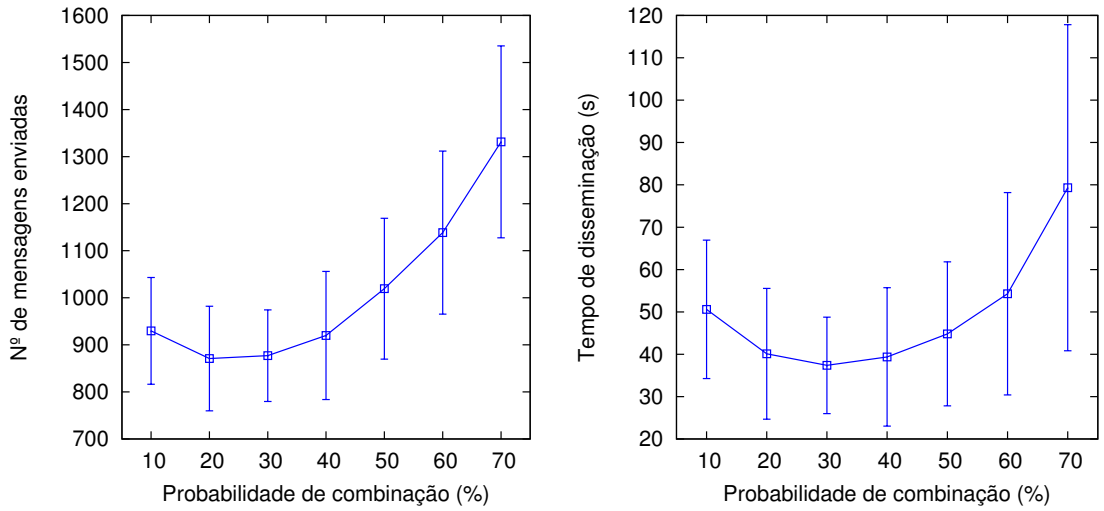


Figura 8. Quantidade de mensagens enviadas e tempo de disseminação em função da probabilidade de combinar pacotes na hora de enviar.

Na figura 9 estão representados resultados dos experimentos feitos aumentando a probabilidade de suprimir o envio de mensagens que são redundantes para aquele nó sensor. O número de mensagens enviadas diminui até a probabilidade de 50% ao preço da disseminação ser um pouco mais lenta. Como essa diferença de tempo não é tão significativa, a economia de mensagens é vantajosa. A partir de 50% o número de mensagens se estabiliza e o tempo de disseminação aumenta. Isso porque mensagens que seriam importantes para um determidado vizinho do nó sensor são suprimidas porque ele recebe essa mesma mensagem de outros vizinhos que a possuem. Sendo assim, a probabilidade que gerará melhores benefícios é de 50%.

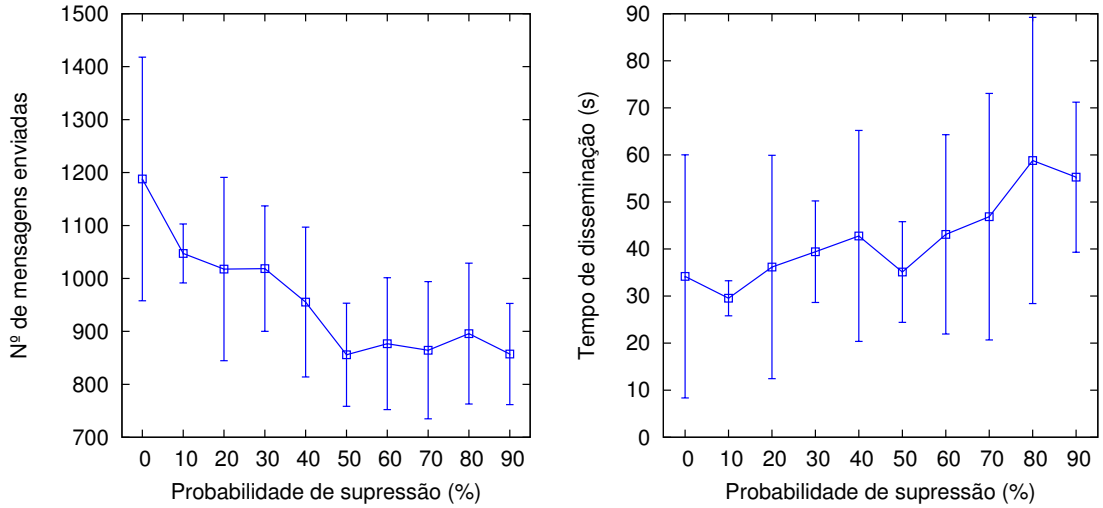


Figura 9. Quantidade de mensagens enviadas e tempo de disseminação em função da probabilidade de suprimir o envio de mensagens redundantes.

7. Conclusão

Neste artigo foi apresentado o CodeDrip, um protocolo de disseminação de dados em redes de sensores sem fio. A ideia principal desse novo protocolo é aplicar a técnica de Codificação em Rede no processo de disseminação, e com isso economizar transmissões de pacotes. Vale lembrar que reduzindo o número de mensagens, reduz-se o consumo de energia da rede, uma importante métrica em RSSFs. Os resultados obtidos nas simulações mostraram que o CodeDrip envia menos mensagens do que o Drip. Além disso, o tempo de disseminação também reduziu, o que mostra que o CodeDrip é mais robusto a falhas de comunicação que o Drip.

O custo adicional necessário para implementar o CodeDrip são identificadores que tem que ser mandados junto com as mensagens para que possa ser possível decodificá-las, e um buffer para armazenar combinações de mensagens que não puderam ser decodificadas. O custo adicional é facilmente controlado definindo um número máximo de mensagens que podem ser combinadas e um tamanho máximo do buffer.

Os ganhos verificados nesse trabalho referem a um custo adicional mínimo no conteúdo de uma mensagem, permitindo a combinação de apenas duas mensagens originais, e um buffer que permite armazenar até três mensagens combinadas. Logo, a economia de energia nos nós sensores por fazer menos transmissões tem o custo de um pouco de memória do dispositivo. Uma aplicação precisa estar utilizando praticamente toda a memória do dispositivo para que esse custo se torne crítico a ponto de não poder usar o CodeDrip por causa disso.

Como possíveis trabalhos futuros, pode-se apontar um estudo mais aprofundado sobre o comportamento do protocolo CodeDrip para casos em que mais mensagens são disseminadas pela rede. Outro possível trabalho é desenvolvimento de novas políticas de combinação de mensagens, fazendo combinações mais complexas a fim de obter um desempenho ainda melhor, analisando as vantagens e desvantagens de cada estratégia.

Agradecimentos

Gostaríamos de agradecer à FAPEMIG, CAPES, CNPq.

Referências

Ahlswede, R., Cai, N., Li, S.-Y., and Yeung, R. (2000). Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216.

Chandanala, R. and Stoleru, R. (2010). Network coding in duty-cycled sensor networks. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pages 203–210.

Cui, T., Chen, L., and Ho, T. (2010). On distributed scheduling in wireless networks exploiting broadcast and network coding. *Trans. Comm.*, 58(4):1223–1234.

Deb, S. (2005). Network coding for wireless applications: a brief tutorial. In *International Workshop on Wireless Ad-hoc Networks (IWWAN)*.

Gay, D., Welsh, M., Levis, P., Brewer, E., Behren, R. V., and Culler, D. (2003). The nesc language: A holistic approach to networked embedded systems. In *In Proceedings of Programming Language Design and Implementation (PLDI)*, pages 1–11.

Hui, J. W. and Culler, D. (2004). The dynamic behavior of a data dissemination protocol for network programming at scale. In *In Proceedings of the 2nd international*, pages 81–94. ACM Press.

Katti, S., Rahul, H., Hu, W., Katabi, D., Medard, M., and Crowcroft, J. (2008). XORs in the Air: Practical Wireless Network Coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510.

Keller, L., Karaagac, A., Fragouli, C., and Argyraki, K. (2011). Evaluation of network coding techniques for a sniper detection application. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2011 International Symposium on*, pages 327–333.

Lee, U., Park, J.-S., Yeh, J., Pau, G., and Gerla, M. (2006). Code torrent: content distribution using network coding in vanet. In *MobiShare '06: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5, New York, NY, USA. ACM.

Levis, P. and Gay, D. (2009). Tinyos programming.

Levis, P., Lee, N., Welsh, M., and Culler, D. (2003). Tossim: Accurate and scalable simulation of entire tinyos applications.

Levis, P., Patel, N., Culler, D., and Shenker, S. (2004). Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, pages 15–28.

Li, S.-Y., Yeung, R., and Cai, N. (2003). Linear network coding. *Information Theory, IEEE Transactions on*, 49(2):371–381.

Ruiz, L. B., Correia, L. H., Vieira, L. F. M., Macedo, D. F., Nakamura, E., Figueiredo, C. M., Vieira, M. A. M., Mechelane, E. H., Camara, D., Loureiro, A. A. F., Nogueira,

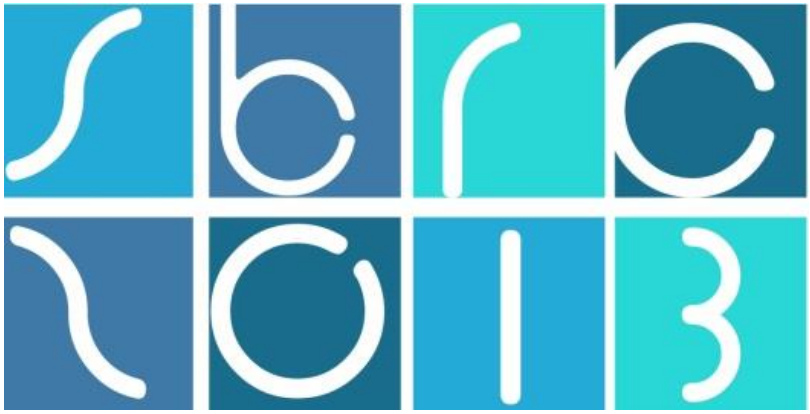
J. M., and da Silva Jr., D. C. (2004). Arquitetura para redes de sensores sem fio. *Minicurso, XXII Congresso da SBRC*.

Tolle, G. and Culler, D. (2005). Design of an application-cooperative management system for wireless sensor networks. In *Second European Workshop on Wireless Sensor Networks (EWSN) Istanbul, Turkey, January 31 - February 2, 2005.*, pages 121–132.

Vieira, L., Misra, A., and Gerla, M. (2007). Performance of network-coding in multi-rate wireless environments for multicast applications. In *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pages 1–6. IEEE.

Wang, X., Wang, J., and Xu, Y. (2010). Data dissemination in wireless sensor networks with network coding. *EURASIP Journal on Wireless Communications and Networking*, 2010(1):465915.

Zhang, J. and Zhang, Q. (2009). Cooperative network coding-aware routing for multi-rate wireless networks. *IEEE INFOCOM 2009 The 28th Conference on Computer Communications*, pages 181–189.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 10

Mobilidade

Localização de Dispositivos Móveis em Redes WiFi usando Variação da Potência de Transmissão e kNN

Helmer A. S. Mourão, Horácio A. B. Fernandes de Oliveira

[1]Instituto de Informática – Universidade Federal do Amazonas (UFAM)
Manaus – AM – Brazil

{helmermourao, horacio}@icomp.ufam.edu.br

Abstract. *Indoor localization systems are usually based in the signal strength of packets received by a set of WiFi routers. Among the most precise solutions, are those based in trainment, that use previously recorded received signal strength indications (RSSI) to allow a classifier, such as kNN (k-Nearest Neighbor) to estimate the position. In these solutions, packets are sent by the mobile devices using the same transmission power. In this paper, we go further and propose the use of different transmission powers, so the information about the signal fading can be taken into consideration by the classifier. Our results clearly indicate significant improvement when compared to the traditional solutions. In some configuration, our solution resulted in less than 1 m error in 96% of the cases.*

Resumo. *Sistemas de localização de dispositivos móveis em ambientes internos são geralmente baseados na potência de recepção de pacotes em um conjunto de roteadores WiFi. Dentre as soluções de localização mais precisas, estão as baseadas em treinamentos, que utilizam informações previamente coletadas de potências de sinais recebidos (RSSI - Received Signal Strength Indication) para permitir que um classificador, como o kNN (k-Nearest Neighbor), estime a posição. Em tais abordagens, os pacotes são enviados pelos dispositivos móveis sempre com a mesma potência de transmissão. Neste artigo, vamos além e propomos o envio dos pacotes em diferentes potências de transmissão, para que a característica de queda do sinal possa também ser levada em consideração pelo classificador. Os resultados obtidos foram animadores e mostraram ganhos significativos quando comparado às abordagens tradicionais encontradas na literatura obtendo, em algumas configurações, erros menores que 1 m em 96% dos casos.*

1. Introdução

Nos últimos anos tem-se observado a massificação dos dispositivos móveis no mercado consumidor tais quais *notebooks*, *netbooks*, aparelhos celulares e *smartphones* [Santos et al. 2011]. Isso, aliado ao aumento da capacidade de processamento e o surgimento de novas tecnologias de comunicação, possibilitou o desenvolvimento de aplicativos orientados a contexto [Moura 2007]. Em tais aplicativos, um dos pontos essenciais é localizar o dispositivo no ambiente em que ele se encontra.

Uma das principais soluções para a localização/posicionamento é a utilização de receptores GPS (*Global Positioning System*), já disponível em muitos dos dispositivos

móveis. Entretanto, tal sistema tem uma baixa performance em ambientes internos devido à falta de visada direta aos satélites, o que impede ou gera erros de localização inaceitáveis [Ni et al. 2004]. Portanto, em tais casos, um sistema de localização local é necessário.

Um desafio para os sistemas móveis é ser o mínimo invasivo possível, através da obtenção das informações necessárias de forma imperceptível. Nesse aspecto, um sistema de localização que se aproveita da infraestrutura de rede WiFi já existente em edifícios e locais públicos torna-se algo bastante desejável. Dentre as principais soluções para localização em ambientes internos encontram-se as soluções baseadas em treinamentos [Moura 2007, Ni et al. 2004, Jan and Lee 2003], Ekahau [1], que utilizam informações previamente coletadas de potências de sinais recebidos (RSSI - *Received Signal Strength Indication*) para permitir que um classificador, como o kNN (*k-Nearest Neighbor*), estime a posição dos nós. Tais soluções têm sido consideradas ultimamente devido a dois motivos principais: (1) disponibilidade das informações de potência do sinal sem necessidade de equipamentos extras; e (2) imunidade do sistema à grande variação e imprevisibilidade do RSSI em ambientes internos.

Nos sistemas baseados em treinamento, é comum que o dispositivo móvel envie alguns pacotes com uma certa potência de transmissão e estes serão recebidos por diversos roteadores com potências de recepção diferentes dependendo do posicionamento deles. Tais informações de potência de recepção serão utilizadas para determinar a posição do nó móvel através do classificador.

Neste trabalho, vamos além e experimentamos tal técnica através da variação da potência de transmissão dos pacotes enviados pelos nós móveis. Desta forma, na nossa solução, chamada SPoT (Sistema de Posicionamento com variação da Potência de Transmissão), ao invés de enviarmos diversos pacotes com a mesma potência de transmissão, enviamos diversos pacotes cada um com uma potência de transmissão diferente. O objetivo dessa variação é que o comportamento da queda dos sinais possa servir como uma característica extra para o classificador usado como, no nosso caso, o kNN [Bahl and Padmanabhan 2000], resultando em erros menores de localização.

O restante deste trabalho está organizado como segue. Na Seção 2, apresentamos os trabalhos relacionados. A Seção 3 descreve o sistema de localização proposto, cuja avaliação de performance é mostrada na Seção 4. Finalmente, na Seção 5, apresentamos nossas conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Nos algoritmos baseados em treinamento, também conhecidos como *fingerprinting*, mapas de RSSI são previamente gerados com base nos valores de RSSI obtidos nos diferentes pontos do local em que se defesa ter a localização. Em seguida, um classificador usa tais informações para estimar a posição dos nós. Um classificador muito utilizado para este fim é o kNN (*k-Nearest Neighbor* – k-Vizinho(s) mais Próximo), conforme utilizado pelo RADAR [Bahl and Padmanabhan 2000].

No RADAR [Bahl and Padmanabhan 2000], um dos primeiros sistemas de localização usando RSSI, leituras do RSSI são realizadas em diversos pontos do local

[1] Ekahau,Inc. Ekahau Positioning Engine 2.0. http://www.ekahau.com/

em que se quer realizar a localização. Cada leitura contêm os valores da potência do sinal recebido pelos diferentes roteadores que receberam o pacote enviado naquele local. Tais informações são registradas em um banco de dados. Após este treinamento, quando um usuário solicita a sua localização, os valores de RSSI da solicitação são comparados com os observados na fase de treinamento usando o classificador kNN, que irá retornar a posição mais provável em que a solicitação foi enviada, informação esta que será retornada para o usuário.

No sistema Horus [Youssef et al. 2003, Youssef and Agrawala 2004], é proposta uma técnica de agrupamento conjunta para a estimativa de localização que usa um método probabilístico. Nesse sistema, cada possível coordenada em que o candidato possa estar é considerada uma classe ou categoria. A fim de minimizar o erro de distância, a localização é escolhida enquanto a sua probabilidade é maior. Aumentando o número de amostras em cada local de amostragem permite melhorar a sua precisão, pois isso melhora a estimativa das médias e desvios-padrão da distribuição Gaussiana utilizada. Sistemas probabilísticos foram propostos também em outros trabalhos [Roos et al. 2002, Castro et al. 2001]

Em [Battiti et al. 2002], é proposto um método de determinação de localização com treinamento usando classificadores baseados em redes neurais. Em seguida, os autores em [Saha et al. 2003] compararam os diversos classificadores citados até o momento: redes neurais, kNN e métodos probabilísticos, mostrando que todas as técnicas tiveram comportamento muito bons.

Em [Hamza 2010] diversos experimentos foram realizados confirmando que as técnicas de localização para ambientes internos baseadas em treinamento (*fingerprinting*) têm desempenho melhor do que as tradicionais baseadas em trilateração/multilateração. O principal motivo apontado foi o fenômeno de *multipath*, em que o comportamento do sinal é afetado pelos diversos obstáculos encontrados em tais cenários. Entretanto, uma observação levantada neste trabalho é que as técnicas de localização baseada em treinamento sofrem degradações na precisão da localização em ambientes dinâmicos, onde as propriedades do local podem mudar com o tempo (rearranjamento de armários, mesas, divisórias, etc).

Levando em consideração os problemas de *multipath* e ambientes dinâmicos, em [Fang and Lin 2010] é proposta uma abordagem com treinamento onde a posição é estimado a partir de um "estado", ao invés do RSSI diretamente. Tal estado é reconstruído a partir de uma sequência temporal de amostras de RSSI. A partir daí, o mais preciso estado de localização é estimado porque o impacto da variação temporal devido ao *multipath* é considerado.

Como pode-se observar, as soluções propostas na literatura são baseadas sempre em pacotes que são enviados sempre com a mesma potência de transmissão. Já no SPoT, solução proposta neste trabalho, os pacotes são enviados com potências diferentes, de forma que essa diferença na queda do sinal possa ser levado em consideração pelo classificador na hora de localizar os dispositivos móveis.

3. SPoT: Sistema de Posicionamento com Variação da Potência de Transmissão

Nesta seção, explicamos o funcionamento do SPoT (Sistema de Posicionamento com variação da Potência de Transmissão), sistema de localização proposto neste trabalho.

Conforme mencionado, nos sistemas encontrados na literatura, são utilizados como informação de entrada os RSSIs recebidos nos diferentes roteadores WiFi ou mesmo uma média de uma janela de RSSIs recebidos (diversos pacotes enviados pelo nó móvel). A nossa proposta é fazer com que se tenha mais informações para o classificador fazer uma inferência de melhor qualidade sobre a localização do dispositivo móvel. Entretanto, precisamos de novas características que possam ser de fácil obtenção em uma infraestrutura WiFi. Desta forma, a ideia da nossa proposta é utilizar a comunicação entre dispositivo e roteadores com diferentes potências de transmissão, visto que isso é um recurso disponível na maioria dos *drivers* de dispositivos WiFi comercializados [Moura 2007].

Tendo essas informações extras, passamos para a segunda fase que é a escolha de um algoritmo para o tratamento dos dados. Como optamos por resolver nosso problema de maneira discreta (conforme explicado mais adiante), escolhemos um algoritmo de classificação que é simples mas que obteve bons resultados em outros trabalhos como em [Bahl and Padmanabhan 2000] e [Ni et al. 2004]. Tal classificador é o kNN (*k-nearest neighbor*).

O kNN foi escolhido por ser um algoritmo que trata bem dados muito ruidosos como é o caso do RSSI, por que diferente do SVM e das RNA‘s que tentam criar uma superfície de separação (que às vezes não existem) ele rotula pelas amostras pelas mais parecidas, e com pequenos valores de k ele consegue bons resultados [Wettschereck et al. 1997].

Na área de reconhecimento de padrões, o algoritmo kNN é considerado um algoritmo de aprendizado supervisionado e foi introduzido por [Cover and Hart 1967]. A ideia geral desse algoritmo “consiste em encontrar os k exemplos rotulados mais próximos do exemplo não classificado e, com base no rótulo desses exemplos mais próximos, é tomada a decisão relativa à classe do exemplo não rotulado” [Ferrero 2009].

Em nossa modelagem usando o kNN, cada indivíduo (amostra coletada ou ponto a ser localizado) é representado em um plano cartesiano onde suas coordenadas correspondem aos valores das suas características, assim uma amostra de um ponto X é representada por um ponto (x_1,x_2,...,x_n) onde n é o numero de características da amostra. Neste trabalho cada coordenada (x_1,x_2,...,x_n) corresponde a um valor do RSSI em uma determinada potência de transmissão.

Ao utilizar o kNN, é necessário ajustar os parâmetros do classificador que, neste caso são o valor de k (que representa o número de vizinhos que serão considerados no cálculo) e como será feita o cálculo da distancia entre 2 pontos.

Para definirmos a melhor configuração para nosso classificador, foram feitos experimentos preliminares para ajustar esses parâmetros, dessa forma obtemos bases preliminares para testar entre varias configurações qual a melhor considerando como métrica a taxa de acerto.

Utilizando técnicas de seleção de parâmetros, foi possível decidir no kNN com

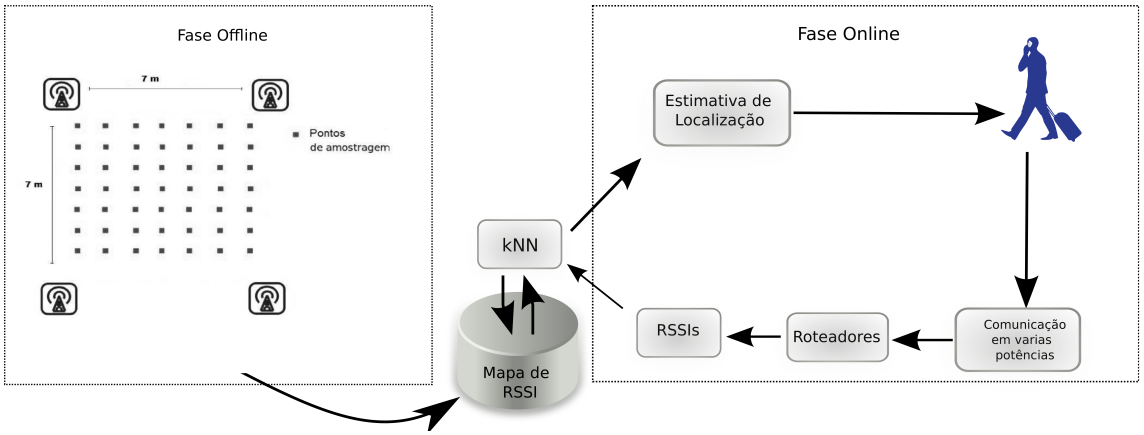


Figura 1. Arquitetura do Sistema de Localização.

k igual a 5 e usando o calculo de distancia "Manhattan", uma vez que configuração se mostrou melhor nos testes preliminares.

Como nas outras abordagens baseadas em treinamento, nossa proposta de localização é discreta. Isso significa que ela não diz exatamente qual a localização do nó, mas sim em que "célula" ele se encontra. Desta forma, o kNN foi treinado com amostras rotuladas pelo centro de uma área de $1\,m^2$ (célula), onde foram feitas as coletas que construíram a sua base de treinamento. Dessa forma, se o classificador acerta a célula em que o dispositivo se encontra é considerado que ele acertou a posição do dispositivo. Tal asserção é bem razoável visto que o erro máximo do centro para qualquer parte da célula será de 0,7 m.

A arquitetura geral do SPoT é ilustrada na Figura 1. Em nossa arquitetura, na fase de treinamento (chamada de fase *offline*), será criado um mapa de RSSI que servirá de base para o treinamento do kNN. Tal mapa é gerado levando um dispositivo móvel para uma localização conhecida da região de interesse e fazendo com que ele envie pacotes de dados em diferentes potências de transmissão. Tais pacotes serão recebidos pelos roteadores (3 ou mais) que guardarão os dados de RSSI em uma base de dados juntamente com a posição conhecida do dispositivo. Tal procedimento deverá ser realizado em cada uma das células que se deseja obter a localização dos dispositivos. Quanto maior o treinamento, maior será a precisão do sistema. Na segunda fase (*online*), um dispositivo móvel poderá requisitar a sua localização enviando dois ou mais pacotes em potencias diferentes e, com base nos RSSIs observados nos roteadores, o classificador kNN irá estimar a localização (célula) do dispositivo.

4. Avaliação de Performance

A avaliação de performance do SPoT foi realizada através de experimentos práticos em um ambiente interno (*indoor*). Em nossos experimentos, utilizamos roteadores sem fio Cisco/Linksys WRT610N com o *firmware* DD-Wrt e um software simples desenvolvido para a captura dos pacotes e obtenção dos RSSIs (baseado na *libpcap*).

Para fins de comparação, foi-se implementado um sistema de localização tradicional baseado em treinamento e usando o kNN. Conforme mencionado anteriormente, tal sistema é baseado na utilização de uma única potência de transmissão. Os experi-

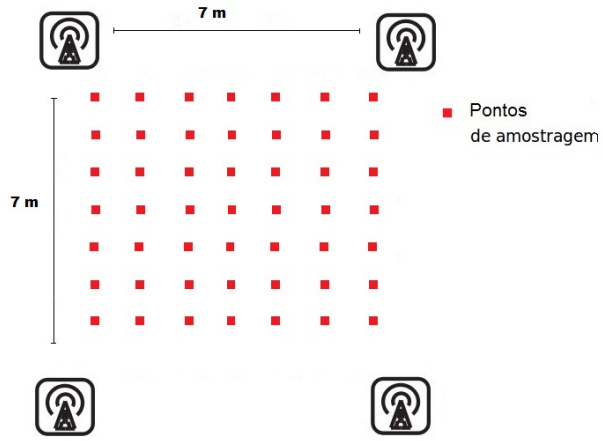


Figura 2. Coleta de Informações.

mentos desse sistema tradicional foram realizados ao mesmo tempo e com os mesmos equipamentos utilizados na nossa abordagem. Desta forma, será possível ver exatamente a influência da utilização de diferentes potências de transmissão no erro de localização, uma vez que todo o restante se manteve igual nas duas abordagens.

A implementação do kNN usada foi a da ferramenta *Waikato Environment for Knowledge Analysis* (WEKA)[2] que é uma coleção de algoritmos de aprendizado de máquina para tarefas de mineração de dados. Escolhemos essa ferramenta por ser simples (apesar de possuir muitas funcionalidades) e ser disponibilizada gratuitamente, além de possuir muita documentação disponível.

4.1. Montando o Mapa de RSSI

Para a primeira fase do sistema (*offline*), foi necessário obter um banco de dados sobre a variação do RSSI para o treinamento do kNN.

Como ambiente de testes escolhemos uma sala fechada com área de 7x7 m^2, essa sala foi usada para simular um ambiente interno tendo alguns obstáculos como mesas e cadeiras.

Em cada extremidade dela foi colocado um roteador que ficou captando os pacotes que estavam sendo enviados a partir de um *notebook*. O *notebook* foi colocado em posições previamente definidas e conhecidas no *grid*, conforme mostra a Figura 2. Foram coletados 200 pacotes por posição alternando a potência de transmissão em 5 valores diferentes. Tais valores foram: 0, 5, 10, 15 e 20 dBm.

Os pacotes de treinamento foram captados pelos quatro roteadores e armazenados para a construção da base de dados, resultando em uma base contendo informações sobre as 49 posições do *grid*. Em seguida, tais informações da base foram utilizadas para o treinamento do kNN.

[2]http://www.cs.waikato.ac.nz/ml/weka/

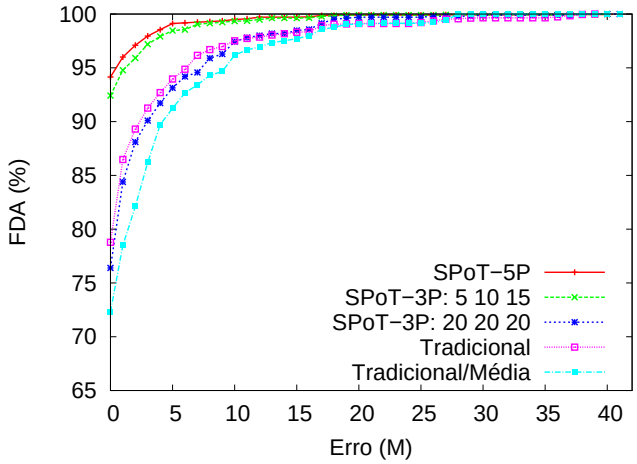


Figura 3. Resultados obtidos usando quatro roteadores.

4.2. Análise dos Resultados usando Quatro Roteadores

Para analisarmos os resultados obtidos pelo SPoT, utilizamos como métrica o erro acumulado em metros (FDA - Função de Distribuição Acumulada). O erro de classificação para cada amostra, nas 49 células de 1 m^2, é calculado com base na posição real da coleta e na posição estimada pelo classificador.

No gráfico da Figura 3, podemos ver o comportamento do erro acumulado para 3 configurações diferentes do SPoT comparadas a 2 variações da abordagem tradicional. A curva "SPoT-5P" mostra o comportamento do erro de localização para o SPoT utilizando as 5 potências de transmissão estudadas, ou seja, o nó móvel enviou 5 pacotes com potências de transmissão diferentes (0, 5, 10, 15 e 20 dBm). Por usar mais características, esta curva foi a que obteve melhores resultados, confirmando as nossas suposições iniciais de que as diferentes potências poderiam ser usadas para melhorar a qualidade do classificador. Como pode-se observar nessa curva, nosso sistema foi capaz de localizar os nós com um erro abaixo de 1 m em mais de 95% dos casos.

Ainda na Figura 3, a curva "SPoT-3P: 5 10 15" mostra o comportamento do SPoT utilizando apenas três potências diferentes, ou seja, o nó móvel enviou três pacotes com potências de transmissão diferentes (5, 10 e 15 dBm). Esta curva se mostrou um pouco inferior à primeira por utilizar menos informações. Entretanto, é fácil de se perceber que os resultados foram bastante próximos e que, assim como na primeira curva, ela ainda se destaca das demais, sendo uma solução aceitável para os casos em que se queira reduzir a quantidade de pacotes enviados pelo dispositivo móvel.

Para melhor entender os resultados obtidos, implementamos uma versão do SPoT que utiliza três pacotes (parecido com a variação anterior) mas que ao invés de usar pacotes com potências diferentes, esta variação utilizou pacotes diferentes mas com a mesma potência de transmissão (20 dBm). Apesar de não ser o algoritmo proposto, essa variação serviu para diferenciar mais facilmente o que é ganho relativo à variação do sinal do que não é. Como pode ser observado na curva "SPoT-3P: 20 20 20", esta variação ficou bem abaixo das outras duas, que utilizaram a variação da potência de transmissão. Este resultado mostra claramente que os ganhos obtidos pelo SPoT devem-se basicamente à mudança da potência de transmissão.

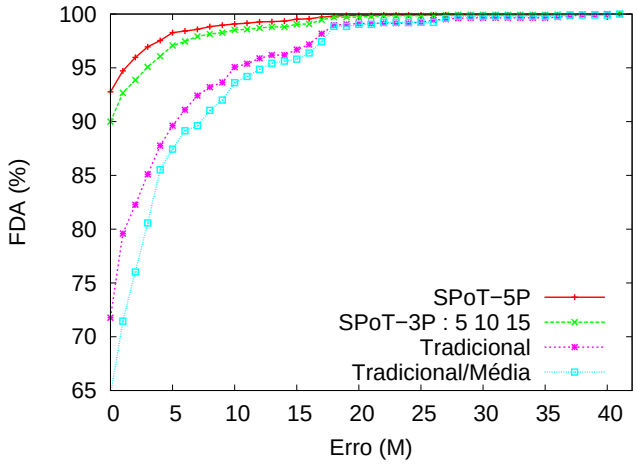


Figura 4. Resultados obtidos usando três roteadores.

Finalmente, a curva "Tradicional" ilustra o comportamento de um sistema de localização tradicional, através de um classificador kNN que utiliza os dados de RSSIs obtidos a partir de um único pacote sempre usando a mesma potência de transmissão. Já a curva "Tradicional/Média" mostra novamente a abordagem tradicional mas com o envio de três pacotes (usando a mesma potência de transmissão) e tirando a média dos RSSIs destes três pacotes. Como pode se observar, as duas abordagens tradicionais obtiveram uma qualidade de localização bem inferior se comparado ao SPoT.

4.3. Análise dos Resultados usando Três Roteadores

Na seção anterior, mostramos os resultados obtidos pelo SPoT utilizando os quatro roteadores disponíveis. Para entender um pouco mais do seu funcionamento e comportamento em diferentes cenários, resolvemos experimentar a solução proposta utilizando apenas três roteadores, quantidade mínima necessária para obter uma solução única. Como haviam quatro roteadores disponíveis, foram realizados testes com as várias combinações de roteadores, 3 a 3, resultando em 4 possíveis combinações de roteadores. Entretanto, percebemos que todas apresentavam o mesmo comportamento e, por isso, mostramos apenas uma das combinações no gráfico da Figura 4.

Novamente, podemos observar que o SPoT foi capaz de localizar os nós com erros bem abaixo dos obtidos pelas abordagens tradicionais. Comparando este gráfico com o anterior (Figura 3), é possível observar uma queda muito pequena na qualidade da localização, mostrando que nossa abordagem mantém uma qualidade aceitável mesmo com uma quantidade menor de roteadores e mostrando ainda que ao se aumentar a quantidade de roteadores, há um ganho muito pequeno na qualidade da localização, o que pode não compensar na maioria dos casos.

4.4. Análise dos Resultados usando diferentes Combinações das Potências de Transmissão

Conforme mencionado anteriormente, e observado no gráfico da Figura 4, a utilização do SPoT com apenas três pacotes (com três potências diferentes), teve resultados próximos aos obtidos executando o SPoT com 5 pacotes e potências diferentes. Como tínhamos que

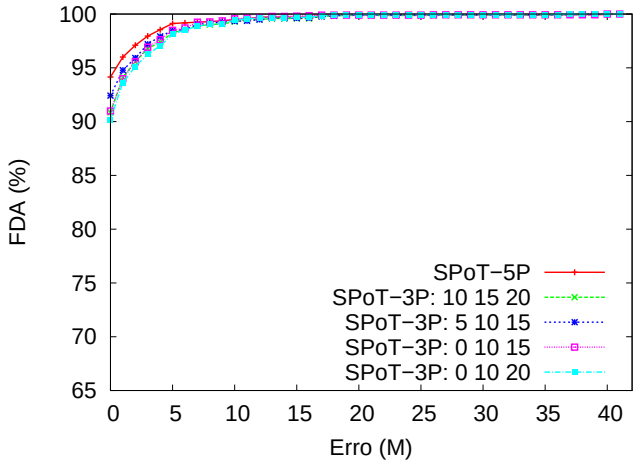


Figura 5. Resultados obtidos variando as potências 3 a 3.

escolher 3 potências de transmissão dentre as 5 disponíveis, resolvemos analisar o comportamento do erro de localização obtido por cada uma dessas possíveis combinações (e utilizando apenas três roteadores). A ideia inicial era avaliar, por exemplo, se a utilização das potências maiores de transmissão seria mais vantajoso em relação às potências menores. Entretanto, o que foi observado e que está ilustrado no gráfico da Figura 5 é que basicamente não houve mudança no desempenho do sistema. Portanto, levando-se em consideração que os pacotes enviados com potências maiores alcançam distâncias maiores, o gráfico da Figura 5 indica que podemos utilizar as maiores potências disponíveis (10, 15 e 20 dBm, neste caso) sem se preocupar com uma degradação perceptível na precisão da localização.

Ainda nessa linha de combinações das potências, e procurando aumentar ainda mais os conhecimentos sobre o comportamento do sistema, resolvemos executar o SPoT fazendo o dispositivo móvel enviar apenas dois pacotes com potências de transmissão diferentes. Assim como no resultado anterior, diversas combinações surgem a partir das 5 potências disponíveis. Dentre as combinações disponíveis, escolhemos as quatro mais relevantes e executamos o SPoT com elas. Os resultados obtidos são mostrados no gráfico da Figura 6. Novamente, como pode-se observar, não houve diferenças significativas nos resultados. Entretanto, comparando este gráfico com o anterior (que usa três potências), é possível perceber que houve uma queda na qualidade da localização. Daí, pode-se observar que o SPoT é bem dependente da quantidade de pacotes enviados em potências diferentes, motivo esse que nos levará a pesquisar ainda mais nossa solução com a utilização de ainda mais potências (trabalhos futuros).

4.5. Análise dos Resultados usando diferentes Quantidades de Potências de Transmissão

No gráfico da Figura 7, podemos observar o comparativo com várias quantidades de potência. Com o uso de apenas 2 potências, já é possível observar uma melhora considerável na qualidade de localização. Aumentando a quantidade de potências utilizadas para 3, 4 e 5 potências, é possível observar um ganho ainda maior na localização. Entretanto, é possível observar que o incremento de ganho de uma quantidade de potências para outra parece diminuir, concluindo-se que se faz necessário uma relação de compro-

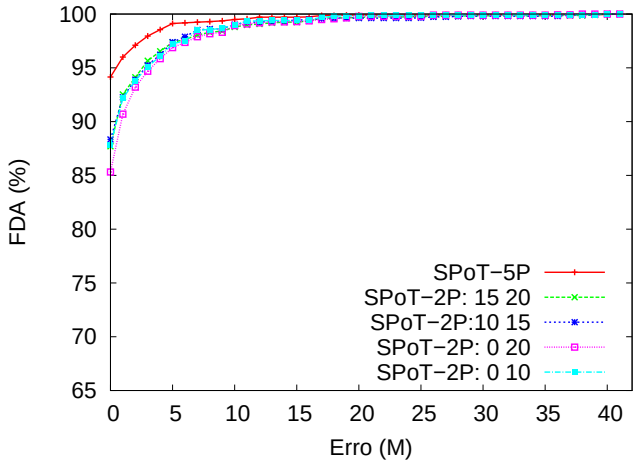


Figura 6. Resultados obtidos variando potências 2 a 2.

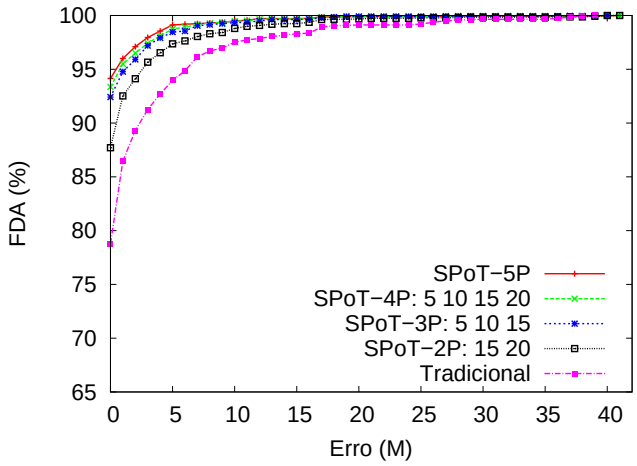


Figura 7. Resultados obtidos variando a quantidade de potências.

misso entre o acerto do classificador e o custo em quantidade de pacotes enviados pelo dispositivo móvel, uma vez que mais potências/características significa o uso de mais pacotes.

5. Conclusão

No presente trabalho, propomos um novo sistema de localização para ambientes internos: o SPoT (Sistema de Posicionamento com variação da Potência de Transmissão). Nossa abordagem proposta é baseada nas técnicas de treinamento e classificação usando kNN, uma das técnicas mais usadas em ambientes internos por levar em consideração a variação e imprevisibilidade do RSSI. Entretanto, nossa abordagem se distingui das demais por utilizar diferentes potências de transmissão como características extras para o classificador, permitindo a este último gerar estimativas mais precisas de localização.

O sistema proposto foi avaliado através de experimentos reais em um ambiente interno utilizando diversas combinações de (1) quantidade de roteadores, (2) quais roteadores, (3) quantidade de potências de transmissão, e (4) quais potências de transmissão, tudo isso comparado com a abordagem tradicionalmente utilizada em localização interna

bem como com sua variante que usa a média do RSSI de vários pacotes. Os resultados obtidos apontam fortes indícios de que essas características extras de diferentes potências são relevantes e melhoram a qualidade da localização obtendo-se erros de menos de 1 m em 96% dos casos em alguns cenários.

Os resultados obtidos são bastante promissores. Como trabalhos futuros, pretendemos experimentar o SPoT em ambientes maiores e com mais obstáculos. Além disso, serão realizados experimentos com uma quantidade maior de pacotes com diferentes potências, para avaliar até quando podemos aumentar a quantidade de pacotes e ainda obter resultados melhores, outra coisa a ser a analisada é a possibilidade do uso de outras características como frequências diferentes e analises de interferências no canal.

Outra parte importante a ser feita é a implementação de fato para dispositivos móveis visto que nesse trabalho utilizamos um notebook para simular o *beacon* móvel.

Referências

Bahl, P. and Padmanabhan, V. (2000). Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784 vol.2.

Battiti, R., Nhat, T. L., and Villani, A. (2002). Location-aware computing: A neural network model for determining location in wireless lans. Technical report.

Castro, P., Chiu, P., Kremenek, T., and Muntz, R. R. (2001). A probabilistic room location service for wireless networked environments. In *Proceedings of the 3rd international conference on Ubiquitous Computing*, UbiComp '01, pages 18–34, London, UK, UK. Springer-Verlag.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

Fang, S.-H. and Lin, T.-N. (2010). A dynamic system approach for radio location fingerprinting in wireless local area networks. *Trans. Comm.*, 58:1020–1025.

Ferrero, C. A. (2009). Algoritmo knn para previsão de dados temporais: funções de previsão e critérios de seleção de vizinhos próximos aplicados a variáveis ambientais em limnologia. Master's thesis, Universidade de São Paulo.

Hamza, L. N. C. (2010). A dynamic system approach for radio location fingerprinting in wireless local area networks. *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pages 253–258.

Jan, R.-H. and Lee, Y. R. (2003). An indoor geolocation system for wireless lans. In *32nd International Conference on Parallel Processing Workshops (ICPP 2003 Workshops), 6-9 October 2003, Kaohsiung, Taiwan*, pages 29–34. "IEEE Computer Society".

Moura, A. I. (2007). Wbls: um sistema de localização de dispositivos móveis em redes wi-fi. Master's thesis, Universidade de São Paulo.

Ni, L. M., Liu, Y., Lau, Y. C., and Patil, A. P. (2004). Landmarc: indoor location sensing using active rfid. *Wireless Networks - Special issue: Pervasive computing and communications*, 10(6):701–710.

Roos, T., Myllymäki, P., Tirri, H., Misikangas, P., and Sievänen, J. (2002). A probabilistic approach to wlan user location estimation. *International Journal of Wireless Information Networks*, 9(3):155–164.

Saha, S., Chaudhuri, K., Sanghi, D., and Bhagwat, P. (2003). Location determination of a mobile device using ieee 802.11b access point signals. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1987–1992 vol.3.

Santos, D., Perkusich, A., and Almeida, H. O. (2011). Infraestrutura para o desenvolvimento de aplicações pervasivas cientes de redes sociais. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 501–514, Campo Grande, MS.

Wettschereck, D., Aha, D. W., and Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for aclass of lazy learning algorithms. *Artif. Intell. Rev.*, 11(1-5):273–314.

Youssef, M. and Agrawala, A. (2004). Handling samples correlation in the horus system. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1023–1031 vol.2.

Youssef, M. A., Agrawala, A., and Shankar, A. U. (2003). Wlan location determination via clustering and probability distributions. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, PERCOM '03, pages 143–, Washington, DC, USA. IEEE Computer Society.

Padrões de Mobilidade de Vizinhança em Redes de Contato Intermitente

Tiphaine Phe-Neau[1], **Miguel Elias M. Campista**[1,2], **Marcelo Dias de Amorim**[1] **e Vania Conan**[3]

[1] UPMC Sorbonne Universités, Paris – França
[2] Universidade Federal do Rio de Janeiro, RJ – Brasil
[3] Thales Communications, Paris – França

Resumo. *A modelagem dinâmica de redes oportunísticas se baseia no conhecimento dual dos contatos e dos intercontatos. Este trabalho propõe o uso de uma visão estendida, na qual os nós rastreiam sua vizinhança estendida (a alguns saltos) e não somente seus vizinhos diretos. Para tal, é introduzido um método que permite aos nós preverem se outros estarão em alcance dada a posição atual e os movimentos anteriores. Essa abordagem é contrária às existentes nas quais os padrões de contato são extraídos da mobilidade espacial dos nós. O método proposto é aplicado a vários traços reais e sintéticos. Inicialmente, um novo algoritmo é provido bem como uma modelagem intuitiva para compreender o entorno de um nó. Em seguida, são destacadas duas cadeias principais de comportamento da vizinhança. Finalmente, três principais tipos de movimento (nascimento, morte e sequencial) são identificados assim como os seus padrões predominantes.*

Abstract. *Modeling the dynamics of opportunistic networks relies on the dual notion of contacts and intercontacts. We propose the use of an extended view in which nodes track their extended vicinity (up to a few hops) and not only their direct neighbors. We introduce a method that allows nodes predicting whether other nodes will be within reach given their current position and previous movements. This approach is contrary to existing ones where contact patterns are derived from the spatial mobility of nodes. We apply our method to several real-world and synthetic datasets. Firstly, we provide a novel algorithm and an intuitive modeling to understand a node's surroundings. Then, we highlight two main behaviors of vicinity chains. Finally, three main types of movements (birth, death, and sequential) are identified as well as their predominant patterns.*

1. Introdução

A compreensão dos padrões de mobilidade no contexto das redes móveis com conexão intermitente é fundamental para o desenvolvimento de protocolos e algoritmos de rede eficientes. A literatura nessa área tem gerado um número significativo de contribuições que oferecem respostas a questões relacionadas à frequência com que os nós se encontram e como isso ocorre [Conan et al. 2007, Gonzalez et al. 2008, Passarella and Conti 2011]. Uma característica comum nesses trabalhos é que todos se baseiam no conhecimento dos *contatos* e dos *intercontatos*. Um contato ocorre quando dois nós estão em alcance mútuo enquanto um intercontato ocorre quando os dois nós *não estão* em contato.

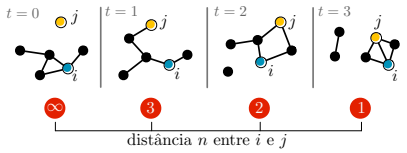


Figura 1. Em $t = 0$, o nó j está fora da vizinhança de i, embora esteja se aproximando. Em $t = 1$, j surge na vizinhança de i com uma distância de três saltos. Em $t = 2$, j se aproxima ainda mais, chegando a uma distância de dois saltos. O nó j entra em contato com i em $t = 3$.

Este trabalho defende que os nós devem considerar uma visão estendida de suas vizinhanças ao incluir nós que não estão em contato, mas que mesmo assim podem ser considerados próximos. Tais nós podem ser alcançados em até κ saltos, sendo o conjunto desses nós referido como κ-*vizinhança* [Phe-Neau et al. 2012]. Acredita-se que nenhum trabalho anterior tenha investigado a evolução no tempo da vizinhança estendida de um nó. A Figura 1 ilustra tal evolução em uma pequena rede. Note que a definição tradicional de contato e intercontato consideraria os primeiros três primeiros intervalos de tempo como iguais para i e j, já que os nós estão em intercontato. Ao contrário, neste trabalho, distingue-se as quatro situações e o impacto dessa definição é investigado através das seguintes questões: Dado que os dois nós i e j estejam separados por n saltos, qual a probabilidade deles estarem separados por m saltos ($m \neq n$) assim que a distância mudar? Ou ainda, é possível identificar padrões nessa dinâmica tal que a movimentação possa ser antecipada?

A movimentação da vizinhança é modelada como um processo Markoviano de tempo contínuo calculado a partir de uma estrutura de dados contendo a evolução da vizinhança de um nó, chamada de "linha de tempo". A ideia é rastrear a vizinhança de um nó até uma distância κ. O limiar κ é ajustado de acordo com a extensão necessária do monitoramento e pela quantidade de sobrecarga tolerada, uma vez que o monitoramento da vizinhança pode gerar sobrecarga de controle. Ao final, a rede pode se basear na movimentação da vizinhança para realizar estimativas de parâmetros fundamentais como o atraso de entrega e as relações sociais entre os nós. O trabalho conhecido mais próximo a este é o CTG (*Connectivity Trace Generator*) que propõe um modelo de conectividade para redes oportunísticas [Calegari et al. 2007]. Entretanto, o CTG também foca na mera noção de contatos e intercontatos, negligenciando os eventos no entorno dos nós.

Este trabalho apresenta as seguintes contribuições, confirmadas a partir de diferentes análises usando traços reais e sintéticos:

- **Um modelo para compreensão do comportamento da vizinhança móvel.** É definida a vizinhança de um nó com a noção de κ-vizinhança assim como é proposto um arcabouço para analisá-la conforme a movimentação. É ainda proposto um fluxo de trabalho (*workflow*) correspondente para geração de cadeias de vizinhança que capturam a evolução estatística da distância entre os nós.
- **Dois tipos de cadeias de vizinhança.** São identificadas duas cadeias principais, curta e estendida, que diferem em função dos estados atravessados. As cadeias estendidas representam uma vizinhança com muitos saltos em potencial dentro do limiar κ, enquanto as curtas possuem vizinhanças de até dois saltos (conforme os resultados obtidos).

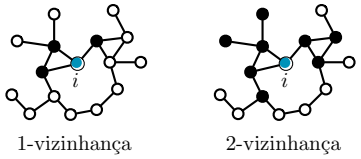


Figura 2. Na esquerda: 1-vizinhança do nó i representa o conjunto de vizinhos a um salto de distância de i. Na direita: 2-vizinhança do nó i representa o conjunto de nós a dois saltos de distância de i. As distâncias mais curtas não são discriminadas em uma dada κ-vizinhança.

- **Três tendências principais de movimento.** Nas cadeias estendidas, são identificados três tipos predominantes de movimentos. Os movimentos de nascimento, de morte e de sequência podem representar 87% de todos os movimentos de um dado traço. Logo, considerar que um desses movimentos é a provável próxima decisão de um nó na vizinhança torna-se razoável.

A metodologia proposta neste trabalho tem como objetivo servir de base para o desenvolvimento de geradores de conectividade, que poderão ser usados na criação de traços realistas com um número qualquer de nós. Este artigo está organizado da seguinte forma. A Seção 2 introduz as informação necessária para a definição de vizinhança. A Seção 3 propõe o fluxo de trabalho para a geração de movimentação da vizinhança. Já as Seções 4, 5 e 6 apresentam os resultados obtidos da análise dos padrão de vizinhança. Finalmente, a Seção 7 conclui este trabalho e apresenta as direções futuras.

2. Definições

Antes de continuar com a análise da movimentação da vizinhança, é importante introduzir formalmente as definições utilizadas.

2.1. κ-vizinhança

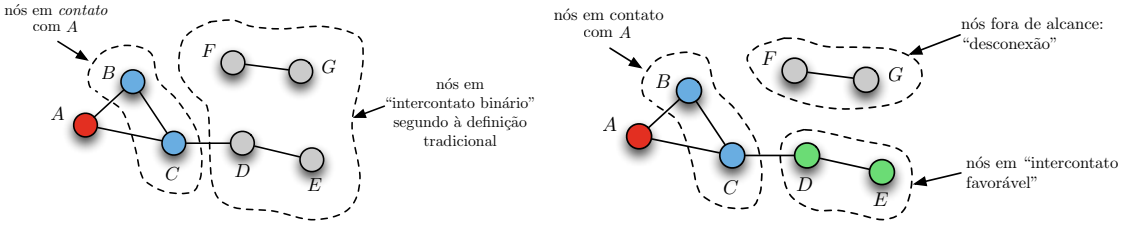
O conceito de κ-vizinhança, definido em um artigo precedente [Phe-Neau et al. 2012], é fundamental já que ele define a extensão na qual a análise se aplica. Logo, discrimina-se a vizinhança de i conforme o número de saltos entre i e os seus vizinhos.

Definição 1. ***κ-vizinhança.*** *A κ-vizinhança $\mathcal{V}_\kappa^i$ de um nó i é o conjunto de todos os nós cujo o caminho mais curto de i é de no máximo κ saltos.*

A partir da Definição 1, conclui-se que $\mathcal{V}_{\kappa-1}^A \subseteq \mathcal{V}_\kappa^A$. A Figura 2 ilustra a 1-vizinhança e a 2-vizinhança do nó i. Neste trabalho, as análises propostas focam em movimentos na κ-vizinhança de um dado nó.

2.2. Intercontato favorável e desconexão

Como dito anteriormente, nas redes móveis com conexão intermitente, esforços para caracterizar a dinâmica da rede estão relacionados à noção dual de *contato* e *intercontato*. Um exemplo é ilustrado na Figura 3(a).Este trabalho distingue os nós que não estão em contato direto, mas que possuem um caminho que os conecta, dos nós que não possuem nenhuma possibilidade de comunicação [Phe-Neau et al. 2011a]. Um par de



(a) Definição tradicional. (b) Definição estendida.

Figura 3. Definição de vizinhança segundo o método tradicional e a proposta.

nós está em intercontato favorável com parâmetro n sempre que há um caminho entre eles com distância mais curta de n saltos. Vale mencionar que dois nós a um salto estão em contato. Formalmente, define-se intercontatos favoráveis e desconexão como se segue:

Definição 2. ***Intercontato favorável.*** *Um intercontato é considerado "favorável" com parâmetro n quando há um caminho mais curto de comprimento n separando os dois nós em questão, tal que $\{n \in \mathbb{N}^* \mid 2 \leq n < \infty\}$.*

Definição 3. ***Desconexão.*** *Ao contrário das situações favoráveis, a "desconexão" indica a falta de caminhos entre um par de nós. Logo, $n \to \infty$.*

Essa visão, que chamamos de "estendida", é ilustrada na Figura 3(b). Os movimentos partindo de contato ou de qualquer um dos n intercontatos favoráveis para qualquer outro estado possível são investigados neste trabalho. Essa análise ajuda a compreender o comportamento da κ-vizinhança de um dado nó.

3. Mobilidade da Vizinhança: Metodologia

A mobilidade da vizinhança considera todos os movimentos dentro da κ-vizinhança de um nó. Para isso, são oferecidos elementos que respondam questões como "*quando a distância n entre os nós i e j muda, qual a probabilidade da distância se tornar m, onde $m \neq n$?*". A resposta é obtida a partir de uma metodologia em duas etapas:

1. **Geração da linha de tempo.** Calcula-se a *linha de tempo* da vizinhança, que é a progressão da menor distância entre quaisquer dois nós ao longo do tempo. Através do uso dessas linhas é possível realizar diferentes análises probabilísticas.
2. **Análise da vizinhança.** A linha de tempo provê a informação necessária para caracterizar a probabilidade de transição entre distâncias quaisquer.

3.1. Geração da linha de tempo

O método proposto usa os traços de contato como entrada, organizados de forma cronológica de eventos instantâneos. Os eventos podem ser tanto o surgimento quanto o desaparecimento de um enlace entre um par de nós (i, j) no instante t. Denota-se esse tipo de evento como $e = \langle t, i, j, \texttt{UP/DOWN} \rangle$, onde `UP` e `DOWN` indicam, respectivamente, o surgimento e o desaparecimento de um enlace entre i e j.

Para um dado par de nós (i, j), uma linha de tempo consiste em uma sequência de distâncias mais curtas entre eles ao longo do tempo (Figura 4(a)). Formalmente, a linha de tempo é representada como uma sequência de tuplas $\langle n, i, j, t_{início}, t_{fim} \rangle$. Isso significa que entre $t_{início}$ e t_{fim}, os nós i e j estiveram a uma distância de n saltos.

Algoritmo 1: Geração da linha de tempo (LT).

Requer: $\mathcal{C}, \mathcal{N}$ // traço de contatos, número de nós
Garante: $\mathcal{N} \times (\mathcal{N}-1)$ linha de tempo (LT)
Local: $\{adj\}$ // matriz de adjacências de tamanho $\mathcal{N}^2$

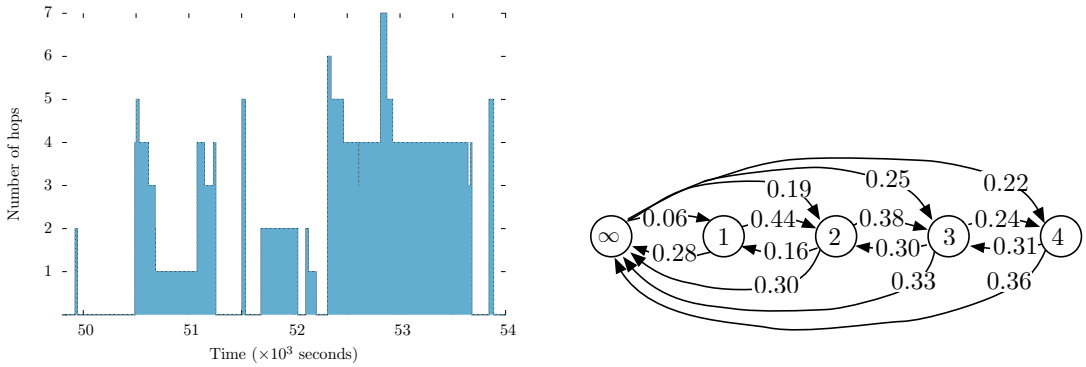
```
1  inicialização ; // todas as linhas de tempo são inicializadas com <∞, 0>
2  while size of C ≠ 0 do
3      t_atual, i, j, evento = remoção da primeira tupla de C;
4      if evento = UP then
5          adj_{i,j} = 1;
6          adj_{j,i} = 1;
7      else if evento = DOWN then
8          adj_{i,j} = 0;
9          adj_{j,i} = 0;
10     for i ← 1 to N do
11         for j ← 1 to N do
12             if i ≠ j then
13                 d_atual = caminho mais curto(i, j);
14                 if comprimento de LT_{(i,j)} = 1 then
15                     anexa (d_atual, t_atual) to LT_{(i,j)};
16                 else
17                     (d_último, t_último) = pegar última tupla de LT_{(i,j)};
18                     if d_último ≠ d_atual then
19                         anexa (d_atual, t_último) to TL_{(i,j)};
20 for i ← 1 to N do
21     for j ← 1 to N do
22         if i ≠ j then
23             formata e imprime LT_{(i,j)};
```

A geração de linhas de tempo requer como entrada o traço de contatos ($\mathcal{C}$) e o número de nós na rede ($\mathcal{N}$), conforme apresentado no Algoritmo 1. Todas as linhas de tempo são inicializadas com a tupla $\langle \infty, 0 \rangle$, indicando que os nós estão desconectados no momento 0. Todas as tuplas seguintes indicam uma mudança no estado e o momento na qual ela ocorreu. Todos os eventos nos traços são lidos e a matriz de adjacências atualizada antes de computar todas as distâncias mais curtas entre os pares de nós. Os dados obtidos são formatados e impressos nas linhas de tempo.

3.2. Análise da vizinhança

A mobilidade da vizinhança é modelada através de um processo Markoviano de tempo contínuo (*continuous time Markov process – CTMP*) para *cada* par de nós. Para um dado nó i, considere que $X_{i,j}^{s}$ seja a variável aleatória representando a distância entre os nós i e j no passo s. O CTMP é usado ao invés de um processo Markoviano simples para permitir a observação da evolução dos movimentos independentemente da amostragem de tempo. A etapa da análise da vizinhança recebe como entrada as linhas de tempo e oferece como saída as probabilidades de transição correspondentes das cadeias de vizinhança.

Estados. Os estados do CTMP dependem da escolha de κ, isto é, do tamanho da vizinhança que se queira monitorar. O número de estados é $\kappa + 1$; o primeiro estado,



(a) (b)

Figura 4. Nesta figura, são apresentadas a linha de tempo de um dos pares do traço *Unimi* (Figura 4(a)) e a movimentação média da vizinhança de um par (i, j) com $\kappa = 4$ no traço *Infocom05* (Figura 4(b)). Por simplicidade, poucas transições são apresentadas (até 4). Nessa κ-vizinhança, a probabilidade de um nó entrar em contato $\{\infty \to 1\}$ é de 6%. Já considerando os nós a uma distância de três saltos, a probabilidade para que eles estejam a dois saltos é de 30%.

representado por '∞', corresponde ao caso onde os dois nós estão desconectados. Já o estado $\{1\}$ representa o contato e os estados remanescentes $\{2, \ldots, \mathsf{M} = \kappa\}$ correspondem a uma situação de intercontato favorável. Note que é considerado cada movimento de um par de nós como um único passo. Não são considerados intervalos de tempo específicos para evitar a dependência do traço com o tempo. Assume-se que X satisfaz as propriedades da cadeia de Markov e que $X^s_{i,j}$ é independente de $X^{s-1}_{i,j}$. Essa premissa pode parecer questionável, mas a natureza dos traços leva à independência dos movimentos dos pares, como será visto na Seção 6.3.

Probabilidades de transição. A compreensão da mobilidade da vizinhança requer foco nas taxas de transição CTMP entre estados, isto é, a probabilidade de dois nós estarem a uma distância m no passo s sabendo que eles estavam a uma distância n no passo precedente: $\mathbb{P}(X^s_{i,j} = m \mid X^{s-1}_{i,j} = n)$, $m \neq n$. Para a caracterização completa do CMTP, são usadas probabilidades de transição entre estados e o tempo médio de permanência em cada um deles. Os tempos de permanência média em cada estado são dados na Tabela 1.

A Figura 4(b) mostra um exemplo da probabilidade de transição média do traço *Infocom05*, que será descrito em maiores detalhes na Seção 3.3. Algumas transições estão omitidas para aumentar a clareza da figura. Como pode ser observado, quando os nós i e j estão desconectados (∞), a probabilidade de que eles se encontrem diretamente é de 6%, enquanto a probabilidade de um intercontato favorável de três saltos é de 25%. O fluxo de trabalho proposto da movimentação da vizinhança pode ser visto na Figura 5.

3.3. Traços

A análise da proposta é baseada na movimentação da vizinhança obtida em traços de experimentos reais e sintéticos como descritos a seguir. Os parâmetros usados foram escolhidos para representar situações particulares.

- **Infocom05.** Baseado em medidas conduzidas em uma conferência de cinco dias em 2005 [Chaintreau et al. 2007], onde 41 participantes carregaram iMotes para

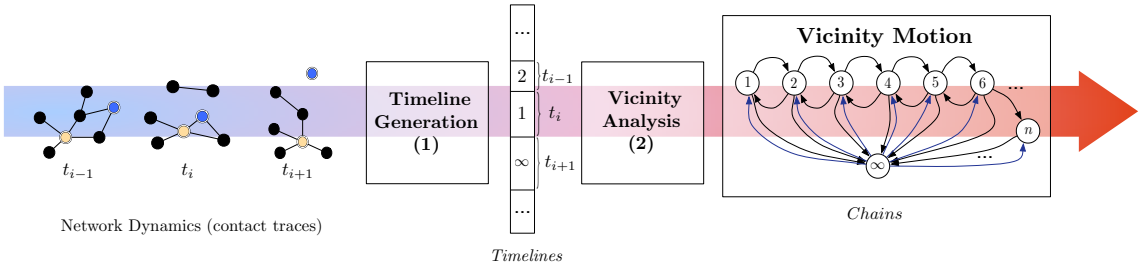


Figura 5. O fluxo de trabalho para a geração da movimentação da vizinhança. Inicia-se através da leitura dos traços de contato que descrevem a conectividade da rede ao longo do tempo. Essa entrada é processada usando o módulo de geração de linhas de tempo (1). As linhas de tempo são sequências de distâncias mais curtas entre todos os pares de nós. A etapa (2), chamada de análise da vizinhança, examina essas sequências para calcular as probabilidade de transição e as *cadeias* correspondentes da movimentação da vizinhança.

coletar informações sobre outros em um raio de 10m. O estudo deste trabalho foca em um trecho de 12 horas com maior atividade da rede. Cada iMote envia sondas a cada 120 segundos. Esse traço representa um encontro profissional.

- **Rollernet.** Conta com 62 participantes medindo a conectividade mútua entre iMotes enquanto patinavam pelas ruas de Paris em um encontro de 3 horas [Tournoux et al. 2011]. Os pesquisadores ajustaram os dispositivos para enviar sondas a cada 30 segundos. Esse traço representa um cenário esportivo.
- **Unimi.** É um traço capturado por estudantes, professores e funcionários da Universidade de Milão em 2008 [Gaito et al. 2009]. Foram envolvidos 48 pessoas com dispositivos especiais que enviavam uma sonda por segundo para a vizinhança. Esse traço representa um cenário escolar e de trabalho.
- **StanfordHigh.** Conta com 789 pessoas em uma escola nos EUA carregando motes TelosB para detectar contatos até 3m de alcance [Salathé et al. 2010]. Os motes que enviavam sondas a cada 20 segundos foram fornecidos aos estudantes, professores e funcionários durante um dia inteiro. Este trabalho usa um subconjunto de 200 participantes por questões de limitação na geração da linha de tempo. O traço *StanfordHigh* possui uma configuração com maioria de adolescentes com tendência de relacionamento em grupos de interesse.
- **Sassy.** Gerado na Universidade de Saint Andrews por pesquisadores que usaram 27 T-motes para capturar contatos entre os alunos e cientistas [Parris et al. 2010]. Os T-motes enviavam sondas a cada 6,67 segundos durante 79 dias. O traço *Sassy* possui uma configuração acadêmica esparsa com longa duração.
- **RT.** É um modelo de mobilidade que corrige falhas do modelo *Random Waypoint* [Pal Chaudhuri et al. 2005]. Foram amostrados o comportamento de vinte nós seguindo esse modelo em uma superfície de $50\times60\text{m}^2$ utilizando 10m de alcance com velocidades entre 0 e 7m/s.
- **Community.** É um modelo de mobilidade baseado em comportamento social [Musolesi and Mascolo 2007]. Ele coloca nós com relações sociais em uma posição específica ao mesmo tempo, como grupos de amigos. São simulados 50 nós com um raio de 10m em um plano de $1.500\times2.500\text{m}^2$ durante 9 horas.

Tabela 1. Tempo médio de permanência em cada estado em segundos.

Traço	Estado								
	∞	1	2	3	4	5	6	7	≥ 8
Infocom05	2.029	399	296	224	175	131	154	212	229
Rollernet	167	51	74	86	102	117	127	142	166
Sassy	157.504	2.315	53.871	1	–	–	–	–	–
StanfordHigh	2.972	1	1	0	–	–	–	–	–
Unimi	18.041	1.300	447	305	214	155	208	74	35
Community	5.210	108	120	114	118	326	1.330	295	15
RT	203	221	117	82	61	47	40	34	35

4. Cenário *vs.* κ

Um requisito importante antes de abordar a movimentação da vizinhança é entender as características gerais da rede para saber qual o valor mais apropriado de κ. Neste trabalho, não se pretende ajustar um valor específico para cada traço, mas apenas estimar um valor genérico que permita alcançar conclusões não equivocadas. Na prática, deseja-se encontrar um κ específico que sirva bem no cenário e que tenha uma boa relação entre sobrecarga e conhecimento da vizinhança.

A primeira análise tem como objetivo encontrar um valor de κ que seja suficientemente grande para cobrir todos os componentes conectados dos nós. Para cada par de nós, é computado a distância max-min do traço, que é diâmetro do componente conectado. No traço *Infocom05*, observa-se que a maior proporção de pares (aproximadamente 75%) tem entre 7 e 9 saltos de distância no máximo, enquanto apenas 13% têm entre 4 e 6 saltos, e 9% têm entre 10 e 12 saltos. Poucos pares têm um caminho entre 1 e 3 saltos de distância. O traço *Unimi* apresenta a maior proporção de nós com caminhos com distância máxima entre 7 e 9 saltos.

Esses resultados confirmam que é importante discriminar o comportamento dos nós, mesmo considerando um alto nível de movimentação de sua vizinhança. O restante deste trabalho foca em cadeias de vizinhança com estados até $\{7, 8, 9\}$, já que elas representam a maior parte das situações observadas.

5. Cadeias de Vizinhança

5.1. Tempo médio de permanência em cada estado

A Tabela 1 apresenta a duração média de permanência no estado κ em segundos. Nos traços *RT* e *Unimi*, é observada uma redução gradual nas durações. Por outro lado, o traço *Rollernet* tem uma tendência de crescimento, enquanto o *Infocom05* e o *Community* têm comportamentos combinados. O status específico do *Rollernet* como um esporte dinâmico pode explicar o aumento dos valores. Pequenas distâncias têm uma pequena duração por causa da conectividade variável e da dinâmica da configuração. Já as distâncias mais longas são absorvidas pela multidão (note que não são discriminadas as mudanças de caminhos se elas forem do mesmo comprimento).

5.2. Distribuições estacionárias

Observa-se em todos os traços que a cadeia de Markov embutida (*embedded Markov chain* - EMC) é irredutível. Logo, uma distribuição estacionária existe em todos os

Tabela 2. Distribuição estacionária em porcentagem.

Traço	Estado								
	∞	1	2	3	4	5	6	7	≥ 8
Infocom05	25,3	5,5	15,4	20,0	16,0	9,7	5,1	2,2	0,8
Rollernet	28,2	2,3	7,7	11,5	12,5	11,5	9,5	7,3	9,5
Sassy	49,2	34,8	15,5	0,5	0,0	0,0	0,0	0,0	0,0
StanfordHigh	45,0	48,0	6,9	0,1	0,0	0,0	0,0	0,0	0,0
Unimi	35,0	9,0	14,0	15,0	12,0	8,0	4,0	2,0	1,0
Community	24,6	9,7	26,3	26,1	10,1	2,5	0,4	0,0	0,3
RT	29,1	5,0	10,6	14,1	14,3	11,5	7,7	4,5	3,2

casos. A Tabela 2 apresenta as distribuições estacionárias quando $\kappa \geq 8$. No traço *Infocom05*, há 25,3% de chance do nó procurado não pertencer à κ-vinzinhança em questão; 5,5% de chance do nó estar em contato; 15,4% dele estar a dois saltos; 20% a três saltos e assim por diante. Note que com a observação da vizinhança para $\kappa = 4$, tem-se 77% de chances de encontrar o nó procurado. Tal conhecimento posterior é útil para avaliar a probabilidade de encontrar um nó rapidamente na chegada ou até mesmo quantificar os limiares de sondagem para manter os custos de manutenção reduzidos.

5.3. Cadeias curtas e estendidas

Existem dois tipos de cadeias observadas: as estendidas que podem variar até estados com dez ou doze saltos e as curtas com movimentos de apenas um ou dois saltos.

Cadeias curtas. As cadeias curtas reforçam a premissa anterior na qual os nós ou estão em contato ou desconectados; a diferença nesse caso é que elas podem ter até dois saltos de distância. Nota-se tal configuração para dois dos traços usados: *Sassy* e *StanfordHigh*. A cadeia observada consiste em estados $\{\infty, 1, 2\}$. Como resultado, tal configuração não tira nenhum ou quase nenhum proveito dos intercontatos favoráveis. Na maior parte do tempo, quando se detecta um nó, seu próximo movimento é quase sempre de desaparecimento da vizinhança. Protocolos oportunísticos devem também considerar esses comportamentos quando necessário.

Cadeias estendidas. Os traços *Infocom05*, *Community*, *RT*, *Rollernet* e *Unimi* mostram cadeias estendidas de vizinhança. As cadeias estendidas possuem maior potencial em mudanças de estados. Alguns com até doze ou mais saltos. As cadeias estendidas permitem estados de interconexão favorável e, portanto, maiores possibilidade de transmissões fim-a-fim. As cadeias estendidas podem também exibir uma larga quantidade de movimentos internos. Será visto que há três tipos de movimentos que dominam as tendências. Com apenas poucos padrões de movimento, será demonstrado que é possível abstrair muitos dos movimentos dos nós.

6. Padrões em Cadeias Estendidas

Os traços que contêm mais cadeias estendidas oferecem mais possibilidades de transições para o próximo salto. Como consequência, por economia de espaço, não serão apresentados de agora em diante os resultados obtidos com os traços *Sassy* e *StanfordHigh*. Nos traços analisados, são observados três tipos principais de transições, chamados de *nascimento*, de *morte*, e de *movimentos sequenciais*.

Tabela 3. Valores de nascimento.

	Traço				
	Infocom05	Unimi	Community	RT	Rollernet
Estado	3	2	2	4	4
Probabilidade	0,25	0,22	0,37	0,19	0,15
Acumulado	0,50	0,40	0,52	0,59	0,44

6.1. Nascimento na κ-vizinhança

O fenômeno de nascimento é caracterizado pelo surgimento de um nó na κ-vizinhança após um período de desconexão. O conhecimento desse fenômeno permite que um nó ou um protocolo saiba em qual distância outro nó pode aparecer. Considere que o nó i queira enviar uma mensagem ao nó j no traço *Infocom05*. Se j estiver atualmente fora da κ-vizinhança do nó i, ele não precisará confiar em um encaminhamento totalmente oportunístico. Dados os valores estacionários calculados na Figura 4(b), sabe-se que j irá surgir com uma probabilidade de 25% a uma distância de três saltos.

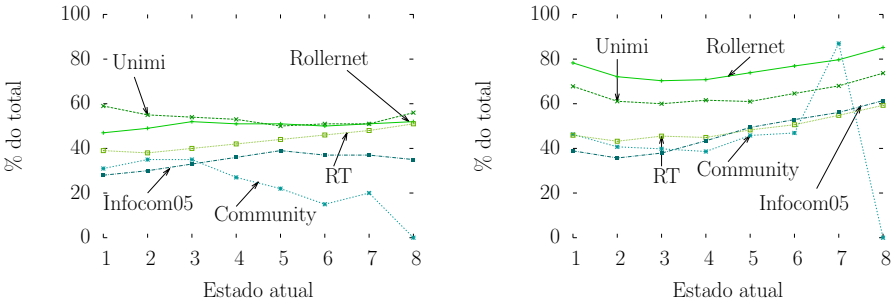
Na Tabela 3, são apresentados os valores relacionados ao evento nascimento nos traços analisados. A linha "estado" descreve o estado com a maior probabilidade de nascimento, enquanto a linha "probabilidade" indica a probabilidade de nascimento desse estado em particular. A linha "acumulado" representa a probabilidade de nascimento acumulada em qualquer distância dentro da κ-vizinhança para $\kappa =$"estado" (isto é, para qualquer valor entre 1 e "estado"). Nota-se que o estado com maior probabilidade de nascimento pode cobrir de 40 até 59% das chegadas (somente uma distância de dois ou três saltos nos traços *Unimi*, *Community* e *Infocom05*). Para todos os traços, a maior probabilidade de nascimento está no conjunto $\{1, 2, 3, 4\}$. Se for escolhido estender os limites para envio de sondas somente até o estado 4, a probabilidade acumulada se torna 50 até 70%, dependendo dos traços. Logo, o envio de sondas até 4-vizinhança é suficiente para alcançar a maioria dos padrões de nascimento no entorno de um nó [Phe-Neau et al. 2011b].

6.2. Morte na κ-vizinhança

Em oposição ao nascimento para padrões de chegada, define-se a morte como o fenômeno de saída dos nós da κ-vizinhança. Os traços são analisados em dois diferentes aspectos: a proporção de mortes relacionadas à cadeia inteira (absoluta) e a comparação com somente movimentos naturais (o que exclui transições entre estados não consecutivos exceto para ∞).

Na Figura 6(a), a evolução das probabilidades é mostrada para os diferentes estados da cadeia. Todos os traços, exceto o *Community*, têm taxas de morte aproximadamente estáveis, cuja máxima variação absoluta é 12%. Na Figura 6(b), são mostrados os resultados no caso dos movimentos naturais. Observa-se um fenômeno interessante: todos os traços possuem a mesma evolução na taxa de morte relativa. Relacionada a movimentos naturais, a proporção de movimentos de morte tem um padrão similar (decréscimo leve seguido de acréscimo leve). A principal diferença vem a ser os valores iniciais no eixo y para cada traço.

Os eventos de nascimento e morte e os sequenciais, apresentados a seguir, representam uma grande parte dos movimentos identificados.



(a) Cadeias completas. (b) Movimentos naturais.

Figura 6. Proporção de mortes.

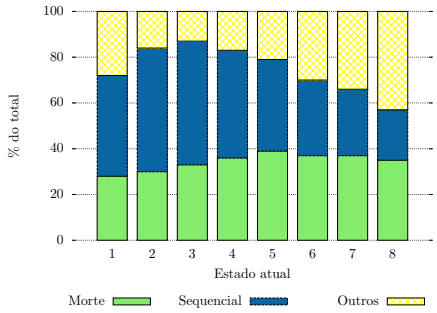
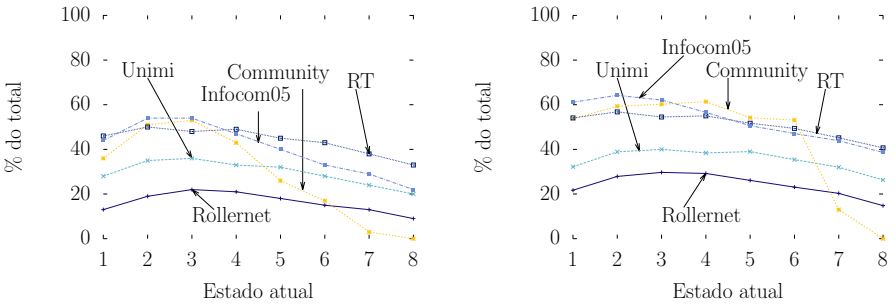


Figura 7. A movimentação da vizinhança (morte, sequencial e errático) para o traço *Infocom05*.

6.3. Movimentos sequenciais

Define-se como um movimento sequencial para dois nós, o processo de se aproximar ou se distanciar um do outro em estados adjacentes da cadeia: quando os nós (i, j) estão a uma distância de n saltos (para $n > 1$), eles se movem sequencialmente para mais próximo se estiverem a exatamente $n - 1$ saltos no passo seguinte. Em oposição, eles sequencialmente se distanciariam se eles estivessem a $n + 1$ saltos.

Observa-se que uma parte não desprezível de movimentos da vizinhança surge de comportamentos sequenciais. Nos traços *Unimi* e *Infocom05*, considerando que os nós permaneçam na κ-vizinhança, os movimentos sequenciais representam entre 50 e 80%



(a) Cadeias completas. (b) Movimentos naturais.

Figura 8. Proporção de movimentos sequenciais.

dos movimentos. Outra observação é que quanto maior a distância do par de nós, maior é a proporção de movimentos erráticos (movimentos que não são nem nascimento, nem morte e nem sequenciais). Entretanto, movimentos sequenciais continuam a predominar. Na Figura 7, é apresentada a proporção de movimentos sequenciais, nascimento e erráticos dentre todos os movimentos da vizinhança. Os movimentos erráticos aumentam com a distância entre os nós enquanto os de morte continuam estacionários em torno de 30%. A presença desse tipo de movimento está de acordo com a premissa de independência de movimentos dos pares. Já os movimentos sequenciais estão concentrados na 4-vizinhança. O efeito dos padrões sequenciais influencia menos em distâncias maiores devido às perturbações do ambiente.

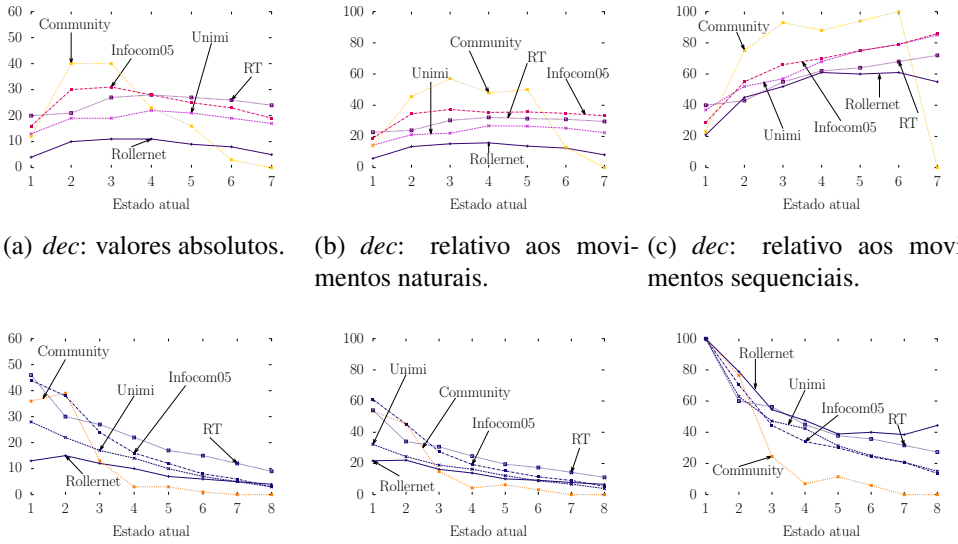
A análise em maiores detalhes dos movimentos sequenciais permite ainda dividilos em mais duas subclasses: movimentos *incrementais (inc)* e *decrementais (dec)*. Enquanto os movimentos *inc* consistem em movimentos onde as distâncias aumentam para o estado imediatamente mais alto, os *dec* consistem no oposto, ou seja, as distâncias diminuem para o estado imediatamente mais baixo. Similarmente ao realizado para as taxas de morte, investiga-se os movimentos *inc* e *dec* usando escalas diferentes: absoluta, relativa para movimentos naturais e relativamente à proporção de movimentos sequenciais. Sobre os movimentos *dec*, todos os traços que mostram cadeias estendidas (*Infocom05*, *Rollernet*, *Unimi* e *RT*) apresentam um aumento lento até a distância de dois saltos, seguida por um suave decréscimo (Figura 8(a)). A proporção de movimentos sequenciais é então simples de predizer. A mesma dedução pode ser feita em geral sobre a proporção de movimentos sequenciais relacionados aos movimentos naturais (Figura 8(b)).

Aumentando a granularidade da observação sobre os movimentos sequenciais, é possível entender quais os padrões de movimento são predominantes: aproximação (*inc*) ou afastamento (*dec*))? Na Figura 9, as proporções *dec* e *inc* são apresentadas. Os valores absolutos de *dec* e *inc* (Figuras 9(a) e 9(d)) mostram que o padrão de aproximação tem uma distribuição estacionária em todos os traços, exceto o *Community*. O *dec* não varia muito em torno do seu valor inicial enquanto o padrão de distanciamento (*inc*) rapidamente diminui nos estados mais altos. Isso pode ser explicado pela conectividade variável em distâncias mais longas que resultam em morte ao invés de movimentos de distanciamento.

Considerando os movimentos naturais e sequenciais, a proporção de movimentos *dec* e *inc* entre os movimentos naturais é diferente em termos de valores, mas possui a mesma evolução que as absolutas. O *dec* mantém valores quase estacionários enquanto o *inc* mostra rápida redução nas Figuras 9(b) e 9(e), respectivamente. A comparação dos movimentos *dec* e *inc*, vistos nas Figuras 9(c) e 9(f), com os movimentos sequenciais mostra padrões claros. Para todos os traços, exceto o *Community*, a proporção de *dec* possui um crescimento aproximadamente linear enquanto o *inc* decresce linearmente. Como o *dec* e o *inc* representam uma partição completa dos movimentos sequenciais, as observações parecem lógicas. O crescimento dos movimentos *dec* tem origem no declínio do *inc* no caso natural, atribuindo uma maior proporção de *dec* no caso sequencial.

6.4. Traço *Community*: caso excepcional

O traço *Community* não se encaixa nas principais observações. Isso pode ser uma consequência direta da natureza do processo de geração, já que o objetivo principal do



(a) *dec*: valores absolutos. (b) *dec*: relativo aos movimentos naturais. (c) *dec*: relativo aos movimentos sequenciais.

(d) *inc*: valores absolutos. (e) *inc*: relativo aos movimentos naturais. (f) *inc*: relativo aos movimentos sequenciais.

Figura 9. Proporções médias de movimentos *inc* e *dec*. Os movimentos *inc* indicam movimentos de distanciamento incremental e os *dec* indicam o oposto. O eixo-x apresenta os estados atuais dos nós envolvidos e o eixo-y a porcentagem de movimentos da vizinhança.

modelo de mobilidade é gerar padrões de movimento onde grupos de pessoas estejam no mesmo lugar ao mesmo tempo. Apesar de ser um esforço válido e resultar em padrões plausíveis, ele gera movimentos não naturais. O traço *Community* é mantido mesmo assim para mostrar o seu comportamento original.

7. Conclusão

Este trabalho modela a vizinhança de um nó usando a noção de κ-vizinhança assim como propõe um fluxo de trabalho para compreensão do seu comportamento. Esse fluxo de trabalho gera informações como a *linha de tempo* e as *probabilidades de transição*. As linhas de tempo permitem a análise da dinâmica das distâncias entre os pares de nós enquanto as probabilidades de transição detalham como os nós se movimentam uns em relação aos outros. Este estudo apresentou dois tipos principais de cadeias de vizinhança: estendidas e curtas. Cada tipo foi discriminado de acordo com os estados alcançáveis. Além das cadeias identificadas, a predominância de apenas poucos tipos de movimentos na rede foi identificada. Esses movimentos, chamados de nascimento, morte e sequencial cobrem até 87% de todos os padrões analisados. Os padrões de movimentação de vizinhança ajudam a entender como é o comportamento da vizinhança, que é fundamental em redes oportunistas. Adicionalmente, os padrões de movimentação ainda apresentam um padrão de linha de tempo que pode ser útil na previsão dos próximos movimentos dos nós assim como pode levar a um novo tipo de geração realística de conectividade.

Como trabalho futuro, planeja-se aproveitar o modelo proposto para desenvolver um gerador de conectividade. O gerador vai integrar as análise de padrões de vizinhança propostas e permitir múltiplos tipos de geração de traços de contato. Tal gerador também permitirá a análise de vizinhança de traços gerados por usuários, seguido pela geração de

traços com características similares de vários tamanhos.

Referências

Calegari, R., Musolesi, M., Raimondi, F., and Mascolo, C. (2007). CTG: A Connectivity Trace Generator for Testing the Performance of Opportunistic Mobile Systems. In *ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Dubrovnik, Croatia.

Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., and Scott, J. (2007). Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620.

Conan, V., Leguay, J., and Friedman, T. (2007). Characterizing Pairwise Inter-contact Patterns in Delay Tolerant Networks. In *International Conference on Autonomic Computing and Communication Systems*, Rome, Italy.

Gaito, S., Pagani, E., and Rossi, G. P. (2009). Fine-Grained Tracking of Human Mobility in Dense Scenarios. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Rome, Italy.

Gonzalez, M. C., Hidalgo, C. A., and Barabasi, A.-L. (2008). Understanding individual human mobility patterns. *Nature*, 453(7196):779–782.

Musolesi, M. and Mascolo, C. (2007). Designing mobility models based on social network theory. *SIGMOBILE Mob. Comput. Commun. Rev.*, 11:59–70.

Pal Chaudhuri, S., Le Boudec, J.-Y., and Vojnovic, M. (2005). Perfect Simulations for Random Trip Mobility Models. In *IEEE Infocom*, Miami, Florida, USA.

Parris, I., Bigwood, G., and Henderson, T. (2010). Privacy-enhanced social network routing in opportunistic networks. In *IEEE International Conference on Pervasive Computing and Communications*, Mannheim, Germany.

Passarella, A. and Conti, M. (2011). Characterising aggregate inter-contact times in heterogeneous opportunistic networks. In *IFIP Networking*, Valencia, Spain.

Phe-Neau, T., Dias de Amorim, M., and Conan, V. (2011a). Fine-Grained Intercontact Characterization in Disruption-Tolerant Networks. In *IEEE Symposium on Computers and Communication*, Kerkyra, Greece.

Phe-Neau, T., Dias de Amorim, M., and Conan, V. (2011b). Using neighborhood beyond one hop in disruption-tolerant networks. `http://arxiv.org/abs/1111.0882v1`.

Phe-Neau, T., Dias de Amorim, M., and Conan, V. (2012). Vicinity-based DTN Characterization. In *ACM MobiOpp*, Zurich, Switzerland.

Salathé, M., Kazandjieva, M., Lee, J. W., Levis, P., Feldman, M. W., and Jones, J. H. (2010). A high-resolution human contact network for infectious disease transmission. *PNAS*, 107(50):pp. 22020–22025.

Tournoux, P.-U., Leguay, J., Benbadis, F., Whitbeck, J., Conan, V., and de Amorim, M. D. (2011). Density-aware routing in highly dynamic DTNs: The rollernet case. *IEEE Transactions on Mobile Computing*, 10:1755–1768.

Uma Estratégia de Tentativas de *Handover* Vertical em Grupo

Nivia Cruz Quental, Paulo André da S. Gonçalves

Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE)
50.740-540 – Recife – PE – Brasil

{ncq, pasg}@cin.ufpe.br

Abstract. *Interoperability has been a keyword in the context of applications running over wireless networks in order to guarantee good user experience. From voice calls to real-time conferences, mobility management makes possible service continuity in case of changing the coverage area. The rising of heterogeneous networks brings vertical handover as the key process in related research. Current challenges include proposing new handover schemes or adapting the classic existing ones. In this paper, we present a handover try strategy for the scheme described in [Lee and Cho 2011]. That scheme intends to handle vertical group handovers based on a given threshold for the handover block probability. Results show that the proposed strategy reduces the total handover latency, when compared with the referred scheme while maintains equivalent handover block probability.*

Resumo. *No contexto de aplicativos executados sobre arquiteturas de redes sem fio, interoperabilidade tem sido a palavra-chave para a garantia de uma boa experiência de usuário. Desde chamadas de voz até conferências em tempo real, a continuidade do serviço diante da mudança de área de cobertura de uma tecnologia é garantida por mecanismos de gerência de mobilidade. Com o crescimento das arquiteturas heterogêneas, o handover vertical é o processo chave dentro das pesquisas relacionadas à área. Os desafios relacionados consistem em adaptar esquemas de handover já conhecidos na literatura ou na criação de novas técnicas. Neste trabalho, é apresentada uma estratégia de tentativas de handover para o esquema proposto em [Lee and Cho 2011]. O referido esquema foi projetado para lidar com handovers verticais em grupo com base em uma probabilidade de bloqueio máxima estabelecida. Os resultados mostram que a estratégia proposta reduz a latência total de handover do esquema de referência enquanto mantém uma probabilidade equivalente de bloqueio de handover.*

1. Introdução

Handover (ou *handoff*) é um processo de transferência de uma estação móvel (*Mobile Station* - MS) de um canal para outro atribuído a uma estação-base (*Base Station* - BS) alvo [Zekri et al. 2012]. Na maioria das tecnologias sem fio, esse processo ocorre quando a potência recebida pela MS decai abaixo de um limiar ou quando a MS se distancia excessivamente da sua BS atual. Dependendo da implementação, o *handover* pode ser iniciado pela própria MS ou pela rede onde se encontra. O principal objetivo do *handover* é manter a continuidade do serviço ao mesmo tempo que efetua mudanças de rede de maneira

transparente para o usuário. Recentemente, o conceito de *handover* não tem sido mais apenas ligado à continuidade de uma chamada telefônica, mas também à continuidade de sessões de *streaming*, à manutenção de QoS e do acesso à Internet.

Essa extensão do conceito de *handover* ocorre devido à popularização dos *tablets* e *smartphones*, os quais têm permitido a experiência coletiva de usuários que compartilham uma mesma área de cobertura. Recentemente, o cenário de mobilidade em variadas velocidades com aplicativos em uso tem sido cada vez mais comum. Assim, cenários como trens e ônibus com pessoas utilizando os seus dispositivos levam à reflexão quanto ao problema do *handover* em grupo (*Group Handover* - GHO) [Jeong et al. 2012]. O GHO se faz necessário quando várias MSs entram em uma mesma nova área de cobertura simultaneamente, sem necessariamente estarem cientes das demais. A escolha de uma BS-alvo em um esquema de GHO precisa garantir o balanceamento de carga e pode considerar adicionalmente outros critérios, como economia de energia, classes de serviço, entre outros [Lee and Cho 2011].

A chegada dos *tablets* e *smartphones* ao mercado também trouxe ao usuário o suporte a diversas tecnologias sem fio como 3G, WiMax, Wi-Fi e LTE em um mesmo aparelho. Isso permite extrapolar o conceito de transferência de um canal para outro em uma mesma tecnologia para a transferência entre tecnologias diferentes. Nesse novo cenário, a necessidade de se manter a sessão do usuário permanece, encorajando novas pesquisas na área de *handover* vertical (*Vertical Handover* - VHO)[Shen and Zeng 2008] [Stevens-Navarro et al. 2008] [Lee et al. 2009] [Taniuchi et al. 2009] [Cicconetti et al. 2010] [Andersson et al. 2010] [Kim and Kim 2011] [Zekri et al. 2012]. Quando se alia a esse contexto um cenário de múltiplos usuários em uma mesma área de cobertura, define-se o problema de *handover* vertical em grupo (*Group Vertical Handover - GVHO)* [Cai and Liu 2008]. Em geral, o processo de *handover* se divide em três etapas: descoberta, decisão e execução [Zekri et al. 2012]. O presente artigo está particularmente interessado no processo de decisão, pois ele exerce maior influência sobre o desempenho do GVHO. Diversas soluções tem sido propostas nesse âmbito, entre as quais, as baseadas em entidades centralizadoras [Niyato and Hossain 2009], algoritmos distribuídos [Lei et al. 2010], atrasos aleatórios [Cai and Liu 2008], aprendizado por reforço [Niyato and Hossain 2009], teoria de jogos [Cai and Liu 2008] e problemas de otimização [Lee and Cho 2011].

Dentre os trabalhos previamente citados, em [Lee and Cho 2011], o objetivo é o de modelar a decisão de GVHO como um problema de otimização por meio da minimização da equação da latência sob a condição de se manter a probabilidade de bloqueio igual ou abaixo de um limiar. A probabilidade de bloqueio neste contexto é a probabilidade com que uma MS tem a sua requisição de *handover* rejeitada pela BS-alvo. Entretanto, apesar do referido esquema utilizar uma abordagem adequada, ele prevê uma estratégia de tentativas de *handover* bastante rudimentar, que não permite maiores variações na ocupação de *slots* de tempo por parte das MSs que precisam tentar o *handover* mais de uma vez. Adicionalmente, a estratégia de tentativas de *handover* adotada no referido trabalho faz com que a latência cresça de forma cada vez mais acentuada à medida que se fazem necessárias mais tentativas. Dessa forma, a estratégia de tentativas de *handover* usada em [Lee and Cho 2011] não tira proveito do potencial do próprio esquema.

Este artigo propõe uma estratégia de tentativas de *handover* baseada em *backoff exponencial* para reduzir a latência média em relação aos resultados obtidos em [Lee and Cho 2011]. Adicionalmente, a estratégia proposta permite suavizar a curva de crescimento da latência com o número de MSs, conseguindo ainda manter o critério de limitação da probabilidade de bloqueio. Assim, é possível melhorar os resultados, alocando as tentativas de *handover* de forma otimizada e explorando as informações providas pelo referido esquema.

Este artigo está organizado como segue. A Seção 2 apresenta os conceitos relacionados ao *handover* vertical em grupo. Os trabalhos relacionados são apresentados na Seção 3. A Seção 4 mostra o esquema de referência descrito em [Lee and Cho 2011], bem como uma análise crítica. A estratégia proposta e os resultados da avaliação de desempenho comparando as implementações são apresentados na Seção 5. Finalmente, a Seção 6 apresenta as conclusões do artigo.

2. *Handover* Vertical em Grupo (Group Vertical Handover - GVHO)

A evolução tecnológica e o barateamento dos *gadgets* recentemente têm sido uma das forças motrizes para a pesquisa em gerência de mobilidade. O cenário de dezenas de pessoas manuseando dispositivos capazes de se conectar a diferentes tecnologias em um mesmo espaço público cria um novo desafio para a manutenção de conexões. As tecnologias sem fio precisam agora considerar dois novos aspectos: o alto número de usuários entrando em uma mesma área de cobertura e a variedade de tecnologias disponíveis. O primeiro aspecto é a principal preocupação das pesquisas de GHO [Jeong et al. 2012]. No segundo aspecto, a continuidade de serviço e a manutenção de contexto são obtidas por meio do VHO [Zekri et al. 2012]. A forma de se tratar as decisões coletivas, levando em consideração as diferentes tecnologias disponíveis, constitui objeto de estudo do GVHO [Lee et al. 2009].

A Figura 1 ilustra um cenário onde dispositivos móveis de diferentes plataformas se aproximam de uma mesma rede com uma tecnologia diferente da atual enquanto se movem simultaneamente. Os critérios utilizados para realização de mudança de área podem ser os mais diversos, como a largura de banda disponível nas redes candidatas, Qualidade de Serviço esperada, consumo de bateria, entre outros. O tipo de tráfego (voz ou dados) utilizado no momento é um fator determinante na adoção desses critérios.

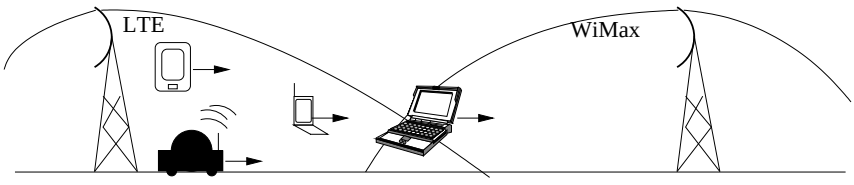


Figura 1. Um cenário de GVHO.

Quando não são consideradas as consequências advindas da escolha de uma tecnologia em favor de outra ou quando se restringe a decisão de *handover* à preferência individual, os resultados podem ser desastrosos do ponto de vista de desempenho. Decisões equivocadas podem levar várias estações móveis a optar pela mesma rede ou uma rede que não seja adequada às suas necessidades, causando uma alocação ineficiente de recursos,

além de prejudicar o desempenho dos demais usuários. Assim, é atualmente uma necessidade crítica a proposta de algoritmos eficientes e eficazes para gerenciamento de conexão, decisões de *handover* e alocação ótima de recursos [Lee et al. 2009]. Essas propostas podem envolver as três etapas do *handover* enumeradas a seguir [Zekri et al. 2012]:

- *Descoberta* - Momento em que se faz a descoberta de serviços e a coleta de informações do estado da rede. Nessa etapa também se utiliza um critério específico para se detectar a necessidade de *handover*, como potência, taxa de transmissão, carga sobre uma BS, nível de bateria, entre outros. O padrão IEEE 802.21 [Taniuchi et al. 2009], o qual descreve o *handover* independente de mídia (*Media Independent Handover* - MIH), determina os principais requisitos nessa etapa de descoberta. Os detalhes das demais etapas devem ser providos pelos grupos de interesse que adotarem o padrão;
- *Decisão* - Durante o processo de decisão, as redes disponíveis são avaliadas com base nos dados coletados na etapa anterior. Esta é a etapa de maior interesse deste artigo, uma vez que a escolha por uma técnica de decisão que utilize os dados coletados de forma ótima e a estratégia adotada para verificação da necessidade de adiamento do *handover* têm consequências diretas no desempenho de toda a operação;
- Execução - Nessa etapa, mensagens de controle são trocadas entre MS, BS-alvo e BS atual para realizar a mudança de canal. A execução deve causar o mínimo de interrupção possível na comunicação, de modo a parecer imperceptível ao usuário. Essa etapa é fortemente dependente das tecnologias envolvidas.

As principais abordagens de decisão de *handover* adotadas nas pesquisas de GVHO envolvem:

- *Uso de uma entidade centralizadora* [Niyato and Hossain 2009] - A gerência do GVHO é responsabilidade de uma *relay station*, retirando a complexidade das estações móveis, além de diminuir o grau de incerteza e garantir um desempenho melhor que a abordagem descentralizada. Um ponto negativo é a pouca tolerância a falhas;
- *Algoritmos distribuídos* [Lei et al. 2010] - Utiliza técnicas conhecidas de paralelismo e sincronização. São geralmente de fácil compreensão, porém limitados quanto à adaptabilidade a novos cenários;
- *Atrasos aleatórios* [Cai and Liu 2008] - Estações móveis optam por tentar o *handover* após um intervalo de tempo aleatório, evitando que tentativas simultâneas de *handover* ocorram. Trata-se de um subtipo de algoritmo distribuído muito empregado nas pesquisas da área;
- *Aprendizado por reforço* [Niyato and Hossain 2009] - O emprego de técnicas de Inteligência Artificial (IA) permite que as estações móveis aprendam mais sobre a rede à medida que realizam tentativas de *handover*. Esta abordagem não requer interação com outros usuários, porém pode gerar problemas de desempenho;
- *Teoria de jogos* [Cai and Liu 2008] [Niyato and Hossain 2009] - Mapeia os cenários de *handover* em jogos cooperativos ou não-cooperativos, onde as estações móveis são jogadores interessados em obter o seu *payoff*. Este pode ser uma maior largura de banda, um gasto menor de energia ou maior segurança. O equilíbrio de Nash é a situação desejada, onde todas as estações móveis não

possuem mais estratégias para obter um melhor *payoff*, garantindo o equilíbrio. A principal vantagem é o mapeamento quase perfeito de um cenário de GVHO em modelos competitivos da Teoria de Jogos. Por outro lado, nem sempre é possível a consideração de parâmetros adicionais no modelo;

- *Derivação de problemas de otimização* [Lee and Cho 2011] - A decisão é modelada com equações matemáticas acompanhadas de uma condição pré-determinada. A partir daí, o problema é solucionado para encontrar o valor ideal para as variáveis dessas equações. Requerendo uma modelagem mais complexa, a derivação para problemas de otimização é mais flexível que a abordagem de Teoria de Jogos.

Em qualquer abordagem para decisão de GVHO, a MS ou sua BS atual podem determinar se é possível requisitar o *handover* em um dado instante ou se essa requisição deve ser adiada para evitar o bloqueio de *handover* dada as condições da rede. Essa determinação é a estratégia de tentativas de *handover*, a qual tem influência direta no desempenho total do esquema adotado. Assim, para se obter melhores resultados, a estratégia de tentativas deve se utilizar das informações providas pelo próprio esquema e pela rede-alvo.

3. Trabalhos Relacionados

Em [Shan et al. 2008] é proposto o uso de uma *relay station* como entidade centralizadora que coordena o GVHO de usuários dentro de um trem levando em conta a localização e a direção na qual as MSs seguem. São avaliadas a probabilidade de bloqueio e a de interrupção do *handover* em função do número de chamadas por minuto. Os resultados são comparados a um esquema sem a inteligência na *relay station* e o trabalho conclui que há uma redução significativa nas probabilidades de bloqueio e de interrupção de *handover* com a adoção do esquema. A estratégia de tentativas é totalmente coordenada pela *relay station*. Nesse caso, fica clara a necessidade de uma entidade centralizadora e de condições especiais para que a mesma opere, o que limita os cenários de aplicação desse esquema.

Em [Cai and Liu 2008] são propostos três algoritmos descentralizados de GVHO: o primeiro é baseado em equilíbrio de Nash, onde a lógica de tentativas reside na estratégia de cada jogador; o segundo se baseia em atrasos aleatórios com variação em um intervalo constante ao se fazer requisições de *handover*, tendo nesse caso uma estratégia de tentativas mais simples; o terceiro é uma versão modificada do anterior considerando a latência do *handover* como base do cálculo, o que torna a estratégia de tentativas mais refinada. As avaliações de desempenho mostram resultados semelhantes entre as abordagens em relação à latência. Apesar da contribuição apresentada nos resultados, não foi levado em consideração que implicações o esquema teria em relação à probabilidade de bloqueio, não sendo possível garantir um limiar tolerável.

Em [Niyato and Hossain 2009], um modelo baseado em jogos evolucionários é proposto. São consideradas duas abordagens: a primeira com o uso de uma entidade central para fornecer informações de vizinhança, sendo essa entidade quem controla a estratégia de tentativas; a segunda é uma abordagem descentralizada que utiliza aprendizado por reforço, permitindo que as MSs infiram o estado da rede para escolha de BSs. Na segunda abordagem, a estratégia de tentativas de *handover* é gerenciada pela

experiência adquirida pela própria MS. A porcentagem de MSs que escolhem a mesma BS foi a métrica escolhida. Conclui-se que cada abordagem possui vantagens dependendo do cenário onde é empregada. Apesar da contribuição com a avaliação do balanceamento de carga, a implicação na latência como consequência da escolha entre uma das abordagens propostas não é apresentada.

Em [Lei et al. 2010], três esquemas para resolver o problema de GVHO são comparados: o primeiro procura dividir as requisições simultâneas por meio de atrasos aleatórios, tendo assim uma estratégia de tentativas bastante simples; no segundo esquema, as MSs selecionam as BSs de acordo com uma probabilidade pré-definida, sendo a estratégia de tentativas baseada nesta probabilidade; no terceiro, a rede coleta informações do estado da rede e gerencia o *handover*, sendo a mesma a responsável pela estratégia de tentativas. As métricas avaliadas foram a latência, a taxa de perda de pacotes e a taxa de rejeição. Os resultados mostraram que a terceira abordagem obteve os melhores resultados na maioria dos cenários avaliados. Entretanto, observa-se que essa solução mais eficiente depende da implementação em uma infraestrutura da tecnologia de rede de acesso.

Em [Lee and Cho 2011], o foco é a proposta de um esquema de GVHO baseado na solução de um problema de otimização com o objetivo de minimizar a latência limitando a probabilidade de bloqueio de *handover*. Além da avaliação da latência, o destaque do referido trabalho é a proposta de uma solução que não requer a presença de uma *relay station* para coordenar o GVHO. A solução também provê balanceamento de carga e prevê a convivência com dispositivos que não suportem o esquema. Contudo, a estratégia de tentativas de *handover* adotada é bastante rudimentar, baseada em atraso constante, o que prejudica o desempenho quando se aumenta o número de MSs envolvidas. Os detalhes são apresentados na Seção 4.

O diferencial do presente artigo consiste em apresentar uma estratégia de tentativas de *handover* para redução da latência da abordagem descrita em [Lee and Cho 2011], usada como referência. Ao mesmo tempo, objetiva-se manter baixos valores mesmo com o aumento do número de MSs. Neste artigo, é apresentado como a abordagem de atrasos aleatórios baseada em *backoff exponencial* pode ser usada em conjunto com a abordagem de derivação do problema de otimização, melhorando o desempenho e mantendo as condições de probabilidade de bloqueio dentro de um limiar pré-determinado.

4. Esquema de GVHO de Referência

Esta seção descreve o esquema de GVHO tomado como referência neste artigo e faz uma avaliação crítica de seu desempenho.

4.1. Descrição do Esquema

O principal objetivo do esquema de GVHO proposto em [Lee and Cho 2011] consiste em evitar que as MSs em uma mesma área sofram bloqueio de requisições de *handover*, ou seja, que essas requisições sejam recusadas pela BS-alvo. Com essa premissa, os usuários realizam inferências para avaliar se devem requisitar o *handover* imediatamente ou se devem tentar posteriormente dependendo da probabilidade de bloqueio calculada. Esta decisão influencia na quantidade de tentativas de *handover*, aumentando ou reduzindo a latência. Assim, é estabelecido o seguinte problema de otimização para a latência total,

representada por L:

$$\textbf{Minimizar}\ L$$
$$\textbf{Sujeito a}\ P_{HoBlock}(t) \leq P_{HoBlockThreshold}\ ,$$

onde $P_{HoBlock}(t)$ é a probabilidade de bloqueio calculada em um dado instante t e $P_{HoBlockThreshold}$ é o seu valor máximo tolerável. A latência é calculada de acordo com a seguinte equação:

$$L = N_{HO}.\Delta t, \tag{1}$$

onde N_{HO} é o número de tentativas até que o usuário decida requisitar o *handover* e Δt é o tempo entre tentativas. No caso da MS decidir requisitar o *handover* na primeira tentativa, a latência total seria Δt, pois no referido trabalho é considerado que o tempo de execução de *handover* é equivalente ao valor de Δt.

A Equação (2) apresenta o cálculo de $P_{HoBlock}(t)$. Essa probabilidade está relacionada à quantidade de BSs, a largura de banda disponível em cada uma e o número de MSs que requisitam o *handover* simultaneamente. Considera-se que as informações sobre o estado das BSs podem ser obtidas por meio de um protocolo que implemente o padrão IEEE 802.21.

$$P_{HoBlock}(t) = \sum_{k=1}^{K} \sum_{i=C_k(t)}^{M-1} \frac{(i+1-C_k(t)).(M-1)!}{(i+1)!.(M-1-i)!} \times ((P_{sel}^k)^{i+1}.(1-P_{sel}^k)^{M-1-i}), \tag{2}$$

onde:

- M: é o número de MSs interessadas;
- K : é o número de BSs envolvidas e com áreas de coberturas sobrepostas;
- $C_k(t)$: a largura de banda disponível na BS_k no instante *t*, considerado no modelo um valor inteiro; cada MS requer uma unidade (1) para realizar o *handover*;
- P_{sel}^k : é a probabilidade de se selecionar a BS_k.

Utilizando a Equação (2), aplica-se a condição de KKT (Karush-Kuhn-Tucker), utilizada em problemas de otimização para determinar o valor de P_{sel}^k. Contudo, o valor de P_{sel}^k pode ser obtido de forma simplificada, através da Equação (3):

$$P_{sel}^k(t) = C_k(t) / \sum_{k=1}^{K} C_k(t). \tag{3}$$

Uma vez estabelecidos esses valores, pode-se encontrar um valor máximo para M no tempo t que garanta a condição do problema de otimização. Esse valor, denominado $M_{ótimo}(t)$, é obtido incrementando-se o seu valor em uma unidade a partir de 1, observando se o $P_{HoBlock}(t)$ permanece menor ou igual a $P_{HoBlockThreshold}$. Baseado no $M_{ótimo}(t)$, uma MS pode isoladamente decidir enviar a requisição com probabilidade $P_{HO}(t)$, dada por:

$$P_{HO}(t) = M_{ótimo}(t)/M. \tag{4}$$

Caso opte por não requisitar o *handover*, uma nova tentativa pode ser realizada após um intervalo de tempo. Assim, a MS requer um número de tentativas necessário para garantir que, quando enviar a requisição, só

terá seu *handover* bloqueado com uma probabilidade igual ou inferior a $P_{HoBlockThreshold}$. O Algoritmo (1) resume o funcionamento do esquema de referência:

Algoritmo 1: Esquema de referência de GVHO.

```
latencia = 0;
c_tentativas = 1;
Mtotal = número de nós participando do GVHO;
Mrestante = Mtotal;
while Mrestante ≤ 0 do
    escolher Mótimo em função de PHOBlock(Equação(2));
    calcular PHO de acordo com a Equação(4);
    if decisao(PHO) then
        escolher BSk com probabilidade P^k_sel (Equação(3));
        NHO = c_tentativas;
        break ;
    else
        latencia += t_tentativas(c_tentativas);
        c_tentativas++;
    end
    Mrestante = Mrestante - Mótimo
end
latencia += LHOexec;
```

onde:

- M_{total} - Número total de MSs envolvidas no GVHO. Essa informação pode ser obtida por meio de comunicação *ad hoc* entre vizinhos ou informações da BS atual, caso suporte envio de informações sobre vizinhança;
- $M_{restante}$ - Contador utilizado para verificação da finalização do algoritmo;
- `decisao()` - Função que retorna `true` com probabilidade $P_{HO}(t)$;
- L_{HOexec} - Tempo de execução do *handover*, o que equivale a Δt;
- `t_tentativas()` - Função de tempo entre tentativas. Depende do número de tentativas já realizadas. No esquema de referência, o valor é sempre igual a Δt;
- `c_tentativas` - Contador de tentativas realizadas. Quando se decide requisitar o *handover* na primeira tentativa, N_{HO} é igual a 1. Nesse caso, a latência total é L_{HOexec}, mantendo-se a fidelidade à Equação (1).

4.2. Avaliação do Esquema

As métricas avaliadas são as mesmas de [Lee and Cho 2011]: latência e probabilidade de bloqueio em função do número de MSs. O número de MSs varia entre 20 e 100, diferentemente de [Lee and Cho 2011] que varia até 65. O valor de Δt é fixado em *0,1s*. Os limiares adotados para $P_{HoBlockThreshold}$ permanecem iguais, em *0,02* e *0,05*. O número de BSs, assim como em [Lee and Cho 2011] é 3. Considera-se dois cenários:

- **Cenário 1** - todas as BSs possuem a mesma capacidade de 18 unidades de largura de banda;
- **Cenário 2** - as BSs possuem as capacidades de 5, 13 e 18 unidades, respectivamente.

Em [Lee and Cho 2011], a possibilidade de se trabalhar com diferentes disponibilidades de largura de banda por parte das BSs é usada como forma de caracterizar a rede como heterogênea. Apesar disso, em [Lee and Cho 2011] o Cenário 2, heterogêneo, é apenas utilizado na validação do simulador. O Cenário 2 também é usado no caso em que é avaliada a interferência de um *handover* individual no GVHO. A análise dessa situação está fora do escopo do presente artigo. Na presente avaliação, o Cenário 2 é incluído na situação do *handover* em grupo.

O esquema de referência é reimplementado em um simulador próprio, onde é possível estudar melhor o seu comportamento. O simulador foi escrito na linguagem C++. Os valores dos gráficos representam a média de 120 execuções calculando intervalo de confiança com nível de 99%. Nos gráficos, os intervalos de confiança aparecem imperceptíveis.

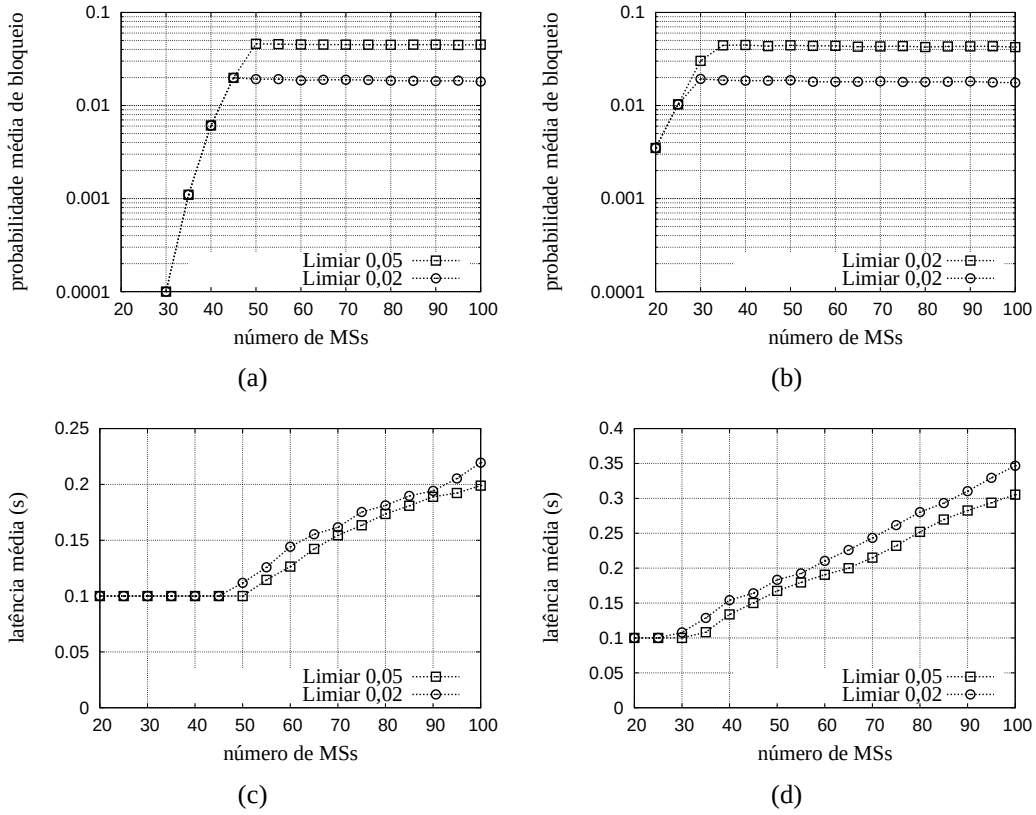


Figura 2. Avaliação do esquema de GVHO de referência em função do número de MSs: (a) Probabilidade de bloqueio no Cenário 1 (b) Probabilidade de bloqueio no Cenário 2 (c) Latência no Cenário 1 (d) Latência no Cenário 2.

A Figura 2(a) apresenta os resultados para a probabilidade de bloqueio no Cenário 1. A probabilidade de bloqueio aumenta até 45 MSs para o limiar de *0,02* e 50 MSs, para o limiar de *0,05*. A partir desses valores, o gráfico se estabiliza. Isso ocorre porque a probabilidade se aproxima dos limiares e é mantida com valor igual ou inferior a $P_{HoBlockThreshold}$ devido à condição do problema de otimização.

A Figura 2(b) apresenta os resultados para a probabilidade de bloqueio no Cenário 2. Aqui a curva da probabilidade de bloqueio se estabiliza mais antecipadamente,

a partir de 30 MSs para o limiar de *0,02* e 35 MSs, para o limiar de *0,05*. Esta antecipação se deve à quantidade de largura de banda total disponível, a qual é menor do que no Cenário 1. Com menos largura de banda, a probabilidade de bloqueio aumenta de forma mais acentuada, porém não deixa de se estabilizar, respeitando a condição estabelecida pelo problema de otimização.

A Figura 2(c) apresenta os resultados para a latência média no Cenário 1. Observa-se que para o limiar *0,02*, a latência começa a crescer a partir de 45 MSs, permanecendo maior do que no caso do limiar *0,05*, onde a latência cresce a partir de 50 MSs. Isso se deve ao fato de haver um número maior de tentativas de *handover* quando se estabelece um limiar mais baixo. Isso significa que as MSs, no caso do limiar *0,02*, tendem a esperar mais antes da requisição de *handover*, uma vez que o critério de probabilidade de bloqueio é mais rigoroso. Assim, observa-se que quanto menor o limiar, maior será o número de tentativas e, por conseguinte, maior será a latência média.

A Figura 2(d) apresenta os resultados para a latência média no Cenário 2. Da mesma forma que no Cenário 1, a curva do limiar *0,02* apresenta maiores valores na latência. Nesse caso, ela começa a aumentar a partir de 25 MSs para o limiar de *0,02* e 30 MSs para o limiar de *0,05*. Nesse cenário, onde a largura de banda disponível é menor, observa-se um crescimento acentuado da latência à medida que se aumenta o número de MSs. O aumento em si é esperado, uma vez que está associado à disputa cada vez maior por recursos. Porém, a inclinação da curva chama a atenção, mostrando que a latência dobra em torno de 60 MSs no caso do limiar *0,02*. Nessa mesma curva, em 100 MSs, tem-se uma latência de 350 ms, onde menos de um terço desse tempo é causado pela execução do *handover*. O tempo gasto em decisão é, em média, de 250 ms.

A função `t_tentativas()`, referenciada no Algoritmo (1), caracteriza a estratégia de tentativas de *handover* adotada. Em [Lee and Cho 2011], ela é constante e seu valor é igual à latência de execução L_{HOexec}. Percebe-se que o crescimento da latência possui uma relação direta com o número de tentativas, fazendo com que a latência sempre cresça em um fator constante. Conclui-se que esta função não está tirando proveito das informações providas pelo próprio esquema, perdendo a oportunidade de se obter uma latência menor mantendo a condição estabelecida no problema de otimização. Assim, observa-se uma oportunidade de se melhorar a estratégia de tentativas de *handover* de modo a otimizar o desempenho, fazendo com que o crescimento da latência em função do número de MSs seja menos acentuado.

5. Estratégia de Tentativas de *Handover* para Melhoria do Desempenho

Nesta seção, é descrita uma estratégia de tentativas de *handover* para reduzir a latência no esquema de GVHO de referência e uma comparação dos resultados é apresentada.

5.1. Estratégia de Tentativas de Handover Proposta

Para se ter uma melhor estratégia de tentativas de *handover*, é necessário propor uma forma mais adequada de se calcular o tempo entre tentativas, representado pela função `t_tentativas()` do Algoritmo(1). Assim, o algoritmo de *backoff exponencial* [Kwak et al. 2005] pode ser empregado na estimativa desse tempo. Assim, o tempo entre tentativas é calculado em função do contador `c_tentativas` e de um valor de

referência para *slot* de tempo. A Equação (5) descreve a proposta para esta nova função:

$$t\_tentativas(c\_tentativas) = \begin{cases} random[0..2^{c\_tentativas} - 1] \text{ . } slotTempo \text{ ,} \\ \quad \text{se } c\_tentativas \leq FatorLimBack \\ random[0..2^{FatorLimBack} - 1] \text{ . } slotTempo \text{ ,} \\ \quad \text{caso contrário} \end{cases} \quad (5)$$

onde $FatorLimBack$ é um fator de limite de *backoff*, e $slotTempo$ é o *slot* de tempo de referência, cuja duração é dependente da tecnologia da BS para a qual a MS pretende migrar. Esse valor pode ser obtido por meio da descoberta da tecnologia da BS, informação que pode ser recebida via MIH do padrão IEEE 802.21. A função $random$ define a escolha aleatória em um intervalo seguindo a distribuição uniforme.

A latência depende diretamente do número de tentativas realizadas que, por sua vez, depende do retorno de `decisao()`. Quando se usa o *backoff* exponencial em `t_tentativas()`, é dada uma oportunidade à MS de realizar uma nova tentativa em um tempo menor que Δt, inclusive de realizar a nova tentativa imediatamente. Quando uma MS escolhe não requisitar o *handover*, outras MSs podem realizá-lo, diminuindo a concorrência e aumentando a probabilidade de sucesso da MS que deixou de fazer a requisição antes, mas que a fará em uma próxima tentativa. Assim, a técnica de atrasos aleatórios pode ser associada à abordagem de derivação de problema de otimização de [Lee and Cho 2011], tirando maior proveito de seus recursos e reduzindo a latência.

5.2. Avaliação de Desempenho e Comparação das Estratégias

A estratégia de tentativas de *handover* proposta foi implementada no mesmo simulador utilizado na avaliação do esquema de referência. Os parâmetros de simulação são os mesmos utilizados na Seção 4.2. Os parâmetro $FatorLimBack$ é fixado em *10*. Este valor foi escolhido por meio de observações preliminares. O parâmetro $slotTempo$ é fixado em $9.10^{-6}s$. Este valor é equivalente ao tempo SIFS do padrão IEEE 802.11, o qual é usado para definir intervalos entre mensagens da camada enlace. Nesta simulação considera-se que as MSs estão prestes a entrar em uma área de cobertura seguindo o padrão IEEE 802.11.

A Figura 3(a) apresenta os resultados de probabilidade de bloqueio para o Cenário 1. Observa-se que os resultados são semelhantes entre o esquema de referência e a estratégia proposta, uma vez que esta mantém a condição de otimização.

A equivalência entre os resultados também é observada na Figura 3(b), a qual apresenta a probabilidade de bloqueio no Cenário 2. Igualmente ao esquema de referência, a estabilização da curva ocorre mais antecipadamente, uma vez que a largura de banda total disponível é menor que no Cenário 1. Mais uma vez observa-se que, em ambos os casos, a condição de otimização é respeitada.

A Figura 3(c) apresenta os resultados de latência para o Cenário 1, onde o impacto da estratégia de tentativas de *handover* pode ser observado. A curva de crescimento torna-se mais suave com a adoção da estratégia. No limiar *0,05* observa-se uma redução de 20% no caso de 65 MSs e uma redução de 28% para 100 MSs. No caso do limiar *0,02*, as curvas apresentam latência maior, tanto no esquema de referência quanto na estratégia proposta. Isso ocorre porque, dada a imposição de um limiar menor, mais tentativas são necessárias para que as MSs finalizem o *handover*, aumentando a latência média. A redução obtida

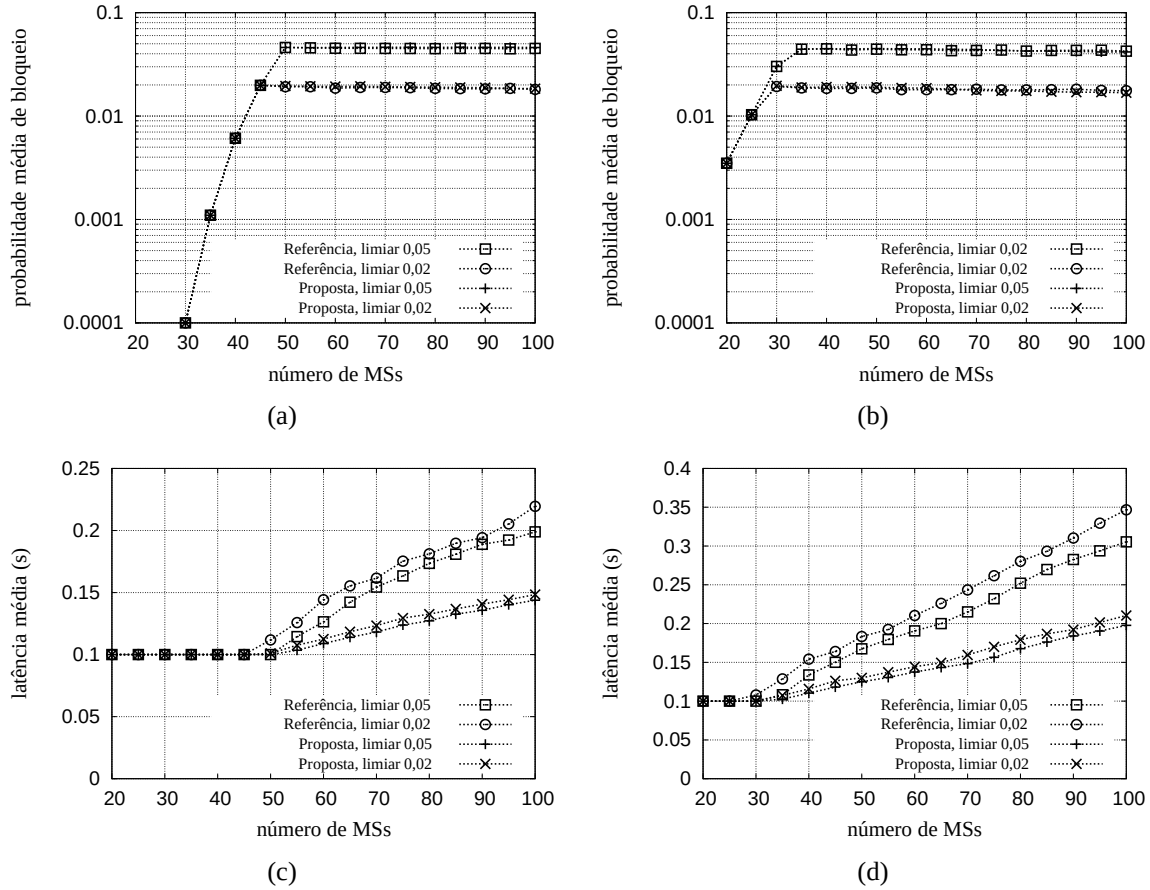


Figura 3. Avaliação do esquema de referência e da adição da estratégia de tentativas de *handover* proposta em função do número de MSs: (a) Probabilidade de bloqueio no Cenário 1 (b) Probabilidade de bloqueio no Cenário 2 (c) Latência no Cenário 1 (d) Latência no Cenário 2.

com a estratégia neste caso foi de 24% para 65 MSs e 33% para 100 MSs. Essa queda da latência se deve à estratégia de tentativas de *handover* proposta, a qual não limita o tempo de espera para um valor igual a latência de execução. A randomização também garantiu que não houvesse necessidade de se realizar mais tentativas de requisição, uma vez que se conseguiu distribuir as MSs em *slots* de tempo separados.

Os resultados da latência para o Cenário 2 são mostrados na Figura 3(d). A redução na largura de banda total disponível se reflete no aumento da latência para ambas as abordagens e a diferença entre os limiares de *0,05* para *0,02* também influencia na obtenção de uma latência maior, assim como ocorre na Figura 3(c). Mais uma vez, a estratégia de tentativas de *handover* proposta causou um efeito de suavizar a curva da latência, trazendo com o limiar *0,05* uma diminuição de 29% para 65 MSs e de 36% no caso de 100 MSs. Com o limiar *0,02* ocorre uma diminuição de 24% para 65 MSs e de 40% no caso de 100 MSs.

6. Conclusões

Este trabalho propôs uma estratégia de tentativas de *handover* sobre um esquema de *handover* vertical em grupo tomado como referência com o objetivo de reduzir a latência

total de *handover* e suavizar o seu perfil de crescimento diante do aumento do número de estações móveis. Observou-se que a utilização de um esquema de *backoff* exponencial contribuiu para a melhor distribuição de estações móveis em *slots* de tempo. Foi possível obter reduções de até 40% dentro dos cenários estudados.

A necessidade de se propor e aperfeiçoar novos esquemas de *handover* adaptáveis aos cenários de transferência de grupos para diferentes tecnologias é imediata, uma vez que a popularização cada vez maior dos *tablets* e *smartphones* tem trazido requisitos de qualidade de serviço, segurança e manutenção de contexto antes não demandadas pelos clássicos sistemas celulares.

Os resultados do presente trabalho encorajam a avaliar mais cenários, como por exemplo utilizar um maior número de BSs, simulações com tráfego de dados, mudança de parâmetros, diferentes infraestruturas, além de novas métricas de avaliação. Outros esquemas podem ser adaptados para utilizar a estratégia proposta e ter seus resultados analisados. Adicionalmente, os mecanismos de MIH podem ser integrados à implementação a fim de se incluir o processo de obtenção de dados da rede na avaliação de desempenho.

7. Agradecimentos

Aos autores de [Lee and Cho 2011] pelas informações complementares sobre a implementação do esquema de GVHO de referência, além de validarem sua reimplementação.

Referências

Andersson, K., Forte, A., and Schulzrinne, H. (2010). Enhanced Mobility Support for Roaming Users: Extending the IEEE 802.21 Information Service. In *Proc. of the 8th International Conference on Wired/Wireless Internet Communications*, pages 52–63, Berlin.

Cai, X. and Liu, F. (2008). Network Selection for Group Handover in Multi-access Networks. In *Proc. of the IEEE International Conference on Communications*, pages 2164–2168, Beijing.

Cicconetti, C., Galeassi, F., and Mambrini, R. (2010). Network-assisted Handover for Heterogeneous Wireless Networks. In *Proc. of the IEEE GLOBECOM Workshops*, pages 1–5, Miami.

Jeong, H., Choi, J., Kang, H., and Youn, H. (2012). An Efficient Group-Based Channel Scanning Scheme for Handover with IEEE 802.16e. In *Proc. of the 26th IEEE International Conference on Advanced Information Networking and Applications Workshops*, pages 639–644, Fukuoka.

Kim, I. and Kim, Y. (2011). Performance Evaluation and Improvement of TCP Throughput over PFMIPv6 with MIH. In *Proc. of the 12th IFIP/IEEE International Symposium on Integrated Network Management*, pages 997–1004, Dublin.

Kwak, B., Song, N., and Miller, L. (2005). Performance Analysis of Exponential Backoff. *IEEE/ACM Transactions on Networking*, 13(2):343–355.

Lee, S., Sriram, K., Kim, K., Kim, Y., and Golmie, N. (2009). Vertical Handoff Decision Algorithms for Providing Optimized Performance in Heterogeneous Wireless Networks. *IEEE Transactions on Vehicular Technology*, 58(2):865–881.

Lee, W. and Cho, D. (2011). Enhanced Group Handover Scheme in Multi-Access Networks. *IEEE Transactions on Vehicular Technology*, 60(5):2389–2395.

Lei, S., Hui, T., and Zheng, H. (2010). Group Vertical Handover in Heterogeneous Radio Access Networks. In *Proc. of the 72nd IEEE Vehicular Technology Conference Fall*, pages 1–5, Ottawa.

Niyato, D. and Hossain, E. (2009). Dynamics of Network Selection in Heterogeneous Wireless Networks: an Evolutionary Game Approach. *IEEE Transactions on Vehicular Technology*, 58(4):2008–2017.

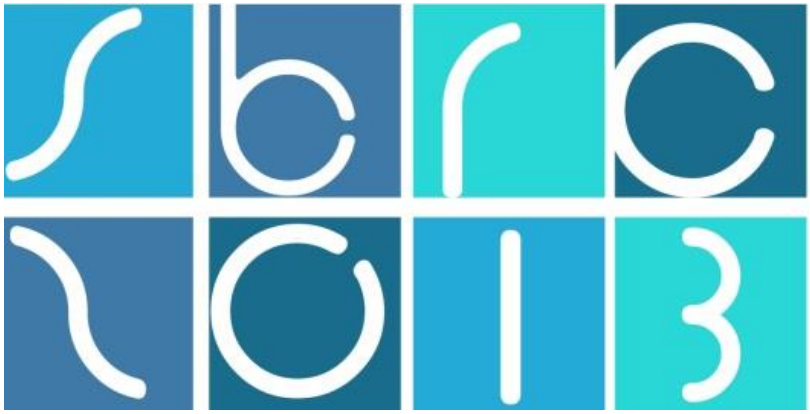
Shan, L., Liu, F., Wang, L., and Ji, Y. (2008). Predictive Group Handover Scheme with Channel Borrowing for Mobile Relay Systems. In *Proc. of the International Wireless Communications and Mobile Computing Conference*, pages 153–158, Crete Island.

Shen, W. and Zeng, Q. (2008). Cost-function-based network selection strategy in integrated wireless and mobile networks. *IEEE Transactions on Vehicular Technology*, 57(6):3778–3788.

Stevens-Navarro, E., Lin, Y., and Wong, V. (2008). An MDP-Based Vertical Handoff Decision Algorithm for Heterogeneous Wireless Networks. *IEEE Transactions on Vehicular Technology*, 57(2):1243–1254.

Taniuchi, K., Ohba, Y., Fajardo, V., Das, S., Tauil, M., Cheng, Y., Dutta, A., Baker, D., Yajnik, M., and Famolari, D. (2009). IEEE 802.21: Media Independent Handover: Features, Applicability, and Realization. *IEEE Communications Magazine*, 47(1):112–120.

Zekri, M., Jouaber, B., and Zeghlache, D. (2012). A Review on Mobility Management and Vertical Handover Solutions over Heterogeneous Wireless Networks. *Computer Communications*, 35(17):2055–2068.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 11

Multimídia Distribuída

Troca Rápida de Canais na Arquitetura iPeer TV

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. *Apesar da crescente popularização dos serviços de IPTV, há ainda diversas funcionalidades a serem melhoradas. Um dos principais desafios nesta área é a redução do tempo de início de reprodução dos fluxos, particularmente nas operações de troca de canais, problema este que é bem relevante em sistemas P2P IPTV. Este trabalho avalia a arquitetura* iPeer TV*, que emprega redes P2P e provê serviços de troca rápida de canais. Três contribuições são oferecidas. Primeiramente, propõe-se uma forma de integração de uma arquitetura P2P IPTV com um esquema de troca rápida de canais, ambos propostos anteriormente e individualmente pelos autores. Em segundo lugar, propõe-se uma otimização simples e eficaz para o esquema de troca rápida de canais, que é capaz de melhorar o seu desempenho. Em terceiro lugar, conduz-se uma análise dos requisitos mínimos de banda de rede para a camada de infraestrutura da arquitetura, de forma que o sistema possa alcançar o seu melhor desempenho.*

Abstract. *In spite of the increasing deployment of IPTV services, various functionalities still need to be improved. One of the main challenges is a reduction in startup delays, especially in channel switchings, a problem which is quite relevant in P2P IPTV systems. This paper evaluates* iPeer TV*, a P2P IPTV architecture with fast channel switching. Our contribution is threefold. Firstly, we propose how our P2P IPTV architecture can be integrated with a previously evaluated channel switching scheme. Secondly, we propose a simple yet effective optimization for such scheme which is capable of improving its performance. Thirdly, we provide an analysis of the minimum bandwidth requirements for the architecture infrastructure layer to ensure the best system performance.*

1. Introdução

Apesar da crescente popularização dos serviços de IPTV, há ainda diversas funcionalidades a serem melhoradas [Manzato and da Fonseca 2008, Hei et al. 2007]. Um dos principais desafios nesta área é a redução do tempo de início de reprodução dos fluxos, em particular nas operações de troca de canais [Manzato and da Fonseca 2008]. Este problema não existe na televisão convencional porque as trocas de canais simplesmente selecionam, através da frequência de transmissão, um conteúdo que já está disponível nas premissas do usuário. Em sistemas IPTV, e particularmente em sistemas P2P IPTV, transmite-se apenas parte do conteúdo servido pela rede, devido à limitação de banda de rede dos usuários. Além disso, *buffers* e estruturas de distribuição são empregados para se amenizar os problemas decorrentes de flutuações de banda de rede e de falhas nas conexões; essas técnicas, apesar de garantirem a continuidade da reprodução dos fluxos, ocasionam latências e degradam a usabilidade do sistema, principalmente quando

se tem diversos canais e se deseja fazer uma navegação rápida entre eles [Manzato and da Fonseca 2011].

Este trabalho avalia a arquitetura *iPeer TV*, que emprega redes P2P e provê serviços de troca rápida de canais. Em um trabalho anterior [Manzato and da Fonseca 2008], os autores propuseram uma nova arquitetura P2P IPTV projetada para superar os principais desafios desta área. Entretanto, a sua avaliação foi deixada como trabalho futuro, pois na ocasião da publicação o sistema ainda estava em desenvolvimento. Em trabalhos posteriores [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011], os autores focaram no problema da redução do tempo de início de reprodução dos fluxos, tendo proposto quatro novos esquemas para se prover troca rápida de canais em sistemas P2P IPTV. Como esses esquemas são independentes de arquitetura, as suas avaliações não consideraram a dinâmica de nenhum sistema específico. No presente trabalho, o esquema que foi melhor avaliado anteriormente, denominado *Rápido*, é integrado à arquitetura P2P IPTV, denominada *iPeer TV*, e uma avaliação de desempenho da operação conjunta de ambos é conduzida. Três contribuições são oferecidas. Primeiramente, propõe-se como a referida integração pode ser realizada. Em segundo lugar, propõe-se uma otimização simples e eficaz para o esquema de troca rápida de canais, que é capaz de melhorar o seu desempenho mesmo com a dinâmica imposta pela arquitetura. Em terceiro lugar, conduz-se uma análise dos requisitos mínimos de banda de rede para a camada de infraestrutura da arquitetura, de forma que o sistema possa alcançar o seu melhor desempenho.

Trabalhos relacionados recaem em sistemas P2P de distribuição de vídeo e esquemas de troca rápida de canais. Os sistemas P2P de vídeo empregam, em geral, múltiplas árvores de distribuição [Padmanabhan et al. 2003] ou estruturas em *mesh* [PPLive 2013], sendo que a principal vantagem dos primeiros é a diminuição natural do tempo de início de reprodução dos fluxos, já que eles mantêm estruturas ativas e sempre prontas para uso. Uma comparação detalhada da arquitetura *iPeer TV* com outros sistemas P2P de vídeo pode ser vista em [Manzato and da Fonseca 2008]. Os esquemas de troca rápida de canais diferem entre si de acordo com os componentes de latência endereçados. Alguns tratam um único componente, podendo ser a latência de rede [Chen et al. 2012] ou a de sincronização do vídeo [Boyce and Tourapis 2005]. Outros tratam múltiplos componentes, podendo ser as latências de sincronização e *buferização* [Banodkar et al. 2008] ou todas em conjunto [Manzato and da Fonseca 2012]. Uma discussão completa sobre os esquemas de troca rápida de canais e os componentes de latência existentes pode ser encontrada em [Manzato and da Fonseca 2013].

O restante deste artigo está organizado da seguinte forma. Na Seção 2, a arquitetura *iPeer TV* e o esquema *Rápido* são revisados. Na Seção 3, a integração deles e a otimização para o esquema *Rápido* são propostos. Na Seção 4, os experimentos de simulação são descritos. Na Seção 5, o sistema final integrado é avaliado e a análise dos requisitos de banda para a arquitetura é conduzida. Na Seção 6, o trabalho é concluído.

2. Contextualização

Nesta seção, a arquitetura *iPeer TV* e o esquema *Rápido* são revisados.

2.1. Arquitetura *iPeer TV*

A Figura 1 apresenta a arquitetura *iPeer TV*, que emprega redes P2P e múltiplas árvores de distribuição [Manzato and da Fonseca 2008]. 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]. 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 propagarem e disponibilizarem 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 [Manzato and da Fonseca 2008].

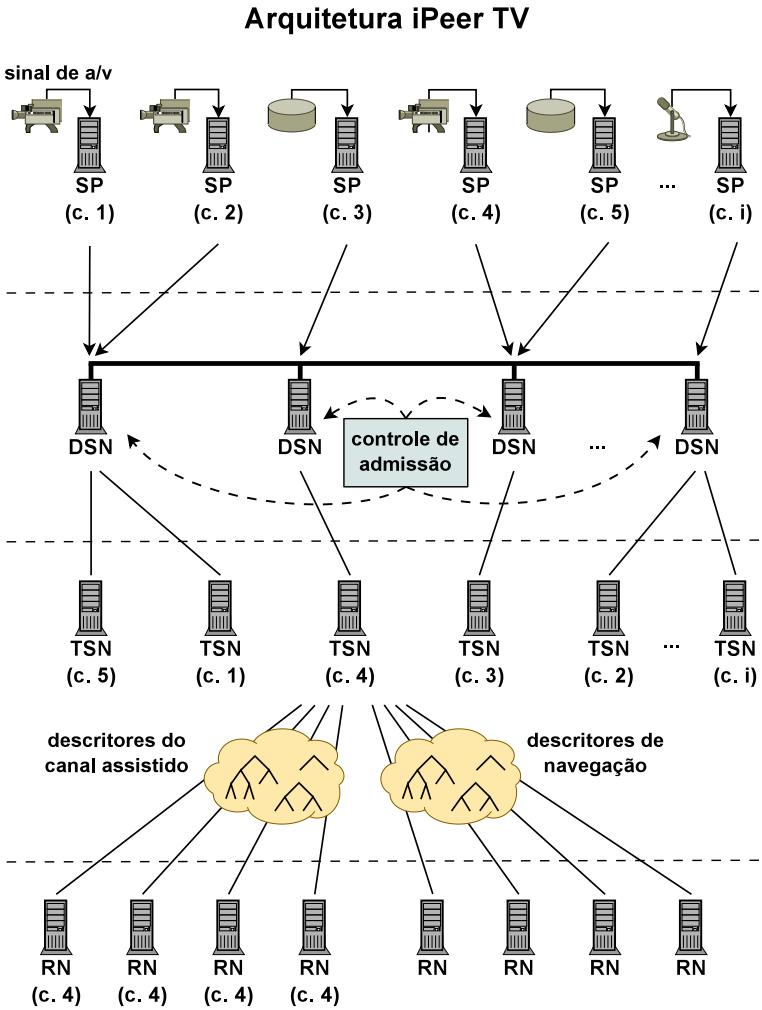


Figura 1. Arquitetura *iPeer TV*, que emprega redes P2P e múltiplas árvores de distribuição [Manzato and da Fonseca 2008].

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 e são admitidos pelos TSNs, através de múltiplas árvores de distribuição, com cada árvore sendo usada para a transmissão de um descritor específico. 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.

2.2. Esquema *Rápido*

A idéia por trás do esquema *Rápido* [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011] é que fluxos com qualidades básicas são suficientes para um usuário identificar o conteúdo que está sendo transmitido em um canal. Dado que esses fluxos apresentam larguras de banda reduzidas, pode-se transmitir múltiplos canais simultaneamente. Com isso, os reprodutores de mídia podem manter múltiplos *buffers* alimentados continuamente para a reprodução imediata dos respectivos canais. A codificação MDC é utilizada para se dividir um canal em múltiplos subfluxos (ou descritores), que apresentam qualidades básicas e são transmitidos aos usuários através de múltiplas árvores de distribuição. Dependendo de se o usuário estiver navegando em canais ou não, a seleção dos descritores enviados a ele pode variar. Sendo assim, o esquema *Rápido* define dois estados nos quais cada usuário pode estar: *assistindo a programação* e *navegando em canais*. No estado *assistindo a programação*, a banda de entrada do usuário é utilizada principalmente para a recepção dos descritores do canal assistido, enquanto que no estado *navegando em canais*, a banda é usada principalmente para a recepção de descritores de canais diferentes, ou seja, os diversos subfluxos com qualidades básicas. A Figura 2(a) ilustra a utilização da banda de entrada do usuário em cada um dos estados.

Para se diminuir as chances de ocorrência de latência no início de uma operação de troca de canais, reserva-se, adicionalmente, no estado *assistindo a programação*, uma parte da banda de entrada do usuário para que um conjunto seleto de fluxos com qualidades básicas seja também transmitido. Reserva-se, também, no estado *navegando em canais*, uma parte da banda para se manter parcialmente a recepção do canal antigo, pois caso o usuário volte a selecioná-lo durante a navegação, ele já estará disponível e com uma qualidade intermediária. Sugere-se, em [Qiu et al. 2009, Cha et al. 2008], que a banda reservada para navegação (em ambos os estados) seja preenchida por canais adjacentes ao assistido e por canais populares, que apresentam altas probabilidades de serem selecionados. Considerando-se esses padrões, três estratégias distintas são aplicadas pelo esquema *Rápido* para a seleção de descritores de navegação: canal antigo, canais populares e canais adjacentes.

A Figura 2(b) elucida as transições de estado do esquema *Rápido*. Quando o usuário está no estado *assistindo a programação* e inicia uma operação de troca de canais, a maior parte da sua banda de entrada é liberada, dispensando-se um subconjunto dos descritores do canal assistido. No espaço que se abre, ele começa a receber descritores adicionais de navegação de outros canais, e o seu estado é alterado para *navegando em canais*. Neste momento, um *timer* individual é iniciado, que conta o tempo decorrido desde a última troca de canais. Durante um período inicial, i.e., enquanto o valor do *timer* estiver no intervalo $[0, TransitionStartTime)$ segundos, nenhuma alteração ocorre nos descritores recebidos por aquele usuário. Durante o período seguinte, i.e., enquanto o valor do *timer* estiver no intervalo $[TransitionStartTime, TransitionFinishTime]$ segundos, uma transição gradual de volta ao estado *assistindo a programação* acontece. Se o *timer* alcançar o valor $TransitionFinishTime$ segundos, o estado do usuário é alterado de volta para *assistindo a programação*; caso contrário, o usuário permanece no estado *navegando em canais*.

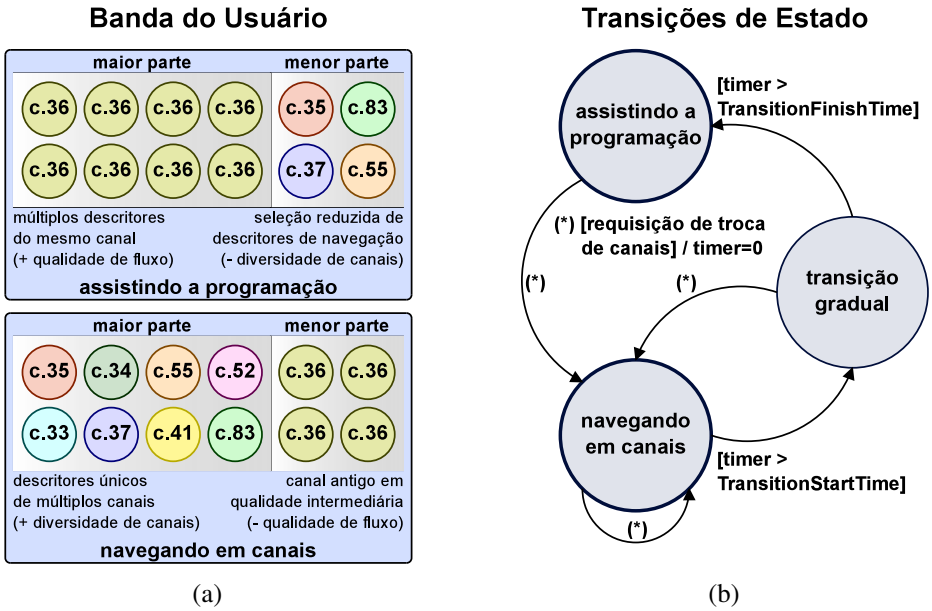


Figura 2. Utilização da banda de entrada do usuário e transições de estado do esquema *Rápido* [Manzato and da Fonseca 2012].

A transição gradual de estados consiste em aumentar periodicamente o número de descritores recebidos para o canal assistido, até que a qualidade do fluxo alcance o valor máximo (estado *assistindo a programação*). Para que a banda de rede agregada não exceda a banda de entrada do usuário, o esquema *Rápido* libera gradualmente os descritores dos canais não assistidos. No final da transição, o usuário estará recebendo todos os descritores do canal assistido, terá o seu estado alterado de volta para *assistindo a programação* e todos os descritores do canal antigo já terão sido liberados.

3. Proposta de Integração e Otimização

Nesta seção, propõe-se como o esquema *Rápido* pode ser integrado à arquitetura *iPeer TV*. Adicionalmente, propõe-se uma otimização simples para o esquema *Rápido*, que visa compensar a sobrecarga causada pela dinâmica da arquitetura.

3.1. Adaptações na Arquitetura *iPeer TV*

O esquema *Rápido* requer que cada usuário seja admitido em pelo menos uma árvore de distribuição de cada canal, bem como em todas as árvores de distribuição do canal assistido, quando ele não estiver navegando [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011]. Além disso, devido à transição gradual de estados, um usuário pode ser temporariamente admitido em múltiplas árvores de *dois* canais ao mesmo tempo: o canal assistido e o canal antigo. Entretanto, devido às limitações de banda de rede dos usuários, na arquitetura *iPeer TV* cada TSN pode servir múltiplos descritores apenas de um *único* canal [Manzato and da Fonseca 2008]. Consequentemente, na integração proposta, um usuário precisa ser admitido simultaneamente por dois TSNs, temporariamente durante suas transições graduais de estados, sendo um TSN primário e o outro secundário.

O TSN primário é responsável por prover os canais de navegação (descritores únicos de cada canal), bem como todos os descritores do canal antigo, enquanto que o TSN

secundário é responsável por prover $n - 1$ descritores do canal assistido (todos os descritores daquele canal exceto um de navegação). Quando um usuário inicia uma operação de troca de canais, durante o período inicial definido pelo parâmetro $TransitionStartTime$ do esquema, apenas o TSN primário serve todos os descritores que ele recebe. Durante o período de tempo seguinte, definido pelo parâmetro $TransitionFinishTime$ do esquema, a transição gradual de estados transcorre, com a seleção de um TSN secundário para aquele usuário. Conforme o usuário solicita descritores adicionais para o canal assistido, ele é admitido em novas árvores de distribuição pelo TSN secundário, de forma que a carga de banda de rede é gradualmente deslocada do TSN primário para o secundário. No final da transição, o TSN primário é dispensado, juntamente com todos os descritores do canal antigo, e o TSN secundário se torna o primário daquele usuário, passando a prover também todos os descritores de navegação.

Na integração proposta, há dois casos nos quais um novo TSN precisa ser alocado no sistema: quando não existir nenhum TSN servindo o canal requisitado e quando todos os TSNs que o servem estiverem com suas bandas de saída esgotadas para a admissão do usuário solicitante. Em ambos os casos, um RN melhor provisionado será selecionado de uma lista de candidatos e então promovido a TSN. Nessa lista, apenas os usuários com capacidades mínimas para se tornarem TSNs estão registrados. Logo antes de se tornar um TSN, o RN selecionado é desligado de todas as árvores de distribuição gerenciadas por seus TSNs primário e secundário. O usuário passa então a ser admitido diretamente por um DSN, começando a receber os descritores do canal servido juntamente com os descritores do canal assistido. Como a maioria dos usuários apresenta restrições de banda de entrada menos severas do que as de saída, a banda de entrada adicional requerida para a recepção deste segundo fluxo não representa um problema. Além disso, os descritores de navegação servidos pelo TSN são compartilhados, já que ele os usa para as suas próprias trocas de canais. Sendo assim, a banda de entrada adicional necessária para que um TSN receba os descritores de navegação a serem servidos é moderada, sendo constituída apenas por aqueles descritores que o próprio TSN já não esteja utilizando para si próprio.

Enquanto na arquitetura original um TSN pode ser rebaixado a RN quando ele deixa de ter descendentes, na integração proposta um TSN permanece neste papel até a sua desconexão do sistema. O motivo desta adaptação é evitar rupturas de fluxo desnecessárias, que poderiam impactar o desempenho do esquema de troca rápida de canais e, com isso, prejudicar a experiência do usuário. Além disso, procura-se promover justiça: os usuários que cooperaram com o sistema (TSNs) deveriam receber as melhores qualidades de fluxo e o melhor desempenho possível nas operações de trocas de canais. Adicionalmente, quando um TSN for necessário para o mesmo canal novamente, o sistema já terá um alocado e pronto para uso, evitando assim a ocorrência de novas rupturas de fluxo. Uma desvantagem desta abordagem é que uma banda de rede maior é requerida na camada de infraestrutura da arquitetura para se manter este TSN admitido diretamente, ao invés de se tê-lo admitido por um outro TSN em uso no sistema. Para compensar esta sobrecarga, propõe-se uma outra adaptação: quando o TSN não estiver repassando os descritores do canal servido ou os de navegação, ele interrompe a transmissão desses descritores a partir do DSN que o admite, com exceção dos descritores de navegação que ele esteja utilizando para si próprio.

3.2. Otimização para o Esquema *Rápido*

Como o esquema *Rápido* é independente de arquitetura, a sua avaliação original [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011] não considerou a dinâmica de nenhum sistema específico. Apesar das bandas de entrada dos usuários terem sido consideradas para se limitar o número máximo de descritores que cada um deles poderia receber, tanto para o canal assistido quanto para os canais de navegação, nenhuma falha foi simulada nas operações de admissão em árvores. Consequentemente, quando o esquema *Rápido* foi integrado pela primeira vez na arquitetura *iPeer TV*, os resultados obtidos foram modestamente piores do que aqueles publicados em [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011].

Na arquitetura *iPeer TV*, as falhas nas operações de admissão em árvores ocorrem devido às limitações de banda de saída dos usuários, bem como pelo esgotamento da banda de rede dos servidores na camada de infraestrutura da arquitetura. Para se compensar essas falhas, decorrentes da dinâmica do sistema, uma otimização simples é proposta para o esquema *Rápido*. Ao se selecionar os descritores de navegação a serem transmitidos ao usuário, uma ordem de prioridade para as admissões deve ser seguida, já que as probabilidades de falhas decorrentes do esgotamento de banda de rede aumentam conforme mais descritores são requisitados. Isso significa que, mesmo que um usuário tenha banda de entrada suficiente para receber diversos canais simultaneamente, ele deve ser admitido primeiramente nas árvores dos canais com maiores probabilidades de serem requisitados. Com isso, se o TSN ou DSN que o admite ficar sem banda de rede para as admissões seguintes, as falhas resultantes ficarão concentradas nos canais com menores probabilidades de serem requisitados.

Como os descritores do canal assistido (e, consequentemente, os do canal antigo) são sempre requisitados antes dos descritores de navegação, eles já são automaticamente priorizados. Sendo assim, a ordem de prioridade para as admissões inclui somente os descritores dos canais adjacentes e dos populares. Sabe-se que 56–60% das trocas de canais são realizadas para canais adjacentes, sendo 69–72% destas para canais superiores e 28–31% para canais inferiores [Qiu et al. 2009, Cha et al. 2008]. Adicionalmente, sabe-se que a estratégia de canais adjacentes do esquema *Rápido* demanda menos banda de rede para ser eficiente do que a estratégia de canais populares, pois enquanto a primeira tem uma boa chance de acerto simplesmente garantindo que os canais imediatamente superiores e inferiores ao atual estejam disponíveis para as trocas adjacentes, a segunda precisa disponibilizar um número maior de canais populares para as trocas não adjacentes [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011]. Com essas informações em mente, propõe-se a seguinte ordem de prioridade para as admissões: (1) o primeiro canal adjacente superior; (2) o primeiro canal adjacente inferior; (3) os três canais mais populares; (4) o próximo canal adjacente superior; (5) o próximo canal adjacente inferior; (6) os demais canais populares, em ordem de popularidade; e (7) os demais canais adjacentes.

O motivo para se requisitar mais de um canal adjacente ao mesmo tempo é o próprio tempo de início de reprodução dos fluxos: quando um usuário troca de canal para um adjacente, é mais provável que o fluxo deste canal esteja pronto para reprodução se ele tiver sido requisitado em algumas trocas de canais anteriores à atual. Com essa simples otimização, é possível reduzir o impacto das falhas de admissão em árvores no esquema de troca rápida de canais, melhorando o seu desempenho na arquitetura P2P IPTV.

4. Experimentos de Simulação

Um simulador de eventos discretos para sistemas P2P IPTV foi desenvolvido para se avaliar o desempenho do sistema final integrado. É de conhecimento dos autores que nenhum dos simuladores existentes que implementam as operações fundamentais de serviços de IPTV esteja disponível para uso. Apesar de uma ferramenta ter sido desenvolvida em [Qiu et al. 2009] para a geração de tráfego sintético (construída a partir de modelos estatísticos derivados de *traces* reais), ela não estava disponível para uso na ocasião do desenvolvimento deste trabalho. Sendo assim, o simulador desenvolvido demandou a implementação das operações fundamentais de serviços de IPTV a partir dos modelos estatísticos apresentados em [Cha et al. 2008] e [Qiu et al. 2009]. Apesar desses modelos terem sido derivados a partir de um sistema IPTV comercial, eles foram usados pelo fato de não existirem modelos equivalentes para P2P IPTV na ocasião do desenvolvimento deste trabalho. Além disso, como o tráfego de IPTV comercial é mais intenso do que o tráfego de P2P IPTV, tipicamente com taxas maiores de requisições de troca de canais, o seu uso é viável na avaliação de sistemas P2P com altas demandas [Manzato and da Fonseca 2012].

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, reproduziu-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 se 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]. Diferenciou-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 heterogeneidade de banda de rede dos usuários, criou-se diferentes classes de conexão, definidas de acordo com estatísticas de padrões de acesso da Internet Brasileira [Teleco 2013]. A largura de banda dos fluxos variou entre 300 Kbps, 1024 Kbps e 2048 Kbps [Hei et al. 2007]. O número de árvores de distribuição utilizadas variou entre 16 e 32 [Padmanabhan et al. 2003]. Para se modelar os tempos de início de reprodução dos fluxos, compostos pelo período necessário para admissão nas árvores de distribuição e *buferização* 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) *Eventos de troca de canais imediatos*, i.e., percentagens de eventos de troca de canais sem latência; (ii) *Qualidades de fluxo*, i.e., números médios de descritores recebidos para o canal assistido pelos

usuários durante suas sessões; e (iii) *Falhas de admissão em árvores*, i.e., percentagens de falhas nas operações de admissão em árvores.

5. Avaliação do Sistema Final Integrado

Nesta seção, o sistema final integrado é avaliado, juntamente com uma análise dos requisitos de banda de rede para a arquitetura. Nas duas primeiras subseções, os resultados obtidos são comparados aos equivalentes originais publicados anteriormente para o esquema *Rápido* [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011]. A terceira subseção não apresenta este tipo de comparação uma vez que o estudo das falhas de admissão em árvores para esta arquitetura é inédito.

Diferentes cenários foram considerados variando-se a largura de banda dos fluxos dos canais existentes e o número de árvores de distribuição utilizadas. Além disso, para cada cenário, a banda de rede total dos servidores na camada de infraestrutura da arquitetura foi variada (de 8 a 192 Gbps). O objetivo era investigar o desempenho do sistema em diferentes situações de contenção de banda de rede. A Tabela 1 mostra as configurações dos diferentes cenários considerados.

Tabela 1. Cenários considerados, variando-se a largura de banda dos fluxos (Str.) e o número de árvores de distribuição (Trs.).

Cenário	Str.	Trs.
S300T16	300 Kbps	16
S300T32		32
S1024T16	1024 Kbps	16
S1024T32		32
S2048T16	2048 Kbps	16
S2048T32		32

Como a largura de banda de cada descritor é proporcional à largura de banda total do fluxo e inversamente proporcional ao número de árvores de distribuição utilizadas, o cenário com a maior contenção de banda de rede (pior caso) é aquele com o maior fluxo e o menor número de árvores, ou seja, o cenário S2048T16 (com fluxo de 2048 Kbps e 16 árvores de distribuição). Analogamente, o cenário com a menor contenção de banda de rede (melhor caso) é aquele com o menor fluxo e o maior número de árvores, ou seja, o cenário S300T32 (com fluxo de 300 Kbps e 32 árvores de distribuição). O cenário S1024T16 (com fluxo de 1024 Kbps e 16 árvores de distribuição) é usado como referência de caso médio, já que ele possui um valor intermediário para a largura de banda do fluxo.

5.1. Eventos de Troca de Canais Imediatos

A Figura 3 ilustra as percentagens de eventos de troca de canais sem latência, em cada cenário considerado. O uso do esquema *Rápido* resultou em diferentes percentagens de eventos imediatos, que são maiores nos cenários com menor contenção de banda de rede. Isso se deve ao fato de que o número de canais de navegação que cada usuário pode receber é determinado pela largura de banda dos descritores; quanto mais descritores de navegação são recebidos, maior é a probabilidade de um canal requisitado já estar disponível para reprodução.

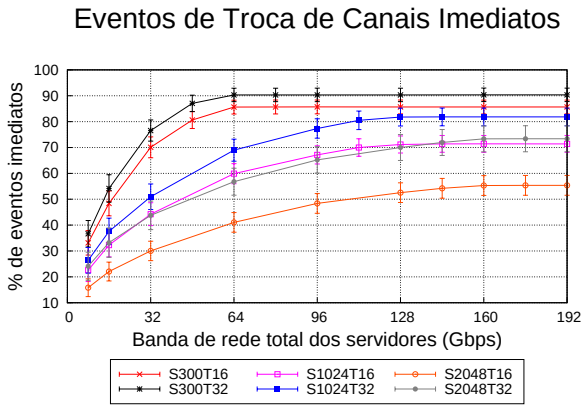


Figura 3. Percentagens de eventos de troca de canais sem latência, em cada cenário considerado.

Os cenários com fluxo de 300 Kbps (S300T16 e S300T32) requereram uma banda de rede total mínima de 64 Gbps nos servidores da camada de infraestrutura da arquitetura. Os cenários com fluxo de 1024 Kbps (S1024T16 e S1024T32) requereram 128 Gbps, enquanto que os cenários com fluxo de 2048 Kbps (S2048T16 e S2048T32) requereram 160 Gbps. De acordo com os resultados, o número de árvores de distribuição utilizadas não afeta a demanda de banda de rede total dos servidores. A razão disso é que, quando o número de árvores aumenta, a largura de banda dos descritores diminui proporcionalmente, mantendo o consumo de banda de admissão equivalente.

No melhor caso (cenário S300T32, com fluxo de 300 Kbps e 32 árvores de distribuição), o uso do esquema *Rápido* resultou em uma redução de 90.31% no número de eventos com latência. No pior caso (cenário S2048T16, com fluxo de 2048 Kbps e 16 árvores de distribuição), a redução foi de 55.30%. No caso médio (cenário S1024T16, com fluxo de 1024 Kbps e 16 árvores de distribuição), uma redução de 71.17% foi obtida. Esses resultados são discretamente melhores do que os equivalentes publicados anteriormente para o esquema *Rápido* [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011]: 89.94%, 48.17% e 67.74%, respectivamente. Apesar dessa melhoria não ter sido significante (considerando-se os valores dessa métrica), ela mostra que o desempenho do esquema *Rápido* não foi afetado pela dinâmica da arquitetura *iPeer TV*, em particular pelas operações de admissão em árvores (analisadas posteriormente nesta seção). Sendo assim, a otimização realizada para o esquema de troca rápida de canais compensou a sobrecarga introduzida pela dinâmica da arquitetura, produzindo resultados até melhores do que os originais em alguns cenários, como foi o caso dos cenários S1024T16 e S2048T16.

5.2. Qualidades de Fluxo

As Figuras 4, 5 e 6 apresentam os números médios de descritores recebidos para o canal assistido pelos usuários durante suas sessões, em cada cenário considerado. As Figuras 4 e 5 mostram essa métrica para os estados *navegando em canais* e *assistindo a programação*, respectivamente, enquanto que a Figura 6 mostra resultados gerais para ambos os estados. Nos cenários com fluxo pequeno (300 Kbps), o número de descritores recebidos aproxima-se do valor máximo (16 ou 32), ao passo que nos cenários com fluxos grandes (1024 e 2048 Kbps), uma queda neste número ocorre. Há dois motivos para este fato. O primeiro, que é independente da adoção de qualquer esquema de troca rápida de

canais, é que, devido ao aumento da contenção de banda de rede, há mais usuários que não conseguem receber todos os descritores dos canais. O segundo motivo é que a adoção do esquema *Rápido* implica em um custo que reduz as qualidades de fluxo, já que parte da banda de rede dos usuários é utilizada para a recepção dos descritores de navegação. Este custo é proporcional à contenção de banda de rede [Manzato and da Fonseca 2012, Manzato and da Fonseca 2011].

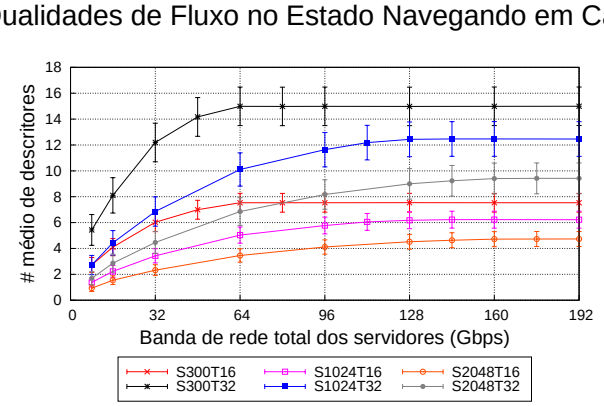


Figura 4. Números médios de descritores recebidos para o canal assistido pelos usuários durante suas sessões no estado *navegando em canais*, em cada cenário considerado.

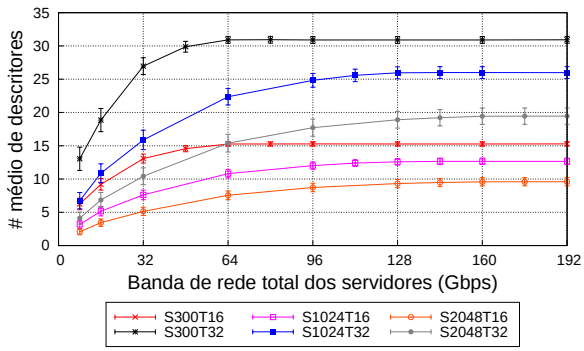


Figura 5. Números médios de descritores recebidos para o canal assistido pelos usuários durante suas sessões no estado *assistindo a programação*, em cada cenário considerado.

Como pode ser visto nas Figuras 4, 5 e 6, a banda de rede total mínima demandada nos servidores da camada de infraestrutura da arquitetura é a mesma que a discutida anteriormente, para cada grupo de cenários: 64 Gbps para os cenários com fluxo de 300 Kbps (S300T16 e S300T32), 128 Gbps para aqueles com fluxo de 1024 Kbps (S1024T16 e S1024T32) e 160 Gbps para aqueles com fluxo de 2048 Kbps (S2048T16 e S2048T32). Além disso, o número de árvores de distribuição utilizadas também não afeta a referida demanda.

No melhor caso (cenário S300T32, com fluxo de 300 Kbps e 32 árvores de distribuição), as qualidades de fluxo no estado *navegando em canais*, estado *assistindo a*

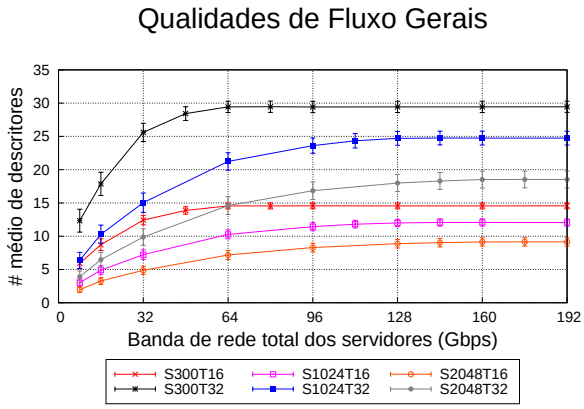


Figura 6. Números médios de descritores recebidos para o canal assistido pelos usuários durante suas sessões em ambos os estados, em cada cenário considerado.

programação e ambos os estados alcançaram, respectivamente, 14.99, 30.91 e 29.44 descritores. Os respectivos resultados originais (publicados em [Manzato and da Fonseca 2012] e [Manzato and da Fonseca 2011]) foram, respectivamente, 15.20, 31.28 e 30.26. No pior caso (cenário S2048T16, com fluxo de 2048 Kbps e 16 árvores de distribuição), as qualidades de fluxo foram, respectivamente, 4.72, 9.58 e 9.13, enquanto que os respectivos resultados originais foram 4.04, 8.12 e 7.86. No caso médio (cenário S1024T16, com fluxo de 1024 Kbps e 16 árvores de distribuição), as qualidades de fluxo foram, respectivamente, 6.18, 12.58 e 11.99, enquanto que os respectivos resultados originais foram 5.97, 12.20 e 11.81. Sendo assim, os resultados para essa métrica são praticamente equivalentes aos publicados anteriormente para o esquema *Rápido*. A razão disso é que, como os descritores do canal assistido são sempre requisitados antes dos descritores de navegação (Subseção 3.2), a sobrecarga introduzida pela dinâmica da arquitetura quase não afeta essa métrica. De maneira análoga, a otimização realizada para o esquema de troca rápida de canais não produz efeito sobre essa métrica, dado que somente considera os canais de navegação.

5.3. Falhas de Admissão em Árvores

Na arquitetura *iPeer TV*, as falhas de admissão em árvores podem ser classificadas em duas categorias. A primeira delas é quando o sistema precisa alocar um novo TSN para admissão de um RN e a banda de rede total dos servidores na camada de infraestrutura da arquitetura está esgotada. Neste caso, como também nenhum dos TSN existentes pode admitir o RN, a operação falha. A segunda categoria é quando um RN já está admitido por um TSN mas a banda de saída deste TSN não permite a transmissão de novos descritores. Neste caso, a banda de rede total dos servidores não afeta a operação, dado que um RN não poderia requisitar descritores diretamente a um DSN.

A Figura 7 ilustra as percentagens de falhas nas operações de admissão em árvores, em cada cenário considerado. A banda de rede total mínima requerida nos servidores da camada de infraestrutura da arquitetura é a mesma que a descrita anteriormente, para cada grupo de cenários. Quando este valor mínimo não é garantido (por exemplo, quando uma banda menor que 64/128/160 Gbps é provisionada nos cenários com fluxos de 300/1024/2048 Kbps, respectivamente), as falhas de admissão em árvores da primeira

categoria ocorrem. Quando a banda de rede total dos servidores é maior do que esses valores mínimos, apenas as falhas da segunda categoria permanecem. Além disso, as percentagens de falhas são similares nos cenários com o mesmo número de árvores, sendo que números menores de árvores produzem percentagens maiores de falhas. Isso ocorre porque os descritores com larguras de banda maiores são mais difíceis de serem servidos pelas bandas de saída limitadas dos usuários.

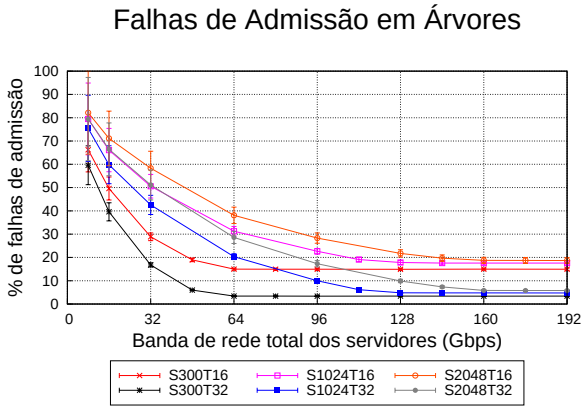


Figura 7. Percentagens de falhas nas operações de admissão em árvores, em cada cenário considerado.

As falhas de admissão em árvores da segunda categoria são permanentes e constituem a sobrecarga introduzida pela dinâmica da arquitetura. Entretanto, a otimização proposta para o esquema de troca rápida de canais compensou essa sobrecarga, produzindo resultados até mesmo melhores do que os originais. Logo, pode-se dizer que o sistema final integrado apresentou um desempenho satisfatório e que a arquitetura *iPeer TV* está agora equipada com um esquema eficaz para se prover serviços de troca rápida de canais.

6. Conclusão

Neste trabalho, a integração do esquema de troca de canais *Rápido* com a arquitetura *iPeer TV* foi avaliada. Algumas mudanças arquiteturais foram necessárias para que essa integração ocorresse. Além disso, uma otimização foi proposta para o esquema *Rápido*, de forma a compensar a sobrecarga introduzida pela dinâmica da arquitetura. O sistema final integrado produziu resultados até mesmo melhores do que aqueles publicados anteriormente para o esquema *Rápido* individualmente. Isso sugere que as decisões de projeto tomadas neste trabalho foram benéficas. Em geral, o sistema final é capaz de realizar, em média, 71% de todas as trocas de canais instantaneamente.

Uma análise dos requisitos mínimos de banda de rede para a arquitetura também foi realizada. Os cenários com fluxo de 300 Kbps requereram uma banda de rede total mínima de 64 Gbps nos servidores da camada de infraestrutura da arquitetura. Os cenários com fluxo de 1024 Kbps requereram 128 Gbps, enquanto que os cenários com fluxo de 2048 Kbps requereram 160 Gbps. Adicionalmente, de acordo com os resultados, o número de árvores de distribuição utilizadas não afeta a demanda de banda de rede da arquitetura.

Referências

Banodkar, D., Ramakrishnan, K. K., Kalyanaraman, S., Gerber, A., and Spatscheck, O. (2008). Multicast instant channel change in IPTV systems. In *Proceedings of the Third International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE '08)*, pages 370–379. IEEE.

Boyce, J. M. and Tourapis, A. M. (2005). Fast efficient channel change. In *Proceedings of the 2005 International Conference on Consumer Electronics (ICCE '05)*, pages 1–2.

Cha, M., Rodriguez, P., Crowcroft, J., Moon, S., and Amatriain, X. (2008). Watching television over an ip network. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement (IMC '08)*, pages 71–84, New York, NY, USA. ACM.

Chen, Y., Merrer, E. L., Li, Z., Liu, Y., and Simon, G. (2012). OAZE: A network-friendly distributed zapping system for P2P IPTV. *Computer Networks*, 56(1):365–377.

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

Hei, X., Liang, C., Liang, J., Liu, Y., and Ross, K. W. (2007). A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):1672–1687.

Liu, J., Rao, S. G., Li, B., and Zhang, H. (2008). Opportunities and challenges of peer-to-peer internet video broadcast. *Proceedings of the IEEE*, 96(1):11–24.

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

Manzato, D. A. G. and da Fonseca, N. L. S. (2011). Esquemas para troca rápida de canais em sistemas IPTV. In *Proceedings of the 29th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2011)*, pages 515–528.

Manzato, D. A. G. and da Fonseca, N. L. S. (2012). Providing fast channel switching in P2P IPTV systems. *To appear in IEEE Journal on Selected Areas in Communications – 2012 Special Issue on Emerging Technologies in Communications (2012 JSAC-SI-ET)*.

Manzato, D. A. G. and da Fonseca, N. L. S. (2013). A survey of channel switching schemes for IPTV. *To appear in IEEE Communications Magazine – 2013 Special Issue on IP-Based TV Technologies, Services and Multidisciplinary Applications*.

Padmanabhan, V. N., Wang, H. J., and Chou, P. A. (2003). Resilient peer-to-peer streaming. In *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, pages 16–27. IEEE Computer Society.

PPLive (2013). PPLive. http://www.pplive.com/en/index.html. (Access Date).

Qiu, T., Ge, Z., Lee, S., Wang, J., Xu, J., and Zhao, Q. (2009). Modeling user activities in a large IPTV system. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement (IMC '09)*, pages 430–441, New York, NY, USA. ACM.

Teleco (2013). Teleco. http://www.teleco.com.br/. (Access Date).

Veloso, E., Almeida, V., Meira, W., Bestavros, A., and Jin, S. (2002). A hierarchical characterization of a live streaming media workload. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment (IMW '02)*, pages 117–130. ACM Press.

Uma Nova Estratégia Completamente Distribuída para Combate à Poluição de Conteúdo em Transmissões ao Vivo

Roverli P. Ziwich[1], **Glaucio P. Silveira**[2], **Elias P. Duarte Jr.**[1]

Universidade Federal do Paraná (UFPR), Curitiba, PR, Brasil
[1]Dept. de Informática, P.O.Box 19018, CEP 81531-980
[2]Centro de Computação Eletrônica, P.O.Box 19037, CEP 81531-980

roverli@inf.ufpr.br, glaucio@ufpr.br, elias@inf.ufpr.br

Resumo. *É notável o crescente uso da Internet para transmissões ao vivo. Por outro lado a poluição de conteúdo nas transmissões ao vivo em redes P2P continua sendo um desafio, já que soluções incorrem necessariamente em sobrecarga de processamento, de uso de rede, ou até atrasos na transmissão. Este trabalho apresenta uma nova estratégia distribuída e descentralizada com o objetivo de combater a propagação de conteúdo poluído em transmissões ao vivo. Para impedir a propagação da poluição, cada peer do sistema executa comparações periódicas sobre determinados chunks de seus vizinhos. Com base nos resultados das comparações, cada peer, de forma independente dos demais, deixa de solicitar chunks aos seus vizinhos considerados poluidores. A solução proposta foi implementada no Fireflies, um protocolo escalável para redes overlay. Resultados experimentais mostram que esta estratégia adiciona baixa sobrecarga no tráfego da rede, e é uma solução viável para tratar a poluição de conteúdo em transmissões ao vivo: em vários casos a solução foi capaz de eliminar a poluição no decorrer das transmissões.*

Abstract. *Live streaming transmissions are becoming increasingly frequent in the Internet. One of the main challenges of those transmissions is to detect and prevent content pollution. This work presents a novel strategy to combat content pollution in P2P-based live streaming that is fully distributed. In order to prevent the propagation of polluted content, each peer independently compares and classifies its neighbors, and stops requesting chunks from those neighbors identified as polluters. The proposed solution was implemented using Fireflies, a scalable overlay network protocol. Experimental results evaluate the impact of the proposed strategy in terms of the network bandwidth overhead, and show that the strategy is an effective solution to combat content pollution in realistic scenarios of live streaming transmissions.*

1. Introdução

Transmissões ao vivo, notadamente de vídeos, estão se tornando cada vez mais populares na Internet [Lin et al. 2010] e diversos sistemas para transmissões ao vivo que utilizam redes P2P surgiram nos últimos anos – como por exemplo o PPLive[1], o SopCast[2], e o

[1]PPLive - *http://www.pplive.com/en*
[2]SopCast - *http://www.sopcast.com*

PPStream[3]. As redes P2P possuem algumas vantagens sobre o formato tradicional cliente-servidor para transmissões ao vivo pois os próprios *peers* podem compartilhar o conteúdo que é transmitido. Desta forma a quantidade, capacidade de processamento e de largura de banda dos servidores que distribuem o conteúdo transmitido pode ser significativamente menor do que a dos mesmos servidores nas redes que utilizam o formato tradicional.

Por outro lado, como os próprios *peers* são responsáveis pelo conteúdo transmitido, a poluição de conteúdo nas transmissões ao vivo em redes P2P é um dos desafios que continuam relevantes. Um ataque de poluição de conteúdo consiste na modificação não autorizada do conteúdo transmitido – que é dividido em pedaços, chamados *chunks*. A modificação dos *chunks* pode ser de diferentes tipos [Gheorghe et al. 2010, Dhungel et al. 2009, Lin et al. 2010], que incluem: a troca de conteúdo; a criação de novos dados; e até a destruição ou omissão dos *chunks* transmitidos.

Outras características que agravam o problema da poluição de conteúdo nas transmissões ao vivo incluem o limite de tempo no qual o conteúdo transmitido tem para alcançar os *peers* da rede, e o *churn*, isto é, o fato de *peers* entrarem e saírem da rede continuamente durante a transmissão. Estas características são relevantes pois a detecção de conteúdo poluído e a consequente criação de novas solicitações pode causar atrasos e até saltos na transmissão assistida pelos usuário [Yang et al. 2008, Zhang and Helvik 2011].

Algumas soluções que tratam o problema da poluição de conteúdo em transmissões ao vivo assumem que todos os *peers* sabem previamente, ou recebem durante a própria transmissão o valor *hash* dos respectivos *chunks* [Wong and Lam 1999]. Esta estratégia é bastante usada para tratar falhas físicas nos canais de comunicação, mas ainda permite a um *peer* malicioso modificar indevidamente um *chunk* e retransmiti-lo juntamente com um novo valor *hash* correspondente. Outras soluções ainda propõem o uso de assinaturas digitais, ou seja, criptografia de chave pública, para todos os *chunks* que são transmitidos [Haridasan and van Renesse 2006]. A assinatura digital é gerada pelo servidor fonte e transmitida juntamente com os *chunks* pela rede. Nesta estratégia, cada *peer* que recebe um *chunk* deve conferir se a assinatura digital é válida. Por outro lado este é um procedimento que pode ser considerado computacionalmente custoso, dependendo dos dispositivos usados pelos usuários da transmissão.

Recentemente, em [Schimidt et al. 2011], uma solução alternativa é apresentada, que utiliza o diagnóstico baseado em comparações para detectar poluição de conteúdo em transmissões ao vivo em redes P2P. Esta solução não utiliza criptografia de chave pública e não utiliza o envio de valores *hash* junto à transmissão. Por outro lado, a solução apresentada assume a existência de uma unidade central – um *tracker* – que é a unidade que realiza o diagnóstico da poluição na rede. Este *tracker* central, por sua vez, pode representar um problema de escalabilidade, caso a solução seja utilizada em redes extremamente grandes. Além disso a solução atua apenas na identificação da existência de poluição na rede, sem combatê-la.

Este trabalho apresenta uma nova estratégia que também utiliza o diagnóstico baseado em comparações e que também não utiliza criptografia de chave pública e não pressupõe o envio de valores *hash* juntamente com a transmissão dos *chunks*. Por outro lado, a nova estratégia proposta neste trabalho, tem por objetivo o combate à propagação

[3]PPStream - *http://www.ppstream.com*

de conteúdo poluído nas transmissões ao vivo em redes *overlay*. Outra contribuição deste trabalho é que a estratégia proposta, diferente da anterior [Schimidt et al. 2011], é completamente distribuída e descentralizada, ou seja, esta solução não utiliza um *tracker* central. Para combater a propagação da poluição, cada *peer* do sistema executa comparações periódicas sobre determinados *chunks* de seus vizinhos. Com base nos resultados das comparações, cada *peer*, de forma independente dos demais, deixa de solicitar *chunks* aos seus vizinhos considerados poluidores.

A solução proposta foi implementada no Fireflies, um protocolo escalável para redes *overlay* [Johansen et al. 2006]. O Fireflies usa a estratégia *pull-based* para a transmissão dos dados e emprega a topologia *mesh*. A implementação foi realizada usando o mesmo simulador Fireflies descrito em [Haridasan and van Renesse 2006]. Resultados experimentais mostram que a estratégia proposta é uma solução viável para identificar e combater a poluição de conteúdo em transmissões ao vivo. Em diversas configurações a solução foi capaz de reduzir consideravelmente a poluição de conteúdo, em vários casos chegando a eliminá-la no decorrer das transmissões. Além disso, os resultados ainda mostram que a solução adiciona baixa sobrecarga no tráfego da rede.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 realiza uma descrição sobre o protocolo Fireflies e sobre o diagnóstico baseado em comparações. A Seção 3 apresenta a solução proposta para o combate à propagação de poluição de conteúdo em transmissões ao vivo. Já na Seção 4 resultados experimentais são apresentados. A Seção 5 descreve trabalhos relacionados e, por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Definições Preliminares

Esta seção apresenta inicialmente uma descrição do protocolo Fireflies – um protocolo escalável para redes *overlay* no qual a solução proposta neste trabalho foi implementada. Na sequência é apresentada uma visão geral sobre o modelo de diagnóstico baseado em comparações que é utilizado pela solução proposta.

2.1. O Fireflies

O protocolo Fireflies é um protocolo escalável que cria uma rede *overlay* tolerante a intrusões [Johansen et al. 2006]. Todos os *peers* da rede executam o protocolo Fireflies usando a estratégia *pull-based* para a transmissão dos dados [Loocher et al. 2007], e os *peers* são organizados em uma topologia *mesh* [Pai et al. 2005]. Além dos *peers*, também existe um *servidor fonte* que gera os *chunks* que são transmitidos na rede. O servidor fonte é considerado uma unidade confiável e que nunca falha.

No Fireflies os *chunks* são enviados pelo servidor fonte para um número configurável de *peers* da rede. Os *peers* então compartilham os *chunks* entre si, com o objetivo de que todos os *peers* da rede recebam todos os *chunks* transmitidos pelo servidor fonte. No protocolo Fireflies todos os *peers* possuem um identificador sequencial e são organizados em múltiplos anéis [Johansen et al. 2006]. O número de anéis é configurável e cada anel contem todos os *peers* do sistema. Estes anéis têm o simples propósito de determinar o conjunto de vizinhos de cada *peer*. Além disso, é possível que um determinado *peer* possua vizinhos em comum em mais de um anel. Portanto cada *peer* da rede sempre possui pelo menos 2 vizinhos e no máximo $(2 * \lambda)$, onde λ representa o número de anéis configurados.

Como exemplo, a Figura 1 ilustra um sistema com 12 *peers* organizados em 4 anéis. Note que os vizinhos do *peer* 1 são os *peers* 3, 4, 5, 6, 7 e 9. A figura também ilustra a situação onde um *peer* possui vizinhos em comum em mais de uma anel: o *peer* 1 possui os *peers* 3 e 9 como vizinhos em dois diferentes anéis. No Fireflies o servidor fonte recebe o identificador 0 e não participa da configuração dos anéis.

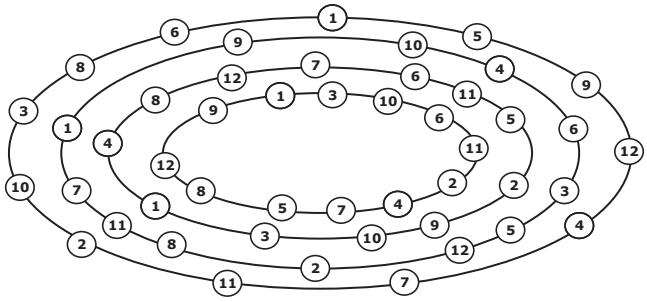


Figura 1. Um exemplo de rede Fireflies com 12 *peers* organizados em 4 anéis.

O protocolo Fireflies ainda configura em cada *peer* uma *janela de disponibilidade* e uma *janela de interesse*. A janela de disponibilidade é uma lista que indica quais *chunks* cada *peer* possui disponíveis para envio a seus vizinhos. Já a janela de interesse indica quais *chunks* cada *peer* ainda precisa receber. Quando um *peer* recebe um *chunk*, ele notifica a todos os seus vizinhos sobre a disponibilidade daquele chunk. Com base nestas notificações cada *peer* é capaz de manter uma lista de quais *chunks* estão disponíveis em cada um dos seus vizinhos. Desta forma, se um *peer* i for notificado por um de seus vizinhos v sobre a disponibilidade de *chunk* c, e se este *chunk* c estiver na janela de interesse do *peer* i, este *peer* requisita o *chunk* c ao vizinho v. Quando o *peer* v receber a requisição, se o *chunk* c ainda estiver na sua janela de disponibilidade, o *peer* v envia o *chunk* c ao *peer* i; caso contrário o *peer* v simplesmente ignora aquela requisição. Este é exatamente o procedimento que também ocorre com todos os *chunks* gerados pelo servidor fonte: quando o fonte gera um novo *chunk* ele notifica seus vizinhos sobre a disponibilidade daquele *chunk*, e então a sua difusão se inicia pelos *peers* da rede.

2.2. Diagnóstico Baseado em Comparações

O diagnóstico baseado em comparações [Duarte Jr. et al. 2011] utiliza a comparação do resultado de tarefas para determinar o estado de cada unidade do sistema, que por sua vez pode ser *falho* ou *sem-falha*. O modelo MM de diagnóstico baseado em comparações foi proposto por Maeng e Malek em [Maeng and Malek 1981] para detectar falhas em sistemas multiprocessados compostos por unidades homogêneas. Neste modelo o sistema é representado por um grafo $G = (V, E)$, onde V é o conjunto de unidades e E é o conjunto de *links* de comunicação. Uma unidade i é uma unidade *comparadora* – ou *testadora* – de outras duas unidades j e k se e somente se $(i, j) \in E$ e $(i, k) \in E$, e também se $i \neq j$ e $i \neq k$. Um *teste*, denotado por $(j, k)_i$, é definido como o envio de uma tarefa a partir de uma unidade comparadora i para outras duas unidades do sistema, unidades j e k. As unidades j e k executam a tarefa recebida e devolvem o resultado da tarefa à unidade testadora que então realiza a comparação do resultado das tarefas.

Se a unidade testadora for uma unidade sem-falha e o resultado da comparação das tarefas indicar *igualdade*, ambas as unidades comparadas são consideradas sem-falha. Por

outro lado, se a comparação indicar *diferença*, ao menos uma das três unidades i, j e k é falha. Este modelo ainda assume que a saída de tarefas produzidas por duas unidades falhas é sempre diferente. O conjunto com todos os resultados das comparações – ou seja, o conjunto de todos os resultados dos testes – é chamado de *síndrome* do sistema.

Em [Ziwich et al. 2005] um modelo distribuído de diagnóstico baseado em comparações é apresentado no qual as próprias unidades são capazes de realizar o diagnóstico. Além disso, outra característica importante é que este modelo assume que a comparação do resultado de tarefas produzidas por duas unidades falhas pode resultar em igualdade. A estratégia proposta neste trabalho emprega este modelo distribuído de diagnóstico [Ziwich et al. 2005], ou seja, dois diferentes *peers* do sistema podem eventualmente possuir uma mesma cópia poluída de um determinado *chunk*. Além disso, na solução proposta, um determinado *peer* testador realiza testes através da solicitação de um determinado *chunk* a todos os seus vizinhos. O resultado da tarefa, que é o conteúdo do próprio *chunk* recebido, é então comparado, em pares. Com base no resultado das comparações os *peers* são agrupados (ou classificados) em conjuntos de acordo com o conteúdo dos *chunks* recebidos.

3. A Estratégia Proposta para Combate à Poluição de Conteúdo

Esta seção apresenta a estratégia proposta para combater a propagação de poluição de conteúdo em transmissões ao vivo em redes P2P. A estratégia proposta utiliza o diagnóstico baseado em comparações para detectar *peers* poluidores e é baseada no protocolo Fireflies. Além do servidor fonte e dos *peers* – que já são componentes da arquitetura do Fireflies – a solução proposta implementa um novo componente, chamado de *módulo comparador*.

O módulo comparador é um componente que é executado por cada *peer*, e é integrado ao próprio código do protocolo Fireflies. Como o módulo comparador é integrado ao próprio protocolo Fireflies, ele possui acesso aos *chunks* recebidos e às janelas de disponibilidade daquele *peer*. Além disso o módulo comparador é o componente responsável por realizar as comparações de determinados *chunks*. Para isso cada *peer*, de forma independente dos demais, escolhe aleatoriamente um *chunk* para ser comparado, dentro dos chamados intervalos de monitoramento. Por sua vez, o intervalo de monitoramento é uma configuração do módulo comparador que indica o tempo máximo no qual o módulo comparador de cada *peer* deve escolher aleatoriamente um *chunk* para ser comparado.

De forma resumida, o procedimento executado por cada *peer* da rede é o descrito a seguir. Assim que um *peer* i recebe um *chunk* de identificador cid de um *peer* vizinho v, este *peer* i verifica se o identificador cid é o identificador de um dos *chunks* que foi escolhido aleatoriamente para ser comparado. Caso este seja um dos *chunks* que devem ser comparados, o módulo comparador do *peer* i o requisita a cada um dos vizinhos que informou sua disponibilidade. É importante ressaltar que estas requisições adicionais realizadas pelo modulo comparador, são requisições regulares do protocolo Fireflies, e são realizadas pelo próprio *peer* e através de conexões do próprio sistema. Em outras palavras, um *peer* que receber esta requisição vinda do módulo comparador do *peer* i não consegue diferenciá-la de qualquer outra requisição recebida de qualquer outro *peer*, e portanto não é possível tratá-la de forma diferenciada.

Assim que o *peer* i concluir a requisição e receber as respostas com o *chunk* re-

quisitado de identificador cid de cada um de seus vizinhos, este *peer* i compara os *chunks* recebidos em pares, e de acordo com o resultado das comparações classifica cada um dos *peers* em um conjunto $U_{i,cid}$. Cada conjunto $U_{i,cid}$ é um conjunto criado no *peer* i e contém cada um um dos *peers* vizinhos do *peer* i, classificados de acordo com o conteúdo do *chunk* cid, e tem o seguinte formato:

$$U_{i,cid} = \{(chunk_a, \{peer_j, peer_k, ...\}), (chunk_b, \{peer_m, peer_n, ...\}), ...\}.$$

Como cada *peer* tem um tempo limite para responder a cada requisição de *chunks*, se por qualquer motivo um *peer* não responder a uma requisição sobre um determinado *chunk*, ele é inserido em um conjunto especifico dos *peers* que não responderam às requisições. Destaca-se que este tempo máximo para cada *peer* esperar as respostas de seus vizinhos é o mesmo tempo configurado como a janela de interesse, ou seja, é o mesmo tempo em que cada *peer* normalmente já espera um determinado *chunk* da transmissão.

Assim que cada *peer* i finalizar um conjunto $U_{i,cid}$, ou seja, o conjunto $U_{i,cid}$ está completo e contém todos os vizinhos do *peer* i, o seguinte procedimento é executado. Se neste conjunto $U_{i,cid}$ existir um subconjunto de *peers* que responderam, e cuja quantidade de $peers$ neste subconjunto seja maior que a metade do número de vizinhos do *peer* i, então: aquele *peer* i (e apenas quele *peer* i) passa a partir daquele momento a ignorar todos os *chunks* e notificações de disponibilidade de *chunks* de qualquer um dos *peers* que não estiverem neste maior conjunto. Esta lista de *peers* bloqueados é constantemente atualizada por cada um dos *peers* do sistema, sempre que cada módulo comparador finalizar um novo conjunto U.

Como exemplo, a Figura 2 ilustra as requisições realizadas pela estratégia proposta. Este exemplo considera que o *peer* 20 escolheu o *chunk* com identificador 325 como um dos *chunks* para serem comparados. Como os vizinhos do *peer* 20 na rede são os *peers* 5, 12, 32, 43 e 57, assim que estes *peers* vizinhos notificarem ao *peer* 20 que possuem o *chunk* 325 disponível, o *peer* 20 irá solicitar a cada um de seus vizinhos este *chunk* 325. Nesta figura as setas direcionadas representam o envio do *chunk* 325 para o *peer* 20, a partir de cada um dos seus vizinhos; as demais arestas representam *links* de comunicação entre os próprios *peers* ou entre os *peers* e o servidor fonte. Ainda neste exemplo, o conteúdo original do *chunk* 325 é ilustrado por "OrigData" e o conteúdo modificado (poluído) indevidamente deste *chunk* é ilustrador por "PollData". O exemplo considera que apenas o *peer* 43 é um *peer* poluidor e que neste momento os *peers* 5, 12, 32 e 57 possuem uma cópia correta do *chunk* 325.

Ainda com base na Figura 2, assim que o *peer* 20 possuir informações de todos os seus vizinhos a respeito do *chunk* 325, o *peer* 20 irá realizar as comparações dos *chunks* recebidos, em pares, e classificará cada um dos seus *peers* vizinhos no conjunto $U_{20,325}$. Especificamente neste exemplo o conjunto gerado, após finalizado, será $U_{20,325} = \{(OrigData, \{5, 12, 32, 57\}), (PollData, \{43\})\}$. Assim que este conjunto $U_{20,325}$ for finalizado pelo *peer* 20, será possível identificar que existe um conjunto com mais da metade do número de vizinhos do *peer* 20. Em outras palavras, o conjunto $(OrigData, \{5, 12, 32, 57\})$ possui 4 *peers*, e $4 \geq (5/2)$, onde $5/2$ é a metade do número de vizinhos do *peer* 20. A partir deste momento o *peer* 20 irá parar de solicitar *chunks* ao *peer* 43.

De forma detalhada e complementando o resumo apresentado acima, a Figura 3 mostra o algoritmo em pseudocódigo que implementa o módulo comparador. A Fi-

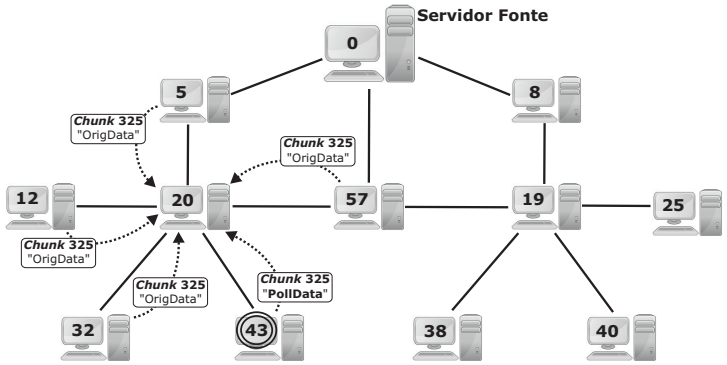


Figura 2. Transmissão do *chunk* 325 para o *peer* 20; o *peer* 43 é poluidor.

gura 3 mostra o algoritmo em pseudocódigo executado pelo módulo comparador, ou seja, é o código responsável pelas solicitações adicionais de *chunks*, pela realização das comparações e pela geração da lista de *peers* vizinhos que terão notificações de *chunks* ignoradas – a lista dos *peers* bloqueados.

Inicialmente (na linha 2) o módulo comparador que executa no *peer* i determina aleatoriamente os identificadores dos primeiros *chunks* para serem comparados por aquele *peer* i. Os identificadores de *chunks* para comparação são escolhidos aleatoriamente, da seguinte forma: $proximo\_cid\_para\_comparar \leftarrow ultimo\_cid\_gerado + mod(random, (monitoring\_interval * mcast\_rate))$, onde $random$ representa um número aleatório, $monitoring\_interval$ é o tempo máximo (em segundos) configurado como o intervalo de monitoramento da solução, e $mcast\_rate$ é o número de *chunks* gerados por segundo pelo fonte. Em outras palavras, o próximo *chunk* a ser monitorado (ou comparado) será qualquer um dos *chunks* gerados pelo servidor fonte nos próximos $monitoring\_interval$ segundos após o momento de geração do último *chunk*. Como exemplo, caso $monitoring\_interval = 15$, $mcast\_rate = 30$ e $ultimo\_cid\_gerado = 5674$, o valor do $proximo\_cid\_para\_comparar$ será um número aleatório entre 5674 e $(5674 + (15 * 30))$. Assim que os próximos identificadores para comparação forem sendo escolhidos eles são inseridos na lista $lista\_de\_cids$.

```
Algoritmo: ModuloComparador /* executando no peer i */
 1: inicio
 2:    lista_de_cids ← gera e atualiza a lista de chunks aleatórios para serem comparados
 3:                    (considerando o intervalo de monitoramento configurado);
 4:    sempre que um vizinho v disponibilizar um novo chunk cid faça
 5:       se cid ∈ lista_de_cids então
 6:          requisitar o chunk cid ao peer v;
 7:          atualizar o U_{i,cid} correspondente com a resposta do peer v;
 8:       fim se
 9:    fim sempre que
10:
11:    sempre que ((U_{i,cid} possuir informação sobre todos os peers vizinhos do i) ou
12:                (acabou o tempo limite para obtenção de informações sobre aquele chunk cid)) faça
13:       se (acabou o tempo limite para obtenção de informações sobre aquele chunk cid) então
14:          incluir vizinhos que não responderam em um subconjunto específico do U_{i,cid};
15:       fim se
16:       se U_{i,cid} possui um subconjunto com mais de N(i)/2 de peers então
17:          lista_de_peers_bloqueados ← ∅; /* limpa a lista de peers bloqueados */
18:          lista_de_peers_bloqueados ← peers que não estão no maior conjunto;
19:       fim se
20:       list_de_cids ← atualiza a lista com novos chunks aleatórios para serem comparados;
21:    fim sempre que
22: fim
```

Figura 3. Algoritmo em pseudocódigo implementado pelo módulo comparador.

O módulo comparador então espera por notificações dos seus vizinhos sobre a disponibilidade de novos *chunks*. Sempre que um vizinho v notificar a disponibilidade de requisição de um novo *chunk* cid (linha 4), o *peer* i executa o código das linhas 5–8: se o *chunk* de identificador cid é um *chunk* para ser monitorado, uma requisição do *chunk* cid é realizada pelo *peer* i ao *peer* v (linha 6). Assim que receber a resposta da requisição, ou seja, uma cópia do próprio *chunk* cid enviada pelo *peer* v, o conjunto $U_{i,cid}$ é atualizado (linha 7) e o *peer* v é classificado de acordo com o resultado da comparação do *chunk* v recebido.

Já as linhas 11–21 são executadas sempre que o conjunto $U_{i,cid}$ estiver completo, ou seja, já possuir informações sobre todos os vizinhos do *peer* i, ou ainda se o tempo limite para obtenção de informações sobre o *chunk* cid estiver esgotado. Caso tenha ocorrido este último caso (o tempo limite foi esgotado), os *peers* que não enviaram nenhuma informação a respeito do *chunk* cid são classificados em um subconjunto específico (linha 14). Por sua vez, o teste da linha 16 verifica se o conjunto $U_{i,cid}$ em questão possui algum subconjunto com mais de $N(i)/2$ *peers* que responderam, onde $N(i)$ é o número de vizinhos do *peer* i. Caso ocorra esta condição, o módulo comparador atualiza a lista $lista\_de\_peers\_bloqueados$ (linhas 17–18) que é usada pelo *peer* i como a lista dos *peers* dos quais o *peer* i irá a partir daquele momento ignorar a disponibilização de novos *chunks*. A lista $lista\_de\_peers\_bloqueados$ é sempre atualizada com todos os *peers* que não estão no maior subconjunto de *peers*.

Finalmente, cada *peer* do sistema Fireflies deve realizar um simples teste adicional para verificar se as notificações de disponibilidade de *chunks* devem ou não ser descartadas: sempre que um vizinho v notificar a disponibilidade de um novo *chunk*, o *peer* i verifica se este vizinho v está ou não na lista $lista\_de\_peers\_bloqueados$. Caso ocorra este caso, o *peer* i simplesmente ignora aquela notificação. Caso contrário, o *peer* i executa os procedimentos previstos pelo protocolo Fireflies para aquele evento da rede *overlay*.

4. Resultados Experimentais

A estratégia proposta foi implementada usando o simulador Fireflies descrito em [Haridasan and van Renesse 2006]. Um grande número de simulações foi executado para sistemas com 200 *peers*. Cada um dos experimentos simulou uma transmissão ao vivo por um período de 180 segundos. O servidor fonte gerou 30 *chunks*/segundo e o Fireflies foi configurado para organizar os *peers* em 15 anéis. O tamanho do *chunk* foi de 10KB. Ambas as janelas de disponibilidade e de interesse de todos os *peers* foram configuradas com 3000 *chunks*. Além disso, em todos os experimentos o intervalo de monitoramento configurado para o módulo comparador foi de 15 segundos, ou seja, no máximo a cada 15 segundos o módulo comparador de cada *peer* escolhia aleatoriamente um *chunk* para ser comparado entre todos os seus vizinhos. Os experimentos foram executados em um computador com processador AMD Phenom 9500 quad-core x64 e 4GB de memória RAM, executando o sistema operacional Linux 64-bits com kernel versão 2.6.18-238.el5.

Os principais propósitos dos experimentos foram (a) verificar qual foi o impacto da poluição nas transmissões, para diferentes quantidades de *peers* poluidores, e (b) computar o *overhead* adicionado pela solução proposta, em termos do número de *chunks* adicionais requisitados pelo módulo comparador. Os principais parâmetros variados nas simulações foram:

(1) a quantidade de *peers* poluidores, variando entre 0%, 5%, 10%, 15%, 20% e 25% do total de *peers* na rede;
(2) foram experimentadas simulações com e sem *churn*: para os experimentos com *churn*, 100 *peers* entraram na rede e outros 100 *peers* saíram da rede. O momento de entrada dos *peers* seguiu uma distribuição normal com média 100 e desvio padrão 20. Para modelar a saída dos *peers* da rede foi utilizada uma distribuição de Poisson com média 100;
(3) também foram realizadas simulações com o módulo comparador ativo e inativo, com o objetivo de comparar o efeito da poluição em redes com a solução proposta e em transmissões sem nenhuma solução para tratar a poluição.

Foram executadas um total de 1.000 simulações. Os resultados foram sumarizados e são apresentados nos gráficos das Figuras 4 a 7. As linhas dos gráficos representam os valores médios, enquanto que as linhas verticais de intervalo mostram o intervalo de confiança de 95% para a amostra de dados correspondente.

A Figura 4(a) mostra o número médio de *chunks* enviados normalmente pelo Fireflies durante as transmissões, sem a solução proposta, para as simulações com *churn* e sem *churn*. Pode-se notar que o número médio de *chunks* enviados pelo Fireflies esteve sempre entre 1 e 1.15 milhões de *chunks*. Já a Figura 4(b) mostra o número médio de *chunks* adicionais requisitados pela solução proposta, isto é, por todos os módulos comparadores de todos os *peers*. Pode-se notar que o número de *chunks* adicionais requisitados esteve sempre em torno de 70.000 a 100.000 *chunks*.

A proporção entre as informações mostradas em ambos os gráficos da Figura 4 indica a percentagem do *overhead* gerado pela solução proposta, em termos da quantidade de *chunks* adicionais requisitados pelo módulos comparadores. Pode-se notar que o módulo comparador adicionou uma sobrecarga de 7% a 8% ao uso do tráfego de rede. É importante lembrar que o intervalo de monitoramento configurado foi de até 15 segundos, e dependendo da largura de banda de rede disponível esta frequência pode ser aumentada ou diminuída.

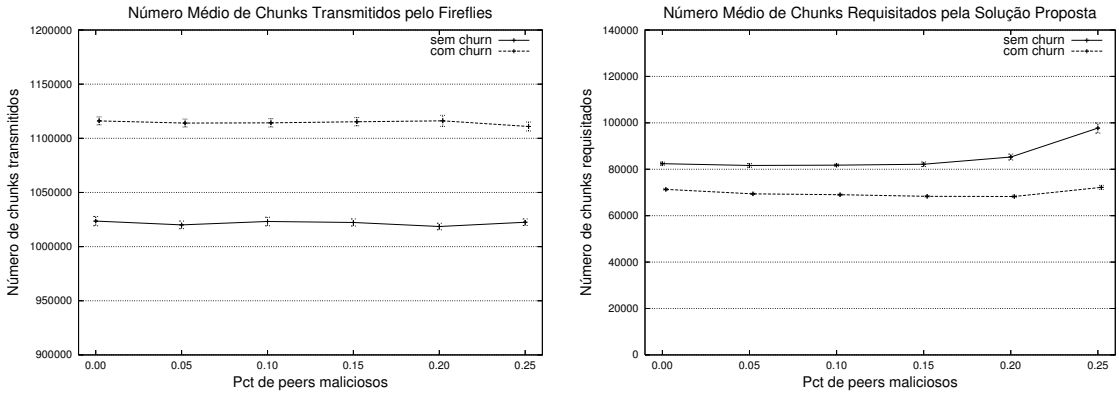
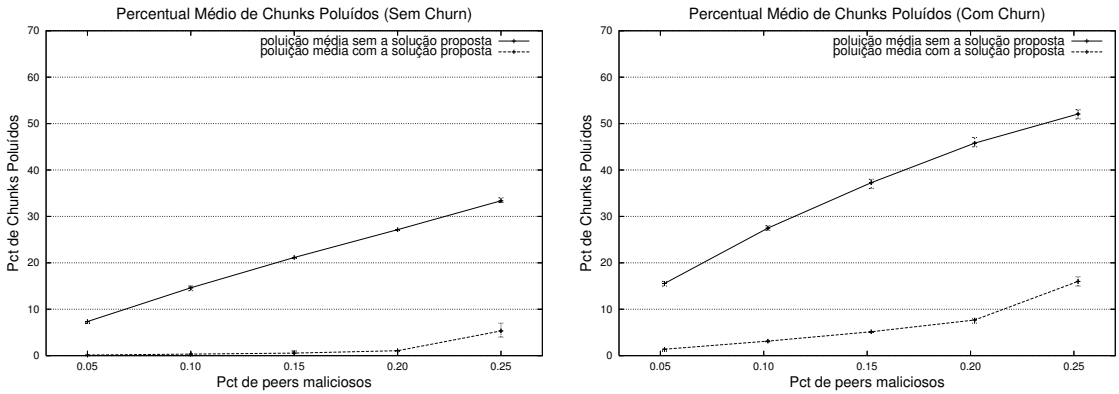


Figura 4. *Chunks* transmitidos normalmente pelo Fireflies e *chunks* adicionais requisitados pelo módulo comparador.

A Figura 5 mostra a quantidade média de *chunks* poluídos durante toda a transmissão para redes sem a solução proposta e para transmissões com a solução proposta

implementada. Pode-se notar que para experimentos sem *churn* com a solução proposta implementada, o percentual médio de *chunks* poluídos durante toda a transmissão caiu de 27.1% para 1% em experimentos onde 20% dos *peers* da rede foram configurados como poluidores, e caiu de 33.3% para 5.3% nos experimentos onde $1/4$ dos *peers* da rede eram *peers* poluidores. Já para experimentos com *churn* o percentual de poluição caiu de 45.7% para 7% em experimentos onde 20% dos *peers* eram poluidores, e caiu de 52% para 16% nos experimentos onde 25% dos *peers* eram poluidores.



(a) (b)

Figura 5. Percentual médio de *chunks* poluídos durante as transmissões em redes sem a solução proposta e com a solução proposta.

As próximas duas figuras mostram o percentual da quantidade de *chunks* poluídos, também para redes com e sem a solução proposta, mas agora durante cada segundo do tempo das transmissões: a Figura 6 mostra a poluição em redes sem a solução proposta, e a Figura 7 mostra a poluição durante as transmissões com a solução proposta ativa.

Em transmissões sem *churn* e sem a solução proposta – Figura 6(a) – pode-se notar que o percentual de *chunks* poluídos tem valores diferentes para cada quantidade de *peers* poluidores na rede, mas de forma geral possui dispersão pequena durante toda a transmissão. Já para as simulações com *churn* – Figura 6(b) – pode-se notar um aumento do percentual de *chunks* poluídos, conforme o tempo de simulação passa da metade do tempo de transmissão – chegando a percentuais próximos de 70% da quantidade de *chunks* poluídos, em transmissões com 25% de *peers* poluidores.

Com base na Figura 7(a) (experimentos sem *churn*) nota-se que, nas transmissões com a solução proposta ativa a poluição praticamente acabou após cerca de 40 segundos de transmissão para as simulações com até 20% de *peers* configurados como poluidores, e nas simulações com 25% de *peers* poluidores, o percentual de *peers* poluídos caiu para cerca de 2% após 80 segundos de transmissão. Casos como este das simulações com 25% de *peers* poluidores onde a poluição ainda continuou com valores acima de 0% ocorrem pois o aumento da quantidade de *peers* poluidores na rede aumenta também a probabilidade da maior parte dos vizinhos de um determinado *peer* serem *peers* poluidores.

A Figura 7(a) ainda mostra que a poluição no sistema foi mais alta apenas nos segundos iniciais das transmissões, ou seja, após as primeiras comparações serem realizadas pelos módulos comparadores, cada *peer* foi capaz de identificar e não solicitar mais dados aos *peers* identificados como poluidores. Além disso, também com base nestes

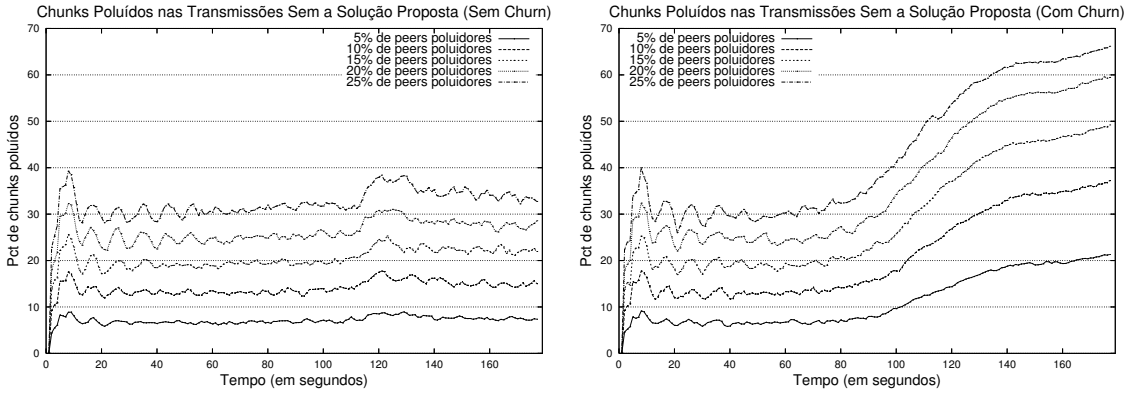


Figura 6. *Chunks* poluídos transmitidos em redes sem a solução proposta, em cada segunda da transmissão.

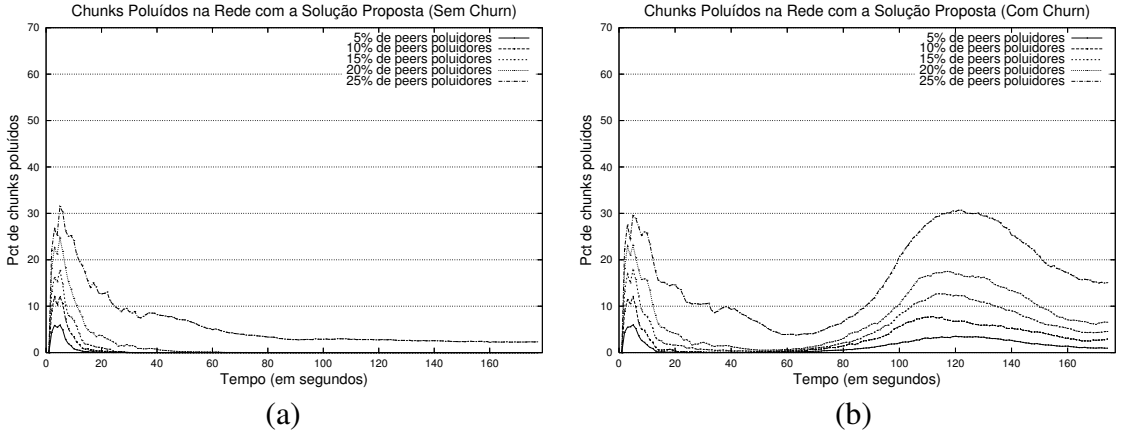


Figura 7. *Chunks* poluídos transmitidos em redes com a solução proposta, em cada segunda da transmissão.

dados, se os 30 ou 40 segundos iniciais de cada transmissão fossem retirados do cálculo da média, o percentual de *chunks* poluídos de toda a transmissão – informação que foi mostrada na Figura 5(b) – reduziria ainda mais significativamente.

Finalmente com base na Figura 7(b) – que mostra o percentual de poluição nas transmissões com a solução proposta nos experimentos com *churn* – nota-se que nos momentos onde ocorrem maiores taxas de entrada de novos *peers* na rede (entre os tempos 100 e 120 segundos) ocorre um aumento no percentual de *chunks* poluídos. Este comportamento ocorre pois os novos *peers* que entram na rede não possuem conhecimento inicial sobre quais dos seus vizinhos são *peers* poluidores. Por outro lado, conforme o tempo de simulação avança e estes novos *peers* iniciam as suas comparações, o percentual de poluição volta a cair novamente. Neste sentido, o experimento sem *churn* mostrado na Figura 7(a) reflete melhor o comportamento geral de cada *peer* quando ele inicia sua participação na transmissão, ou seja, nos momentos iniciais, o *peer* acaba recebendo uma maior quantidade de *chunks* poluídos por desconhecer quais dos seus vizinhos são poluidores, mas conforme o tempo avança – e o módulo comparador realiza e finaliza as comparações – este *peer* identifica e para de solicitar *chunks* a vizinhos poluidores.

5. Trabalhos Relacionados

Diversas estratégias que tratam a poluição de conteúdo – ou *poisoning* [Christin et al. 2005] – em redes P2P foram publicadas na literatura. Algumas das mais relevantes são brevemente descritas a seguir. Uma estratégia básica para combate à poluição de conteúdo, empregada pelo BitTorrent [Dhungel et al. 2007] é permitir que os *peers* obtenham previamente os valores *hash* de todos os *chunks* [Wong and Lam 1999]. Desta forma quando um *chunk* é recebido cada *peer* pode verificar a integridade daqueles dados. Em transmissões ao vivo um dos problemas dessa técnica está em receber previamente o valor *hash* de conteúdos que são gerados durante a própria transmissão.

Outra estratégia consiste na aplicação da criptografia de chave pública, ou seja, disseminar todo *chunk* juntamente com uma assinatura digital correspondente gerada pelo servidor fonte [Haridasan and van Renesse 2006]. A desvantagem dessa estratégia é que principalmente para transmissões ao vivo, dependendo dos dispositivos dos usuários envolvidos nas transmissões, a criptografia de chave publica pode ser proibitiva. Uma variante desta estratégia, chamada de *Linear Digests* também é apresentada em [Haridasan and van Renesse 2006] e agrupa os valores *hash* de um conjunto de *chunks* em uma mesma assinatura digital, que também é gerada pelo servidor fonte.

Algumas outras ferramentas [Liang et al. 2006, Li et al. 2012] ainda aplicam criptografia simétrica a todos os *chunks* usando uma chave secreta compartilhada pré-definida. Em [Li et al. 2012] também é proposto um mecanismo seguro de gerenciamento de chaves no qual o servidor fonte periodicamente recria e retransmite a nova chave compartilhada para um número limitado de *peers* da rede. Já nas técnicas de listas negras [Liang et al. 2005] utilizam faixas de IPs para englobar o maior número possível de *peers* que disseminaram conteúdo poluído. O desafio desta técnica é englobar o menor número de *peers* não poluidores nestas faixas de IPs. Além disso [Gheorghe et al. 2010] aponta que esta técnica se mostra custosa quando aplicadas para transmissões ao vivo.

Em [Chen et al. 2008] é apresentada uma outra solução que emprega o conceito de grupos de *peers* que são mantenedores da integridade do conteúdo transmitido. Nesta solução o servidor fonte publica o conteúdo a este grupo de peers. Todos os demais *peers* podem realizar requisições a este grupo de *peers* para verificar a integridade de um *chunk* específico. Já em [Walsh and Sirer 2006] os autores apresentam um sistema P2P para compartilhamento de arquivos, baseado em reputação. Neste sistema os próprios *peers* da rede classificam outros *peers* como honestos, que por sua vez adquirem acesso ao conteúdo compartilhado. Em [Borges et al. 2008] uma outra solução também baseada no mesmo princípio de reputação é apresentada, mas agora aplicada a transmissões ao vivo.

Em [Dhungel et al. 2009] os autores realizam a avaliação de quatro técnicas mencionadas acima: criptografia simétrica, verificação de *hashes*, assinaturas digitais e lista negra. Os autores concluem que o uso de árvores de Merkle é um dos mecanismos mais eficientes em termos da sobrecarga computacional adicionada. Mais recentemente em [Lin et al. 2010] uma avaliação do impacto de ataques de poluição é realizada. O trabalho mostra que o impacto de um ataque de poluição não está diretamente relacionado ao tamanho da rede em si, mas que depende fortemente dos níveis de *churn* e da banda de rede disponível nos *peers* maliciosos e no servidor fonte.

Uma estratégia baseada em *network coding* [Feng and Li 2008] é apresentada em

[Wang et al. 2010] para identificar e limitar a poluição de conteúdo em transmissões ao vivo em redes P2P. Cada *chunk* transmitido pelo servidor fonte é dividido em blocos. Por sua vez cada bloco é subdividido em palavras (ou *codewords*), que acabam convertendo cada um dos *chunks* em uma matriz de elementos de um campo de Galois (*Galois Field*, ou GF). Por fim blocos codificados – que são criados baseados nas matrizes GF e combinam um vetor de coeficientes aos blocos originais – são as informações transmitidas pelo servidor fonte para os peers. Cada *peer* que recebe estes blocos codificados, os decodificam para reconstruir os *chunks* originais.

Já em [Borges et al. 2012, Oliveira et al. 2009] uma caracterização do tráfego do SopCast é apresentada. Uma das conclusões do trabalho é que um *peer* malicioso foi capaz de comprometer 50% dos *peers* da rede e 30% da banda de rede. Por fim, recentemente em [Coelho et al. 2011] os autores apresentam uma avaliação dos mecanismos de autenticação de conteúdo em redes P2P para transmissão ao vivo. Os autores comparam a sobrecarga gerada e a segurança adicionada por diversas técnicas e mostram que, para transmissões ao vivo de alta resolução, os mecanismos com sobrecarga aceitáveis avaliados não foram resilientes aos ataques de poluição.

6. Conclusão

Este trabalho apresentou uma nova estratégia que utiliza o diagnóstico baseado em comparações para combater a poluição de conteúdo em transmissões ao vivo em redes *overlay*. Cada *peer* do sistema executa comparações periódicas sobre determinados *chunks* de seus vizinhos. Com base nos resultados das comparações, cada *peer*, de forma independente dos demais, deixa de solicitar *chunks* aos seus vizinhos poluidores. A estratégia proposta foi implementada usando o protocolo Fireflies e um grande número de experimentos através de simulações foram conduzidos. Os resultados dos experimentos mostraram que a solução adiciona cerca de 7% a 8% de sobrecarga ao tráfego de rede, e que é uma solução viável para a detecção e contenção da poluição em transmissões ao vivo em redes P2P. Trabalhos futuros incluem implementar a estratégia em um serviço real de transmissões ao vivo na Internet, avaliar a estratégia proposta para ataques do tipo omissão, com uma maior quantidade de peers e com outros intervalos de monitoramento, além de avaliar a estratégia em outros tipos de redes *overlay*.

Referências

Borges, A., Almeida, J. M., and Campos, S. V. A. (2008). Fighting Pollution in P2P Live Streaming Systems. *Proc. of the IEEE Intl. Conf. on Multimedia and Expo*, pages 481–484.

Borges, A., Gomes, P., Nacif, J., Mantini, R., Almeida, J. M., and Campos, S. V. A. (2012). Characterizing SopCast Client Behavior. *Computer Communications*, 35(8):1004–1016.

Chen, R., Lua, E. K., Crowcroft, J., Guo, W., Tang, L., and Chen, Z. (2008). Securing Peer-to-Peer Content Sharing Service from Poisoning Attacks. *Proc. of the 8th IEEE Intl. Conf. on Peer-to-Peer Computing*, pages 22–29.

Christin, N., Weigend, A. S., and Chuang, J. (2005). Content Availability, Pollution and Poisoning in File Sharing Peer-to-Peer Networks. *Proc. of the 6th ACM Conf. on Electronic Commerce*, pages 68–77.

Coelho, R. V., Pastro, J. T., Antunes, R. S., Barcellos, M. P., Jansch-Pôrto, I., and Gaspary, L. P. (2011). Challenging the Feasibility of Authentication Mechanisms for P2P Live Streaming. *Proc. of the 6th Latin America Networking Conference*, pages 55–63.

Dhungel, P., Hei, X., Ross, K. W., and Saxena, N. (2007). The Pollution Attack in P2P Live Video Streaming: Measurement Results and Defenses. *Proc. of the Workshop on Peer-to-peer Streaming and IP-TV*, pages 323–328.

Dhungel, P., Hei, X., Ross, K. W., and Saxena, N. (2009). Pollution in P2P Live Video Streaming. *Intl. Journal of Computer Networks and Communications*, 1(2):99–110.

Duarte Jr., E. P., Ziwich, R. P., and Albini, L. C. P. (2011). A Survey of Comparison-Based System-Level Diagnosis. *ACM Computing Surveys*, 43(3):22:1–22:56.

Feng, C. and Li, B. (2008). On Large-Scale Peer-to-Peer Streaming Systems with Network Coding. *Proc. of the 16th ACM Intl. Conf. on Multimedia*, pages 269–278.

Gheorghe, G., Cigno, R. L., and Montresor, A. (2010). Security and Privacy Issues in P2P Streaming Systems: A Survey. *Peer-to-Peer Networking and Applications*, 4(2):75–91.

Haridasan, M. and van Renesse, R. (2006). Defense Against Intrusion in a Live Streaming Multicast System. *Proc. of the 6th IEEE Intl. Conf. on Peer-to-Peer Computing*, pages 185–192.

Johansen, H., Allavena, A., and van Renesse, R. (2006). Fireflies: Scalable Support for Intrusion-Tolerant Network Overlays. *Proc. of the 1st ACM SIGOPS/EuroSys European Conf. on Computer Systems*, 40(4):3–13.

Li, J.-S., Hsieh, C.-J., and Wang, Y.-K. (2012). Distributed Key Management Scheme for Peer-to-Peer Live Streaming Services. *Intl. Journal of Communication Systems*.

Liang, J., Kumar, R., and Ross, K. W. (2006). The FastTrack Overlay: A Measurement Study. *Computer Networks*, 50(6):842–858.

Liang, J., Naoumov, N., and Ross, K. W. (2005). Efficient Blacklisting and Pollution-Level Estimation in P2P File-Sharing Systems. *Proc. of the Asian Internet Engineering Conf.*, pages 173–175.

Lin, E., de Castro, D. M. N., Wang, M., and Aycock, J. (2010). SPoIM: A Close Look at Pollution Attacks in P2P Live Streaming. *Proc. of the 18th Intl. Workshop on Quality of Service*, pages 1–9.

Loocher, T., Meier, R., Schmid, S., and Wattenhofer, R. (2007). Push-to-Pull Peer-to-Peer Live Streaming. *Proc. of the 21st Intl. Symp. on Distributed Computing*, pages 388–402.

Maeng, J. and Malek, M. (1981). A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems. *Proc. of the 11th IEEE Fault-Tolerant Computing Symp.*, pages 173–175.

Oliveira, J., Borges, A., and Campos, S. V. A. (2009). Content Pollution on P2P Live Streaming Systems. *Proc. of the 15th Brazilian Symp. on Multimedia and the Web*, (50).

Pai, V., Kumar, K., Tamilmani, K., Sambamurthy, V., Mohr, A. E., and Mohr, E. E. (2005). Chainsaw: Eliminating Trees from Overlay Multicast. *Proc. of the 4th Intl. Workshop on Peer-To-Peer Systems*, pages 127–140.

Schimidt, E. A., Ziwich, R. P., Duarte Jr., E. P., and Jansch-Pôrto, I. (2011). Diagnóstico de Poluição de Conteúdo em Redes P2P para Transmissões de Mídia Contínua ao Vivo. *Proc. of the 17th Brazilian Symp. on Multimedia and the Web*, pages 221–228.

Walsh, K. and Sirer, E. G. (2006). Experience with an Object Reputation System for Peer-to-Peer Filesharing. *Proc. of the 3rd USENIX Symp. on Networked Systems Design and Implementation*, 3:1–14.

Wang, Q., Vu, L., Nahrstedt, K., and Khurana, H. (2010). MIS: Malicious Nodes Identification Scheme in Network-Coding-Based Peer-to-Peer Streaming. *Proc. of the 29th IEEE Intl. Conf. on Computer Communications*, pages 1–5.

Wong, C. K. and Lam, S. S. (1999). Digital Signatures for Flows and Multicasts. *IEEE/ACM Trans. on Networking*, 7(4):502–513.

Yang, S., Jin, H., Li, B., Liao, X., Yao, H., and Tu, X. (2008). The Content Pollution in Peer-to-Peer Live Streaming Systems: Analysis and Implications. *Proc. of the 37th Intl. Conf. on Parallel Processing*, pages 652–659.

Zhang, P. and Helvik, B. E. (2011). Modeling and Analysis of P2P Content Distribution under Coordinated Attack Strategies. *IEEE Consumer Communications and Networking Conf.*, pages 131–135.

Ziwich, R. P., Duarte Jr., E. P., and Albini, L. C. P. (2005). Distributed Integrity Checking for System with Replicated Data. *Proc. of the 11th IEEE Intl. Conf. on Parallel and Distributed Systems*, pages 363–369.

Uma Fotografia do Instagram: Caracterização e Aplicação

Thiago H. Silva[1], **Pedro O. S. Vaz de Melo**[1], **Jussara M. Almeida**[1], **Antonio A. F. Loureiro**[1]

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

{thiagohs, olmo, jussara, loureiro}@dcc.ufmg.br

Abstract. *Participatory sensing systems (PSS) have the potential to become fundamental tools for supporting the study, in large scale, of urban social behavior and city dynamics. To that end, this work characterizes the photo sharing system Instagram, considered one of the currently most popular PSSs on the Internet. Based on a dataset of approximately 2.3 million shared photos, we characterize user behavior in the system showing that there are several advantages and opportunities for large scale sensing, such as a global coverage at low cost, but also challenges, such as a very unequal photo sharing frequency, both spatially and temporally. Moreover, we present an application based on data obtained from Instagram to identify regions of interest in a city, which illustrates the promising potential of PSSs for the study of city dynamics.*

Resumo. *Sistemas de sensoriamento participativo (SSP) (participatory sensing systems) têm o potencial de se tornarem ferramentas fundamentais para o estudo em larga escala do comportamento social urbano e da dinâmica de cidades. Nessa direção, este trabalho analisa o Instagram, um sistema para compartilhamento de fotos que é considerado um dos mais populares SSPs disponíveis na Internet atualmente. Baseado em um conjunto de dados de aproximadamente 2,3 milhões de fotos compartilhadas, caracterizamos o comportamento de usuários desse sistema, mostrando que existem muitas vantagens e oportunidades para sensoriamento em grande escala, tais como uma abrangência global a baixo custo, mas também desafios, como uma frequência de compartilhamento de fotos espaço-temporal altamente desigual. Além disso, apresentamos uma aplicação baseada em dados obtidos do Instagram para identificar regiões de interesse dentro de uma cidade. Essa aplicação ilustra o potencial promissor de SSPs para o estudo da dinâmica das cidades.*

1. Introdução

Mark Weiser, no seu trabalho clássico intitulado "*The computer for the 21st century*" publicado na *Scientific American* [Weiser 1991], popularizou o conceito da *computação ubíqua*, que prevê o acesso a ambientes de computação por qualquer pessoa, em qualquer lugar, a qualquer instante, com dispositivos computacionais acoplados aos mais triviais objetos, como etiquetas de roupas, copos de café, canetas ou qualquer objeto pessoal. Embora essa ainda não seja a realidade e esse conceito tenha sido estendido para incluir, por exemplo, Internet das Coisas (*Internet of Things*), muito foi feito nessa direção nestes últimos 20 anos após a publicação do trabalho de Weiser. Por exemplo, as Redes de Sensores Sem Fio (RSSFs) [Akyildiz et al. 2002], que são um tipo especial de rede *ad hoc*, são projetadas para coletar dados referentes a grandezas físicas dos mais variados ambientes em que estão inseridas e a fornecer tais informações para o usuário final. Além disso, há o uso crescente de sistemas de sensoriamento participativo

(SSPs) [Burke et al. 2006], que permitem a pessoas conectadas à Internet fornecerem dados de contexto sobre o ambiente em que estão em um determinado momento.

De fato, os SSPs têm o potencial para complementar as RSSFs em diversos aspectos. Enquanto as RSSFs foram projetadas para sensoriar áreas de tamanho limitado, como florestas e vulcões, os SSPs podem alcançar áreas de tamanhos variados e de larga escala, como grandes metrópoles, países ou até mesmo todo o planeta [Silva et al. 2012a]. Além disso, uma RSSF está sujeita a falhas, uma vez que o seu funcionamento depende da correta coordenação das ações dos seus nós sensores, que possuem severas restrições de energia, processamento e memória. Por outro lado, SSPs são formados por entidades autônomas e independentes, os seres humanos, o que torna a tarefa de sensoriamento altamente resiliente a falhas individuais.

O sucesso dos SSPs está diretamente ligado à popularização do *smartphone*, que se tornou o dispositivo computacional pessoal mais amplamente adotado e onipresente [Krumm 2009]. Os *smartphones* possuem um rico conjunto de sensores embutidos, tais como GPS, acelerômetro, microfone, câmera, giroscópio e bússola digital. Entretanto, o sensoriamento não depende apenas dos dados gerados por esses sensores, podendo vir também das observações subjetivas do seu usuário. É possível encontrar vários exemplos de SSPs já implantados e usados através de *smartphones*, como o Waze[1], para relatar condições de tráfego em tempo real, e o Weddar[2], para relatar condições meteorológicas. Além disso, há serviços de compartilhamento de localização, como o Foursquare[3], ou de fotos, como o Instagram[4], nos quais os usuários podem enviar imagens em tempo real para o sistema. Em particular, o Instagram é um dos mais populares SSPs atuais, com quase 100 milhões de usuários e mais de 1 bilhão de fotos recebidas, sendo que, a cada segundo, um novo usuário se registra no sistema e 58 novas fotos são inseridas [Daniells 2012].

O objetivo principal deste trabalho é caracterizar a rede de participação do Instagram, visando mostrar os desafios e as oportunidades que emergem do sensoriamento participativo realizado pelos usuários desta aplicação. Baseado em um conjunto de dados de aproximadamente 2,3 milhões de fotos, mostramos a abrangência planetária da rede, assim como a frequência altamente desigual do compartilhamento de fotos, tanto espacial quanto temporalmente, que é bastante correlacionada com as rotinas de atividades humanas. Além disso, mostramos também como é possível projetar aplicações a partir de sistemas como o Instagram, apresentando uma aplicação para identificar regiões de interesse dentro de uma cidade. Essa aplicação ilustra o potencial de SSPs para o estudo da dinâmica de cidades. Até onde sabemos, este é o primeiro trabalho de caracterização do uso do Instagram, particularmente com o foco no seu potencial, como um sistema de sensoriamento participativo no projeto de novas aplicações e serviços.

O restante deste trabalho é organizado como segue. A seção 2 apresenta os trabalhos relacionados. A seção 3 discute a participação dos seres humanos no processo de sensoriamento, abordando os sistemas participativos de sensoriamento e as redes de sensores participativos (RSPs), advindas de SSPs. A seção 4 apresenta a caracterização da RSP derivada do Instagram. A seção 5 descreve uma aplicação de classificação de regiões usando esta RSP. Finalmente, a seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O processo de sensoriamento do meio ambiente pode envolver seres humanos como: (i) alvos do processo [Larson et al. 2011], ou (ii) responsáveis por eles a partir do compartilhamento

[1]http://www.waze.com
[2]http://www.weddar.com
[3]http://www.foursquare.com
[4]http://www.instagram.com

de dados locais [Srivastava et al. 2012, Goodchild 2007]. Neste trabalho, focamos no segundo caso, considerando sistemas que utilizam dispositivos móveis do dia a dia, como *smartphones*, para construir uma rede de sensoriamento participativa, que é descrita na seção 3.2. Na literatura, existem diversas propostas de sistemas que consideram a participação de humanos no processo de sensoriamento, como descrito anteriormente. Tais sistemas são chamados de sistemas de sensoriamento participativo (SSPs) e incluem, por exemplo, sistemas de monitoramento de tráfego [Eisenman et al. 2010] e monitoramento de ruídos [Rana et al. 2010].

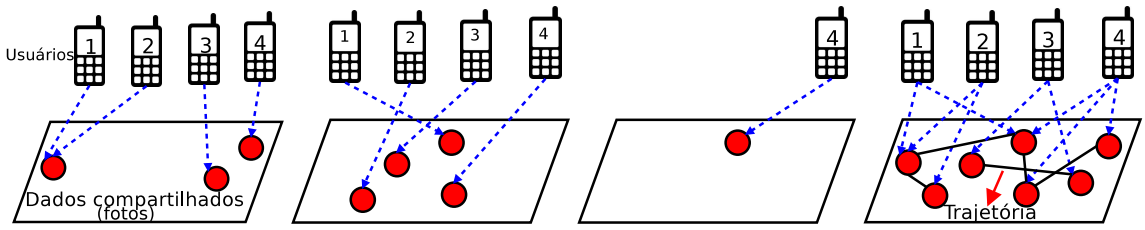
O sucesso de SSPs depende fundamentalmente da participação sustentável dos usuários ao longo do tempo. Em [Reddy et al. 2010], os autores propõem mecanismos de incentivos baseados em micro-pagamentos (*micro-payments*), que são pequenas quantias de dinheiro dadas ao usuário quando ele realiza determinadas atividades no sistema. Além da participação sustentável, é necessário garantir a qualidade dos dados compartilhados pelos usuários [Mashhadi and Capra 2011]. Por exemplo, em diversos SSPs os usuários podem fabricar dados falsos, que supostamente foram sensoriados, a baixo custo. Logo, a integridade dos dados não é sempre garantida [Saroiu and Wolman 2010].

Existem vários trabalhos dedicados ao estudo das características de SSPs específicos. Por exemplo, em serviços de compartilhamento de localização, como o Foursquare, os autores de [Cheng et al. 2011] observaram que os usuários seguem um padrão de mobilidade simples e factível de ser reproduzido. Nessa direção, os autores de [Cho et al. 2011] observaram que viagens de curta distância são periódicas no espaço e no tempo e não são afetadas pela estrutura social da rede, que, por sua vez, influencia somente as viagens de longa distância. De fato, Scellato et al. [Scellato et al. 2011] mostraram que 40% das relações sociais no sistema analisado acontecem a menos de 100 km. Em [Noulas et al. 2011a], os autores analisaram a dinâmica de compartilhamento dos usuários de serviços de compartilhamento de localização mostrando, por exemplo, que a distribuição do número de check-ins é altamente desigual, sendo bem modelada por um comportamento de lei de potência (*power-law*). Em [Vasconcelos et al. 2012], os autores analisaram o uso de *tips* no Foursquare, que são comentários curtos sobre determinado local, caracterizando como os usuários interagem entre si utilizando esta funcionalidade e propuseram um algoritmo para estimar a influência de usuários. Diferentemente dos demais trabalhos, os autores não exploraram características geográficas.

Outros trabalhos propõem utilizar dados derivados de SSPs em novas aplicações, já que esse tipo de dado auxilia no melhor entendimento das fronteiras físicas e noções de espaço [Bilandzic and Foth 2012]. Nessa direção, os autores de [Cranshaw et al. 2012] apresentaram um modelo que classifica regiões de uma cidade a partir de padrões de atividades coletivas, enquanto Noulas et al. [Noulas et al. 2011b] propuseram classificar áreas e usuários de uma cidade usando as categorias dos locais registrados no Foursquare.

Em trabalho anterior [Silva et al. 2012a], analisamos as propriedades de RSPs derivadas de duas aplicações de compartilhamento de localização: Gowalla e Brightkite. Nós analisamos as distribuições espacial e temporal dos *check-ins* realizados pelos usuários desses sistemas, visando levantar evidências relevantes para o projeto de novos serviços e aplicações. Já em [Silva et al. 2012b], propusemos uma nova forma de visualizar a dinâmica de cidades a partir de hábitos e rotinas de usuários, derivados dos *check-ins* no Foursquare.

O trabalho aqui apresentado diferencia-se dos anteriores (incluindo os nossos) por focar em um novo sistema de grande popularidade atualmente – o Instagram. Até onde sabemos, esta é a primeira caracterização de uma aplicação de compartilhamento de fotos acessada, majoritariamente, a partir de *smartphones*. Mais ainda, dando continuidade aos trabalhos recentes [Silva



(a) Tempo 1 (b) Tempo 2 (c) Tempo 3 (d) Tempo total

Figura 1. RSP analisada: serviço de compartilhamento de fotos

et al. 2012a, Silva et al. 2012b], este trabalho também aborda o estudo da dinâmica das cidades através de SSPs, mostrando que sistemas de compartilhamento de fotos, particularmente o Instagram, também podem ser utilizados para este propósito.

3. Humanos no Processo de Sensoriamento

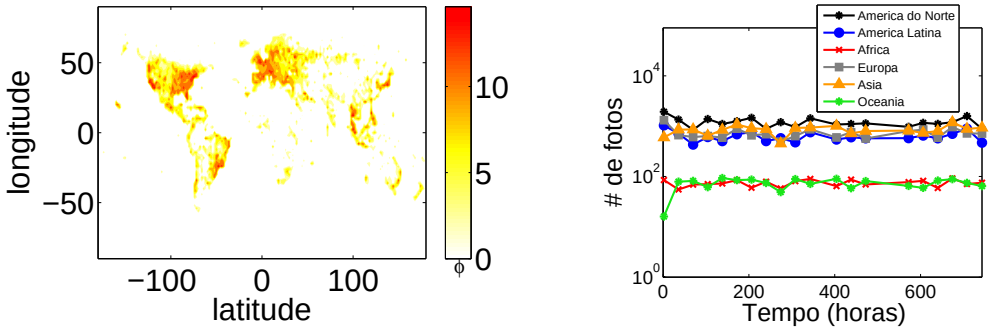
O foco deste trabalho é em sistemas que dependem de humanos no processo de sensoriamento, sendo eles responsáveis pelo compartilhamento de dados locais. Tais dados podem ser obtidos com o auxílio de dispositivos de sensoriamento, como sensores embutidos em celulares (por exemplo, GPS), ou através de sensores humanos (por exemplo, visão), compartilhando informações produzidas por eles próprios.

3.1. Sensoriamento Participativo

Sensoriamento participativo é o processo em que seres humanos usam dispositivos móveis e serviços de computação em nuvem para compartilhar dados sensoriados [Burke et al. 2006]. Usualmente, sistemas de sensoriamento participativo consideram que o compartilhamento dos dados é gerado automaticamente, ou passivamente, por sensores embutidos no dispositivo móvel. Porém, neste trabalho consideramos também observações geradas pelos usuários de forma manual, ou proativa. Sensoriamento participativo com essas características também pode ser chamado de *crowdsourcing* ubíquo (*ubiquitous crowdsourcing*) [Mashhadi and Capra 2011]. A popularidade de sistemas para sensoriamento participativo cresceu rapidamente com o aumento do uso de celulares com sensores embutidos e capacidade de acesso à Internet, ou seja, os chamados *smartphones*. Esses dispositivos se tornaram uma poderosa plataforma que inclui capacidades de sensoriamento, computação e comunicação.

Um dado sensoriado em uma aplicação de sensoriamento participativo é: (i) obtido através de sensores físicos (por exemplo, GPS) ou observações humanas (por exemplo, congestionamento na rodovia); (ii) definido no tempo e no espaço; (iii) obtido automaticamente ou manualmente; (iv) estruturado ou não; (v) compartilhado voluntariamente ou não. Para ilustrar esse tipo de sistema, considere uma aplicação para monitoramento de trânsito, como o Waze. Usuários podem compartilhar observações sobre acidentes ou congestionamentos manualmente. Uma aplicação poderia ainda calcular e compartilhar automaticamente a velocidade de um carro com o auxílio de dados obtidos com o GPS. Com as medidas da velocidade de diferentes veículos amostrados em uma área particular, é possível inferir, por exemplo, congestionamentos. Como nesse caso específico usuários operam uma aplicação que foi criada para esse propósito, o dado sensoriado é estruturado. Mas, caso os usuários usem um serviço de *microblogging*, como o Twitter, o dado sensoriado seria não estruturado. Por exemplo, o usuário "João" envia uma mensagem "trânsito agora está muito lento próximo da portaria do campus".

Serviços de compartilhamento de fotos, como o Instagram, são exemplos de aplicações para sensoriamento participativo. O dado sensoriado é uma foto de um lugar específico. Pode-



(a) Número de fotos n por pixel dado pelo valor de ϕ mostrado na figura, em que $n = 2^{\phi} - 1$.

(b) Variação temporal do número de fotos compartilhadas por continente.

Figura 2. Cobertura da RSP do Instagram.

mos extrair informação desse tipo de dado de diversas maneiras. Por exemplo, é possível visualizar em tempo real como está a situação de uma certa área da cidade.

3.2. Rede de Sensoriamento Participativo

Em uma rede de sensoriamento participativo (RSP), 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. De forma similar às RSSFs, o dado sensoriado é enviado para o servidor, ou "nó sorvedouro". Mas, diferentemente das RSSFs, RSPs têm as seguintes características: (a) nós são entidades móveis autônomas, mas uma pessoa com um dispositivo móvel; (b) o custo da rede é distribuído entre os nós, proporcionando uma escalabilidade global; (c) o sensoriamento depende da vontade das pessoas participarem no processo de sensoriamento; (d) nós transmitem o dado sensoriado diretamente para o sorvedouro; (e) nós não sofrem de severas limitações de energia; (f) o nó sorvedouro só recebe dados e não tem controle direto sobre os nós.

A figura 1 mostra um exemplo de RSP formada a partir de serviços de compartilhamento de fotos, que é a RSP analisada nas seções seguintes. As figuras 1-a, 1-b e 1-c representam quatro usuários em três diferentes momentos. Fotos compartilhadas pelos usuários a cada momento são marcadas por uma seta pontilhada. Observe que nem todos os usuários realizam atividades em todos os momentos. Depois de um certo intervalo, podemos analisar os dados de diversas maneiras. Por exemplo, a figura 1-d mostra um grafo onde os vértices representam os locais onde as fotos foram compartilhadas e as arestas conectam fotos compartilhadas pelo mesmo usuário. Com esse grafo é possível extrair várias informações interessantes de diferentes partes do mundo, fornecendo uma notável escala global a uma infraestrutura de baixo custo, como ilustrado na figura 2-a.

4. Caracterização do Instagram

Nesta seção nós analisamos uma rede de sensores participativa (RSP) derivada do Instagram.

4.1. Descrição dos Dados

A RSP analisada é derivada de um conjunto de dados do Instagram, que é um serviço de compartilhamento de fotos online. Os dados do Instagram foram coletados através do Twitter[5], que é um serviço de *microblogging*, ou seja, ele permite que os seus usuários enviem e recebam atualizações pessoais de outros contatos em textos de até 140 caracteres, conhecidos como

[5]http://www.twitter.com

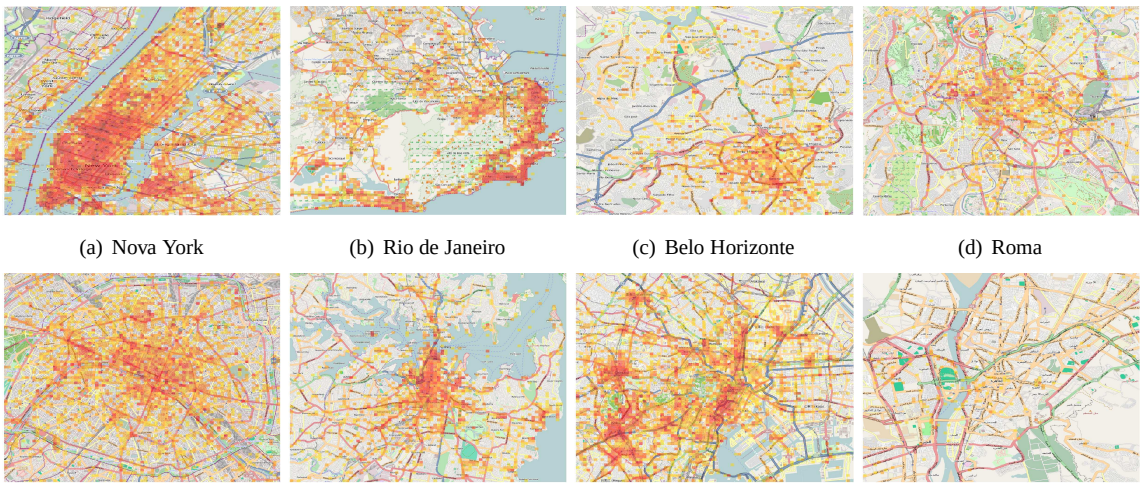
"*tweets*". Além de *tweets* de texto simples, os usuários também podem compartilhar fotos a partir de uma integração com o Instagram. Neste caso, fotos do Instagram anunciadas no Twitter passam a ficar disponíveis publicamente, o que por padrão não acontece quando a foto é publicada unicamente no sistema do Instagram.

Entre 30 de junho e 31 de julho de 2012, foram coletados 2 272 556 *tweets* contendo fotos georeferenciadas, postadas por 482 629 usuários. Cada *tweet* é composto de coordenadas GPS (latitude e longitude) e o horário do compartilhamento da foto.

4.2. Cobertura da Rede

Nesta seção, analisamos a cobertura da RSP do Instagram em diferentes granularidades espaciais, começando por todo o planeta, depois por continentes e cidades e, por fim, até áreas específicas de uma cidade. A figura 2-a mostra a cobertura no planeta da RSP do Instagram na forma de um mapa de calor da participação dos usuários: cores mais escuras representam um maior número de fotos compartilhadas em determinada área. Apesar da cobertura ser bastante abrangente na escala planetária, ela não é homogênea. A figura 2-b mostra o número de fotos compartilhadas por continente ao longo do tempo. Note que a atividade de sensoriamento nas Américas, Europa e Ásia é pelo menos uma ordem de magnitude maior que na África e Oceania. Além disso, pode-se observar ainda que a participação dos usuários da América do Norte é levemente superior a dos usuários da América Latina, Europa e Ásia.

Avaliamos agora a participação dos usuários do Instagram em oito grandes e populosas cidades localizadas em cinco continentes: Nova York, Rio de Janeiro, Belo Horizonte (BH), Roma, Paris, Sydney, Tokyo, e Cairo. A figura 3 mostra o mapa de calor da atividade de sensoriamento para cada uma dessas cidades. Mais uma vez, cores mais escuras representam um maior número de fotos em determinada área. Pode-se observar uma alta cobertura para algumas cidades, como mostrado nas figuras 3-a (Nova York), 3-e (Paris) e 3-g (Tokyo). No entanto, pode-se observar na figura 3-h que o sensoriamento no Cairo, que também possui um número elevado de habitantes, é significativamente mais baixo. Tamanha diferença na cobertura pode ser explicada por diversos fatores. Além dos aspectos econômicos, diferenças na cultura dos habitantes desta cidade quando comparadas com as culturas presentes nas outras cidades estudadas podem ter um impacto significativo na adoção e uso do Instagram [Barth 1969].



(a) Nova York (b) Rio de Janeiro (c) Belo Horizonte (d) Roma

(e) Paris (f) Sydney (g) Tokyo (h) Cairo

Figura 3. Cobertura espacial da RSP do Instagram em 8 cidades: todas as fotos compartilhadas. O número de fotos em cada área é representado por um mapa de cores, onde a escala vai de amarelo a vermelho (atividade mais intensa).

Além disso, pode-se observar que a cobertura no Rio de Janeiro e em Sydney é bem mais heterogênea quando comparada com a cobertura em Paris, Tóquio e Nova York. Isto ocorre provavelmente por causa dos aspectos geográficos que estas cidades têm em comum, ou seja, grandes áreas verdes e grandes porções d'água. Rio de Janeiro, por exemplo, tem a maior floresta urbana do mundo, localizada no meio da cidade, além de muitas colinas de difícil acesso humano. Estes aspectos geográficos limitam a cobertura do sensoriamento. Além disso, em ambas as cidades os pontos de interesse público, tais como pontos turísticos e centros comerciais, são distribuídos de forma desigual pela cidade. Há grandes áreas residenciais com poucos pontos desse tipo, enquanto outras áreas têm grande concentração desses pontos. Estes resultados são qualitativamente semelhantes aos reportados em [Silva et al. 2012a, Silva et al. 2012b] para RSPs derivadas de três aplicações de compartilhamento de localização e para diferentes cidades, o que demonstra o potencial do Instagram como ambiente para sensoriamento participativo em grandes regiões.

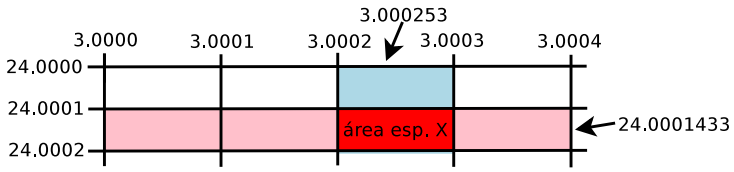


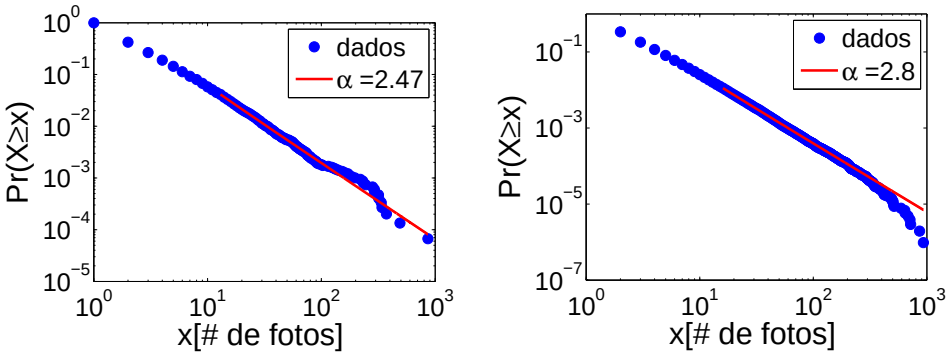
Figura 4. Exemplo de identificação de uma área específica

Uma vez que a atividade de participação pode ser bastante heterogênea dentro de uma cidade, propomos dividir a área de cidades em espaços retangulares menores, como em uma grade[6]. Chamaremos cada área retangular de uma *área específica* dentro de uma cidade e, a partir disso, analisaremos o número de fotos compartilhadas nessas áreas específicas. Neste trabalho, consideramos que uma área específica possui a seguinte delimitação: $1{\cdot}10^{-4^\circ}$ (latitude) $\times$ $1{\cdot}10^{-4^\circ}$ (longitude). Isso representa uma área de aproximadamente 8×11 metros em NY e 10×11 metros no Rio de Janeiro. Para outras cidades, as áreas também podem variar um pouco, mas não a ponto de afetar significativamente as análises realizadas. A figura 4 ilustra o processo de divisão da área de uma cidade em áreas específicas e de como é feita a associação da coordenada geográfica (24,0001433; 3, 000253) a uma área específica X.

A figura 5 apresenta a função de distribuição acumulada complementar (CCDF) do número de fotos compartilhadas por área específica da cidade de Nova York (figura 5-a) e de todas as localidades em nossa base de dados (figura 5-b). Primeiramente, observe que, em ambos os casos, uma lei de potência[7] descreve bem esta distribuição. Isso implica que, na maioria das áreas específicas, há poucas fotos compartilhadas, enquanto existem algumas poucas áreas com centenas de fotos compartilhadas. Estes resultados estão consistentes com os resultados apresentados em [Noulas et al. 2011a, Silva et al. 2012a], que estudaram a participação de usuários em sistemas de compartilhamento de localização. Em sistemas para compartilhamento de fotos, assim como em sistemas de compartilhamento de localizações, é natural que algumas áreas possuam mais atividade que outras. Por exemplo, em áreas turísticas o número de fotos compartilhadas tende a ser maior do que em um supermercado, apesar de um supermercado ser geralmente um local bastante popular. Se uma determinada aplicação requer uma cobertura mais abrangente, é necessário incentivar os usuários a participarem em locais que eles usualmente não o fariam. Micro-pagamentos ou sistemas de pontuação são exemplos de alternativas que poderiam funcionar nesse caso.

[6]Note que nas áreas selecionadas não é considerado fronteiras.

[7]Matematicamente, uma quantidade x segue uma lei de potência se ela pode ser obtida de uma distribuição de probabilidade $p(x) \propto x^{-\alpha}$, onde α é um parâmetro constante conhecido como expoente ou parâmetro escalar, e é um valor tipicamente entre $2 < \alpha < 3$.



(a) NY (b) Todas as localidades

Figura 5. Distribuição do número de fotos em áreas específicas

Como foi mostrado anteriormente, uma RSP pode ter uma cobertura em escala planetária. No entanto, foi mostrado também que essa cobertura pode ser bastante heterogênea, em que grandes áreas ficam praticamente descobertas. A figura 6 mostra a cobertura da rede total considerando a dimensão temporal, ou seja, o número de localidades que estão ativas (i.e., sensoriadas) em um determinado intervalo de tempo, considerando todos os dados disponíveis. O número máximo de áreas específicas sensoriadas por hora corresponde a aproximadamente somente 0,2% do número total de áreas em nossa base de dados (1 030 558). Em outras palavras, a cobertura instantânea da RSP do Instagram é muito limitada quando consideramos todas as localidades que poderiam ser sensoriadas no planeta[8]. Isso significa que a probabilidade de uma área específica aleatória ser sensoriada em um horário aleatório é bem baixa.

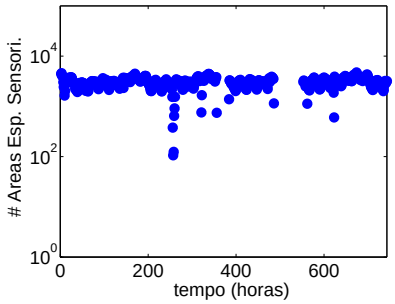


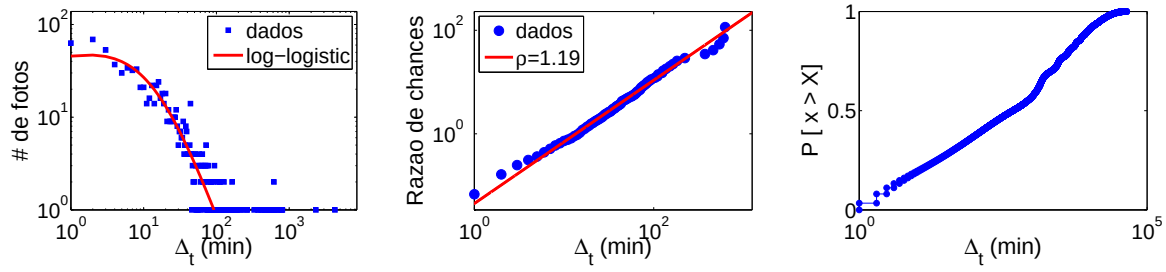
Figura 6. Variação temporal do número de áreas específicas sensoriadas.

4.3. Intervalo de Sensoriamento

Redes de sensoriamento participativo são bastante escaláveis porque seus nós são autônomos, ou seja, os usuários são responsáveis pela sua própria operação e funcionamento. Como o custo da infraestrutura é distribuído entre os participantes, esta enorme escalabilidade e cobertura é alcançada mais facilmente. O sucesso desse tipo de rede consiste em ter participação sustentável e de alta qualidade. Em outras palavras, o sensoriamento é eficiente desde que os usuários sejam mantidos motivados a compartilharem seus recursos e dados sensoriados frequentemente.

Investigamos agora a frequência com que usuários do Instagram realizam o compartilhamento de fotos. A figura 7-a mostra o histograma do intervalo de tempo Δ_t entre compartilhamentos de fotos consecutivos em uma determinada área específica tipicamente popular. Note que o histograma é bem ajustado por uma distribuição *log-logistic* [Fisk 1961]. Observe as rajadas de atividade e os longos períodos de inatividade: há momentos em que muitas fotos são

[8]Considerando nesse caso todas as localidades já sensoriadas pelo menos uma vez.



(a) Histograma (b) Razão de chances (c) Distribuição acumulada

Figura 7. Distribuição do intervalo de tempo entre compartilhamentos de fotos em uma área específica popular.

compartilhadas em intervalos de poucos minutos e momentos em que não há compartilhamento por horas. Isso pode indicar que a maioria do compartilhamento de fotos, nesta área popular (assim como em outras), acontece em intervalos específicos, provavelmente relacionados ao horário em que as pessoas usualmente as visitam. Por exemplo, o compartilhamento de fotos em restaurantes tende a acontecer mais nos horários de almoço e jantar. Aplicações baseadas nesse tipo de sensoriamento devem considerar que a participação do usuário pode variar significativamente ao longo do tempo.

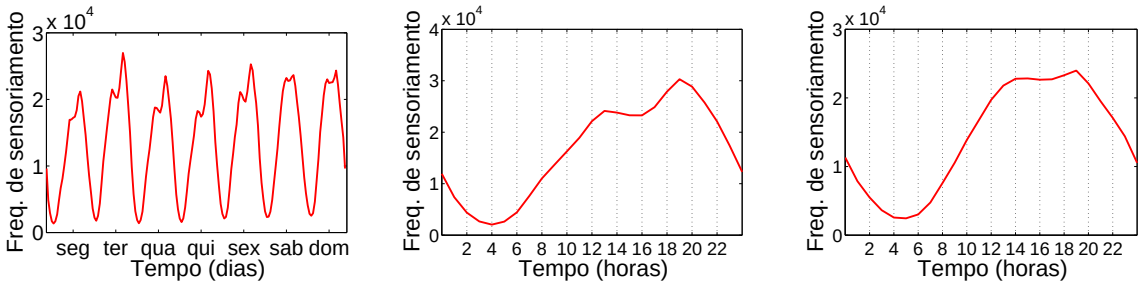
Outra observação interessante relacionada ao intervalo de tempo Δ_t entre compartilhamentos pode ser extraída da figura 7-b, que mostra a função razão de chances[9] (RC) desses intervalos. A RC é uma função acumulada que mostra claramente o comportamento cumulativo de uma dada distribuição na cabeça quanto na cauda. Sua fórmula é $RC(\Delta_t) = \frac{CDF(\Delta_t)}{1-CDF(\Delta_t)}$, onde $CDF(\Delta_t)$ é a função de densidade acumulada da distribuição sendo analisada, no caso a distribuição dos intervalos de tempo Δ_t entre compartilhamentos. Como em [Vaz de Melo et al. 2011], a RC do intervalo de tempo entre fotos compartilhadas também mostra um comportamento de lei de potência com inclinação $\rho \approx 1$. Isso sugere que os mecanismos por trás das atividades humanas podem ser mais simples e gerais do que aqueles propostos na literatura, pois dependem de uma grande quantidade de parâmetros [Malmgren et al. 2008]. A figura 7-c mostra a distribuição do intervalo entre eventos. Podemos observar que uma fatia significativa dos usuários realiza compartilhamento consecutivo de fotos em um curto intervalo de tempo. Cerca de 20% de todo compartilhamento observado acontece em até 10 minutos. Como será discutido na seção 4.5, isso sugere que os nós tendem a compartilhar mais de uma foto na mesma área.

4.4. Sazonalidade

Analisamos agora como a rotina dos humanos afeta o compartilhamento dos dados. A figura 8-a mostra o padrão semanal de compartilhamento de fotos do Instagram[10]. Como esperado, a atuação da rede de participação apresenta um padrão diurno, o que implica que durante a madrugada a atividade de sensoriamento é bastante baixa. Considerando dias de semana, é possível observar um ligeiro aumento da atividade ao longo da semana, com exceção de terça-feira, quando há um pico de atividade. No trabalho [Cheng et al. 2011], que analisou sistemas para compartilhamento de localização, foi observado esse mesmo comportamento, sem nenhum dia como exceção. Isso sugere que durante o período de coleta pode ter ocorrido um evento atípico durante a terça-feira que gerou muitos compartilhamento de fotos. Por fim, pode-se

[9] *Odds ratio function.*

[10] O horário do compartilhamento foi normalizado de acordo com o local onde a foto foi tirada, utilizando para isso a informação geográfica do local.



(a) Padrão semanal (b) Agregado - Seg. a Sex. (c) Agregado - Sáb. a Dom.

Figura 8. Padrão do compartilhamento de fotos durante os dias da semana

observar dois picos de atividades ao longo do dia, por volta dos horários de almoço e jantar. Diferentemente do comportamento observado para o compartilhamento de localizações [Cheng et al. 2011], não foi observado picos de atividade no compartilhamento de fotos por volta do horário do café da manhã.

Analisamos ainda os diferentes padrões de comportamento para dias de semana e final de semana. A figura 8-b mostra o número médio de fotos compartilhadas por hora, de segunda-feira a sexta-feira. A figura 8-c também mostra a mesma informação para sábado e domingo. Como podemos observar, os picos durante os dias de semana acontecem por volta de 13:00 (almoço) e 19:00 (jantar). Já no final de semana não é observado um pico de atividade claro durante o horário do almoço. Pelo contrário, a atividade permanece intensa durante toda a tarde até o início da noite, com um ligeiro aumento por volta das 19:00.

4.5. Comportamento dos Nós

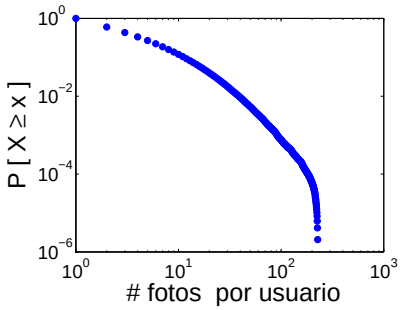
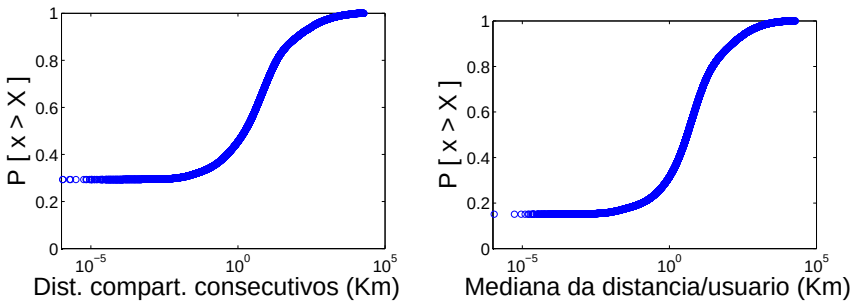


Figura 9. Distribuição do número de fotos compartilhadas pelos usuários



(a) Todas as distâncias (b) Mediana da distância por usuário

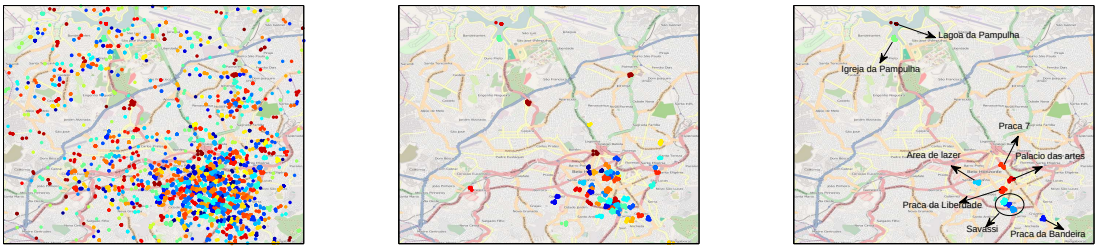
Figura 10. Distribuição da distância geográfica entre fotos consecutivas de um mesmo usuário.

Nesta seção é analisado o desempenho dos nós da RSP (i.e., dos usuários) quanto ao

compartilhamento de fotos. A figura 9 mostra que a distribuição do número de fotos compartilhadas por cada usuário da nossa base de dados possui cauda pesada, significando que a participação dos usuários pode variar muito. Por exemplo, aproximadamente 40% dos usuários contribuíram com apenas uma foto no período considerado, enquanto que somente 17% e 0.1% dos usuários contribuíram com mais que 10 e 100 fotos, respectivamente. É natural que essa variabilidade aconteça por diversos motivos. Por exemplo, alguns usuários podem dar mais importância para quesitos de privacidade do que outros.

Analisamos também a distância geográfica entre dois compartilhamentos de fotos consecutivos pelo mesmo usuário, usando, para tal, as coordenadas geográficas associadas a cada foto. A figura 10-a mostra a função de densidade acumulada da distância geográfica entre cada par de fotos consecutivas compartilhadas por cada usuário do nosso conjunto de dados. Pode-se observar que uma fatia significativa (aproximadamente 30%) das distâncias entre fotos consecutivas são muito curtas (menos de 1 metro). Isso indica que os usuários tendem a compartilhar várias fotos no mesmo local. Essa hipótese é reforçada pela significativa fatia de intervalos de tempo entre fotos consecutivas de curta duração mostrada na figura 7-c: 20% destes intervalos (Δ_t) não ultrapassam 10 minutos. Isso não foi observado na mesma proporção para o compartilhamento de localização. Em [Noulas et al. 2011a], por exemplo, foi observado que 20% dos compartilhamentos de localizações acontece em até 1 km de distância. Para o compartilhamento de fotos, esse valor chega a aproximadamente 45%. Esse resultado pode ser explicado pelo simples fato de que uma foto pode conter muito mais informações que uma localização. Por exemplo, em um restaurante os usuários poderiam compartilhar fotos dos amigos presentes, da comida, ou de uma situação particular, mas tenderiam a compartilhar sua localização apenas uma única vez.

Por fim, analisamos cada usuário separadamente. A figura 10-b mostra a distribuição das medianas das distâncias entre compartilhamentos consecutivos computadas para cada usuário. Note que pelo menos 50% das fotos consecutivas de uma parcela significativa de usuários (aproximadamente 20%) são tiradas a uma distância muito pequena ($\approx$1 metro).



(a) Todos os clusters (b) Pontos de interesse (c) Pontos turísticos

Figura 11. Pontos de Interesse de Belo Horizonte

5. Aplicação

É bastante comum existirem áreas em uma cidade que despertam um maior interesse dos residentes ou visitantes, os aqui denominados *pontos de interesse* (PDI). Dentre os PDIs mais visitados, podemos mencionar os pontos turísticos da cidade. No entanto, nem todos os PDIs de uma cidade são pontos turísticos. Por exemplo, uma área de bares pode ser bastante popular entre os residentes da cidade, mas sem atrativos turísticos. Além disso, PDIs são dinâmicos, ou seja, áreas que são populares hoje podem não o ser amanhã.

Assim, uma aplicação que emerge naturalmente a partir da análise de dados do Instagram é de identificação de PDIs em uma cidade. Isso é possível porque cada foto representa, implicitamente, um interesse de um indivíduo em um determinado instante. Com isso, quando muitas fotos de um determinado local são compartilhadas em um determinado instante, pode-se inferir que esse local é um PDI (observe a figura 5). Mais especificamente, o processo de identificação de PDIs envolve os seguintes passos:

1. Considera-se que cada par i de coordenadas (longitude, latitude) $(x, y)_i$ está associada a um ponto p_i;
2. calcula-se a distância [Sinnott 1984] entre cada par de pontos (p_i, p_j);
3. agrupa-se todos os pontos p_i que possuem uma distância inferior a 250 metros em um *cluster* C_k. Essa distância limite foi obtida através do método Acoplamento Completo (Complete-Linkage) [Sørensen 1948]. O resultado desse procedimento é exibido na figura 11-a. Nessa figura cores diferentes[11] representam diferentes clusters C_k para a cidade de Belo Horizonte;
4. para cada cluster C_k, consideramos apenas um ponto (foto) por usuário. Com isso, a popularidade de um cluster se baseia no número de diferentes usuários que compartilharam uma foto na área do cluster. Este procedimento evita considerar as áreas visitadas por poucos usuários, por exemplo casas, como populares;
5. para cada cluster C_k, cria-se um cluster alternativo C_r, assim $r = k$. A princípio cada cluster alternativo não possui nenhuma foto associada a ele. Em seguida, para cada foto f_i, escolhemos um cluster alternativo C_r de forma aleatória e atribuímos f_i a C_r. Após distribuir todas as fotos f_i seguindo esse procedimento, comprovamos que a distribuição do número de fotos atribuídas para cada cluster C_r é explicada por uma distribuição normal com média μ e desvio padrão σ;
6. por fim, dos clusters originais C_k encontrados a partir do passo 3, excluímos todos aqueles em que o número de fotos do mesmo está a uma distância 2σ da média μ, ou seja, está no intervalo $[\mu - 2\sigma; \mu + 2\sigma]$. De acordo com a regra dos três sigmas (*three-sigma rule*) esse intervalo representa ≈95% da distribuição de fotos aleatórias nos clusters C_r. A ideia deste passo é excluir aqueles clusters que podem ter sido gerados por situações aleatórias, ou seja, clusters que provavelmente não refletem um PDI da cidade.

Os PDIs obtidos através desse processo são mostrados na figura 11-b. Além de identificar PDIs em uma cidade, podemos separar dos PDIs os pontos turísticos. Para isso, primeiramente geramos um grafo $G(V, E)$, onde os vértices $v_i \in V$ são todos os PDIs e uma aresta (i, j) existe do vértice v_i para o vértice v_j se em um determinado momento um usuário compartilhou uma foto em um PDI v_j, logo após ter compartilhado uma foto no PDI v_i. O peso $w(i, j)$ de uma aresta representa o número total de transições realizadas do PDI v_i para o PDI v_j, considerando as transições de todos os usuários. Para identificar pontos turísticos, consideramos que grande parte dos turistas seguem uma trajetória bem conhecida dentro da cidade, sendo guiada pelos principais pontos turísticos da mesma. Além disso, em cada ponto turístico ele tira uma ou mais fotos e parte para o ponto turístico seguinte. Dessa maneira, consideramos que arestas (i, j) com pesos $w(i, j)$ altos denotam essas transições frequentes de um ponto turístico para outro em uma cidade.

Feito isso, excluímos de G todas as arestas (i, j) com peso $w(i, j)$ menor que um limiar t, que é dado pela probabilidade de gerar $w(i, j)$ aleatoriamente em um grafo aleatório $G_R(V, E_R)$. A identificação do valor que separa arestas de peso alto das de peso baixo é feita da seguinte maneira. Primeiro, criamos um grafo aleatório $G_R(V, E_R)$ contendo os mesmos

[11] Devido ao grande número de clusters algumas cores se repetem, mas isso não possui nenhum significado especial.

nós de G. Depois, para cada sequência de n_u fotos $f_u^1, f_u^2, ..., f_u^{n_u}$ de cada usuário u, atribuímos PDIs aleatoriamente para cada foto e geramos as arestas aleatórias E_R de G_R a partir dessa nova atribuição. Assim, a sequência das fotos é aleatória, o que gera arestas e pesos de arestas aleatórios em G_R, mas preserva o número total de fotos que foi tirada em um determinado local. A ideia é simular trajetórias aleatórias na cidade. Desta maneira aleatória, a distribuição dos pesos das arestas segue uma distribuição normal $N_w(\mu_w, \sigma_w)$ com média μ_w e desvio padrão σ_w.

De acordo com $N_w(\mu_w, \sigma_w)$, quando a probabilidade de gerar $w(i, j)$ em $G_R(V, E_R)$ é próxima de zero, ou seja, quando for pouco provável que o grafo aleatório $G_R(V, E_R)$ tenha um peso de aresta $w(i, j)$, então a transição de $v_i \rightarrow v_j$ é uma transição popular na cidade sendo, de acordo com a nossa conjectura, uma transição entre pontos turísticos. Para o nosso conjunto de dados, o valor de t que fornece uma probabilidade próxima de 0 é $t = 10$. Como podemos observar na figura 11-c, os vértices (PDIs) do grafo resultante representam a maioria dos pontos turísticos de Belo Horizonte. As áreas dos PDIs resultantes cobrem sete de todos os oito pontos recomendados pelo TripAdvisor[12], sendo as áreas culturais e de lazer mais importantes de BH.

É interessante notar a diferença entre as figuras 11-b e 11-c, a primeira contendo todos os PDIs e a segunda somente os pontos turísticos da cidade de BH. Novamente, esta aplicação é interessante porque ela é capaz de identificar os PDIs em um contexto espaço-temporal, o que é fundamental, uma vez que os PDIs são dinâmicos e mudam ao longo do tempo.

6. Conclusão e Trabalhos Futuros

Neste trabalho apresentamos o que é, no melhor de nosso conhecimento, a primeira caracterização do Instagram. A análise do sistema foi feita tratando-o como uma rede de sensoriamento participativa. Assim, discutimos a cobertura espacial e temporal desta rede mostrando a sua abrangência planetária bem como uma frequência de compartilhamento de fotos espaço-temporal muito desigual e correlacionada com as rotinas de atividades humanas. Também discutimos uma aplicação que demonstra o potencial de uma RSP derivada do Instagram para o estudo da dinâmica de cidades.

Como trabalhos futuros, pretendemos analisar outras RSPs e desenvolver novas aplicações que exploram estas redes. Por exemplo, podemos imaginar aplicações que consideram conjuntamente dados provenientes de outros sistemas de sensoriamento participativo como o Waze (condições de tráfego) e o Weddar (condições meteorológicas), considerando inclusive, diferentes categorias/interesses das pessoas.

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

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422.

Barth, F. (1969). *Ethnic groups and boundaries: the social organization of culture difference*. Scandinavian university books. Little, Brown.

Bilandzic, M. and Foth, M. (2012). A review of locative media, mobile and embodied spatial interaction. *International Journal of Human-Computer Studies*, 70(1):66–71.

[12]www.tripadvisor.com

Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., and Srivastava, M. B. (2006). Participatory sensing. In *Proc. Workshop on World-Sensor-Web (WSW)*.

Cheng, Z., Caverlee, J., Lee, K., and Sui, D. Z. (2011). Exploring Millions of Footprints in Location Sharing Services. In *Proc. 5th Int'l Conference on Weblogs and Social Media*.

Cho, E., Myers, S. A., and Leskovec, J. (2011). Friendship and mobility: user movement in location-based social networks. In *Proc. 17th ACM Int'l Conference on Knowledge Discovery and Data Mining*.

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. 6th International Conference on Weblogs and Social Media*.

Daniells, K. (2012). Infographic: Instagram statistics 2012. *Digital Buzz Blog*.

Eisenman, S. B., Miluzzo, E., Lane, N. D., Peterson, R. A., Ahn, G.-S., and Campbell, A. T. (2010). Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks*, 6(1).

Fisk, P. R. (1961). The graduation of income distributions. *Econometrica*, 29(2):171–185.

Goodchild, M. (2007). Citizens as sensors: The world of volunteered geography. *GeoJournal*, 69(4):211–221.

Krumm, J. (2009). *Ubiquitous Computing Fundamentals*. Chapman & Hall/CRC, 1st ed.

Larson, E. C., Lee, T., Liu, S., Rosenfeld, M., and Patel, S. N. (2011). Accurate and privacy preserving cough sensing using a low-cost microphone. In *Proc. 13th International Conference on Ubiquitous Computing*.

Malmgren, R. D., Stouffer, D. B., Motter, A. E., and Amaral, L. A. N. (2008). A poissonian explanation for heavy tails in e-mail communication. *Proc. National Academy of Sciences*, 105(47):18153–18158.

Mashhadi, A. J. and Capra, L. (2011). Quality Control for Real-time Ubiquitous Crowdsourcing. In *Proc. 2nd International Workshop on Ubiquitous Crowdsouring*.

Noulas, A., Scellato, S., Mascolo, C., and Pontil, M. (2011a). An Empirical Study of Geographic User Activity Patterns in Foursquare. In *Proc. of the Fifth Int'l Conf. on Weblogs and Social Media (ICWSM'11)*.

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. 5th Int. Conf. on Weblogs and Social Media*.

Rana, R. K., Chou, C. T., Kanhere, S. S., Bulusu, N., and Hu, W. (2010). Ear-phone: an end-to-end participatory urban noise mapping system. In *Proc. 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*.

Reddy, S., Estrin, D., Hansen, M., and Srivastava, M. (2010). Examining micro-payments for participatory sensing data collections. In *Proc. 12th ACM International Conference on Ubiquitous Computing*.

Saroiu, S. and Wolman, A. (2010). I am a sensor, and i approve this message. In *Proc. 11th Workshop on Mobile Computing Systems and Applications*.

Scellato, S., Noulas, A., Lambiotte, R., and Mascolo, C. (2011). Socio-spatial Properties of Online Location-based Social Networks. In *Proc. 5th International Conference on Weblogs and Social Media*.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., and Loureiro, A. A. F. (2012a). Uncovering Properties in Participatory Sensor Networks. In *Proc. 4th ACM International Workshop on Hot Topics in Planet-scale Measurement*.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., and Loureiro, A. A. F. (2012b). Visualizing the invisible image of cities. In *Proc. IEEE International Conference on Cyber, Physical and Social Computing*.

Sinnott, R. W. (1984). Virtues of the Haversine. *Sky and Telescope*, 68(2):159+.

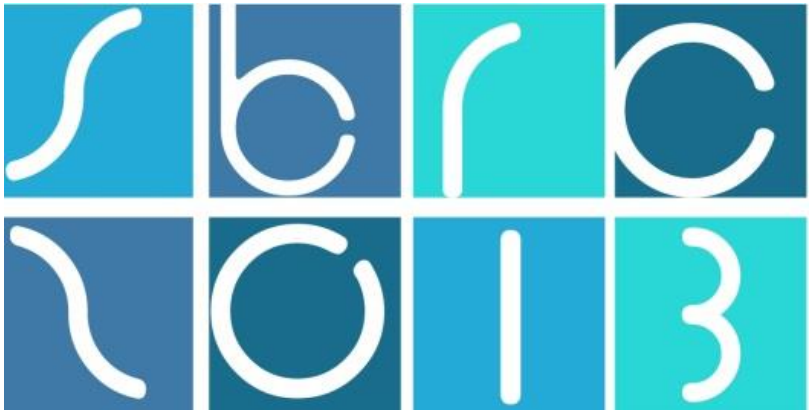
Sørensen, T. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biologiske Skrifter*, 5(4).

Srivastava, M., Abdelzaher, T., and Szymanski, B. (2012). Human-centric sensing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1958):176–197.

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. XXX SBRC*.

Vaz de Melo, P. O. S., Faloutsos, C., and Loureiro, A. A. (2011). Human dynamics in large communication networks. In *Proc. SDM*.

Weiser, M. (1991). The Computer in the 21st Century. *Scientific American*, 265(3):94–104.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 12

Redes Ópticas

Seleção de paradigmas em redes ópticas híbridas integradas

Lucas P. Melo[1], **Messias Bittencourt**[1], **Gustavo B. Figueiredo**[1], **Tertuliano Franco**[2]

[1]Departamento de Ciência Computação – Universidade Federal da Bahia (UFBA)

[2]Departamento de Matemática – Universidade Federal da Bahia (UFBA)

{lucaspm, messiasbf, gustavo}@dcc.ufba.br, tertuliano@ufba.br

Abstract. *This paper proposes a decision function aimed at choosing the most appropriated optical switching paradigm to transport flows in a hybrid optical network. The decision function has the advantage of being a simple function that considers parameters like utilization, statistical multiplexing gain and signaling cost associated with each paradigm considered. Moreover, a simulation based performance evaluation that shows the impact of the chosen criteria in the network state is presented.*

Resumo. *Este artigo propõe uma função para decisão sobre o paradigma a ser usado no transporte de fluxos em redes ópticas híbridas integradas. A função de decisão tem como vantagem ser uma função simples que considera como parâmetros a utilização, o ganho em multiplexação estatística e o custo associado à sinalização em cada paradigma considerado. Adicionalmente, é apresentada uma avaliação de desempenho realizada via simulação que mostra o impacto do critério de decisão no estado da rede.*

1. Introdução

A pesquisa realizada sobre paradigmas de comutação óptica visa a proposição ou melhoria de um dado paradigma, dadas as ineficiências observadas em outros paradigmas. Isso tem levado à proposição de diversos paradigmas de comutação óptica tais como comutação de circuitos ópticos (do inglês *Optical Circuit Switching* - OCS) [Barker et al. 2005], comutação de rajadas ópticas (do inglês *Optical Burst Switching* -OBS) [Qiao and Yoo 1999], comutação de pacotes ópticos (do inglês *Optical Packet Switching* -OPS) [Qiao and Yoo 1999], além de inúmeros protocolos e algoritmos sempre visando ajustá-los a determinado cenário. Contudo, recentemente, uma segunda via tem sido adotada que é a integração de diferentes paradigmas em uma arquitetura de rede visando obter os benefícios desses paradigmas, enquanto minimiza as suas deficiências, dando origem a um novo modelo de rede denominado Redes Ópticas Híbridas (ROH).

Uma dessas arquiteturas é chamada de rede híbrida integrada. Nessa arquitetura, a mesma infraestrutura de rede é utilizada para a realizar a transmissão do tráfego tanto usando o paradigma OCS, quanto usando o paradigma OBS. Assim, o mesmo conjunto de equipamentos pode ser utilizado, diminuindo os custos associados. Para garantir a interoperabilidade entre os paradigmas, um plano de controle eletrônico utilizado para a

reserva dos recursos na rede é formado pela combinação dos planos de controle da rede OCS e da rede OBS.

O custo reduzido torna o modelo integrado mais vantajoso que os demais do ponto de vista financeiro [Buysse et al. 2009]. Entretanto, diversos desafios precisam ser transpostos de forma que a rede experimente a menor probabilidade de bloqueio e utilização ao transportar diferentes tipos de fluxos pela mesma infraestrutura física usando diferentes paradigmas de comutação óptica. Um desses desafios é decidir sobre qual paradigma será usado para a transmissão de um dado fluxo na rede óptica.

Este artigo propõe uma função de decisão que é responsável pela indicação de qual paradigma utilizar, entre os paradigmas OBS e OCS, em função das características do fluxo a ser transportado na rede e do efeito que este fluxo terá sobre a rede. Para tal, são consideradas a utilização provocada na rede ao se transportar o fluxo em cada paradigma, o ganho em multiplexação estatística e o custo de sinalização de cada um dos paradigmas. Resultados mostram que a operação da rede pode guiar a escolha dos paradigmas ao usar pesos que mais adequados aos critérios considerados importantes pela administradora da rede.

O resto do artigo está organizado como segue: a Seção 2 apresenta a arquitetura das redes híbridas e trabalhos relacionados, a Seção 3 apresenta a abordagem proposta para seleção de paradigmas em redes ópticas híbridas integradas, a Seção 4 mostra os exemplos numéricos e por fim a Seção 5 apresenta as conclusões e trabalhos futuros.

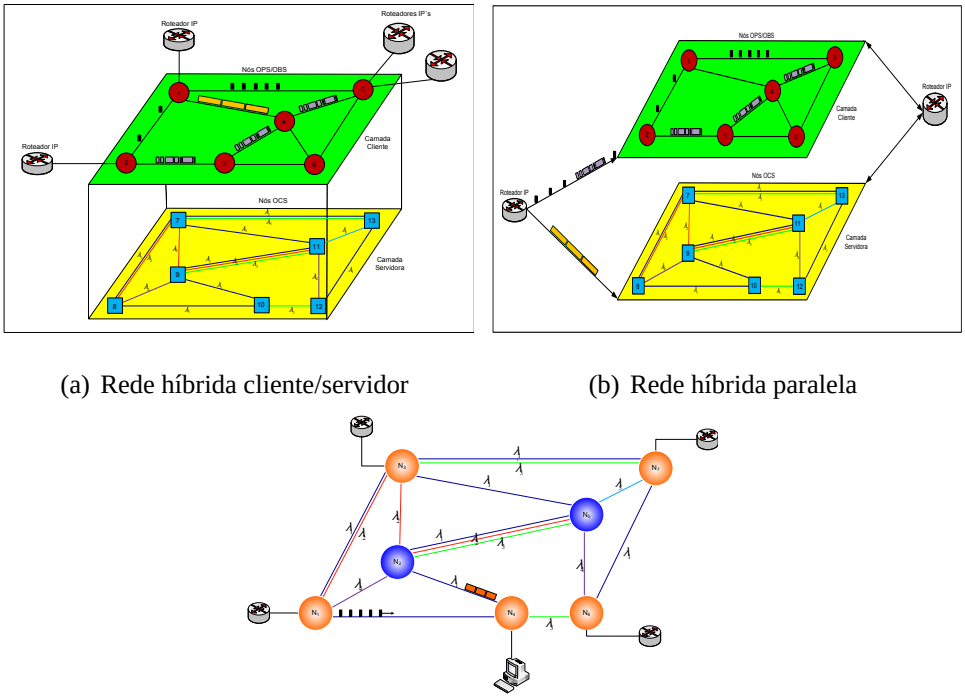
2. Redes ópticas híbridas

O estudo de redes ópticas híbridas WDM é um tema que tem despertado a atenção da comunidade de redes ópticas e obviamente, diversos trabalhos tem sido propostos na literatura recentemente.

Buysse et. al em [Buysse et al. 2009], propõe uma classificação baseada no grau de interação e integração das tecnologias de redes OBS e OCS. Em tal classificação, as Redes Ópticas Híbridas são divididas em três arquiteturas: Redes Ópticas Híbridas Cliente/Servidor; Redes Ópticas Híbridas Paralelas e Redes Ópticas Híbridas Integradas.

Na primeira categoria, a estrutura de comutação da rede óptica é agrupada em uma hierarquia de duas camadas, onde a camada superior é a camada cliente, representada pela tecnologia de comutação óptica OBS. Esta camada solicita um determinado serviço de transporte óptico de dados para a camada inferior, denominada de camada servidora, e que é representada pela tecnologia de comutação óptica OCS, como pode ser observado na Figura 1(a), que ilustra o comportamento de tráfego da rede óptica híbrida cliente/servidor.

A segunda categoria refere-se ao modelo de rede óptica híbrida paralela. Neste modelo, a rede óptica é formada por duas camadas paralelas e independentes, como pode ser observado na ilustração da Figura 1(b). Observe neste modelo que a estrutura de comutação da camada superior é constituída única e exclusivamente pela tecnologia óptica OBS ao passo que na camada inferior essa mesma estrutura é formada pela rede OCS. Para a seleção do tráfego que será direcionado para cada uma das camadas, um nó periférico inteligente seleciona a camada de comutação com base na solicitação do usuário, no modelo de tráfego de entrada ou ambos.



(a) Rede híbrida cliente/servidor (b) Rede híbrida paralela

(c) Rede híbrida integrada

Figura 1. Modelos de redes ópticas híbridas.

A terceira categoria refere-se às Redes Ópticas Híbridas Integradas, onde duas ou mais tecnologias de redes ópticas são integradas em uma só para oferecer um serviços de transporte de dados. Desta forma, todos os nós da rede óptica híbrida integrada compartilham os mesmos recursos de largura de banda, compartilhamento de recursos, protocolos, comprimentos de onda, protocolo de sinalização. Nesta categoria, um nó externo solicita a um nó de borda da rede óptica um serviço de transporte de dados. O nó de borda, baseado nas características do tráfego ou nos parâmetros de qualidade de serviço, configura um paradigma óptico (OCS ou OBS) para o transporte do tráfego solicitado. Em um nível lógico, os comutadores OBS e OCS não têm conhecimento entre si, e a escolha entre eles é feita exclusivamente no ponto de entrada das unidades de dados na borda da rede.

2.1. Trabalhos relacionados

Em [Khodashenas et al. 2011] uma arquitetura de nó híbrido OBS/OCS é proposta. Esta arquitetura parte do conceito de MG-OXC (do inglês *Multi-Granular Optical Cross Connect*) que são comutadores que englobam duas matrizes de comutação em um único nó.

Em [Wang et al. 2009] um modelo de rede híbrida integrada, denominado HyLABS (do inglês *Hybrid Lightpath and Burst Switching*) foi proposto. Nesta arquitetura, os pacotes são encaminhados prioritariamente pela rede OCS em caminhos ópticos estabelecidos do nó de borda ao nó de destino dos pacotes. A topologia virtual da rede OCS é inicialmente construída usando programação linear com uma matriz de tráfego estática. Assim, quando um pacote chega ao nó de borda, o mesmo realiza uma busca à procura de caminhos ópticos que conectem o nó de borda ao nó de destino. Caso não existam caminhos ópticos diretos, ou os existentes não possuam capacidade para transportar os

pacotes, uma nova busca procurando caminhos com mais saltos é realizada. Caso não obtenha sucesso, os pacotes são encaminhados usando-se o paradigma OBS.

Em [Wong and Zukerman 2008], é proposta uma arquitetura de rede óptica híbrida integrada não preemptiva em que as reservas da rede OBS não são descartadas em função das reservas da rede OCS. Na proposta, quando um pedido de reserva para rede OCS chega ao nó de borda, caso o único comprimento de onda disponível esteja sendo utilizado por rajadas, a requisição do circuito é enfileirada até que as rajadas tenham sido transmitidas.

Em [Vu et al. 2005] são propostos modelos analíticos para um nó da rede óptica híbrida integrada considerando tanto o caso em que os circuitos possuem prioridade sobre as rajadas quanto o caso em que não existe prioridade entre os paradigmas de comutação óptica.

Em [Saha et al. 2012] é apresentada uma arquitetura de rede óptica híbrida para a interconexão de DCN (do inglês *Data Center Networks*), denominada *HyScaleII*. Nessa arquitetura, OCS e OBS são usadas para transmitir fluxos de baixa e alta granularidade, respectivamente. O artigo propõe como a topologia da rede deve ser organizada de forma que seja escalável e tolerante a falhas, além de capaz de suportar o grande volume de dados transmitido. Além disso, os autores apresentam um algoritmo de roteamento capaz de levar vantagem da topologia empregada, aumentando assim o desempenho da rede.

Em [Menon et al. 2009], é proposta uma arquitetura de rede híbrida baseada no modelo cliente/servidor em que o tráfego excedente dos circuitos já provisionados é encaminhado pela rede através do paradigma OBS. Inicialmente circuitos de diferentes classes de serviço, denominados circuitos primários, são criados entre os múltiplos comutadores híbridos de borda. Assim, quando um pacote de controle associado a uma rajada chega a um dos nós, é verificado, usando informações relativas ao seu destino e prioridade, se existe um circuito primário que possa transmitir a rajada. Caso sim, a mesma é encaminhada. Caso não exista, canais adicionais são inspecionados à procura de algum que possa acomodar a demanda. Apesar de simples a proposta, ela não especifica como os recursos devem ser reservados, como as rajadas são criadas e como é o processo de seleção de canal.

Em [Xue et al. 2005] é realizada uma avaliação de desempenho das redes OCS e OBS. Os resultados são bem incipientes para que sejam efetivamente considerados mas apontam para uma maior utilização das redes OBS em relação às redes OCS. Já em [Perelló and et al. 2010] é realizada uma avaliação de desempenho de uma rede híbrida OBS/OCS levando-se em consideração critérios de QoS como perda e atraso de pacotes. O artigo considera uma rede híbrida paralela em que a rede OBS é utilizada para tráfego sensível a atraso enquanto a rede OCS é utilizada para tráfego sensível a perdas.

Em [Moura et al. 2011] é apresentada uma avaliação de desempenho quantificando o impacto produzido pela rede OBS na probabilidade de bloqueio da rede OCS. Além disso, o artigo apresenta uma estratégia para atualização de informações capaz de diminuir as perdas na rede devido a informações desatualizadas.

3. Seleção de paradigmas em Redes Ópticas Híbridas

Quando um fluxo chega ao nó de borda, é necessário que o nó decida a respeito de qual paradigma de comutação óptica será utilizado. Nesta seção, propõe-se uma política para decisão sobre qual paradigma de comutação ótica deverá ser transmitido o fluxo, que considera a utilização da rede, o ganho em multiplexação estatística e o custo de sinalização associado a cada um dos paradigmas.

3.1. Modelo de rede adotado

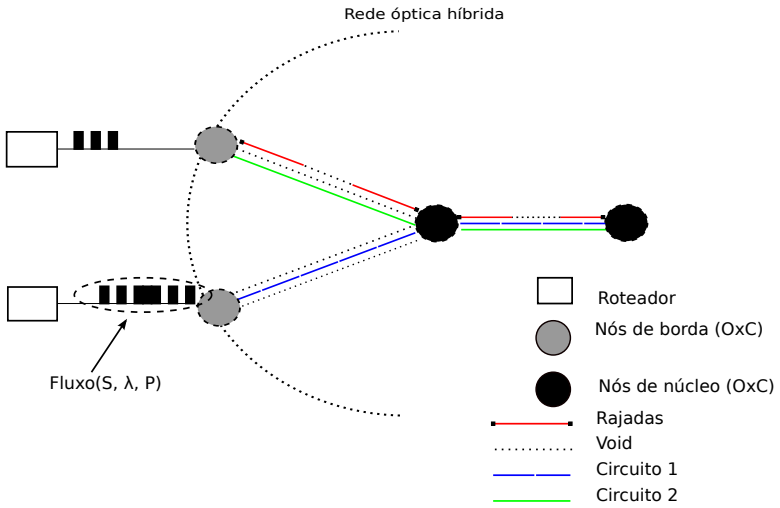


Figura 2. Modelo adotado

A rede considerada nesse trabalho, ilustrada na Figura 2, é uma rede *overlay* IP sobre WDM em que os pacotes são transmitidos na rede óptica através de circuitos ou de rajadas ópticas, conforme o paradigma de comutação óptica selecionado.

Antes dos pacotes serem transmitidos, o nó de borda deve determinar qual paradigma usar. Para tanto, assume-se que cada fluxo f segue um SLA (do inglês *Service Level Agreement*) e é caracterizado pela tupla (S, λ, P), onde S é a quantidade de bytes a ser transmitida pelo fluxo, λ é a taxa média de envio dos pacotes, e P a MTU usada na rede IP. Assume-se que tanto a taxa de chegada de fluxos quanto a taxa de chegadas dos pacotes que compõem os fluxos seguem a distribuição de *Poisson* e que o tamanho dos pacotes segue a distribuição Beta(a,b) [Castro 2011] com parâmetros a e b.

A rede óptica é composta por nós OXC (do inglês *Optical Cross Connects*) que atuam ora como nós de borda e ora como nós de núcleo. Em outras palavras assume-se que todo nó da rede óptica pode estar conectado a redes de acesso trazendo tráfego IP e que todos tem a capacidade de comutar o sinal óptico usando conversão total de comprimentos de onda.

Se o paradigma escolhido para transmissão for o OCS, um circuito dedicado é estabelecido seguindo um processo de reserva bidirecional em que um pacote de controle é enviado ao núcleo da rede a fim de realizar a reserva dos recursos. Ao final do processo, em havendo recursos em todos os nós, o nó de destino envia uma mensagem de confirmação ao nó de origem. Só após a chegada da confirmação ao nó de origem

os dados são transmitidos usando o circuito. Ao término da transmissão, o mesmo processo se repete para a liberação dos recursos previamente reservados. O algoritmo de RWA implementado foi o *first-fit* realizado no menor caminho considerando o número de enlaces.

Caso o paradigma OBS seja selecionado, os pacotes IP são encaminhados a filas de montagem de acordo com seu destino. O algoritmo usado na montagem é o *Min-Burst-Length Max-Assembly-Period* [Cao et al. 2002] que adota um tempo máximo de montagem igual a T unidades de tempo e produz rajadas de tamanho mínimo igual a l bytes e máximo igual a L bytes. Após montada a rajada, os recursos são reservados unidirecionalmente, seguindo o protocolo JET (do inglês *Just Enough Time*) [Qiao and Yoo 1999]. O algoritmo de escalonamento é o LAUC-VF.

3.2. O processo de seleção de paradigmas

O processo de decisão sobre qual paradigma utilizar é proposto baseado num princípio empírico e heurístico de lucro potencial da rede gerado por um dado paradigma de comutação óptica i. Ele é definido por $\mathcal{L}_i$ e apresentado na Equação (1).

$$\mathcal{L}_i = \alpha U_i + \beta G_i - \gamma C_i \tag{1}$$

onde, U_i é a utilização, G_i o ganho e C_i o custo gerados pelo paradigma $i = \{b, c\}$, representando comutação de rajadas e de circuitos, respectivamente. Finalmente, α, β e γ são fatores de operação que podem ser escolhidos pelo operador de rede para relativizar a importância de cada parâmetro da Equação (1) dentro do seu contexto.

O lucro potencial é usado como critério de escolha em que o paradigma que produzir o maior valor será aquele escolhido para transportar o fluxo a ser transmitido. Ele considera três métricas que são utilizadas para inferir o efeito na rede causado pela transmissão de um fluxo f usando cada um dos paradigmas. As métricas adotadas neste trabalho são a utilização da rede, o ganho em multiplexação estatística e o custo de sinalização para reserva dos recursos e seus cálculos serão apresentados nas nas subseções 3.2.1, 3.2.2 e 3.2.3, respectivamente.

3.2.1. Cálculo da utilização para os paradigmas OBS e OCS

A utilização é definida como o tempo médio em que a rede está ocupada realizando a transmissão dos dados, assumindo-se sucesso na reserva dos recursos para o circuito, no caso de OCS e para as rajadas, no caso de OBS. A ideia é que, considerando as características do fluxo, evite-se fazer uma escolha que provoque baixa utilização dos recursos da rede.

Para o paradigma OBS, considerando o protocolo de reservas JET, cada reserva é feita pelo período de duração correspondente a uma rajada. Além disso, os recursos não são reservados no período correspondente à chegada do pacote de controle ao nó e a chegada da rajada. Assim, a utilização da rede do paradigma OBS é igual a 1.

O cálculo da utilização do paradigma OCS é feito modelando-se um circuito como

uma fila M/G/1 com os seguintes parâmetros:

λ	Taxa de chegada de pacote do fluxo
$\mathbb{E}[S]$	Tempo médio de serviço
C_v	coeficiente de variação

Seja X uma variável aleatória que modela o tamanho dos pacotes. Assumindo-se que X segue a distribuição Beta com parâmetros a e b, tem-se:

$$\mathbb{E}[X] = \frac{a}{a+b} \tag{2}$$

$$\text{Var}(X) = \frac{ab}{(a+b)^2(a+b+1)} \tag{3}$$

Dado que a transmissão dos pacotes pelo circuito óptico é feita na taxa de transmissão do canal, V, tem-se:

$$\mathbb{E}[S] = \frac{\mathbb{E}[X]}{V}.$$

Finalmente, a utilização do circuito é dada por:

$$\rho = \lambda \mathbb{E}[S] \tag{4}$$

A Equação (5) sumariza a utilização dos diversos paradigmas:

$$U_i = \begin{cases} 1 & \text{, se } i = b; \\ \lambda \mathbb{E}[S] & \text{, se } i = c. \end{cases} \tag{5}$$

3.2.2. Cálculo do ganho em multiplexação estatística dos paradigmas OBS e OCS

O ganho em multiplexação estatística é definido de forma aproximada como a probabilidade de que outro fluxo f' seja acomodado no mesmo comprimento de onda utilizado para acomodar f (assumindo que todo o fluxo f utilizará o mesmo comprimento de onda), usando um paradigma em particular.

Obviamente, o ganho em multiplexação estatística dado pelo paradigma OCS é nulo, já que uma vez feita a reserva dos recursos, outro fluxo não poderá usar o mesmo comprimento de onda. Assim,

$$G_c = 0 \tag{6}$$

Dessa forma, nesta seção será apresentada uma derivação aproximada para o ganho em multiplexação estatística produzido pelo paradigma OBS.

Sejam r_f e $r_{f'}$ reservas associadas a rajadas oriundas dos fluxos f e f', respectivamente e considerando que r_f foi realizada a partir do tempo 0, para que uma reserva de f' possa ser feita no mesmo comprimento de onda, é necessário que: $i)$ o seu tamanho (denotado aqui por $|r_{f'}|$) seja menor do que os *voids* criados pelo fluxo f e que $ii)$ o ponto médio da reserva correspondente à rajada de $r_{f'}$ atinja distância pelo menos $|r_{f'}|/2$ de cada extremo do intervalo *void* gerado por f, como ilustrado na Figura 3.

Adicionalmente, a figura ilustra o fato de que um outro período de montagem de rajadas do fluxo f (representado pela linha tracejada acima do símbolo T) inicia-se imediatamente após o envio da rajada antecedente. Supondo que o intervalo da reserva

$r_{f'}$ é disposto de maneira uniforme no intervalo T, o que é uma hipótese razoável para tempos longos, já que os processos f e f' são independentes, o ponto médio do intervalo da reserva r'_f é escolhido uniformemente no intervalo $[|r'_f|/2, T - |r'_f|/2]$.

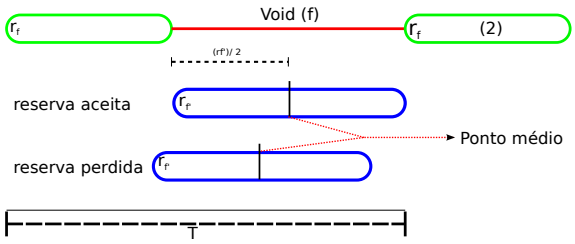


Figura 3. Uso de *voids* por rajadas do fluxo f'.

Para que a reserva r'_f não intercepte r_f nem a próxima reserva oriunda de f, (ilustrada na figura como $r_f(2)$), o ponto médio de r_f deve cair no intervalo $[|r_f| + (|r'_f|/2), T - |r_f|/2]$. Logo, a probabilidade de rajadas de f' não interceptarem rajadas de f pode ser aproximada por:

$$\frac{T - |r_f| - |r'_f|}{T - |r_f|}$$

, se $|r_f| < |v_f|$, onde v_f é o tamanho do *void* e supondo conhecidos os tamanhos de r_f e r'_f.

Assim, uma aproximação da probabilidade de que uma rajada do fluxo f' seja acomodada no mesmo comprimento de onda do fluxo f pode ser dada por[1]:

$$G_b = \begin{cases} \frac{T - \mathbb{E}[B_f] - \mathbb{E}[B'_f]}{T - \mathbb{E}[B_f]} & \text{, se } \mathbb{E}[B_f] + \mathbb{E}[B'_f] < T \\ 0 & \text{, caso contrário} \end{cases} \tag{7}$$

O cálculo da esperança do tamanho das rajadas, $\mathbb{E}[B]$, dos fluxos f e f', considerando serem idênticos, é dada por:

$$\begin{aligned} \mathbb{E}[B] = \sum_{k=0}^{+\infty} \mathbb{E}\left[\sum_{i=1}^{k} \beta_i | N = k\right] \cdot \left(\frac{e^{-\lambda T}(\lambda T)^k}{k!}\right) = \\ = \sum_{k=1}^{+\infty} \frac{ka}{a+b} \cdot \left(\frac{e^{-\lambda T}(\lambda T)^k}{k!}\right) = \\ = \frac{a}{a+b} \sum_{k=1}^{+\infty} \left(\frac{e^{-\lambda T}(\lambda T)^k}{(k-1)!}\right) = \\ = \frac{(\lambda T)a}{a+b} \sum_{k=1}^{+\infty} \left(\frac{e^{-\lambda T}(\lambda T)^{(k-1)}}{(k-1)!}\right) \end{aligned}$$

como $\sum_{k=1}^{+\infty} \left(\frac{e^{-\lambda T}(\lambda T)^{(k-1)}}{k-1!}\right) = 1$ tem-se que:

$$\mathbb{E}[B] = \frac{\lambda T a}{a+b} \tag{8}$$

[1]Os subíndices f e f' na expressão $\mathbb{E}[B]$ foram usados nesta expressão para diferenciar o tamanho médio das rajadas de f e f', respectivamente. Entretanto, quando não existir risco de dupla interpretação, será usado o termo $\mathbb{E}[B]$ para representar o tamanho médio das rajadas

A Equação (9) sumariza o ganho em multiplexação estatística dos diversos paradigmas.

$$G_i = \begin{cases} 0 & \text{, se } i = c; \\ \begin{cases} \frac{T - \mathbb{E}[B_f]\mathbb{E}[B'_f]}{T - \mathbb{E}[B_f]} & \text{, se } \mathbb{E}[B_f] + \mathbb{E}[B'_f] < T \\ 0 & \text{, caso contrário} \end{cases} & \text{, se } i = b. \end{cases} \tag{9}$$

3.2.3. Cálculo do *overhead* de sinalização dos paradigmas OBS e OCS

O *overhead* de sinalização dos paradigmas é definido como a razão entre o tempo gasto para efetuar a reserva dos recursos em cada paradigma e o tempo total de uso da rede, contabilizado a partir do instante de envio do pacote de controle. Sejam t_s o tempo de sinalização e t_x o tempo de transmissão dos dados, o *overhead* de sinalização, C_i, é definido por:

$$C_i = \frac{t_s}{t_s + t_x} \tag{10}$$

Na rede OBS, t_s corresponde ao tempo gasto no processamento dos pacotes de controle em cada nó usando-se reserva unidirecional. Assim, assumindo-se uma rota com H nós, e que cada nó j da rota utilizada possui o mesmo tempo de processamento, p_j, tem-se que $t_s = Hp_j$. Adicionalmente, o tempo de transmissão é dado pela razão entre o tamanho da rajada e a velocidade de transmissão, $t_x = \frac{B}{V}$. Além disso, o *overhead* de sinalização do paradigma OBS, deve considerar para um dado fluxo, o número total de rajadas, expresso por $\mathbb{E}[N_b]$, necessário para o envio de todos os S bytes, o que é dado pela Equação (11).

$$\mathbb{E}[N_b] = \frac{S}{\mathbb{E}[B]}, \tag{11}$$

onde $\mathbb{E}[B]$ é dado pela Equação (8). xs Assim, considerando o processo de reserva de recursos para todas as rajadas, tem-se que:

$$C_b = \mathbb{E}[N_b]\frac{VHp_j}{VHp_j + \mathbb{E}[B]}, \tag{12}$$

onde $\mathbb{E}[B]$ é o tamanho médio que aproxima o tamanho da rajada B e V é a velocidade do enlace.

Considerando-se o paradigma OCS, tem-se que:

$$t_s = 2\sum_{j=1}^{H} p_j = t_s = 2Hp_j,$$

se p_j é igual em todos os nós da rota.

Adicionalmente, para o cálculo de t_x é preciso estimar o tempo em que cada pacote utiliza o circuito, além da quantidade de pacotes IP necessária para transmitir o fluxo f. O tempo em que cada pacote utiliza o circuito corresponde ao tempo de resposta da fila M/G/1 e é dado por:

$$r = \frac{\mathbb{E}[S] + \rho\mathbb{E}[S](1 + C_v^2)}{2(1 - \rho)} \tag{13}$$

Já o número estimado de pacotes em cada fluxo, dados que o valor S pode ser obtido pela descrição do fluxo f e $\mathbb{E}[X]$ pela Equação (3), é dado por:

$$\mathbb{E}[N_P] = \frac{S}{\mathbb{E}[X]}$$

Dessa forma, a estimativa de t_x é dada pela Equação (14).

$$t_x = \mathbb{E}[N_p]r \tag{14}$$

Assim:

$$C_c = \left(\frac{2Hp_i}{2Hp_i + \frac{S\mathbb{E}[S]+\rho\mathbb{E}[S](1+C_v^2)}{2(1-\rho)\mathbb{E}[X]}} \right) \tag{15}$$

Finalmente, observando-se as Equações (5) e (9), percebe-se que elas representam funções limitadas no domínio $[0; 1]$. Entretanto, a Equação (12) não representa uma função limitada em $[0; 1]$ devido ao fator $\mathbb{E}[N_b]$, o que pode criar um viés na Equação (1), fazendo com que o custo de sinalização majore a função representada pela Equação (1). Portanto, para evitar tal viés, as Equações (12) e (15) são multiplicadas por $\frac{1}{\mathbb{E}[N_b]}$, produzindo a Equação (16) que sumariza o custo de sinalização dos diversos paradigmas:

$$C_i = \begin{cases} \frac{VHp_i}{VHp_i+\mathbb{E}[B]} & \text{, se } i = b; \\ \left(\frac{2Hp_i}{2Hp_i+\frac{S\mathbb{E}[S]+\rho\mathbb{E}[S](1+C_S^2)}{2(1-\rho)\mathbb{E}[X]}} \right) \frac{1}{\mathbb{E}[N_b]} & \text{, se } i = c. \end{cases} \tag{16}$$

De um modo geral, analisando a função 1, é possível perceber que os fluxos com alta taxa de transmissão e longa duração tendem a usar o paradigma OCS, isso porque, fluxos com alta taxa de transmissão produziriam utilização equivalente à transmissão por rajadas e ainda teriam um menor custo de sinalização, já que no envio por OBS diversas rajadas seriam enviadas. Além disso, para fluxos intensos, o paradigma OBS tem baixo ganho em multiplexação estatística.

Por outro lado, fluxos com menor duração (menos bytes) e/ou baixa taxa de transmissão seriam usados pelo paradigma OBS, já que um número menor de rajadas seria enviado, diminuindo o custo de sinalização associado. Além disso, o paradigma produziria alta utilização, devido ao processo de montagem de rajadas e finalmente produziria alto ganho em multiplexação estatística, já que o tamanho dos *voids* seria maior, assim como o período de montagem.

4. Exemplos Numéricos

Para avaliar o impacto da função de decisão proposta, simulações foram realizadas utilizando simulador proprietário desenvolvido em C++. A Figura 4 mostra a topologia da rede utilizada nas simulações. No experimento, o tempo de reconfiguração da malha de comutação foi considerado insignificante em relação ao tempo de processamento dos pacotes de controle. Os parâmetros utilizados são sumarizados na Tabela 1.

Os fluxos foram gerados de acordo com um processo de Poisson produzindo fluxos de carga distribuída uniformemente entre valor mínimo e máximo (ver Tabela 1).

Comprimentos de onda por enlace	32
Duração da simulação	1 s
Intervalo médio de criação de fluxos	2.22 ms
Período de montagem de rajadas	1 ms
Tempo de processamento de pacote de controle	50 ms
Velocidade de enlace	1 Gbps
Tamanho máximo de rajada	500, 1085, 1670, 2255, 2840, 3425, 4010, 4596 KB
Tamanho mínimo de rajada	Tamanho máximo de rajada / 10
Tamanho mínimo de fluxo	1 KB
Tamanho máximo de fluxo	1 MB

Tabela 1. Parâmetros utilizados nas simulações

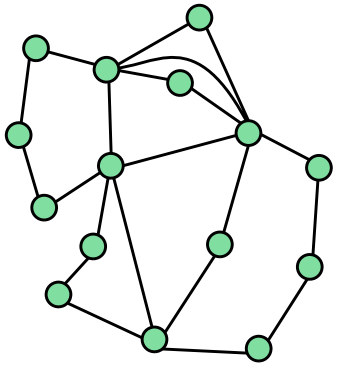


Figura 4. Topologia utilizada nas simulações.

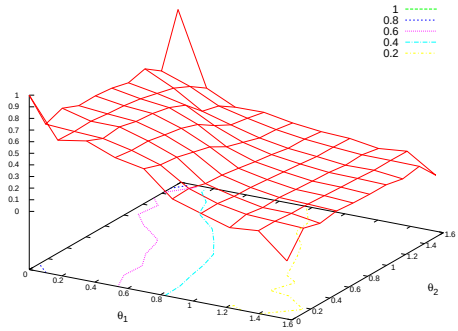
Os parâmetros da função de decisão (Função (1)), α, β e γ, foram dados em função de dois ângulos θ_1 e θ_2 que variam entre 15 pontos igualmente espaçados do intervalo $[0, \pi/2]$. Esta escolha foi feita pois os parâmetros conferem o mesmo comportamento quando multiplicados pelo mesmo fator. Assim:

$$\alpha = \sin(\theta_1)$$
$$\beta = \cos(\theta_1)\sin(\theta_2)$$
$$\gamma = \cos(\theta_1)\cos(\theta_2)$$

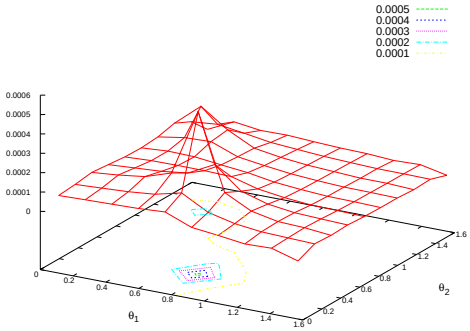
Foram feitas 10 replicações do experimento para cada uma das $8 \times 15 \times 15 = 1800$ combinações dos parâmetros tamanho de rajadas, θ_1 e θ_2 e foram obtidas as seguintes métricas: Proporção de fluxos classificados como OCS; Taxa de bloqueio de fluxos OCS e de rajadas; Taxa de chegada média para pacotes de cada tipo de fluxo; Atraso médio de pacotes de fluxos OBS; Vazão média e Quantidade média de comprimentos de onda reservados em dado instante. A Figura 5 sumariza os resultados obtidos.

Conforme vê-se na Figura 5(a), de um modo geral, a proporção de fluxos classificados como OCS aumenta mais sensivelmente com a redução de θ_1 e mais suavemente com o aumento de θ_2. Valores baixos de θ_1 e θ_2 correspondem a uma região em que a utilização e o ganho em multiplexação estatística possuem menos importância (menores valores de α e β) enquanto o custo de sinalização é mais importante. Isso explica uma proporção maior de fluxos transmitidos usando OCS e mostra como os parâmetros α, β e $gamma$ podem ser usados pelo operador da rede para beneficiar o critério mais apropriado. Além disso, para valores menores do que o tamanho mínimo de rajada (Figura 5(b)), uma chance de bloqueio menor parece estar associada com o maior peso no valor de utilização e multiplexação estatística.

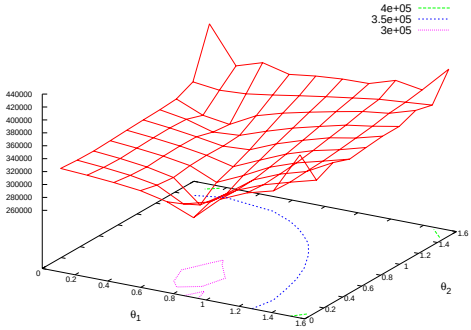
A taxa média de chegada de pacotes selecionados como OBS encontra valor mínimo justamente na vizinhança do ponto $(0.8, 0.1)$ (Figura 5(c)), o que parece estar associado com a diminuição da taxa de bloqueio para tamanhos mínimo de rajada maio-



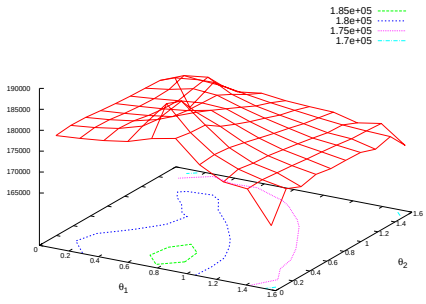
(a) Proporção de fluxos OCS.



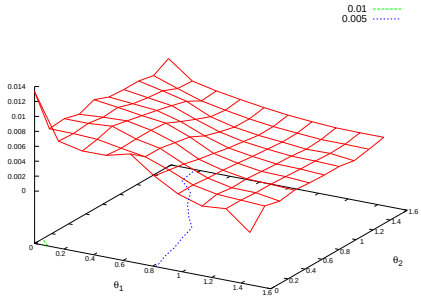
(b) Taxa de bloqueio de fluxos OBS.



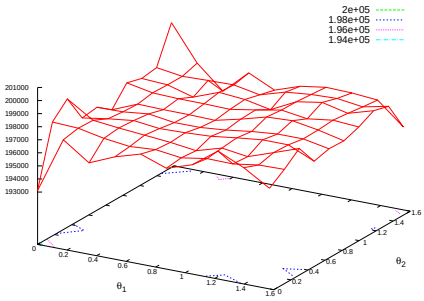
(c) Taxa média de chegada de pacotes para fluxos OBS.



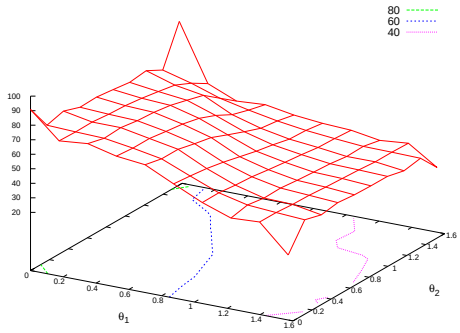
(d) Atraso médio de fluxos OBS (em microsegundos).



(e) Taxa de bloqueio de fluxos OCS.



(f) Vazão média em blocos de 1 ms.



(g) Quantidade média de comprimentos de onda reservados num dado instante.

Figura 5. Medidas obtidas no experimento.

res. Isto pode ser explicado pelos ganhos de multiplexação estatística obtidas no uso do OBS para fluxos esparsos. Note, no entanto, que quando o tamanho do rajada mínimo possui menor valor, a situação é inversa.

O atraso de fluxos OBS (Figura 5(d)) parece ser maior nas regiões onde há menor taxa para fluxos OBS. Isto se deve ao montador preencher o tamanho máximo de rajada com maior frequência, enviando-a antes de completar um período.

Para valores pequenos de rajada mínima, a taxa de bloqueio do OCS (Figura 5(e)) diminui com θ_2 e θ_1 (mais fortemente com este último).Para valores menores de rajada mínima (Figura 5(f)), a vazão foi máxima quando o ganho de multiplexação estatística foi o fator mais preponderante no algoritmo seletor e mínimo quando somente o custo de sinalização foi considerado, apresentando a média dos dois comportamentos em outras situações.O comportamento visto nas taxas de bloqueio se reflete no número médio de comprimentos de onda reservados em cada instante. Para valores menores de tamanho de rajada mínima (Figura 5(g)), há maior economia no uso de comprimentos de onda quando há maior predileção pelo OBS.

5. Conclusões e Trabalhos futuros

Dada a necessidade de um uso eficiente da grande largura de banda disponível na tecnologia WDM, esforços de pesquisa tem sido orientados na construção de uma arquitetura de comutação óptica híbrida que considera as vantagens dos paradigmas OBS e OCS. Dentre os paradigmas adotados, o que possui o menor custo é o paradigma integrado já que o mesmo conjunto de equipamentos pode ser utilizado para a transmissão em ambos os paradigmas.

Este artigo propôs uma função de decisão para a seleção do paradigma mais apropriado para a transmissão de um fluxo de dados em uma rede óptica híbrida integrada. A função determina o lucro potencial de cada paradigma, levando em consideração a utilização, o ganho em multiplexação estatística e o custo de sinalização de cada paradigma. Resultados obtidos via simulação mostram que os parâmetros da função podem ser usados pelo operador da rede para beneficiar o critério mais apropriado para a rede em questão.

Como trabalhos futuros, pretende-se a incorporação de critérios associados à probabilidade de perda de pacotes (mais alta na rede OBS) e/ou outros critérios de QoS. É possível ainda realizar refinamentos nas aproximações dadas na computação do ganho em multiplexação estatística

Referências

Barker, K., Benner, A., Hoare, R., Hoisie, A., Jones, A., Kerbyson, D., Li, D., Melhem, R., Rajamony, R., Schenfeld, E., Shao, S., Stunkel, C., and Walker, P. (2005). On the feasibility of optical circuit switching for high performance computing systems. In *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, page 16.

Buysse, J., De Leenheer, M., Develder, C., Dhoedt, B., and Demeester, P. (2009). Cost-effective burst-over-circuit-switching in a hybrid optical network. In *Networking and Services, 2009. ICNS '09. Fifth International Conference on*, pages 499 –504.

Cao, X., Li, J., Chen, Y., and Qiao, C. (2002). Assembling tcp/ip packets in optical burst switched networks. In *IEEE GLOBECOM*, pages 2808–2812.

Castro, E. R. S. (2011). *Modelo para distribuição de probabilidade do comprimento dos pacotes em redes de computadores*. PhD thesis, Universidade Federal de Campina Grande - Programa de Pós Graduação em Engenharia Elétrica.

Khodashenas, P., Perelló and, J., Spadaro, S., Comellas, J., and Junyent, G. (2011). A feedback-based hybrid obs/ocs architecture with fast-over-slow capability. In *Optical Network Design and Modeling (ONDM), 2011 15th International Conference on*, pages 1 –6.

Menon, P., Cerroni, W., and Reimer, N. (2009). Overflow traffic modeling in hybrid optical circuit/burst switching nodes with service differentiation. In *Optical Fiber Communication - incudes post deadline papers, 2009. OFC 2009. Conference on*, pages 1 –3.

Moura, I., Mazullo, F., Maranhão, J., and Soares, A. (2011). Impacto da comutação obs na probabilidade de bloqueio ocs em redes híbridas ocs/obs. In *SBRC - Simpósio Brasileiro de Redes de Computadores*, pages 698–711.

Perelló and, J., de Guinea, N., Spadaro, S., Junyent, G., and Comellas, J. (2010). Performance evaluation of a hybrid obs/ocs network with qos differentiation based on packet loss/delay requirements. In *Transparent Optical Networks (ICTON), 2010 12th International Conference on*, pages 1 –4.

Qiao, C. and Yoo, M. (1999). Optical burst switching (OBS) - A new paradigm for an optical Internet. *Journal of High-Speed Networks*, 8(1):69–84.

Saha, S., Deogun, J., and Xu, L. (2012). Hyscaleii: A high performance hybrid optical network architecture for data centers. In *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pages 1 –5.

Vu, H. L., Zalesky, A., Wong, E., Rosberg, Z., Bilgrami, S., Zukerman, M., and Tucker, R. (2005). Scalable performance evaluation of a hybrid optical switch. *Lightwave Technology, Journal of*, 23(10):2961 – 2973.

Wang, Y., Wang, S., Xu, S., and Wu, X. (2009). A new hybrid optical network design consisting of lightpath and burst switching. In *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, volume 03, pages 1873 –1876.

Wong, E. and Zukerman, M. (2008). An optical hybrid switch with circuit queueing for burst clearing. *Lightwave Technology, Journal of*, 26(21):3509 –3527.

Xue, F., Yoo, S., Yokoyama, H., and Horiuchi, Y. (2005). Performance comparison of optical burst and circuit switched networks. In *Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC*, volume 3, page 3 pp. Vol. 3.

Heurísticas para Roteamento com Agregação de Tráfego em Redes Ópticas Multi-domínio

Rangel Oliveira[1], **Fernanda Sumika Hojo de Souza**[2], **Geraldo Robson Mateus**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627 – Pampulha – 31270-010
Belo Horizonte – Minas Gerais – Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de São João del-Rei
Av. Visconde do Rio Preto, s/n – Colônia do Bengo – 36301-360
São João del-Rei – Minas Gerais – Brazil

{rangel,mateus}@dcc.ufmg.br, fsumika@ufsj.edu.br

Resumo. *O aumento da implantação de redes ópticas multi-domínio deve-se à diminuição dos custos de fibra óptica e crescente demanda por comunicação multimídia que requer alta largura de banda e baixo atraso nas comunicações. Os usuários dessas redes se comunicam por conexões ópticas estabelecidas fim-a-fim, denominadas de caminhos ópticos, desde um vértice de origem até um vértice de destino. Agregação de Tráfego consiste em combinar uma série de demandas de tráfego tal que a alta capacidade de cada caminho lógico possa ser utilizada da forma mais eficiente possível. Como o cálculo de rotas implementa função chave na agregação de tráfego, o estudo de soluções para o problema de agregação de tráfego no contexto de redes ópticas multi-domínio mostra-se de relevância no atual cenário. Neste trabalho são propostas duas abordagens heurísticas para o problema, sendo a abordagem centralizada capaz de fornecer melhores soluções a um preço computacional mais elevado e falta de escalabilidade e a abordagem descentralizada mais adequada a cenários reais com a desvantagem de uma solução baseada apenas em informações locais.*

Abstract. *The increasing application of multi-domain optical networks is due to the reduction of optical fiber costs and the increasing demand for multimedia, requiring high bandwidth and low delay in communication. The users of these networks can communicate through end-to-end optical connections, called optical paths from a source node to a destination node. Traffic grooming consists in combining a series of traffic demands such that the high capacity of each logical path can be used as efficiently as possible. As the routes' calculation implements a key role in traffic grooming, the study of solutions to the problem of traffic grooming in the context of multi-domain optical networks is very relevant in the present scenario. In this work we propose two heuristic approaches to the problem, such that the centralized approach can provide better solutions with a higher computational cost and lack of scalability and the decentralized approach is more suitable for real scenarios with the disadvantage of a solution based only on local information.*

1. Introdução

Multiplexação por divisão de comprimento de onda (WDM – *Wavelength Division Multiplexing*) [Mukherjee 1997, Mukherjee 2006] e fibra óptica são tecnologias promissoras que surgiram para suprir a necessidade de acomodar o crescimento explosivo do tráfego em redes de telecomunicações. Redes ópticas possuem uma enorme capacidade de transmissão da ordem de dezenas de Tbps. Utilizando a tecnologia WDM, a largura da banda de transmissão da ordem de Tbps pode ser dividida em múltiplos canais de comprimento de onda. Cada canal WDM pode operar em diferentes velocidades, por exemplo, velocidades de alguns Gbps. No entanto, a existência de um meio físico de alta capacidade não implica em alto padrão de qualidade na comunicação se a rede não for utilizada de forma adequada. Assim, técnicas de otimização mostram-se de grande importância no projeto dessas redes a fim de explorar todo o potencial por elas oferecido, melhorando o desempenho das redes e provendo arquiteturas mais confiáveis e de menor custo.

O problema de roteamento com agregação de tráfego em redes multi-domínio pode ser considerado relevante em função do atual cenário, no qual enfrentamos um crescimento explosivo de tráfego em redes de telecomunicações. Embora muitas soluções venham sendo estudadas na literatura, há sempre possibilidade de otimizar o funcionamento de tais redes frente a novos cenários e demandas mais exigentes.

Considerando que as requisições de tráfego ainda são, muitas vezes, uma pequena fração da capacidade total de um canal de comprimento de onda, é possível aumentar ainda mais a utilização do canal, agrupando diversas requisições de tráfego em um mesmo comprimento de onda. Esta técnica é conhecida como *traffic grooming* [Zhu and Mukherjee 2003, Mukherjee 2006]. Neste caso, as requisições de tráfego são expressas em termos de reserva de largura de banda de diferentes granularidades: OC-1, OC-3, OC-12, OC-48, onde 1 OC $\approx 51,85$ Mbps. A agregação ou desagregação de múltiplas requisições de tráfego no vértice origem, vértice destino ou vértice intermediário exige a utilização de multiplexadores ópticos (cada multiplexador tem um custo de instalação) para adicionar/separar o tráfego de um comprimento de onda. Portanto, a rota de uma determinada requisição é composta por "saltos"ópticos definidos por um caminho físico por onde o sinal óptico atravessa vértices intermediários, criando uma conexão virtual entre seus vértices finais. A sequência de "saltos"ópticos seguidos por uma requisição define um *lightpath* [Mukherjee 2006].

Visando minimizar os custos de instalação ou de forma equivalente, o número de canais de comprimento de onda utilizados para garantir que as requisições de tráfego sejam satisfeitas, devemos otimizar o roteamento e alocação de comprimento de onda com agregação de tráfego. Este problema é conhecido como *grooming, routing and wavelength assignment* (GRWA) [Zhu and Mukherjee 2003, Hu and Leida 2004, Barr et al. 2006, Vignac et al. 2009].

Com o expansão das redes de telecomunicações, um novo cenário - baseado em domínios - surgiu e mostrou-se imprescindível a fim de atender as necessidades dos usuários distribuídos por diferentes partes do planeta. Um domínio é um agrupamento lógico de dispositivos de rede que está submetido a regras de roteamento e tráfego. Assim, diversos domínios encontram-se espalhados em diferentes localizações geográficas e muitas vezes demandam comunicação uns com os outros. Para resolver o problema de comunicação entre domínios, foram criados protocolos tais como o EGP e o BGP. Estes

protocolos baseiam-se em políticas de tráfego entre os domínios e obtêm rotas de atendimento de acordo com essas informações. O problema de roteamento em redes multi-domínio é um caso generalizado do problema de roteamento com apenas um domínio, no qual dois tipos de requisições devem ser satisfeitas: o roteamento intra-domínio, no qual a comunicação está restrita ao domínio em questão e o roteamento inter-domínio, onde a comunicação é realizada transpassando roteadores de borda, que são capazes de rotear pacotes para fora do domínio de origem.

Este trabalho tem como objetivo contribuir com novas soluções para o problema de roteamento com agregação de tráfego em redes ópticas multi-domínio. Duas abordagens heurísticas são propostas para o problema, sendo a abordagem centralizada capaz de fornecer melhores soluções a um preço computacional mais elevado e falta de escalabilidade e a abordagem descentralizada mais adequada a cenários reais com a desvantagem de uma solução baseada apenas em informações locais. As duas abordagens são comparadas através de experimentos computacionais, evidenciando as vantagens e desvantagens de cada uma delas.

As seções seguintes, estão organizadas como 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 computacionais e discussão são apresentados na Seção 5. Finalmente, a Seção 6 conclui o trabalho.

2. Trabalhos Relacionados

Técnicas de agregação podem ser classificadas em várias categorias: agregação de tráfego estático vs. agregação de tráfego dinâmico, tráfego não-bifurcado vs. tráfego bifurcado, agregação de tráfego com salto simples vs. agregação de tráfego com múltiplos saltos. A agregação de tráfego estática considera o caso em que todas as requisições são conhecidas a priori e não sofrem alterações por longos períodos de tempo. Em contraste, a agregação de tráfego dinâmica trata o caso em que as demandas surgem dinamicamente, de acordo com uma distribuição de probabilidade. No caso de tráfego não-bifurcado, cada demandas devem seguir por um único caminho desde seu vértice de origem, até seu vértice de destino. No tráfego bifurcado, uma requisição pode ser fracionada e roteada por um ou mais caminhos. O tráfego bifurcado torna o problema menos complexo, uma vez que é mais fácil acomodar demandas fracionadas na capacidade residual dos comprimentos de onda. Finalmente, a agregação de tráfego com salto simples restringe a requisição a usar um único *lightpath*, enquanto a agregação de tráfego com múltiplos saltos permite que uma requisição seja roteada por diversos *lightpaths* concatenados. Neste trabalho, iremos tratar a variação do problema com agregação de tráfego estático, tráfego não-bifurcado e com múltiplos saltos.

O problema GRWA é estudado em [Zhu and Mukherjee 2002, Hu and Leida 2004, Vignac et al. 2009, De et al. 2010]. Para uma revisão sobre agregação de tráfego, veja [Zhu and Mukherjee 2003]. Em [Zhu and Mukherjee 2002], os autores propõem uma formulação de Programação Linear Inteira (PLI), juntamente com duas heurísticas (MST e MRU), com o objetivo de melhorar a vazão da rede. Um método de decomposição que divide o GRWA em problemas menores - problema GR e problema WA - é proposto em [Hu and Leida 2004], mostrando que

estes podem ser resolvidos de forma mais eficiente separadamente. Reformulações e abordagens de decomposição são desenvolvidas em [Vignac et al. 2009], resolvendo instâncias realísticas com 14 e 20 vértices. Melhorias nos resultados propostos por [Zhu and Mukherjee 2002] são mostrados em [De et al. 2010], onde o objetivo de maximizar a vazão da rede é utilizado. No entanto, nenhum destes trabalhos considera o uso de redes multi-domínio.

Uma das dificuldades em se realizar o roteamento com proteção em redes multi-domínio é a escalabilidade. O trabalho desenvolvido em [Truong and Jaumard 2012] propõe uma nova técnica que tem como foco a topologia, chamada *Topology Aggregation*. Essa estratégia leva em consideração apenas algumas rotas intra-domínio para trafegar toda a demanda entre os roteadores de borda. Neste caso, cada rota dentro de um domínio é abstraída como uma aresta virtual no roteamento inter-domínios, tornando assim a rede mais simples de se gerenciar em algoritmos de roteamento. Os testes realizados mostram que as soluções obtidas são bem próximas àquelas obtidas para o problema com apenas um domínio.

Os autores em [Zhu et al. 2003] apresentam três algoritmos de roteamento em redes ópticas para redes de larga escala: *"End-to-End"*, *"Concatenated Shortest Path"* e *"Hierarchical Routing"*. O algoritmo *"End-to-End"* considera todos os elementos conectados como um grafo simples, então algoritmos baseados no menor caminho funcionam de forma satisfatória. O algoritmo *"Concatenated Shortest Path"* também chamado de roteamento segmento por segmento, já que cada domínio decide, baseado em informações locais, qual a melhor forma de alocar as requisições. Por outro lado os roteadores de borda decidem qual é o próximo domínio, segmento, que fará parte da rota como um todo. Diferentemente do *"Concatenated Shortest Path"*, o algoritmo *"Hierarchical Routing"* realiza uma abstração sobre os domínios, e cada um é mapeado para um vértice de uma nova rede que é a primeira camada do modelo. Resolvendo o roteamento nesta camada, cada vértice/domínio selecionado na rota inicial pode ser usado como entrada para um algoritmo de menor caminho que resolve a parte do roteamento, alocação e agregação de comprimentos de onda no domínio. O algoritmo *"End-to-End"* é melhor para baixa probabilidade de bloqueio da rede se o tráfego global é dominante. Em todos os outros casos, a *"Concatenated Shortest Path"* e *"Hierarchical Routing"* podem ser melhores, devido a sua baixa complexidade de tempo e de memória.

Os trabalhos em [Halabi et al. 2011] e [Halabi et al. 2012] abordam o problema de agregação e roteamento em redes ópticas multi-domínio. A especificidade deste problema está no cálculo das rotas de atendimento, que realizado de forma eficiente provê grandes ganhos em termos de recursos utilizados. No contexto multi-domínio há detalhes que precisam ser analisados para que se possa comparar suas soluções com o problema de domínio único. Nos trabalhos, os autores focam em um modelo de roteamento hierárquico usando vértices integrados IP/WDM em um GMPLS na rede óptica multi-domínio, que implementa uma abstração de topologia "Full Mesh"e propõe um esquema interdomínio de agregação de tráfego e roteamento baseado em *lightpath*.

Não foi possível comparar os trabalhos existentes na literatura de forma direta com nossa abordagem, devido ao cenário escolhido e critério utilizado na função objetivo.

3. Definição do Problema e Formulação Matemática

O problema de roteamento e alocação de comprimentos de onda com agregação de tráfego em redes ópticas multi-domínio, o qual chamamos GRWA-MD, é NP-difícil por ser redutível ao RWA [Somani 2005]. As características consideradas são: tráfego estático (previsão das demandas é conhecida *a priori*), tráfego assimétrico, fluxo não bifurcado, agregação de tráfego com múltiplos saltos e todos os vértices equipados com *cross connect switches* ópticos. O problema GRWA-MD pode ser formulado como um problema de programação linear inteira (PLI). A formulação proposta é baseada em variáveis que representam caminhos. Esta formulação possibilita um tratamento especial, através de uma abordagem de Geração de Colunas (GC) [Lasdon 1970, Desaulniers et al. 2005], que pode ser usada em uma estrutura heurística ou exata, garantindo, nesse caso, soluções ótimas.

Este problema é definido como segue. Dados os parâmetros a seguir, o objetivo é minimizar o número total de comprimentos de onda utilizados em todos os *lightpaths*, atendendo as demandas de todas as requisições sem exceder as capacidades dos canais de comprimento de onda. Cada canal pode ser compartilhado por diferentes requisições, isto é, a agregação de tráfego é aplicada contanto que a capacidade de cada comprimento de onda seja respeitada.

Sejam os seguintes parâmetros: $G^P(V, A)$ - Grafo da rede física, onde V é o conjunto de vértices e A é o conjunto de arcos conectando os vértices; $G^L(V, L)$ - Grafo da rede lógica, onde V é o conjunto de vértices e L é o conjunto de *lightpaths* conectando os vértices; D - Conjunto de domínios; L - Conjunto de *lightpaths*; W - Conjunto de comprimentos de onda; K - Conjunto de requisições; P^k - Conjunto de caminhos simples elementares que conectam os vértices da requisição k; r^k - Demanda da requisição $k \in K$; C - Capacidade de um comprimento de onda.

Sejam as constantes: a_l^p, assumindo valor 1se o *lightpath* l está no caminho p (0, caso contrário); b_{ij}^l, assumindo valor 1, se o arco (i, j) está no *lightpath* l (0, caso contrário). Sejam as variáveis da formulação: λ_p^k, assumindo valor 1, se o caminho p é usado para rotear a demanda $k \in K$ e 0, caso contrário; y_l^w assumindo valor 1, se o comprimento de onda w é atribuído ao *lightpath* $l \in L$ e 0, caso contrário.

A formulação baseada em caminhos, denominada F é dada por:

$$\min \left[\sum_{w \in W} \sum_{l \in L} y_l^w \right] \tag{1}$$

sujeito ao seguinte conjunto de restrições.

Restrições de roteamento, tomadas a cada requisição $k \in K$:

$$\sum_{p \in P^k} \lambda_p^k = 1, \qquad \forall k \in K, \tag{2}$$

Restrições de atribuição de comprimento de onda, para cada comprimento de onda w em cada arco (i, j):

$$\sum_{l \in L} b_{ij}^l y_l^w \leq 1, \qquad \forall w \in W, \forall (i, j) \in A, \tag{3}$$

As restrições de capacidade, também acoplando variáveis:

$$\sum_{k \in K} \sum_{p \in P^k} r^k a_l^p \lambda_p^k \leq C \sum_{w \in W} y_l^w \qquad \forall l \in L, \tag{4}$$

Restrições de alocação dos comprimentos de onda:

$$y_l^w - y_l^{w+1} \geq 0 \qquad \forall l \in L, \forall w < |W|, \tag{5}$$

Restrições gerais:

$$y_l^w \in \{0,1\} \qquad \forall w \in W, \forall l \in L, \tag{6}$$

$$\lambda_p^k \in \{0,1\} \qquad \forall k \in K, p \in P^k. \tag{7}$$

A função objetivo (1) minimiza o número total de comprimentos de onda utilizados nos lightpaths da rede. Restrições (2) garantem a existência de um caminho para rotear cada requisição $k \in K$. As desigualdades (3) asseguram que os *lightpaths* que compartilham arcos físicos utilizem comprimentos de onda distintos. Restrições (4) acoplam variáveis de caminho e *lightpaths*, impondo que o somatório das demandas que atravessam um *lightpath* seja menor ou igual ao somatório das capacidades de todos os comprimentos de onda desse *lightpath*. Restrições (5) garantem que os comprimentos de onda serão alocados de forma sequencial (observe que a ausência de tais restrições não inviabiliza o problema). Finalmente, restrições (6) e (7) definem os limites das variáveis. Observe que as restrições para garantir que o roteamento de requisições intra-domínio nunca alcancem domínios externos não é tratada diretamente nesta formulação. Estas serão consideradas no problema de precificação associado à esta formulação, isto é, no subproblema de gerar caminhos viáveis para atender as requisições.

4. Heurísticas

4.1. Heurística Centralizada baseada em Geração de Colunas (HGC)

Apesar da formulação (1)-(7) apresentar um número exponencial de variáveis (também chamadas de colunas no método GC), o uso de um algoritmo de geração de colunas nos permite avaliar limites de Programação Linear (PL) decorrentes da remoção das restrições de integralidade nas variáveis. Como sugerido pelo nome, em um algoritmo de geração de colunas, o conjunto completo de colunas não é considerado de imediato. Ao contrário, apenas um conjunto restrito de colunas ($|Q^k| << |P^k|$) é tomado inicialmente, seguido por passos iterativos de resolução do programa master (também chamado de Programa Master Restrito - PMR) e geração de colunas *on-the-fly*, quando necessário. O procedimento continua enquanto existem colunas atrativas a serem adicionadas ao PMR. Quando não houverem mais colunas a serem adicionadas, a relaxação linear do problema foi resolvida e um limite inferior (em um problema de minimização) para a função objetivo do problema é alcançada. O número total de colunas demandado pelo algoritmo no final do procedimento é tipicamente uma pequena fração do número total de colunas possíveis.

4.1.1. Limites inferiores derivados pela Geração de Colunas

Para entender como os limites de PL derivados por (1)-(7) são avaliados, vamos associar variáveis duais (π, γ, β, ϕ) às restrições (2), (3), (4) e (5), respectivamente. Essas variáveis serão usadas pelo problema de precificação, isto é, o problema capaz de identificar novas colunas a serem inseridas no PMR. O problema de precificação deve encontrar um

caminho mínimo para cada par origem/destino das requisições. Assim, a cada iteração do método de geração de colunas, o problema de precificação é resolvido $|K|$ vezes. Um algoritmo para encontrar o caminho mínimo entre dois vértices em um grafo como *Dijkstra* pode ser aplicado. Para testar se o caminho gerado é atrativo ou não para entrar no PMR, os valores duais da iteração anterior são utilizados para precificar os caminhos. Em termos matemáticos, deve-se verificar se as restrições duais são violadas:

$$\pi^k + \sum_{l \in L} r^k a_p^l \beta_l \leq 0, \forall k \in K, \forall p \in P^k \tag{8}$$

Em nossa implementação, os conjuntos iniciais Q^k são gerados como segue. Para garantir uma solução viável para o problema, um caminho que respeite as restrições de domínios é provido para cada requisição $k \in K$. Assim, é computado o menor caminho conectando os vértices de cada requisição k em termos de número de saltos.

4.1.2. Princípio de Funcionamento

Uma heurística bastante usada para obter soluções primais viáveis para o problema original é conhecida como *Heurística Master Restrito* [Joncour et al. 2010]. A idéia principal é resolver a relaxação de PL do problema através do procedimento de geração de colunas descrito acima e resolver o PMR resultante com restrições de integralidade nas variáveis. Assim, utilizamos a idéia da Heurística Master Restrito, com a diferença de que o PMR não é resolvido apenas uma vez. O PMR é resolvido um número específico de vezes, durante o processo de ramificação ou *branching* empregado. O processo de ramificação consiste em impor limites às variáveis fracionárias, em busca de soluções inteiras para o problema. Este é baseado na dicotomia das variáveis, isto é, a variável pode assumir o valor zero ou o valor um. As variáveis utilizadas neste processo pertencem ao conjunto $\{y_l^w : \forall w \in W, \forall l \in L\}$.

Resolver um programa linear inteiro (PLI) sobre as colunas existentes nos permite alcançar uma solução inteira viável. Este passo pode levar um tempo considerável, dependendo do tamanho do PLI. Assim, foi estabelecido um limite de tempo na execução do PLI Master Restrito, além de um número máximo de iterações deste processo.

4.1.3. Pseudo-código

O algoritmo 1 descreve os passos principais seguidos pela heurística proposta. Na linha 1, o PMR é criado. A linha 2 inicia o conjunto de colunas para garantir uma solução viável para o PMR. A variável $melhora$ na linha 3 indica se uma nova coluna atrativa foi encontrada a cada iteração. A variável $iter$ é iniciada na linha 4. A linha 5 estabelece o critério de parada do método, baseado em um número máximo de iterações. Na linha 6, as variáveis que sofreram *branching* são fixadas no PMR. A variável $coluna$ na linha 7 inicia vazia. Na linha 8, $solucao$ recebe a primeira solução (relaxada) do PMR. A linha 9 indica que enquanto existirem colunas atrativas, o procedimento continua. Da linha 11 à linha 18, o mesmo procedimento é repetido para cada requisição $k \in K$: os valores duais são atualizados para o problema de precificação (onde o algoritmo *Dijkstra* foi empregado), é verificado se a coluna gerada é atrativa (apresenta um custo reduzido negativo) e esta é adicionada ao PMR em caso positivo. Em seguida, nas linhas 20-22, são criados dois

subproblemas baseado na variável fracionária escolhida, fixando seu valor em zero ou um, caso esta solução não seja inteira e tenha um objetivo menor do que a melhor solução corrente. As linhas 23-25 implicam que a cada x iterações, definidas pelo parâmetro $numIteracoes$, o problema PMR é resolvido aplicando restrições de integralidade nas variáveis, para obter uma solução inteira. A linha 26 define que a solução corrente é atualizada caso necessário e finalmente, na linha 28, a melhor solução encontrada é retornada como resposta.

Algorithm 1 HGC

```
1: PMR ← criaPMR();
2: geraColunasIniciais(PMR);
3: melhora ← true;
4: iter ← 0;
5: while iter ≤ maxIter do
6:     fixaVariaveis(PMR);
7:     coluna ← ∅;
8:     solucao ← solverPL(PMR);
9:     while melhora do
10:        melhora ← false;
11:        for all k ∈ K do
12:            atualizaValoresDuais();
13:            coluna ← dijkstra();
14:            if f(coluna) < 0 then
15:                adicionaAoPMR(coluna);
16:                melhora ← true;
17:            end if
18:        end for
19:    end while
20:    if !eInteira(solucao) &&solucao.objetivo < melhorSolucao.objetivo then
21:        criaSubproblemas();
22:    end if
23:    if iter%numIteracoes == 0 then
24:        solution ← solverPLI(PMR);
25:    end if
26:    atualizaMelhorSolucao();
27: end while
28: return melhorSolucao;
```

4.2. Heurística Descentralizada baseada em GRASP

O *Greedy Randomized Adaptive Search Procedure* (GRASP) é uma meta-heurística comumente aplicada em problemas de otimização. O algoritmo consiste em iterações que realizam sucessivas construções gulosas e buscas locais para melhoria da solução atual. Portanto, o algoritmo tem uma estratégia multipartida adaptativa. A fase de construção da solução inicial é chamada fase construtiva, e resume-se a gerar uma solução gulosa guiada por uma lista restrita de candidatos. Esta lista é considerada no momento de escolher qual é o próximo elemento a fazer parte da solução. As seções a seguir explicam o que é feito em cada fase do algoritmo.

4.2.1. Fase construtiva

A fase construtiva do GRASP implementada para o problema de agregação de tráfego em redes ópticas é baseada em uma análise das requisições, que indicará quais delas promovem o menor aumento no valor da função objetivo ao serem atendidas. Uma rota de atendimento para cada requisição é calculada, e é atribuído à ela um valor com o número

de comprimentos de onda necessários para alocar tal requisição nos enlaces da rota calculada. Em seguida, o conjunto de requisições é ordenado de acordo com os valores obtidos anteriormente. Este procedimento representa a parte adaptativa do método (trecho entre as linhas 2 e 3 correspondente no Algoritmo 2), pois como a cada escolha de um item a ser incorporado à solução o estado da rede pode estar alterado, outras requisições, ao serem roteadas, podem utilizar menos recursos da rede. Ao final deste procedimento tem-se uma solução inicial com o atendimento de todas as requisições, mas com uma ordem de atendimento que prioriza a economia dos recursos (trecho entre as linhas 4 e 6). O Algoritmo 2 ilustra a implementação do método:

Algorithm 2 Fase construtiva

Require: $requisicoes$, $rede$, α
Ensure: $solucao\ inicial\ randomizada$
1: **while** $requisicoes \neq \varnothing$ **do**
2: $ordenarequisicoes()$;
3: $menorCusto \leftarrow rand()\%(\alpha \times requisicoes.size())$;
4: $req \leftarrow requisicoes[menorCusto]$;
5: $novaListaRequisicoes \leftarrow novaListaRequisicoes \cup req$;
6: $requisicoes \leftarrow requisicoes - req$;
7: $solucaoCorrente.alocaRequisicao(req)$;
8: **end while**
9: $solucaoCorrente.setRequisicoes(novaListaRequisicoes)$;
10: **return** $solucaoCorrente$;

A condição de parada adotada é baseada em número de iterações e é representada pelo laço entre as linhas 1 e 7. Os passos entre as linhas 4 e 6 atualizam a melhor solução encontrada globalmente. O parâmetro α define a porcentagem da lista de candidatos que será levada em consideração na construção da solução inicial.

Algorithm 3 GRASP

Require: $requisicoes$, $rede$, α
Ensure: $melhor\ solucao\ dentre\ os\ minimos\ locais\ analisados$
1: **while** $condicao\ de\ parada\ nao\ for\ satisfeita$ **do**
2: $solucaoCorrente \leftarrow faseConstrutiva(requisicoes, rede, \alpha)$;
3: $novaSolucao \leftarrow buscaLocal(solucaoCorrente)$;
4: **if** $novaSolucao.objetivo \leq melhorSolucao.objetivo$; **then**
5: $melhorSolucao \leftarrow novaSolucao$;
6: **end if**
7: **end while**
8: **return** $melhorSolucao$;

4.2.2. Algoritmos para o problema de roteamento inter-domínio

O principal aspecto que diferencia o roteamento intra-domínio e o roteamento inter-domínio é o fato de que no primeiro caso, informações sobre o domínio em questão são suficientes para realizar o roteamento; sendo que no segundo caso, não é possível conhecer como o tráfego será tratado internamento no domínios por onde a requisição precise atravessar para chegar ao seu domínio de destino. Na realidade, os domínios possuem políticas internas de roteamento que não são de conhecimento de domínios adjacentes, o que dificulta a resolução do problema de forma integrada.

Uma das abordagens clássicas da literatura para lidar com este problema é baseada em um modelo hierárquico. O modelo hierárquico define que o roteamento será realizado

em vários níveis. No problema em questão temos dois níveis: o mais externo diz respeito às rotas inter-domínios e o mais interno às rotas intra-domínio. O modelo proposto para resolver este problema pode ser chamado de *Abstração Hierárquica*.

Há cenários em que o roteamento inter-domínio se torna bem complexo. Primeiramente tem-se o caso simples: supondo que os pares de domínios adjacentes estão conectados por apenas um enlace, o roteamento de demandas de transbordo é feita simplesmente calculando uma rota entre os roteadores de borda dos dois domínios. Neste caso, cada domínio pode realizar todo o processo de roteamento de forma independente. O problema do roteamento inter-domínio é complexo quando domínios adjacentes estão conectados por mais de um enlace. Utilizando uma abstração em que cada domínio é representado por um único vértice, nota-se que o grafo resultante é um multigrafo. O algoritmo de menor caminho originalmente não resolve de forma satisfatória o cálculo de rotas porque sempre leva em consideração o vértice antecessor. Isso faz com que a informação de qual enlace será utilizado na rota seja perdido. Considerando uma solução para o cálculo do menor caminho em um multigrafo não resolve o problema da melhor forma. Uma vez definido que uma requisição será roteada para fora de um domínio através de um roteador de borda específico, o próximo roteador contido no domínio adjacente já estaria definido como pertencente à rota de atendimento da requisição. Isso torna o problema de roteamento intra-domínio dependente de informações oriundas de um domínio adjacente. Sendo assim, utiliza-se uma representação em que os roteadores de borda também fazem parte da topologia de nível mais externo. Primeiramente porque eles são acessíveis pelas rotas de atendimento inter-domínio e devido a um fato particular, já mencionado anteriormente, que é o de evitar a geração de um multigrafo na abstração.

Dada uma rede com um conjunto de domínios, a abstração será obtida a partir de todos os roteadores de borda presentes na rede original, isto é, cada domínio passa a ser representado pelo conjunto de seus roteadores de borda, que são todos conectados entre si. Para cada requisição com seu par origem-destino em domínios diferentes, uma rota de atendimento será obtida desde o domínio de origem até o domínio de destino através dos dois níveis definidos. Inicialmente, é realizado o roteamento no nível mais externo, ou seja, a partir de um vértice artificial do domínio de origem, a requisição é roteada através dos roteadores de borda da *Abstração Hierárquica*, até atingir o vértice artificial do domínio de destino. Em seguida, são calculadas as rotas internas aos domínios, que possuem apenas informações locais. O custo total da rota de atendimento de uma requisição é o somatório dos custos internos de cada domínio pelo qual a requisição trafegou somado ao custo de cada enlace inter-domínio que a rota utilizou na *Abstração Hierárquica*.

Supondo que dois domínios adjacentes sejam conectados por mais de um enlace, na topologia física haverá vértices distintos que fazem parte da conexão. Tal aspecto torna o problema de roteamento de primeiro nível idêntico ao problema de *grooming* em redes ópticas, logo as rotinas utilizadas para resolver o problema com apenas um domínio poderão ser aplicadas neste ponto, com as devidas adaptações.

Uma vez definidas as rotas de atendimento de todas as requisições que possuem origem e destino em domínios diferentes, é necessário informar a cada um dos domínios as respectivas requisições que serão de sua responsabilidade no roteamento interno. Cada domínio calcula suas próprias rotas de atendimento para as requisições internas. É impor-

tante salientar que a qualidade da solução pode ficar prejudicada ao passo que são adicionadas restrições à topologia da rede. O modelo hierárquico utiliza apenas informações locais; ao contrário da heurística anterior, que considera o conhecimento da rede por completo.

5. Experimentos Computacionais

Nesta seção, apresentamos resultados computacionais para o problema, obtidos através das duas heurísticas propostas. Foram utilizadas topologias de redes reais bastante conhecidas, que encontram-se disponíveis na SNDlib (disponível online em `http://sndlib.zib.de/`). As características das redes utilizadas em nossos experimentos são apresentadas na Tabela 1. As requisições encontram-se disponíveis juntamente com as topologias, na biblioteca da SNDlib. Foram definidos domínios específicos para cada topologia, sendo que para duas delas, apenas um domínio foi adotado. Desta forma, procuramos criar instâncias com diversas características para avaliar o comportamento dos métodos propostos. O número de vértices varia entre 10 e 50, enquanto o número de requisições pode ser bem pequeno, como no caso da rede `di-yuan` que possui apenas 22 requisições até alcançar valores como 1482 requisições, como no caso da rede `janos-us-ca`. A Figura 1 mostra as todas as redes utilizadas neste trabalho. Foi estipulado ainda, que cada comprimento de onda tem a capacidade de 2000 unidades de fluxo, e que cada arco da rede suporta até 200 comprimentos de onda.

Todos os testes computacionais reportados nessa seção foram conduzidos com uma máquina Intel Xeon Core 2 Quad, com 2GHz e 8Gb de memória RAM, executando sob o sistema operacional Linux. Foi utilizada a versão 12.1 do CPLEX na resolução do PMR da HGC.

topologia	$\|V\|$	$\|A\|$	$\|D\|$	$\|K\|$
atlanta	15	44	2	210
dfn-bwin	10	90	1	90
di-yuan	11	84	1	22
france	25	90	3	300
germany	50	176	4	662
janus-us-ca	39	122	3	1482
newyork	16	98	3	240
norway	27	102	2	702
pioro	40	178	5	780
polska	12	36	3	66

Tabela 1. Redes reais da SNDlib

topologia	FO		Tempo (s)	
	HGC	GRASP	HGC	GRASP
atlanta	**149**	212	2323.66	**112.75**
dfn-bwin	303	**297.75**	**388.92**	469.75
di-yuan	11	**10.62**	429.57	**24.60**
france	**141**	159.87	1690.29	**384.92**
germany	82	**72.12**	**1367.19**	1713.67
janos-us-ca	**2952**	3485.25	**3262.99**	10875.62
newyork	25	**21.5**	**29.09**	168.69
norway	41	**38.62**	**76.77**	2863.23
pioro	**241**	306.75	4756.91	**3525.14**
polska	**22**	24.87	102.56	**17.66**

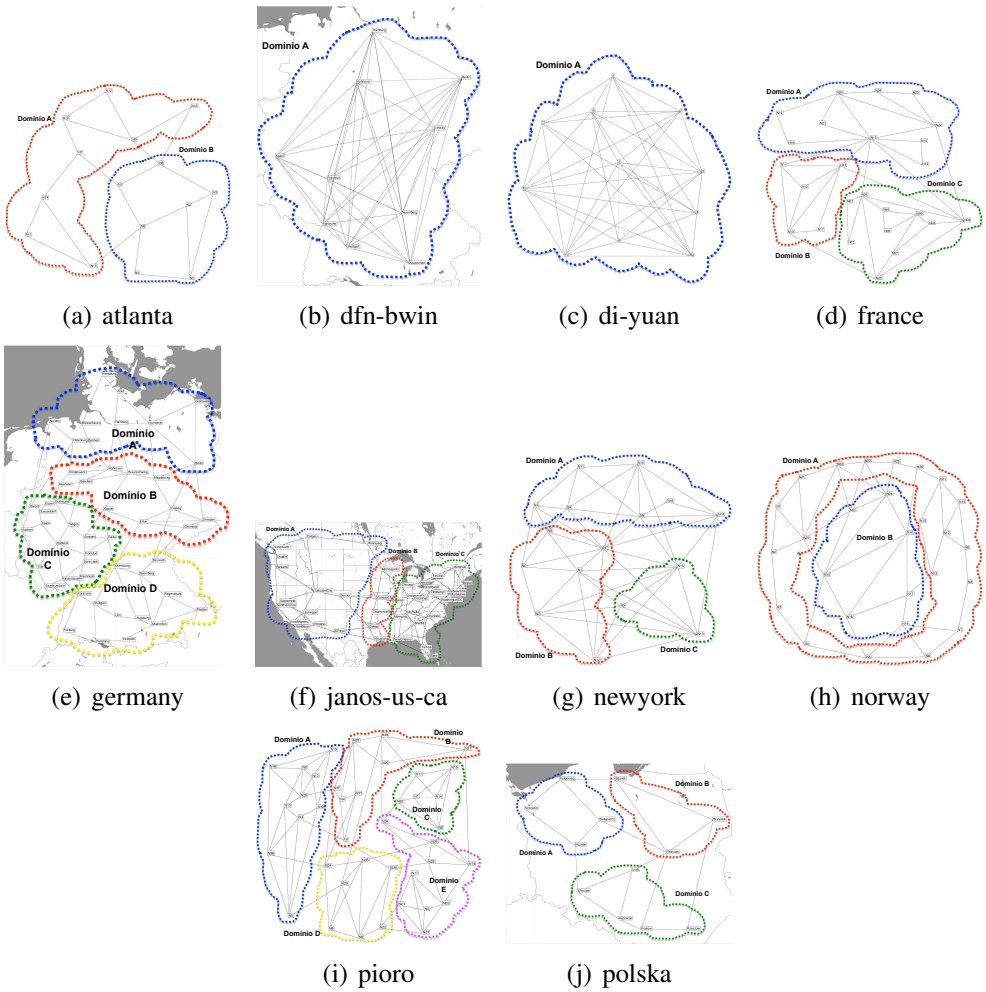
Tabela 2. Resultados computacionais por instância

A Tabela 2 apresenta resultados computacionais para as instâncias analisadas, fornecendo os valores da função objetivo (FO) e tempo em segundos para cada uma das heurísticas propostas. Analisando os dados da Tabela 2, é possível observar que quando a heurística GRASP foi capaz de alcançar resultados melhores em termos da FO para as instâncias em questão, o tempo de execução foi mais elevado, exceto no caso da instância `di-yuan`. Embora o método GRASP tenha como critério de parada um determinado número de iterações, em determinados casos a busca local pode consumir um tempo considerável, com o benefício de melhor explorar o espaço de soluções e assim, alcançar melhores soluções.

Analisando as características das instâncias e os resultados obtidos, pode-se notar

que a heurística HGC apresenta melhores resultados em redes mais esparsas. Contudo, nos casos em que a heurística HGC teve um comportamento inferior, observa-se que os tempos computacionais associados foram baixos, indicando que o número de iterações pode ser aumentado nesse caso, a fim de encontrar melhores soluções em tempos ainda aceitáveis. O método foi avaliado utilizando o mesmo parâmetro para todas as instâncias, para analisar seu comportamento, mas é importante ressaltar que estes parâmetros podem ser ajustados caso a caso.

Ainda, é possível observar que o GRASP possui um comportamento melhor do que a heurística *HGC* quando os domínios possuem um maior de número de vértices e arestas do que a abstração hierárquica gerada. Isso indica que o problema de roteamento e atribuição de comprimentos de onda tem a principal fonte de melhorias na função objetivo na resolução dos subproblemas multi-domínio. Quando é feita a abstração hierárquica, algumas informações não são levadas em consideração para realizar o roteamento nos níveis internos nos domínios. Então, quanto maior for as dimensões dos domínios, maior a chance do GRASP obter resultados mais satisfatórios.



(a) atlanta (b) dfn-bwin (c) di-yuan (d) france

(e) germany (f) janos-us-ca (g) newyork (h) norway

(i) pioro (j) polska

Figura 1. Topologias de Rede da SNDlib

A Tabela 3 apresenta os resultados computacionais obtidos por domínio de cada instância. Observa-se que nos casos em que a heurística GRASP apresentou melhores

resultados de FO, seus resultados individuais para cada domínio também são melhores. De forma geral, as duas heurísticas obtiveram resultados próximos. Assim, embora uma abordagem seja centralizada e a outra descentralizada e baseada em informações locais, percebe-se que não há uma perda significativa na qualidade dos resultados quando o conjunto completo de informações não esta disponível. Podemos concluir que a abordagem descentralizada, que é mais apropriada para cenários reais, teve um desempenho bom em relação à abordagem centralizada, que muitas vezes não é escalável. A comparação das duas abordagens confirma que métodos de otimização são capazes de encontrar soluções de alta qualidade para o problema de agregação de tráfego em redes multi-domínio com eficiência.

Tabela 3. Resultados computacionais por domínio

topologia	Algoritmo	Domínio				
		A	B	C	D	E
atlanta	HGC	40	85	-	-	-
	GRASP	88	100	-	-	-
dfn-bwin	HGC	303	-	-	-	-
	GRASP	297.75	-	-	-	-
di-yuan	HGC	11	-	-	-	-
	GRASP	10.62	-	-	-	-
france	HGC	47	13	45	-	-
	GRASP	50.61	18.32	55.73	-	-
germany	HGC	18	16	29	15	-
	GRASP	17.12	14.35	19.93	16.23	-
janos-us-ca	HGC	652	69	1813	-	-
	GRASP	806.34	275.01	1971.62	-	-
newyork	HGC	11	8	3	-	-
	GRASP	6.74	6.91	2.94	-	-
norway	HGC	34	9	-	-	-
	GRASP	28.45	8.12	-	-	-
pioro	HGC	39	32	17	34	34
	GRASP	50.78	43.32	23.34	35.71	51.06
polska	HGC	5	4	7	-	-
	GRASP	6.13	4.52	9.87	-	-

6. Conclusões

Neste trabalho foram apresentadas duas heurísticas para o problema de roteamento com agregação de tráfego em redes ópticas multi-domínio. O problema em estudo é NP-completo e soluções exatas para o mesmo nem sempre são aplicáveis a redes de tamanho real. Os resultados computacionais indicam que de forma geral, as duas heurísticas obtiveram resultados próximos. A abordagem descentralizada é mais apropriada para cenários reais e teve um desempenho bom em relação à abordagem centralizada, que muitas vezes não é escalável. A comparação das duas abordagens confirma que o uso de métodos de otimização é importante no projeto de redes ópticas multi-domínio, sendo capaz de encontrar soluções de alta qualidade para o problema de agregação de tráfego com eficiência.

Como trabalhos futuros, pretende-se estudar a versão dinâmica do problema. A aplicação de abordagens multi-período e o uso de simulação poderão ser empregados para tornar os cenários avaliados ainda mais reais.

Agradecimentos

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Fundação de Amparo à Pesquisa do estado de Minas Gerais (FAPEMIG).

Referências

Barr, R. S., Kingsley, M. S., and Patterson, R. A. (2006). *Telecommunications Network Grooming*, chapter 29, pages 837–862. Springer US, New York, USA.

De, T., Pal, A., and Sengupta, I. (2010). Traffic grooming, routing, and wavelength assignment in an optical WDM mesh networks based on clique partitioning. *Photonic Netw. Commun.*, 20(2):101–112.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer.

Halabi, W., Steenhaut, K., Goossens, M., and Nowe, A. (2011). Routing and traffic grooming in multi-domain optical networks. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on*, pages 1 –7.

Halabi, W., Steenhaut, K., Goossens, M., Truong, T.-H., and NowÈ, A. (2012). Hierarchical routing and traffic grooming in ip/mpls-based ason/gmpls multi-domain networks. *Photonic Network Communications*, 23:217–229. 10.1007/s11107-011-0352-9.

Hu, J. and Leida, B. (2004). Traffic Grooming, Routing, and Wavelength Assignment in Optical WDM Mesh Networks. In *Proceedings of the IEEE INFOCOM*, pages 495–501.

Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., and Vanderbeck, F. (2010). Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36(0):695–702. ISCO 2010 - International Symposium on Combinatorial Optimization.

Lasdon, L. S. (1970). *Optimization Theory for Large Scale Systems*. McMillan Company, New York, USA.

Mukherjee, B. (1997). *Optical Communication Networks*. McGraw-Hill.

Mukherjee, B. (2006). *Optical WDM Networks*. Springer.

Somani, A. K. (2005). *Survivability and traffic grooming in WDM optical networks*. Cambridge University Press.

Truong, D.-L. and Jaumard, B. (2012). A novel topology aggregation approach for shared protection in multi-domain networks. *Optical Switching and Networking*, 9(2):81 – 96. IEEE ANTS 2010.

Vignac, B., Vanderbeck, F., and Jaumard, B. (2009). Reformulation and decomposition approaches for traffic routing in optical networks. Research report, INRIA.

Zhu, K. and Mukherjee, B. (2002). Traffic grooming in an optical WDM mesh network. *IEEE Journal on Selected Areais in Communications*, 20(1):122–133.

Zhu, K. and Mukherjee, B. (2003). A review of traffic grooming in WDM optical networks: Architectures and challenges. *Optical Networks Magazine*, 4(2):55–64.

Zhu, Y., Jukan, A., and Ammar, M. (2003). Multi-segment wavelength routing in large-scale optical networks. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 2, pages 1381 – 1385 vol.2.

Proteção Parcial para Demandas de Alta Capacidade em Redes WDM Utilizando Mecanismo de Bloqueio Preventivo

Paulo J. S. Júnior[1], **André C. Drummond**[2]

[1] Departamento de Engenharia Elétrica
Faculdade de Tecnologia
Universidade de Brasília

[2]Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade de Brasília

{paulojunior,andred}@cic.unb.br

Abstract. *In high-capacity networks, some services and applications such as HD video (4k and 8k) streams will demand the establishment of connections with extremely high bandwidth requirements, sometimes higher than the capacity of a single wavelength, in WDM optical networks. In order to provide enough bandwidth to this applications and, at the same time to guarantee reliability, it is necessary to route the traffic through multiple parallel lightpaths. This article proposes a preventive dynamic blocking mechanism that avoid bottlenecks in the network leading to a high efficient algorithm for the provisioning of resources with partial protection for requests with extremely high bandwidth requirements in WDM optical networks.*

Resumo. *Em redes de alta capacidade, alguns serviços e aplicações como vídeo HD (2k, 4k e 8k) demandam o estabelecimento de conexões com requisito de largura de banda de alta capacidade, as vezes maior do que a capacidade de um comprimento de onda em redes WDM. Com objetivo de prover banda suficiente para estas aplicações e ao mesmo tempo garantir confiabilidade, é necessário rotear o tráfego através de multiplos caminhos ópticos em paralelo. Este artigo propõem um algoritmo que utiliza um mecanismo dinâmico de bloqueio preventivo para evitar a criação de gargalos na rede aumentando a eficiência do algoritmo no aprovisionamento de recursos com proteção parcial para requisições de alta capacidade em redes ópticas WDM.*

1. Introdução

Nos últimos anos, observa-se o crescimento na implantação de redes ópticas com multiplexação por comprimentos de onda (WDM) na infraestrutura de transporte da Internet [Dutta and Rouskas 2002]. Esta nova tecnologia aumentou vertiginozamente a quantidade de banda passante disponível nas redes e, da mesma forma, 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].

As redes WDM suportam a implantação de múltiplos canais de transmissão paralelos por fibra óptica utilizando-se 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. Além disso, devido a grande capacidade de banda passante destes caminhos, os fluxos da camada superior são agregados de forma a melhor utilizar os recursos da rede. Essa técnica é conhecida como agregação de tráfego (*traffic grooming*) [Dutta and Rouskas 2002], e define o problema de rotear uma demanda de tráfego sobre uma topologia virtual, i.e., sobre a rede de caminhos ópticos estabelecidos na rede.

A tecnologia WDM é amplamente utilizada em redes de transporte de alto nível (*Tier 1*) que possuem enlaces que atravessam longas distâncias, o que aumenta o risco de falhas, seja por causa de cortes acidentais de suas fibras, falhas em equipamentos, ou mesmo devido a ataques à sua infraestrutura. Devido a esse risco, é necessário implementar mecanismos de sobrevivência utilizando-se técnicas de proteção e/ou restauração de caminhos ópticos [Ramamurthy et al. 2003] [Yao 2005].

A proteção total é a forma mais utilizada e tradicional nas redes operacionais, porém, pode não ser indicada para as aplicações de alta capacidade, dado que a demanda de banda passante necessária para se prover redundância de caminhos impactaria significativamente na disponibilidade de recursos para outras aplicações da rede. Nesse contexto, mecanismos de proteção parcial podem ser utilizados, o que reduziria o impacto sobre a rede e, também, permitiria que aplicações com tolerância à perda de dados podessem operar sem que haja prejuízo na qualidade experimentada (QoE) pelos usuários [Kim and Choi 2010] [Padmanabhan et al. 2003]. Como exemplo, pode-se citar as aplicações de distribuição de vídeo de fluxo contínuo que utilizam codificação por múltiplos descritores. A proteção parcial garante que, mediante a ocorrência de uma falha na rede, ou mais especificamente a falha de um enlace, 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]. As técnicas de roteamento multicaminhos e de proteção parcial permitem a obtenção de soluções que apresentam melhor desempenho e menor custo, mitigando o impacto das aplicações de alta capacidade sobre a rede [André Costa Drummond and Barreto 2012]. A proteção parcial busca dividir a demanda de tráfego a ser protegida em múltiplos caminhos ópticos disjuntos (não compartilham enlaces) de forma que cada caminho transporte uma quantidade de tráfego menor ou igual ao nível máximo de tolerância a falhas da aplicação.

Este artigo introduz um algoritmo de roteamento adaptativo que trata do problema de agregação e proteção de tráfego em redes ópticas WDM para aplicações de alta capacidade com proteção parcial. O algoritmo proposto utiliza a técnica de roteamento multicaminhos, motivado pelos resultados do trabalho [Chen et al. 2009] de um dos autores, que mostra a eficiência desta técnica em um cenário com demandas de alta capacidade, e evidencia a potencialidade da mesma na construção de mecanismos de proteção. 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 busca a redução do atraso diferencial.

O artigo está organizado da seguinte forma: na Seção 2 são apresentados trabalhos relacionados e apontadas algumas deficiências em soluções recentes da literatura; a Seção 3 introduz o algoritmo *Best-Cost with Partial Protection 2* - BCPP2; 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 adaptadas para as redes WDM. 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, e apresentam um algoritmo para redes WDM com conversão total de comprimentos de onda. Eles demonstram que a proteção parcial é uma solução mais barata, tão eficiente e confiável quanto a total.

O estudo [Das et al. 2009] propõe um algoritmo que objetiva atingir a máxima proteção parcial possível. Os autores inicialmente propuseram o algoritmo *Min-Cut Disjoint Paths*, MDP, que resolve o problema de aprovisionamento com proteção parcial. O MDP procura por um conjunto de caminhos disjuntos de forma a assegurar que nenhum caminho carregue mais fluxos do que o permitido, ou seja, que nenhum caminho transporte uma quantidade de banda passante superior ao percentual máximo de perda suportado pela proteção parcial contratada. Para achar um bom conjunto de caminhos disjuntos, o MDP usa a técnica de busca de caminhos proposta em [Huang et al. 2011]. 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], os autores exploram uma solução de proteção compartilhada e desenvolvem um algoritmo que também utiliza roteamento multicaminho para provisão de proteção. 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.

Em ambos os estudos [Das et al. 2009, Huang et al. 2011], e também no estudo recente [Das et al. 2011] , é assumido que a rede WDM possui a capacidade de conversão total de comprimentos de onda, ou seja, a capacidade de utilizar quaisquer comprimentos de ondas livres nos enlaces ao longo de um caminho, tal premissa requer a implementação de uma arquitetura de alto custo e, portanto, na prática não é realista. Além disso, ao assumir esta premissa, o problema de RWA (NP-Completo [Dutta and Rouskas 2002]) reduz-se ao problema de roteamento clássico.

3. Algoritmo Proposto

O algoritmo proposto, *Best-Cost with Partial Protection 2* - BCPP2, foi desenvolvido 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, apresenta uma nova técnica de seleção de rotas, propõem um mecanismo de bloqueio preventivo e implementa o controle do retardo diferencial.

Para obter proteção parcial e realizar a alocação de recursos de maneira mais eficiente, o algoritmo BCPP2 adota uma estratégia inicial de obter, de maneira *off-line*, todas as rotas para todos os pares origem-destino da rede. Em seguida, implementa um mecanismo de bloqueio preventivo, criando duas listas de enlaces de interesse, *blackList* e *grayList*, utilizando-as para limitar o conjunto de rotas que serão efetivamente consideradas na busca por caminhos. O algoritmo, então, ordena o conjunto de rotas restantes de acordo com a quantidade de recursos das mesmas, e prossegue para a escolha final de um conjunto de caminhos disjuntos que atenda os requisitos de banda passante, o nível de proteção da chamada e que possua um baixo atraso diferencial.

3.1. BCPP2 - Best-Cost with Partial Protection

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 BCPP2. O algoritmo recebe como entradas as topologias física e virtual da rede e um conjunto de rotas. O conjunto de rotas é obtido através da execução de uma rotina *off-line* que gera uma lista com todas as rotas possíveis para todos os pares origem-destino da rede. O objetivo da geração de todas as rotas é o de permitir ao algoritmo realizar uma escolha mais criteriosa dos caminhos que serão efetivamente utilizados na composição da solução do problema, evitando a criação de gargalos na rede [Ho and Lee 2007a].

Para cada chamada, o algoritmo inicia executando um mecanismo de bloqueio preventivo. Primeiramente, o algoritmo obtêm o estado dos enlaces da rede e estabelece duas listas, chamadas *blackList* e *grayList*. A *blackList* é composta por todos os enlaces que possuam menos de 20% de sua capacidade de banda passante disponível e, portanto, contém os enlaces que estão bastante congestionados. A *grayList*, por sua vez, é composta por enlaces que não entraram na *black-list*, mas que apresentam certas características que indicam que os recursos alocados nos mesmos não serão liberados rapidamente, ou seja, são enlaces que possuem maior probabilidade de entrarem na *black-list*. Os enlaces que se enquadram nesta lista são os enlaces que atendem uma pequena quantidade de fluxos, até 30% da capacidade, e possuem pouca pouca banda passante disponível, menos de 30%

BCPP2 - Best-Cost 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: $P = pairPaths(Rotas, s, d) - (blackList() + grayList())$
3: $P_{glp} = GLP(VT, s, d)$
4: $P_{vlp} = VLP(P, P_{glp}, s, d)$
5: **Enquanto** $verdadeiro$ **faça**
6: $P_{aux} = sortCost(P_{vlp}, P_{glp})$
7: $P' = minDD(P_{aux})$
8: **Se** banda disponível em $P' \geq b$ **então**
9: **Se** $|P'| \geq \left\lceil \frac{b}{b(1-f)} \right\rceil$ **então**
10: $i = 0$
11: **Enquanto** banda alocada $< b$ **faça**
12: $P^* \leftarrow P'[i]$
13: Aloca o máximo de banda até $b(1-f)$
14: i++;
15: **Fim Enquanto**
16: **Fim Se**
17: **Fim Se**
18: **Se** Existir P^* **então**
19: Aceita a requisição
20: **Senão**
21: **Se** $|P_{vlp}| > 0$ **então**
22: $P_{vlp} = P_{vlp} \setminus P_{vlp}[0]$
23: **Senão**
24: break
25: **Fim Se**
26: **Fim Se**
27: **Fim Enquanto**
28: Rejeita a requisição
29: **Fim Para**

Figura 1. Algoritmo BCPP2

da capacidade. A capacidade de fluxos que podem ser atendidos por um enlace é igual a capacidade de um comprimento de onda dividido pela demanda de menor granularidade considerada, multiplicado pelo número total de comprimentos de onda do enlace. Em seguida, o algoritmo cria o conjunto P composto por todas as rotas possíveis entre os pares de comunicação da chamada (Função $pairPaths()$), sendo removidas 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 ou próximos do congestionamento são bloqueados, sendo removidos do conjunto P.

Na Linha 3 é executada a função $GLP()$ para se obter o conjunto P_{glp} de caminhos agregáveis, ou seja, caminhos ópticos já estabelecidos entre a origem e o destino da chamada que possuam alguma capacidade disponível. O próximo passo é executar a função $VLP()$ para se obter o conjunto P_{vlp} de caminhos viáveis, que são todas as rotas, dentre as disponíveis em P, que possuam pelo menos um comprimento de onda disponível fim-a-fim, ou seja, que podem ser utilizadas para a alocação de novos caminhos ópticos.

Após a definição dos conjuntos de caminhos agregáveis e viáveis, entre as Linhas 5 e 27, o algoritmo irá buscar por uma solução que atenda os requisitos da chamada.

Na Linha 6 é executada a função $sortCost()$ para a ordenação dos caminhos e agrupamento dos caminhos disjuntos de menor custo em um único conjunto P_{aux}, em seguida a função $minDD()$ avalia formas de rearranjar e selecionar os caminhos de P_{aux} gerando o conjunto P' de caminhos ópticos disjuntos com baixo retardo diferencial. O retardo diferencial é causado pela utilização de múltiplos caminhos que possuam diferentes retardos de propagação, o que pode acarretar na desordenação dos pacotes, requerendo a alocação de memória para o re-sequenciamento dos pacotes no nó de destino e, portanto, deve ser mitigado. O retardo diferencial entre dois caminhos p_i e p_j pode ser definido como $dd(p_i, p_j) = |d_{p_i} - d_{p_j}|$ [Ahuja et al. 2004].

Neste ponto, o algoritmo definiu o conjunto P', que possui os caminhos candidatos para a solução final de alocação, em seguida, nas Linhas 8 e 9 é avaliado se 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 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 existir uma solução, a chamada será aceita (Linha 19), e seu fluxo de dados será distribuído entre os caminhos ópticos agregáveis e/ou viáveis de P^*. Caso contrário, remove-se o primeiro caminho do conjunto P_{vlp} (Linha 22) e o algoritmo retorna para a Linha 6 para definir um novo conjunto candidato P'. Caso este processo iterativo não resulte em uma solução, ao esgotar os elementos de P_{vlp}, o algoritmo bloqueará a chamada.

3.2. Modelo de Custo

O custo C de um caminho, utilizado na Função $sortCost$, é calculado segundo a Equação 2.

$$C = \alpha_1.C_H + \alpha_2.C_D + \alpha_3.C_W + \alpha_4.C_F + \alpha_5.C_{BW} \tag{2}$$

No total, são utilizadas 5 métricas distintas para a composição do custo total de um caminho. Tal custo pode, ainda, sofrer um ajuste fino através da manipulação dos pesos α_x, o que permite a adaptação da sensibilidade do algoritmo a cenários específicos.

$$C_H = \frac{numHops}{maxHops} \tag{3}$$

$$C_D = \frac{Delay}{maxDelay} \tag{4}$$

$$C_W = 1 - \frac{availableW}{maxW} \tag{5}$$

$$C_F = 1 - \frac{numFlow}{maxFlows} \tag{6}$$

$$C_{BW} = 1 - \frac{availableBW}{maxBW} \tag{7}$$

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 atransparenteo 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, 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

Os exemplos numéricos apresentados nesta seção comparam o algoritmo BCPP2 com o estado da arte da literatuta, o algoritmo Smart-MDP [Das et al. 2009], e também com o algoritmo Smart-Real-MDP [André Costa Drummond and Barreto 2012] proposto anteriormente pelos mesmos autores deste trabalho. Para avaliar o desempenho dos algoritmos propostos foram realizadas simulações utilizando o simulador de redes ópticas WDMSim [Drummond 2010] . As topologias consideradas nas simulações, Figura 2, foram a USA com 24 nós e 43 enlaces bidirecionais, PanEuro, com 27 nós e 81 enlaces bidirecionais e a topologia NSFNet, com 16 nós e 50 enlaces bidirecionais. 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 10Gbps. A arquitetura utilizada é do tipo transparente, ou seja, todos os caminhos ópticos são estabelecidos entre a origem e o destino das chamadas, e não há conversão de comprimentos de onda.

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]. As chamadas foram distribuídas uniformemente para todos os pares de comunicação da rede, e suas demandas por banda passante foram inversamente distribuídas seguindo os pesos 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].

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 é calculada em *Erlang* e definida como *taxaMediaChegada x holdingTime x (bandaRequisitada/Capacidade)* [Ho and Lee 2007b]. As métricas utilizadas foram a taxa média de bloqueio de banda (MBBR), o número médio de caminhos ópticos estabelecidos por chamada aceita na rede (MNLP) e o Índice de Justiça de Jain (JFI) [Jain 1991] calculado sobre a MBBR de todos

Tabela 1. Fluxos de Video de Alta Capacidade

Compactado	Banda (Mbps)	Peso
2k	192	100
4k	382	50
8k	1200	16
não Compactado	Banda Passante	Peso
2k	3056	6
4k	6112	3
8k	19200	1

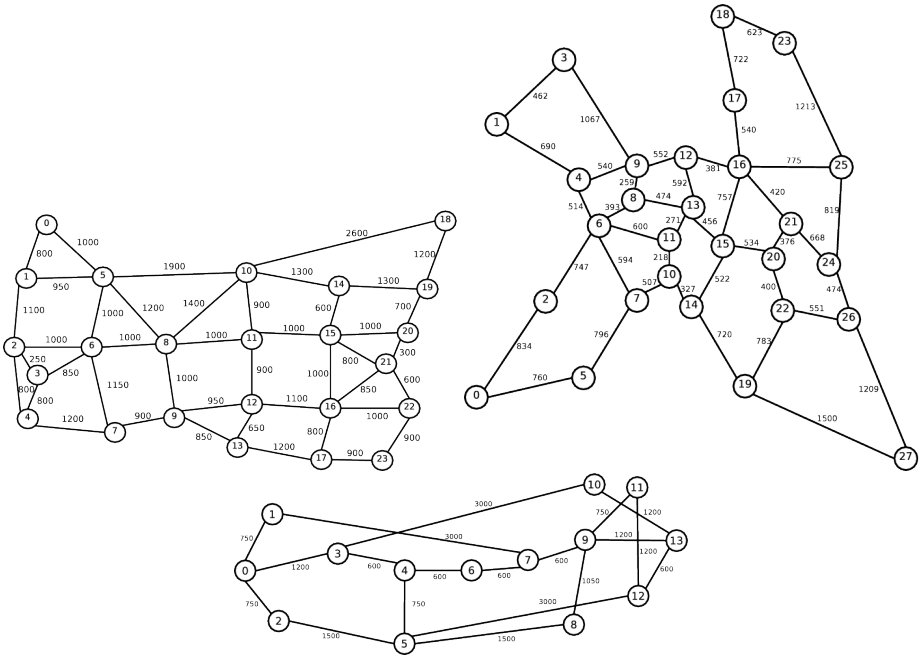


Figura 2. Topologias USA, PanEuro e NSFNet

os pares origem-destino da rede. Para cada evento de chegada simulado, os algoritmos SMART-MDP, Smart-Real-MDP e BCPP foram executados. A faixa de carga considerada neste estudo foi definida de forma que o valor da BBR calculado para os algoritmos varia-se, aproximadamente, de 0 a 10%.

Na Figura 3 são avaliadas a MBBR e o MNLP para a topologia USA. Pode-se observar no gráfico da MBBR que os resultados do algoritmo BCPP2 apresentam valores bastante menores se comparados aos dos outros algoritmos, sendo quase uma ordem de grandeza (10x) em relação ao Smart-Real-MDP e até duas ordens de grandeza (100x) menor que o Smart-MDP. Para uma carga de 0,8 *Erlang*, a BBR do BCPP2 ainda permanece sendo cerca de 37% menor do que a do Smart-Real-MDP e por volta de 57% menor do que a do Smart-MDP. O BCPP2, além de apresentar uma MBBR muito menor do que a dos outros algoritmos, também foi capaz de economizar os recursos da rede, estabelecendo menos caminhos ópticos que os outros algoritmos, sendo até 75% menor do que o Smart-MDP e 18% menor em relação ao Smart-Real-MDP. O decaimento no número de caminhos ópticos estabelecidos em função do aumento da carga na rede demonstra a dificuldade dos algoritmos em encontrar novos caminhos em cenários com carga mais alta, o que é esperado. Estes resultados evidenciam a superioridade do mecanismo de bloqueio preventivo de enlaces do algoritmo BCPP2 que acaba escolhendo de forma mais criteriosa

os caminhos a serem utilizados na rede, o que permite que haja um enorme ganho em termos de bloqueio de chamada a partir de um maior aproveitamento dos caminhos ópticos já estabelecidos, o que fica claro nos resultados que indicam uma baixa necessidade de alocação de novos caminhos ópticos, principalmente em cargas mais baixas.

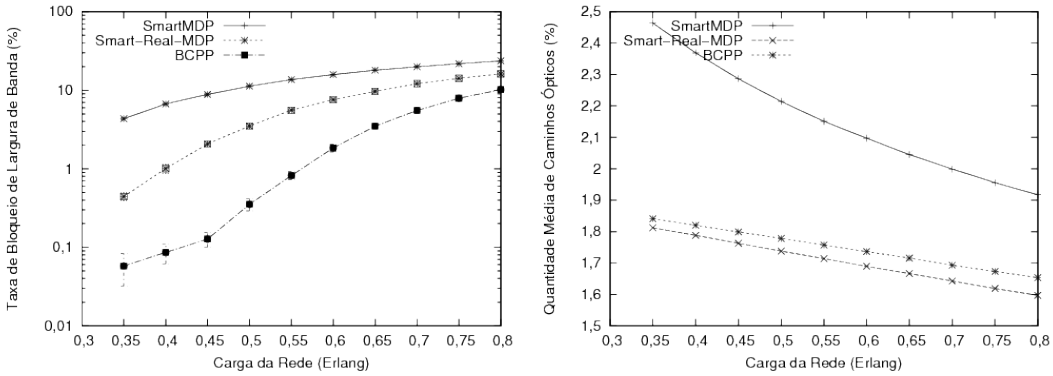


Figura 3. BBR e Número de caminhos ópticos estabelecidos para a rede USA.

A Figura 4 mostra os resultados para a topologia NSFNet, esta topologia tem a característica de ser mais restrita do que a USA e, portanto, exige uma maior capacidade dos algoritmos de alocação em encontrar caminhos alternativos para atender as chamadas. Neste cenário, o BCPP2 apresenta uma MBBR cerca de 50% menor do que a do algoritmo Smart-Real-MDP e aproximadamente 75% menor que a do Smart-MDP, chegando a atingir uma MBBR 22 vezes menor que a do Smart-MDP para 0,3 *Erlang*. Para uma carga de 0,65 *Erlang*, o BCPP2 apresenta um resultado 55% menor do que os resultados do Smart-MDP. Assim como na topologia USA, a economia de recursos realizada pelo algoritmo BCPP2 é evidente, sendo aproximadamente de 24% em relação ao Smart-MDP. Pode-se observa que o algoritmo BCPP2 foi capaz de lidar com a menor conectividade da topologia NSFNet e obteve resultados de BBR bastante inferiores aos dos outros algoritmos.

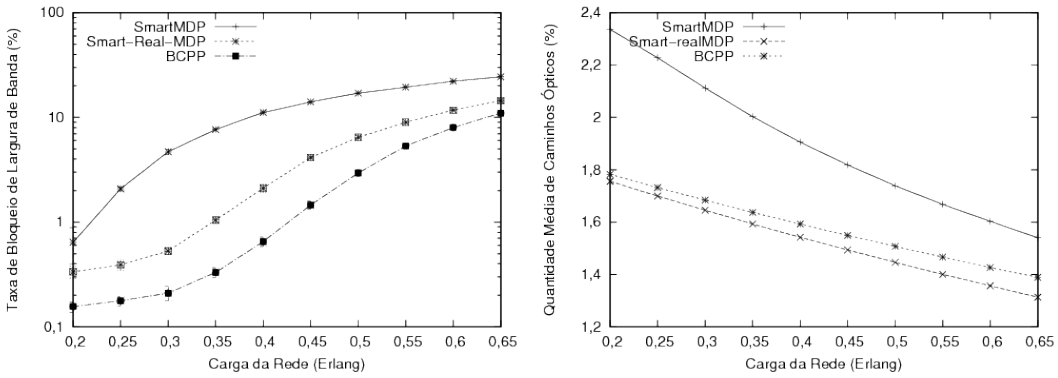


Figura 4. BBR e Número de caminhos ópticos estabelecidos para a rede NSFNet.

Na Figura 5 pode-se observar os resultados para a topologia PanEuro, esta topologia possui uma característica singular por possuir um núcleo bem conectado e 4 anéis externos contendo diversos nós de grau 2, o que dificulta bastante sua utilização no estabelecimento de rotas alternativas. Sendo este o cenário mais restrito avaliado, espera-se que os resultados dos algoritmos sejam mais próximos uns dos outros, porém, o algoritmo

BCPP2 apresentou resultados de MBRR melhores do que ambos os algoritmo, chegando a ser mais de uma ordem de grandeza menores do que os do algoritmo Smart-MDP. Mesmo para a maior carga avaliada, de 0.5 *Erlang*, onde ocorre a menor diferença entre os resultados, o BCPP2 consegue ser 50% mais eficiente do que o Smart-MDP. Na avaliação do número médio de caminhos ópticos estabelecidos, o BCPP2 apresenta resultados quase idênticos ao Smart-Real-MDP, todavia, mostra uma economia em torno de 20% com relação aos resultados do algoritmo Smart-MDP.

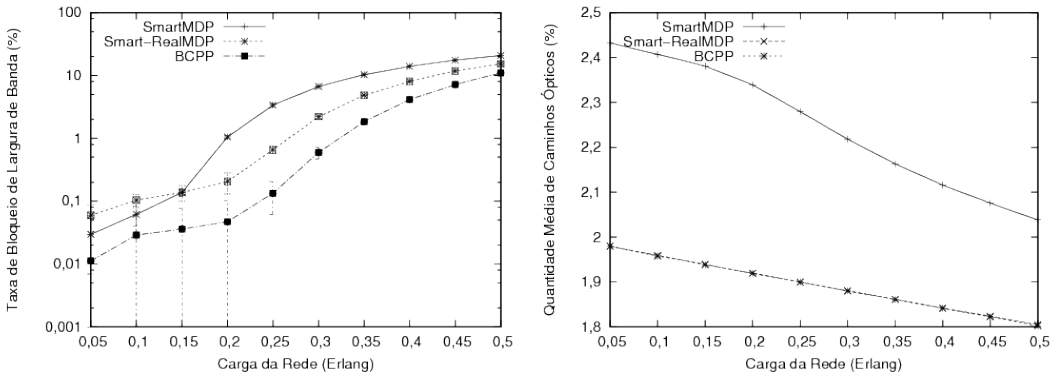


Figura 5. BBR e Número de caminhos ópticos estabelecidos para a rede PanEuro.

Na Tabela 2 estão apresentados os resultados do Índice de Justiça de Jain para as três topologias avaliadas. Comparando-se os resultados dos algoritmos BCPP2 e Smart-MDP, percebe-se um ganho do algoritmo BCPP2 que varia entre 15% até uma diferença máxima de 8 vezes. Estes resultados mostram que, além de apresentar uma taxa média de bloqueio de banda bastante reduzida e uma grande capacidade em agregar o tráfego em caminhos ópticos existentes, evitando assim a alocação de novos caminhos, o algoritmo BCPP2 também foi mais justo na alocação de recursos em relação a cada par origem-destino, o que implica em uma maior eficiência em evitar a criação de gargalos na rede.

5. Conclusão

O surgimento de aplicações com requisições extremas de banda passante tem trazido a necessidade da implementação de esquemas de roteamento adaptativo em redes ópticas WDM. Estas aplicações de alta capacidade tornam o mecanismo de proteção parcial e a técnica de roteamento multicaminho ferramentas essenciais para a provisão de soluções mais eficientes e confiáveis.

Este artigo propôem uma nova solução para o problema de agregação dinâmica de tráfego com proteção parcial que, além de utilizar uma função de custo abrangente, introduz um mecanismo de bloqueio preventivo capaz de prover, simultaneamente, justiça e economia de recursos, o que leva a uma menor taxa de bloqueio na rede.

O algoritmo proposto, *Best-Cost with Partial Protection* - BCPP2, mostrou ser mais eficiente que os algoritmos Smart-Real-MDP e o Smart-MDP, atingindo um nível de eficiência ordens de grandeza superior em relação a solução da literatura. O BCPP2 conseguiu melhorar os resultados com relação as métricas de taxa de bloqueio de banda, número de caminhos ópticos utilizados e justiça. O uso do bloqueio preventivo e a não utilização de algoritmos tradicionais de menor caminho, ajudou a evitar a formação de gargalos na rede, pois permitiram a escolha mais criteriosa de caminhos dentre todos as

Jain Fairness Index (JFI)

Topologia PanEuro

MBBR	2,0	4,0	6,0	8,0	10,0
Smart-MDP	0,01	0,12	0,23	0,30	0,37
Smart-Real-MDP	0,07	0,20	0,27	0,33	0,38
BCPP	0,08	0,21	0,31	0,35	0,43

Topologia NSFNet

MBBR	2,0	4,0	6,0	8,0	10,0
Smart-MDP	0,16	0,31	0,38	0,43	0,50
Smart-Real-MDP	0,33	0,42	0,46	0,51	0,58
BCPP	0,30	0,48	0,53	0,58	0,61

Topologia USA

MBBR	2,0	4,0	6,0	8,0	10,0
Smart-MDP	0,18	0,21	0,32	0,34	0,50
Smart-Real-MDP	0,15	0,27	0,27	0,34	0,58
BCPP	0,20	0,31	0,41	0,44	0,61

Tabela 2. Justiça na alocação de recursos para diferentes cenários de carga nas redes PanEuro, NSFNet e USA.

possibilidades existentes. A aplicação de uma função de custo abrangente foi fundamental para a seleção dos melhores caminhos, ou seja, os que consomem menos recursos. Por fim, a função que mitiga o atraso diferencial, torna o algoritmo proposto uma solução completa para ser implementada na prática.

Finalmente, o algoritmo BCPP2 demonstrou grande robustez ao apresentar resultados similares frente a diferentes cenários de avaliação que empregaram topologias com níveis de conectividade distintos.

Referências

Ahuja, S., Korkmaz, T., and Krunz, M. (2004). Minimizing the differential delay for virtually concatenated ethernet over sonet systems. In *International Conference on Computer Communications and Networks*, pages 205–210.

André Costa Drummond, P. J. d. S. 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.*

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.

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.

Das, A., Martel, C., Mukherjee, B., and Rai, S. (2011). New approach to reliable multipath provisioning. *Optical Communications and Networking, IEEE/OSA Journal of*, 3(1):95 –103.

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).

Dutta, R. and Rouskas, G. (2002). Traffic grooming in wdm networks: past and future. *Network, IEEE*, 16(6):46–56.

Ho, Q.-D. and Lee, M.-S. (2007a). A zone-based approach for scalable dynamic traffic grooming in large wdm mesh networks. *IEEE Journal of Lightwave Technology*, page 261–270.

Ho, Q.-D. and Lee, M.-S. (2007b). A zone-based approach for scalable dynamic traffic grooming in large wdm mesh networks. *IEEE Journal of Lightwave Technology*, 25(1):261–270.

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.

Jain, R. (1991). The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation and modeling.

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.

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).

Adaptação do Algoritmo BSR para Redes Ópticas SLICE

Alex F. Santos[1], Raul C. Almeida Jr[2], Karcius D. R. Assis [1], Gilvan M. Durães[1], André Soares[3] e William F. Giozza[4]

[1]Universidade Federal da Bahia (UFBA)

[2]Universidade Federal de Pernambuco (UFPE)

[3]Universidade Federal do Piauí (UFPI)

[4]Universidade de Brasília (UnB)

{alex.ferreira,karcius.assis}@ufba.br, raul.almeidajunior@ufpe.br, gilvanmd@dcc.ufba.br, andre.soares@ufpi.edu.br e giozza@unb.br

Abstract. *In Spectrum-Sliced Elastic Optical Path Network (SLICE), the problem of Routing and Spectrum Allocation (RSA) is essential for the insertion of optical paths in the network. RSA is an NP-Complete problem and therefore heuristics are used to maximize the use of resources. In order to reduce the complexity of RSA, it is common to initially define the set of paths between source-destination node pairs through which traffic demand will be routed and subsequently perform spectrum allocation. This paper presents an adaptation for SLICE optical networks of a routing algorithm recently proposed for conventional WDM networks, referred to as BSR (Best among the Shortest Routes). Simulations were performed for different topologies and the results suggest great benefits in terms of blocking probability of the new proposal in relation to traditional BSR.*

Resumo. Em uma rede óptica SLICE (*Spectrum-Sliced Elastic Optical Path Network*), o problema de roteamento e alocação de espectro (RSA -*Routing and Spectrum Allocation*) é essencial para a inserção dos caminhos ópticos na rede. RSA é um problema NP-Completo e, para resolvê-lo, empregam-se heurísticas com o intuito de maximizar a utilização dos recursos. A fim de reduzir a complexidade do RSA, é comum inicialmente definir os caminhos entre os nós (fonte, destino) por onde o tráfego (demanda de sub-portadoras) será encaminhado e, posteriormente, realizar a atribuição do espectro. Este artigo apresenta uma adaptação para redes ópticas SLICE de um algoritmo de roteamento proposto recentemente para redes WDM convencionais, denominado BSR (*Best among the Shortest Routes*). As simulações foram realizadas para diversas topologias e os resultados sugerem vantagens em termos de probabilidade de bloqueio da nova proposta em relação ao BSR tradicional.

1. Introdução

Na última década diversas pesquisas sobre planejamento e desempenho de redes ópticas foram realizadas e algumas perspectivas foram apontadas, como por exemplo, na 22ª edição do SBRC [Soares e Giozza 2004].

A tecnologia de redes ópticas com roteamento de comprimento de onda amadureceu e, atualmente, apesar de alguns limites [Essiambre *et al* 2010], é a forma mais apropriada para suportar a crescente demanda de tráfego nas redes de transporte (*backbones*) que compõem as infraestruturas de telecomunicações da Internet. Nessas redes, o tráfego é roteado inteiramente no domínio óptico mediante o uso da multiplexação por divisão de comprimento de onda (WDM – *Wavelength Division Multiplexing*), em que a largura de banda de uma fibra é loteada em diferentes raias espectrais com espaçamentos uniformes, chamadas de comprimentos de onda. Um caminho óptico é formado pela concatenação, nos nós roteadores, de alguns comprimentos de onda em diferentes fibras sem que o sinal saia do domínio fotônico. Com WDM, vários caminhos ópticos em diferentes comprimentos de onda podem ser estabelecidos de forma simultânea em uma mesma fibra óptica, possibilitando o uso da ampla largura de banda da fibra.

Recentemente, tem havido um crescente interesse na investigação de uma arquitetura de rede óptica sem a grade fixa de comprimentos de onda (denominada de *gridless*), onde o gerenciamento e os elementos da rede darão suporte para que a largura de banda dos caminhos ópticos seja flexível, ou seja, possa ocupar uma largura livre do espectro de acordo com o volume de tráfego e as requisições do usuário. Essas redes foram introduzidas em [Jinno *et al* 2009] e são conhecidas na literatura como redes de caminhos ópticos elásticos, redes ópticas elásticas ou, simplesmente, redes SLICE (*Spectrum-Sliced Elastic Optical Path Network*).

Em redes SLICE, o espectro é atualmente assumido como um conjunto de *slots* de frequência de 6,25 GHz ou 12,5 GHz. A diferença maior para as redes WDM é que um caminho óptico pode ser agora estabelecido por meio de um determinado número de slots de frequência a depender da taxa de transmissão desejada e do formato de modulação utilizado. Para suportar o conceito desta nova arquitetura de rede, *transponders* com largura de banda variável (BVs), que podem variar a largura de banda na borda da rede, *Optical Crossconects* (WXCs), que podem variar a largura de banda a partir do núcleo da rede, e tecnologias de modulação eficientes como O-OFDM (*Optical Orthogonal Frequency-Division Multiplexing*) têm sido propostos para permitir a flexibilidade na atribuição dos espectros no domínio óptico [Jinno *el al* 2009].

Para situar o leitor no problema abordado, a Figura 1(a) mostra a alocação de canais WDM, espaçados uniformemente, em que cada caminho óptico tem uma capacidade de 10 Gbps. (Evidentemente, esta capacidade ocupa espectro em Hz pela relação entre a taxa de transmissão e a frequência, a qual depende do formato de modulação e define a eficiência espectral).

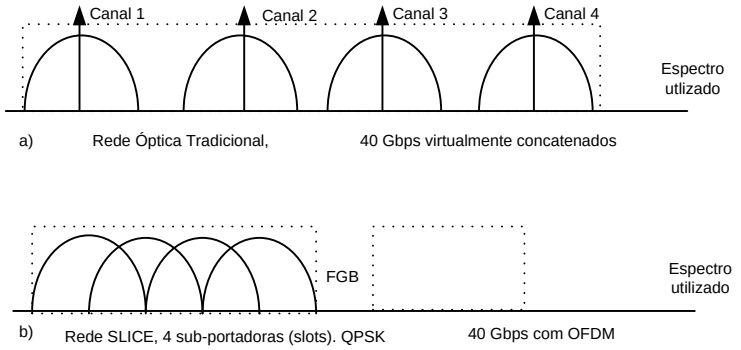


Figura 1. Espectro de redes WDM, a) Tradicional e b) SLICE com FGB (*Filter Guard Band*).

Usando a tecnologia O-OFDM, a largura de banda do canal advém da utilização de múltiplas subportadoras que enviam dados independentemente (por exemplo, QPSK com 4 subportadoras) e formam uma faixa espectral, como mostrado na Fig. 1(b). O uso de OFDM possibilita uma elevada compactação da largura de banda, já que a ortogonalidade permite que os sinais de cada subportadora se estendam para as adjacentes. A largura de banda correspondente às várias subportadoras utilizadas para o caminho óptico será alocada na rede em forma de uma quantidade de slots. A partir daqui, por simplicidade, a demanda de banda será quantificada em termos de número de slots requisitados.

Similar ao problema de roteamento e alocação de comprimentos de onda (RWA – *Routing and Wavelength Assignment*) em redes WDM, na rede SLICE existe o problema de roteamento e atribuição de espectro (RSA – *Routing and Spectrum Allocation*) [Wang, Cao, e Pan, 2011], [Almeida Jr. *et al* 2013]. Neste é alocado uma fatia do espectro ou um conjunto de *slots* para atender à demanda de tráfego. O RSA é diferente e mais desafiador do que o problema RWA, principalmente pelo fato de os caminhos ópticos (*lightpaths*) poderem utilizar diferentes granularidades espectrais. Adicionalmente, numa rede sem conversão espectral, a restrição de continuidade de comprimento de onda é transformada em restrição de continuidade de espectro e a fatia do espectro (número de *slots*) alocada para a conexão deve ser mantida ao longo dos enlaces da rota de forma contínua.

Neste trabalho, é considerada uma arquitetura de rede SLICE com BVs e OXCs de largura de banda variável, sob uma demanda de tráfego dinâmica. O problema RSA dinâmico deve ser resolvido com a rede em operação, onde as requisições de caminhos ópticos chegam aleatoriamente à rede. Caso não haja recursos suficientes para atender a uma determinada requisição de caminho óptico, ocorrerá um bloqueio. Por esta razão, o objetivo de um provedor de serviços de transporte via rede SLICE, face ao problema RSA dinâmico, é atender às requisições atuais de caminhos ópticos visando minimizar a probabilidade de bloqueio de futuras requisições de caminhos ópticos.

No RSA dinâmico, os algoritmos de roteamento, de forma similar ao RWA, podem ser separados em três classes: roteamento fixo, roteamento alternativo ou fixo alternativo, e roteamento exaustivo ou dinâmico [Murthy e Gurusamy 2002].

No roteamento fixo, cada par de nós (origem, destino) da rede óptica dispõe de apenas uma rota que é computada previamente. Assim, antes mesmo de surgir uma requisição de caminho óptico, o Plano de Controle responsável pelo roteamento já sabe qual rota deve ser utilizada. Isto significa que, após o surgimento da requisição, o desafio passa a ser a alocação de uma fatia de espectro para a requisição, levando em consideração a largura desta requisição.

No roteamento alternativo, um conjunto com mais de uma rota é definido previamente para cada par de nós (origem, destino). Isto representa mais de uma alternativa, em termos de rota, na tentativa de estabelecer um circuito óptico. O roteamento alternativo pode ainda ser classificado em roteamento fixo alternativo ou roteamento adaptativo alternativo. A diferença entre eles é a forma como é feita a seleção de uma rota do conjunto de rotas alternativas previamente definidas. No roteamento fixo alternativo [Murthy e Gurusamy 2002], [Lin, Wang e Tsai 2006], as rotas são previamente ordenadas, por exemplo, em função do número de saltos. A seleção da rota é feita seguindo a ordem previamente definida. Se a primeira rota não possui recursos disponíveis, as rotas seguintes são analisadas uma a uma até ser encontrada uma rota com recursos disponíveis. Por exemplo, no problema sob estudo, se nenhuma das alternativas de rotas pré-definidas tiver uma fatia de espectro contínua, livre, e de tamanho adequado em todos os enlaces da rota, a requisição é bloqueada. No roteamento adaptativo alternativo [Birman 1996], a seleção de uma rota do conjunto de rotas definido previamente é feita de acordo com informações do estado atual da rede. Essas informações servem de entrada para um critério desejado, como, por exemplo, alocar a rota com maior fatia espectral disponível.

Os algoritmos da classe de roteamento exaustivo têm como vantagem a capacidade de utilizar qualquer rota possível da topologia na tentativa de estabelecer o caminho óptico [Murthy e Gurusamy 2002]. Com isso, uma requisição de caminho óptico apenas será bloqueada se nenhuma rota, entre sua origem e destino, dispuser de pelo menos uma fatia do espectro contínua, livre, e da largura da requisição.

Em termos gerais, as classes de algoritmos de roteamento apresentam a seguinte ordem crescente de desempenho em termos de probabilidade de bloqueio: fixo, fixo alternativo e exaustivo [Zang, Jue e Mukherjee 2000], [Lin, Wang e Tsai 2006]. Entretanto, esse aumento do desempenho é acompanhado também pelo incremento da complexidade de suas soluções. Algoritmos de roteamento que dependem de informações globais sobre o estado da rede resultam em uma maior complexidade para os protocolos do Plano de Controle. Tal complexidade pode ser traduzida em um maior atraso no estabelecimento de um circuito óptico. Certamente, a solução prévia das rotas, com o uso de um algoritmo representante da classe de roteamento fixo, potencializa a redução do tempo requerido para o estabelecimento dinâmico do caminho óptico. O roteamento fixo proporciona uma menor complexidade para os protocolos do Plano de Controle, uma vez que a escolha da rota para cada par de nós origem e destino é feita na fase de planejamento da rede.

Este trabalho é inspirado em um algoritmo recentemente proposto na literatura [*Durães et al*, 2010], que usa roteamento fixo e que procura balancear a carga da rede, conhecido como BSR (*Best among the Shortest Routes*). Aqui é proposta uma adaptação ao BSR, para dar suporte a redes que usam a arquitetura SLICE. Logo, o chamado BSR

adaptado tenta compor um Plano de Controle adequado para a rede SLICE, sendo que a ideia básica é dar liberdade, na fase de planejamento, a escolha de rotas diferentes para requisições de largura diferente, permitindo assim uma melhor justaposição de tráfegos com diferentes números de *slots* e, portanto, um melhor balanceamento da carga.

Assim como o BSR tradicional, o BSR adaptado também é da classe de roteamento fixo, o que representa uma menor complexidade para o algoritmo proposto.

O restante deste artigo está organizado da seguinte forma. Na próxima seção são apresentados os trabalhos relacionados e as nossas contribuições. Na seção 3 são apresentados o BSR tradicional para redes ópticas WDM e o BSR adaptado para redes SLICE. Na seção 4 é realizado um estudo de avaliação de desempenho comparando o algoritmo BSR adaptado, o algoritmo BSR tradicional e o algoritmo clássico de Dijkstra (DJK). As considerações finais são feitas na seção 5.

2. Trabalhos Relacionados

O problema RSA tem sido bastante estudado nos últimos anos. Várias estratégias foram propostas visando melhorar o desempenho das redes SLICE em termos de probabilidade de bloqueio sob tráfego dinâmico ou outras métricas para tráfego estático [Christodoulopoulos, Tomkos e Varvarigos 2010], [Wang, Cao, e Pan 2011], [Santos *et al* 2012], [Christodoulopoulos, Tomkos e Varvarigos, 2011], [Durán *et al* 2012].

Em [Christodoulopoulos, Tomkos e Varvarigos 2010], o problema RSA é sub-dividido em Roteamento e Alocação de Espectro, onde são apresentadas heurísticas para resolvê-lo de forma separada. Uma formulação exata, através de Programação Linear Inteira, também é apresentada para resolver o problema completo. Entretanto, a mesma só é viável em redes de pequena dimensão. Em [Durán *et al* 2012] foram adaptados algoritmos tradicionais utilizados em redes WDM para aplicação em redes SLICE sob tráfego dinâmico, e os resultados mostraram que a ordem de desempenho dos algoritmos pode diferir nas duas arquiteturas.

A grande maioria dos trabalhos na literatura que estuda o problema RSA baseia-se na classe de roteamento fixo em função de sua menor complexidade. Tais trabalhos consideram o uso de algoritmos de menor caminho (menor número de saltos, isto é, o custo de cada enlace é igual a 1) para definir uma rota fixa para cada par de nós (origem, destino). Dentre os algoritmos de menor caminho, o algoritmo de *Dijkstra* [Dijkstra, 1959] é um dos mais citados. Por simplicidade, o termo algoritmo de menor caminho de *Dijkstra* será denotado por DJK.

A opção pelo uso de algoritmos tradicionais de menor caminho (*e.g.*, *Dijkstra*, *Bellman-Ford*, etc) tende a limitar a capacidade de uma rede óptica, seja esta tradicional ou com a arquitetura SLICE. Nesses algoritmos, a escolha da rota de menor caminho é feita sem avaliar o impacto que essa rota pode ocasionar em outras rotas que compartilhem os mesmos enlaces. Como esses algoritmos tradicionais não têm por objetivo balancear a carga entre os enlaces da rede óptica, é possível o surgimento de enlaces "gargalos" comprometendo o desempenho no atendimento à demanda de caminhos ópticos.

Como visto anteriormente, com o intuito de minimizar este problema, [Durães *et al.* 2010] propuseram o algoritmo BSR para redes ópticas WDM. O BSR é da classe de

roteamento fixo e portanto requer uma menor complexidade computacional. Simulações foram realizadas para redes WDM e os resultados comprovam sua eficiência quando comparado com DJK e uma proposta chamada RRT (*Restricted Routing Technique*), que também se propõe a descongestionar enlaces críticos da rede [Rajalakshmi e Jhunjhunwala 2008]. O BSR é caracterizado por escolher as rotas de menor caminho e, ao mesmo tempo, buscar um melhor balanceamento de carga entre os enlaces da rede. Cada requisição na rede WDM tem uma mesma largura, ou seja, solicita uma fatia do espectro de tamanho fixo. Entretanto, em uma rede SLICE, cada requisição que surgir para um par fonte-destino pode ter número diferente de *slots*, conforme visto na Figura 1. Consequentemente, a escolha da rota adequada para cada requisição sugere uma nova estratégia para o cálculo das rotas, aqui denominada BSR adaptado, e que também pertence à classe de roteamento fixo.

Este artigo apresenta as seguintes contribuições:

a. Um algoritmo **BSR adaptado,** o qual dá liberdade ao planejamento de rotas de acordo com a largura de banda da requisição, visando ao balanceamento de carga entre os enlaces da rede e deixando capacidade aberta para futuras requisições que venham a surgir;

b. Simulações que comparam o desempenho dos algoritmos de roteamento BSR adaptado, BSR e DJK, em termos de probabilidade de bloqueio, considerando as topologias das redes NSFNET, EON e Rede Hipotética Brasileira sob o plano de controle de uma arquitetura SLICE.

3. Algoritmos BSRs

Esta seção apresenta os algoritmos BSR tradicional e BSR adaptado, da classe de roteamento fixo, os quais são propostos para redes ópticas transparentes tradicionais e redes SLICE, respectivamente. Os BSRs fazem uso de resultados de simulações iterativas na tentativa de encontrar a melhor solução para o planejamento da rede. O desafio dos algoritmos é balancear a carga entre os enlaces da rede de modo a diminuir a probabilidade de bloqueio das requisições de caminhos ópticos. É importante destacar que, por se tratarem de algoritmos de roteamento fixo, a execução dos BSRs não é feita com a rede em operação. Os BSRs são executados em uma fase de planejamento. Por isso, o tempo necessário para a realização de suas simulações iterativas não é algo impeditivo. Para uma explicação detalhada do conceito BSR, o leitor pode consultar [*Durães et al*, 2010].

A seguir são listadas algumas notações utilizadas na apresentação dos algoritmos BSRs:

- L é o conjunto de todos os enlaces da rede;
- l é um enlace que pertence a L;
- $c(l)$ é o custo do enlace l;
- $c(l)_i$ é o custo do enlace l na i-ésima iteração;
- $u(l)$ é a utilização do enlace l
- $u(l)_i$ é a utilização do enlace l obtida via simulação na i-ésima iteração;
- T é o número máximo de iterações dos BSRs
- R é o número máximo de slots que pode ser requisitado
- W é o número de slots atual requisitado para cada par fonte-destino

Baseado nas notações acima, uma explicação dos dois algoritmos, BSR tradicional, ou simplesmente BSR, e BSR adaptado é demonstrada nas subseções a seguir.

3.1 BSR

Cada iteração *i* do BSR simula uma solução de roteamento, S_i, do universo de M soluções possíveis. Os resultados da simulação realizada na *i*-ésima iteração são os valores de utilização de cada enlace da rede ($u(l)_i$) e o desempenho da rede em termos de probabilidade de bloqueio obtidos com a solução de roteamento S_i.

A ideia básica deste algoritmo é ajustar, na iteração *i+1*, o custo de cada enlace com uma pequena ponderação (1-α) proporcional ao valor da utilização do enlace obtido na simulação da iteração *i*. O ajuste no custo de cada enlace é dado pela equação,

$$c(l)_{i+1} = \alpha \, . \, c(l)_i + (1 - \alpha) \, . \, u(l)_i \qquad (1)$$

com $1 \leq i \leq T$ e $\alpha = 0{,}9999$. Observe que o valor de α deve ser próximo de 1 para que os custos dos enlaces sejam minimamente alterados, em função da utilização dos mesmos, e para que as novas rotas encontradas continuem sendo rotas com o menor número de saltos. O valor de $\alpha = 0{,}9999$ foi determinado empiricamente, após análises dos resultados obtidos com diferentes valores.

Após obter os custos $c(l)_{i+1}$, utiliza-se um algoritmo de menor caminho simples (DJK) para encontrar a solução de roteamento S_{i+1} que será utilizada na simulação da iteração *i+1*. Esse pequeno ajuste serve como um critério de desempate para que o algoritmo DJK encontre uma solução S_{i+1} com rotas que representem um maior equilíbrio na utilização dos enlaces da rede e, consequentemente, diminua a probabilidade de bloqueio de uma requisição de caminho óptico.

Inicialmente, na iteração 1, aplica-se a solução de roteamento encontrada com o DJK assumindo a topologia de rede em questão com o custo dos seus enlaces igual a 1.

A seguir, é apresentado um resumo das etapas do algoritmo BSR.

Algoritmo BSR

---

1) Atribuir o custo 1 para cada enlace da topologia de rede em questão. Obter o conjunto de rotas fixas S_1 obtida com o algoritmo de roteamento de menor caminho (DJK). Executar a simulação da iteração *i=1* e para isto utilize a solução de roteamento S_1.
2) Atualizar os custos dos enlaces para a iteração *i+1* de acordo com a Equação 1.
3) Encontrar a nova solução de roteamento S_{i+1}. Para isto utilize o algoritmo DJK e considere os custos dos enlaces da topologia da rede em questão obtidos no passo anterior. Simule a rede com a solução de roteamento S_{i+1}.
4) Se $i < T$ volte para o passo 2. Caso contrário vá para o passo 5.
5) Verificar dentre as *T* iterações qual a solução de roteamento que apresentou menor probabilidade de bloqueio. A solução de rotas fixas utilizadas na simulação desta iteração representa as rotas escolhidas pelo algoritmo BSR.

O BSR consegue obter resultados melhores que os algoritmos de DJK e RRT em redes ópticas WDM, ver [Durães *et al.* 2010]. Logo, a sua adaptação para redes SLICE

sugere uma melhora na probabilidade de bloqueio também para esta nova arquitetura, o que foi confirmado pelas simulações. Porém, melhoras adicionais podem ser alcançadas ao realizar um roteamento fixo que gere uma distribuição de carga mais efetiva. A próxima subseção abordará o algoritmo BSR adaptado, proposto neste trabalho.

3.2 BSR Adaptado

O BSR tradicional para redes WDM analisa cada requisição, a qual possui uma largura de banda fixa. O ponto chave é adaptar o BSR para realizar o roteamento (definir o conjunto de caminhos) para uma arquitetura SLICE, em que as requisições têm tamanho variável.

Logo, para cada conexão, é definida a quantidade de slots que deverá ser alocada, os quais devem ser contínuos ao longo da rota. Se esta condição for atendida ao longo da rota, a atribuição de espectro é realizada, caso contrário, a conexão é bloqueada.

A seguir, é apresentado um resumo das etapas do algoritmo BSR adaptado.

Algoritmo BSR adaptado

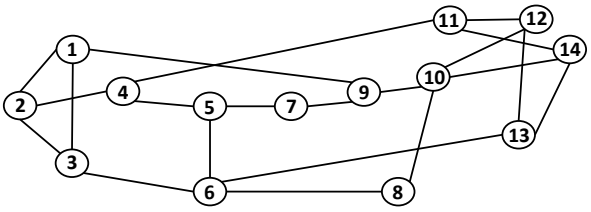
Para a explicação do BSR adaptado, suponha que cada fibra na rede possa transportar no máximo B slots, e que os pedidos de conexão são para $k = 1,2,..., M$ slots, sendo M o número máximo de slots requisitados. Seja $R^k_{(s,d)}$ a rota fixa para todas as requisições entre os nós s e d com pedido de k slots. Visto que o BSR adaptado busca uma melhor distribuição de carga na rede pela definição de um roteamento fixo para cada demanda de tráfego, $R^m_{(s,d)}$ pode ser diferente de $R^n_{(s,d)}$ se $m \neq n$. As etapas do BSR adaptado estão elencadas a seguir:

1) Assuma $i = 1$ e atribua o custo 1 para cada enlace da topologia de rede em questão.
2) Para $k = 1$ até M faça:
   2.1) Obter o conjunto de rotas mais curtas com o algoritmo de roteamento de menor caminho (DJK), para as requisições de k slots entre todos os pares de nós da rede. Defina este conjunto como $\{R^k\}$ e guarde-o na memória.
   2.2) Atualize os custos dos enlaces de acordo com a Equação 1, sendo agora $u(l)$ o número de slots ocupados no enlace l.
3) Simule a rede com a solução de roteamento de todas as menores rotas encontradas no passo 2.1 (S_i=R^1∪R^2∪...∪R^m). Faça $i = i + 1$.
4) Se $i < T$, volte para o passo 2. Caso contrário vá para o passo 5.
5) Verificar dentre as T iterações qual o conjunto de solução de roteamento S_i que apresentou menor probabilidade de bloqueio. A solução de rotas fixas utilizadas na simulação desta iteração representa as rotas escolhidas pelo algoritmo BSR adaptado para as diferentes larguras de requisições.

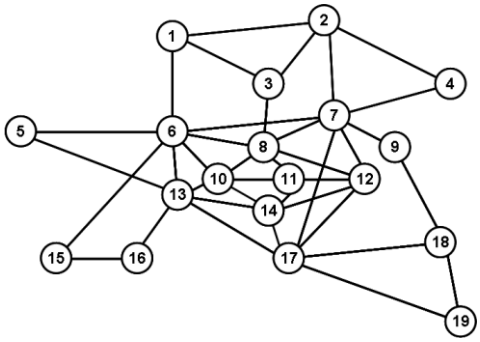
Logo, após serem definidas as rotas por onde os caminhos ópticos poderão passar, o próximo passo é atribuir slots contínuos para estes caminhos ópticos que surgem dinamicamente. Esta atribuição pode ser feita aleatoriamente ou através de algum algoritmo de alocação. Neste artigo, escolheu-se um algoritmo de alocação de prioridade fixa, o *First-Fit*, por ser simples e bastante utilizado na literatura, tanto para redes ópticas WDM quanto para redes SLICE. O objetivo é comparar os algoritmos: BSR adaptado, BSR e DJK em uma arquitetura SLICE em termos da probabilidade de bloqueio. Observe que o foco do trabalho é comparar algoritmos de roteamento e não algoritmos de alocação de espectro, logo a escolha de algoritmos de alocação mais sofisticados se torna marginal.

4. Resultados Numéricos

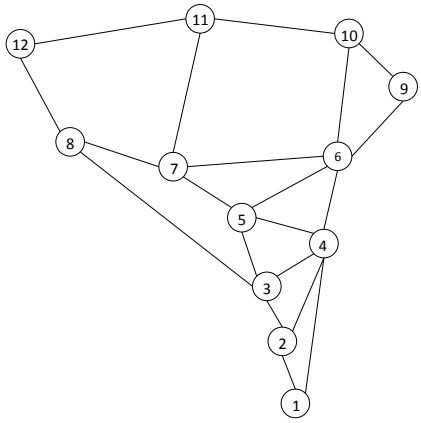
Para verificar a eficiência do BSR adaptado para redes ópticas SLICE, foram realizadas simulações em cenários e topologias distintas como NSFNet [Wang, Cao, e Pan, 2011], EON [Mahony, 1994], Brasileira [Assis *el al.*, 2009] e Abilene [Abilene Network, 2005]. Para cada simulação são realizadas cinco replicações com diferentes sementes de geração de variável aleatória. São geradas um milhão de requisições para cada replicação. Os resultados gráficos apresentam os intervalos de confiança calculados com um nível de confiança de 95%. Cada simulação é realizada em média de 55 segundos, dependendo da topologia da rede, utilizando o computador Pentium Dual Core 1.86GHz, 2GB de RAM com Windows 7.



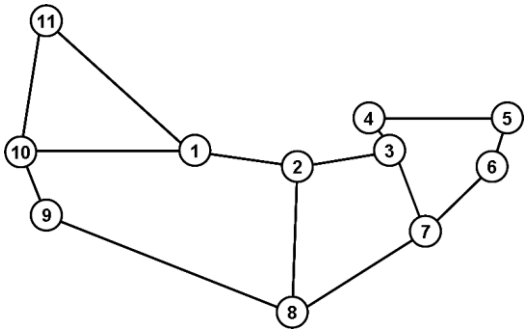
a. Rede NSFNet



b. Rede EON



c. Rede Brasileira



d. Rede Abilene

Figura 2. Topologias de rede estudadas.

Na rede NSFnet (Figura 2a), que contém 14 nós e 42 enlaces, as simulações foram realizadas com requisições de 1, 2, 3,..., 8 slots e um total de 128 slots disponíveis por enlace. A requisição é bloqueada se não existe slots contíguos disponíveis no caminho entre os pares (fonte, destino) para acomodar a mesma. Define-se a probabilidade de bloqueio de caminhos como o número total de requisições bloqueadas divididas pelo número total de requisições geradas. De forma similar define-se a probabilidade de bloqueio de slots.

Então, para a rede NSFNET, a Figura 3a ilustra a probabilidade de bloqueio de caminhos e a Figura 3b a probabilidade de bloqueio de slots. Como se pode observar, o BSR adaptado consegue uma menor probabilidade de bloqueio que o DJK e o BSR. Em média obtêm-se uma melhora na probabilidade de bloqueio de caminhos de 36,1% e 17,5% quando se compara o algoritmo BSR adaptado com o DJK e o BSR, respectivamente. Para a probabilidade de bloqueio de slots, obtêm-se desempenho de forma alinhada com o bloqueio de caminhos, só que os percentuais são de 35,6% e 17,2%.

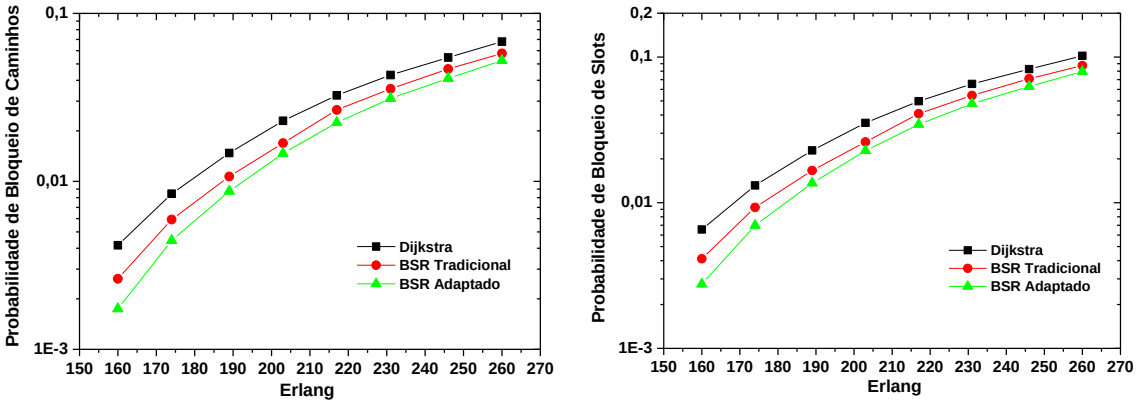


Figura 3. Probabilidade de Bloqueio de Caminhos (a) e Probabilidade de Bloqueio de Slots (b) em função da carga da rede para os algoritmos Dijkstra, BSR tradicional e BSR Adaptado na rede NSFNet. As requisições são de 1, 2, 3, ..., 8 slots e há um total de 128 slots disponíveis por enlace.

Para a rede EON (Figura 2b), que contém 19 nós e 76 enlaces, realizaram-se as simulações num cenário diferente, onde as requisições agora são de 1, 2, 4 ou 8 slots e há um total de 128 slots disponíveis por enlace. As Figuras 4a e 4b apresentam a probabilidade de bloqueio de caminhos e a probabilidade de bloqueio de slots, respectivamente. Para esta topologia, o BSR adaptado consegue novamente uma menor probabilidade de bloqueio do que os algoritmos *DJK* e BSR. Em média, a melhora na probabilidade de bloqueio de caminhos foi de 73,9% e 26,9% quando se compara o BSR adaptado com o DJK e o BSR, respectivamente. Para a probabilidade de bloqueio de slots, os percentuais de desempenho foram semelhantes. Devido sua topologia ser muito mais conectada e, portanto, permitir um conjunto maior de soluções (rotas), o BSR adaptado operando na rede EON obteve melhores percentuais do que na NSFNet.

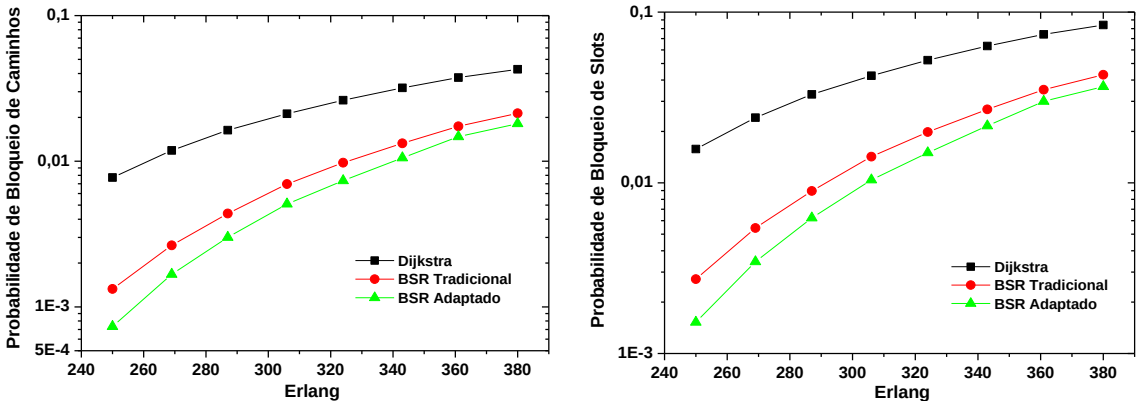


Figura 4. Probabilidade de Bloqueio de Caminhos (a) e Probabilidade de Bloqueio de Slots (b) em função da carga da rede para os algoritmos Dijkstra, BSR tradicional e BSR Adaptado na rede EON. As requisições são de 1, 2, 4 ou 8 slots e há um total de 128 slots disponíveis por enlace.

Na rede Brasileira (Figura 2c), que contém 12 nós e 40 enlaces, realizaram-se simulações com requisições de 1, 2, 3, ..., 16 slots e um total de 256 slots disponíveis por enlace. Novamente, observa-se na Figura 5 bons desempenhos da probabilidade de bloqueio de caminhos e de probabilidade de bloqueio de slots, atingindo ganhos do BSR adaptado, nas duas métricas, de 37% e 18% em relação ao DJK e BSR, respectivamente.

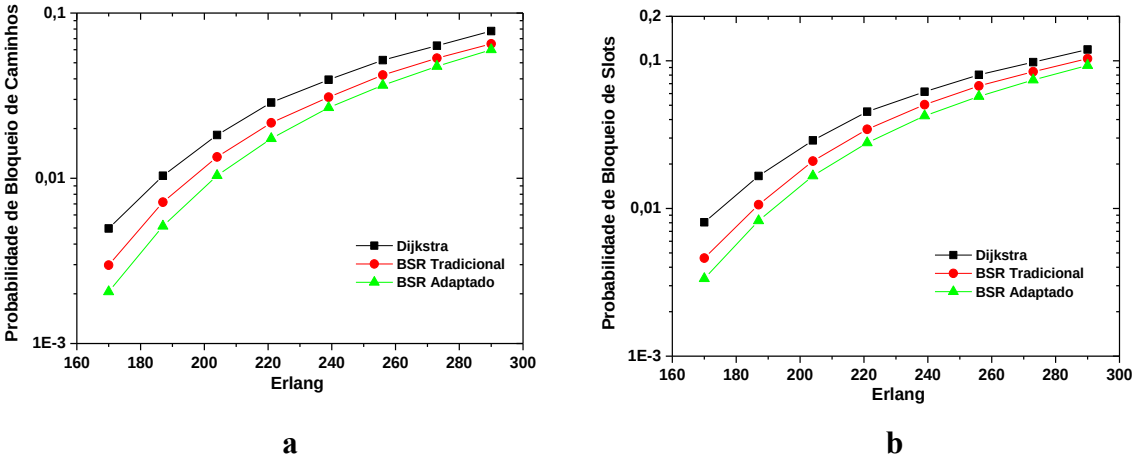
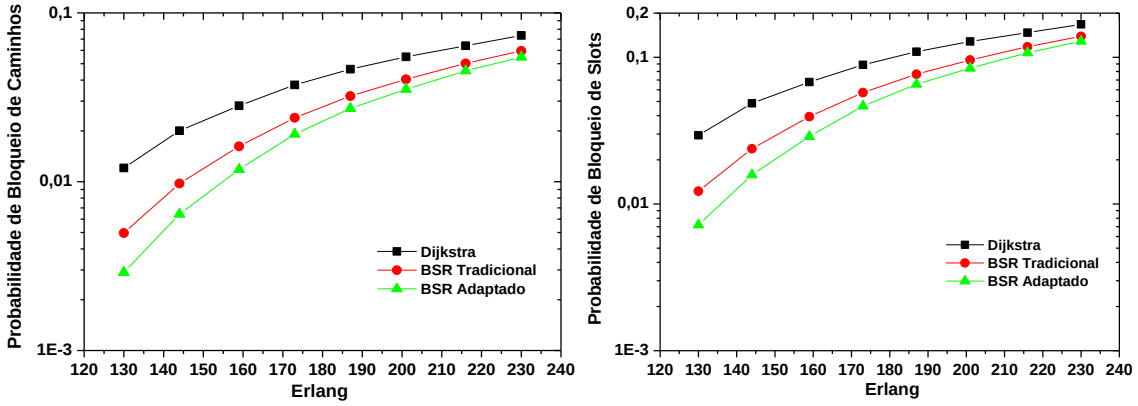


Figura 5. Probabilidade de Bloqueio de Caminhos (a) e Probabilidade de Bloqueio de Slots (b) em função da carga da rede para os algoritmos Dijkstra, BSR tradicional e BSR Adaptado na rede Brasileira. As requisições são de 1, 2, 3,..., 16 slots e há um total de 256 slots disponíveis por enlace.

Por fim, foram realizadas simulações para a rede Abilene (Figura 2d), que contém 11 nós e 28 enlaces, com requisições de 1, 2, 4, 8 ou 16 slots e um total de 256 slots disponíveis por enlace. Em média, a melhora na probabilidade de bloqueio de caminhos, quando se usa o BSR adaptado, foi de 47,9% e 21,2% em relação ao DJK e BSR, respectivamente (Figura 6a). De forma alinhada ao bloqueio de caminhos, o desempenho para a probabilidade de bloqueio de slots obteve percentuais de melhora de 46,6% e 20,5% (Figura 6b).



a **b**

Figura 6. Probabilidade de Bloqueio de Caminhos (a) e Probabilidade de Bloqueio de Slots (b) em função da carga da rede para os algoritmos Dijkstra, BSR tradicional e BSR Adaptado na rede Abilene. As requisições são de 1, 2, 4, 8 ou 16 slots e há um total de 256 slots disponíveis por enlace.

5. Conclusão

Neste artigo, foi apresentado um algoritmo de roteamento para dar suporte ao planejamento de redes ópticas. O BSR adaptado teve sua eficiência demonstrada através de comparações com o BSR tradicional e o algoritmo de *Dijkstra* em diversos cenários. Percebe-se que a estratégia adaptada serve para balancear a carga (em termos de número de slots) melhorando a eficiência e roteando a demanda de tráfego em redes ópticas elásticas de forma apropriada.

Referências

Abilene Network (2005), “Internet2 network,” Internet2, Ann Arbor, MI [Online]. Available: http://www.internet2.edu/pubs/200502-IS-AN.pdf (accessed 26.11.2012).

Almeida Jr, R. C.;Santos, Alex Ferreira dos; K.D.R. Assis; Waldman, H.; Martins Filho, J. F. (2013) “Slot assignment strategy to reduce loss of capacity of contiguous-slot path requests in flexible grid optical networks”. Electronics Letters (Online), v. 49, p. 359-361, 2013.

Assis, K.D.R.; Maranhao, J.; Ferreira, A. e Giozza, William. (2009) “Heuristic to Maximize the Open Capacity of OBS Networks with Initial Static Traffic”. Telecomunicações (Santa Rita do Sapucaí), v. 12, p. 18-23.

Birman, A., (1996) “Computing Approximate Blocking Probabilities for a Class of All-optical Networks,” IEEE Journal on Selected Areas in Communications, vol. 14, ed. 5, pp. 852-857, Junho.

Christodoulopoulos, K., Tomkos, I. e Varvarigos, E. A., (2010) “Routing and Spectrum Allocation in OFDM-based Optical Networks with Elastic Bandwidth Allocation”, IEEE Globecom 2010

Christodoulopoulos, K., Tomkos, I., e Varvarigos, E. A., (2011) "Elastic bandwidth allocation in flexible OFDM-based optical networks", J. Lightw. Technol., vol. 29, no. 9, pp. 1354–1366, May.

Dijkstra, E. W. (1959) "A Note on Two Problems in Connection with Graphs". Numerical Mathematics, 1: 269–271.

Durães, Gilvan M., Soares, André, Amazonas, José R., e Giozza, William. (2010) "The choice of the best among the shortest routes in transparent optical networks". Computer Networks, 54(14):2400 – 2409.

Durán, R.J., Rodríguez, I., Fernández, N., Miguel, I. de, Merayo, N., Fernández, P., Aguado, J.C., Jiménez, T., Lorenzo e R.M., Abril, E.J., (2012) "Performance Comparison of Methods to Solve the Routing and Spectrum Allocation Problem", 14th International Conference on Transparent Optical Networks (ICTON).

Essiambre, R. J. et al., (2010) "Capacity Limits of Optical Fiber Networks," J. Lightwave Technol. 28,662.

Jinno, M., Takara, H., Kozicki, B., Tsukishima, Y., Sone, Y., e Matsuoka, S., (2009) "Spectrum-Efficient and Scalable Elastic Optical Path Network: Architecture, Benefits, and Enabling Technologies," IEEE Comm. Mag., vol.47, pp. 66-73.

Lin, H. C., Wang, S. W. e Tsai, C., (2006) "Traffic Intensity Based Fixed-Alternate Routing in All-Optical WDM Networks", in Proceedings of the IEEE ICC'2006, Istanbul, Turkey, Junho 11 – 15.

Mahony, M. J., (1994) "A european optical network: design considerations", in: IEEE Colloquium on Transparent Optical Networks, pp. 1–16.

Murthy, C. S. R. e Gurusamy, M., (2002) "WDM Optical Networks - Concepts, Design and Algorithms". Prentice Hall PTR.

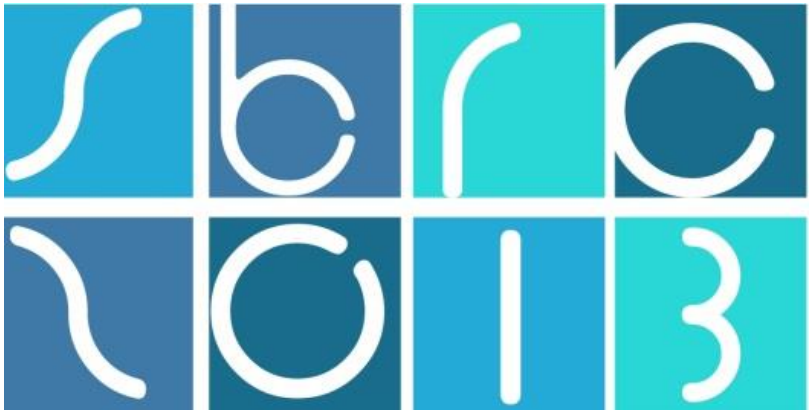
Rajalakshmi, P. e Jhunjhunwala, A., (2008) "Load Balanced Routing to Enhance the Performance of Optical Backbone Networks", in 5th IFIP International Conference on– Wireless and Optical Communications Networks(WOCN 2008), Surabaya, Indonésia.

Santos, Alex Ferreira dos; Santos, C. C. ; Durães, Gilvan Martins; Assis, K.D.R.; Almeida Jr, R. C. (2012) "Roteamento e Alocação de Espectro em Redes Ópticas: O Conceito SLICE". In: XXX Simpósio Brasileiro de Telecomunicações (SBrT'12), 2012, Brasília - DF. SBrT 2012.

Soares, André Castelo Branco; Giozza, W. F. (2004) "Avaliação de Desempenho de Algoritmos para Alocação Dinâmica de Comprimento de Onda em redes Ópticas Transparentes". In: 22º SBRC, 2004, Gramado. Anais de 22º Simpósio Brasileiro de Redes de Computadores. Gramado, RS: UFRGS, LARC, SBC, 2004. v. 1. p. 661-672.

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. of IEEE INFOCOM.

Zang, H., Jue, J. P., e Mukherjee, B., (2000) “A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks,” Opt. Networks Mag., vol. 1, no. 1.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 13

Redes de Sensores sem Fio: Roteamento e Agregação de Dados

Agrupamento Dinâmico de Sensores Baseado na Similaridade de Leitura de Dados

Fernando Henrique Gielow[1], **Aldri L. dos Santos**[1]

[1]NR2 – Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR – Brasil

{fhgielow,aldri}@inf.ufpr.br

Abstract. *Wireless Sensor Networks (WSNs) are an important interface between physical and computational environments, where the logical clustering of sensor nodes is a commonly used technique to organize traffic. Although current clustering protocols treat various kinds of dinamicity on the network, such as mobility or leader rotations, few solutions consider the readings similarity, which would provide benefits in terms of better use of compression techniques and easier reactive detection of anomalous events. This paper proposes a dynamic clustering protocol that handles spatial similarity between nodes readings, called DDFC. Its operation is based on the biological principles of fireflies to ensure distributed synchronization of the cluster's similar readings aggregations, differentiating thus from the classic use of fireflies. Simulations show that DDFC is capable of maintaining the cluster's readings aggregation synchronized, thus clustering nodes dynamically according to their similar readings.*

Resumo. *As Redes de Sensores Sem Fio (RSSFs) são uma interface importante entre o ambiente físico e o computacional. Nelas, o agrupamento lógico dos nós sensores é usado para organizar o tráfego da rede. Embora os protocolos de agrupamento atuais tratem várias dinamicidades na rede, como a de mobilidade ou de líderes, são poucas as soluções que consideram a similaridade dinâmica das leituras dos nós, que traria vantagens como o uso mais eficiente de técnicas de compressão e melhor detecção reativa de eventos anômalos. Este trabalho propõe um protocolo de agrupamento dinâmico que trata a relação de similaridade espacial entre as leituras dos nós da RSSF, chamado DDFC. O seu funcionamento é inspirado nos princípios biológicos de vagalumes para garantir a sincronização distribuída da agregação de leituras similares nos agrupamentos, se diferenciando do uso clássico de vagalumes. Simulações demonstram que o protocolo é capaz de manter a agregação de leituras dos agrupamentos lógicos sincronizadas, agrupando dinamicamente os nós de leituras similares.*

1. Introdução

Atualmente, é comum no meio urbano o uso de sensores e radares para a detecção de velocidade de veículos, ou mesmo para saber se eles estão indevidamente parados sobre faixas de pedestres. Contudo, tais aplicações ainda são primitivas por usufruírem apenas da interpretação *singular e individual* dos dados, sem o estabelecimento de relações entre eles. As relações mais usuais entre leituras de dados do meio são as relações espaciais e temporais [Yoon and Shahabi 2007]. Para diversas grandezas naturais, como temperatura, umidade e luz, tais medições tem tendência de serem parecidas quando tomadas em

regiões próximas, devido à sua **relação espacial**. Ademais, medições sucessivas em uma localidade individual tem tendência de variar gradualmente, possuindo **relação temporal**.

Ao explorar e analisar os dados de maneira *coletiva*, considerando suas possibilidades de relações, aplicações mais robustas podem ser vislumbradas e desenvolvidas. Em um cenário urbano, por exemplo, atualmente os dados vistos coletivamente possibilitam verificar o tráfego nas ruas para se determinar rotas ótimas, ou mesmo analisar padrões espaciais de temperatura a fim de localizar ilhas de calor [Murty et al. 2008]. Com as leituras de luz durante a noite, seria possível avaliar o perigo que determinadas ruas podem oferecer. Leituras de som permitiriam determinar o nível de poluição sonora em dadas regiões ou mesmo determinar como é a propagação do som no meio.

Embora as Redes de Sensores Sem Fio (RSSFs) sejam uma solução existente há diversos anos, seu uso ainda não alcançou o seu potencial máximo quanto à coleta de dados [Partridge 2011]. As RSSFs servem como uma interface de comunicação entre o meio físico e o meio computacional, formado virtualmente por diversos conjuntos de dados. Assim, elas são interface essencial para os *Cyber-Physical Systems* [Rajkumar et al. 2010] e para o advento da Internet das Coisas [Ma 2011, López et al. 2012].

Uma técnica muito utilizada para a organização lógica dos nós nas RSSFs é o agrupamento destes nós [Banerjee et al. 2011]. Essas formações, denominadas *clusters*, correspondem a grupos lógicos hierárquicos entre os nós. Os agrupamentos são muito utilizados por possibilitar agregação de dados e organizar o tráfego de mensagens na rede [Dechene et al. 2007]. Ademais, utilizando técnicas de agrupamento, a organização lógica que mantém os nós com leituras similares traria vantagens como a possibilidade de agregar os dados de maneira mais eficiente, devido à maior similaridade, além de possibilitar a detecção mais robusta de eventos anômalos [Reis et al. 2007].

Entretanto, pouca pesquisa acerca de protocolos de agrupamento que tratem simultaneamente a correlação e a variação dos dados foi desenvolvida em abordagens reativas até o momento. Os protocolos de agrupamento tem sido propostos com diversos objetivos: Alguns visam **se adequar à dinamicidade da mobilidade** [Islam et al. 2011, Brust et al. 2008], outros tentam até mesmo **recriar formações de agrupamentos por completo** [Guo and Li 2007, Villas et al. 2011]. Contudo, poucos são os que **consideram a similaridade espacial dos dados** [Wu et al. 2008, Yoon and Shahabi 2007] e, menos ainda, os que **suportam a natureza dinâmica dos dados** em uma abordagem também dinâmica de agrupamento [Pham et al. 2010]. Assim, se faz necessária uma maneira distribuída e robusta de manter agrupamentos lógicos de nós sensores com leituras similares.

Este trabalho propõe um protocolo de agrupamento lógico de nós sensores que considera a similaridade das leituras realizadas pelos nós, chamado DDFC (***D**ynamic **D**ata-aware **F**irefly-based **C**lustering*). O protocolo agrupa os nós de leituras similares, tendo como base de funcionamento os princípios biológicos dos vagalumes. O DDFC sincroniza agregações de leituras similares nos agrupamentos, suportando a sua manutenção dinâmica e seu roteamento interno. Simulações mostram a eficiência do DDFC para manter os nós de leituras similares agrupados, assim como eleger líderes adequados.

O artigo está organizado desta forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha o funcionamento do DDFC. A Seção 4 mostra a sua avaliação do desempenho. Finalmente, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

As RSSFs são dinâmicas de diversas maneiras, em questões de topologia, rotas e posicionamento dos nós. Desta forma, os mecanismos de agrupamento devem se adequar, sendo adaptativos e reconfiguráveis. Existe a **dinamicidade pela mobilidade**, que visa manter agrupamentos enquanto nós transitam arbitrariamente pela rede, havendo a necessidade de mecanismos específicos para estes cenários. O protocolo SPRP$_G$ [Islam et al. 2011] estabelece uma árvore geradora, criando nós líderes e *gateway* a fim de conectar os seus agrupamentos dentro dela. Por sua vez, o protocolo KHOPCA [Brust et al. 2008], inspirado no jogo da vida, opera proativamente através de um conjunto simples de regras que define agrupamentos de k-saltos até os líderes estabelecidos nesse processo.

Levando em consideração a complexidade e o custo de manter uma estrutura hierárquica de maneira dinâmica e proativa, algumas abordagens optam atender aos requisitos de **dinamicidade por recriação** completa dos agrupamentos, seja ela periódica ou reativa. O mecanismo DCRR [Guo and Li 2007] considera que agrupamentos em uma rede dinâmica são de relevância apenas quando há a detecção de algum evento, sendo que manter a estrutura de agrupamentos de maneira contínua na rede é dispendioso. Da mesma forma, o ESC [Villas et al. 2011] coordena os nós na detecção de um evento relevante para que, com a eleição líderes em células espaciais, não sejam enviadas informações redundantes à estação-base. Entretanto, abordagens que dependam do reagrupamento da rede como um todo introduzem custos em termos de latência e energia.

Destas abordagens, nenhuma oferece suporte apropriado ao agrupamento através da similaridade de dados. Considerando a **similaridade de dados**, e de maneira similar ao DCRR, o CAG [Yoon and Shahabi 2007] cria agrupamentos sob demanda. Seu funcionamento considera consultas realizadas a partir de uma estação-base, que inunda a consulta, tendo como parâmetro um limiar percentual tolerável de diferença entre as leituras dos nós a serem agrupados. Ao final, os nós comuns informam suas leituras aos líderes, que as agregam e enviam à estação-base. Da mesma forma que o CAG, o DACH [Wu et al. 2008] define limiares de diferença quanto à similaridade, visando criar, de maneira centralizada na estação-base, uma hierarquia virtual com vários níveis crescentes de similaridade. Na literatura, o protocolo SCCS [Pham et al. 2010] se destaca ao considerar agrupamentos dinâmicos e reconfiguráveis formados pela similaridade de dados, sem necessitar de inundações tão constantes como o CAG e o DACH.

Contudo, nenhuma destas abordagens oferece suporte apropriado à similaridade de dados. Dos protocolos que consideram dados similares, o protocolo CAG depende de inundações constantes na rede para estabelecer novas hierarquias. O protocolo DACH por sua vez, depende muito da estação-base, que coleta informações da rede inteira para estabelecer a hierarquia. Por fim, embora o SCCS dependa de menos centralização, a sua manutenção não é adequada. Para o SCCS se adequar à variação dos dados, é permitida apenas a operação de quebra de agrupamento e, sem que existam fusões, a estação-base deve disparar novos processos de reagrupamento global.

Assim, um protocolo que atue de maneira mais distribuída e seja capaz de estabelecer agrupamentos lógicos consistentes com a semelhança das leituras de dados dos nós sensores se faz necessário. Os protocolos devem ser capazes de realizar uma espécie de sincronização dinâmica para que os nós possam também ser agrupados ou fragmentados continuamente, sem uma reestruturação completa a partir da estação-base.

3. DDFC

Esta seção apresenta o protocolo DDFC (***D**ynamic **D**ata-aware **F**irefly-based **C**lustering*), para criar e manter agrupamentos lógicos dos nós que possuam leituras de dados similares na RSSF. Através da manutenção de estruturas locais para o **armazenamento da vizinhança**, este protocolo sincroniza localmente a **agregação média de leituras similares** nos nós, possibilitando a determinação precisa de quando um agrupamento de nós deve ser fragmentado ou diferentes agrupamentos devem ser unidos. Uma vez estabelecidos os agrupamentos lógicos, o DDFC define **índices para o roteamento interno** aos agrupamentos, possibilitando que as mensagens dos nós comuns atinjam um líder do agrupamento. No DDFC, os agrupamentos formados podem ser compostos por mais de um líder, devido à extensão espacial das leituras similares. Do mesmo jeito, o roteamento interno nos agrupamentos pode utilizar-se de mais de um salto até atingir um líder.

3.1. Armazenamento de vizinhança

Duas estruturas locais simples de grande importância em cada nó sensor suportam o funcionamento do DDFC, contendo **(i)** informações sobre as leituras dos vizinhos espaciais e **(ii)** o conjunto de vizinhos espaciais que satisfazem os *thresholds* de similaridade de dados. A Figura 1 ilustra um pedaço de uma topologia, à esquerda, e as estruturas de dados armazenadas de cada nó. O Nó selecionado possui no total sete vizinhos, dos quais quatro possuem dados similares, que satisfazem os *thresholds* de similaridade de dados.

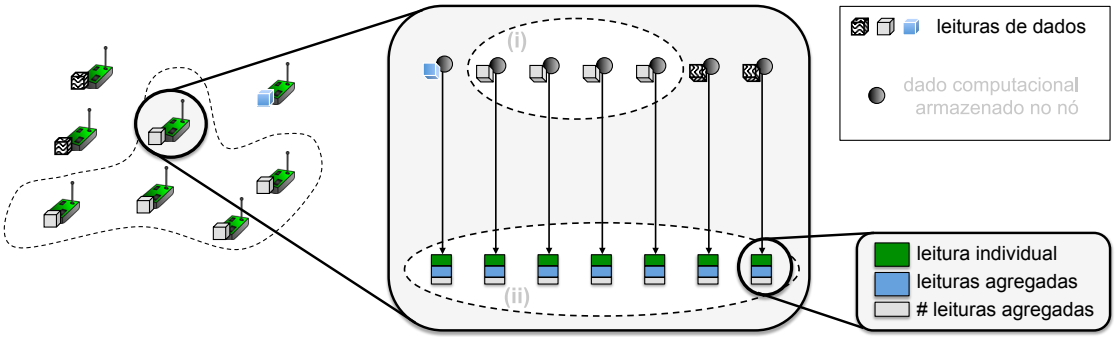


Figura 1. Estruturas de dados que representam a vizinhança dos nós.

Este conjunto de nós de leituras similares é mantido em uma estrutura **(i)** $SNeigh$, conjunto indicado por linhas pontilhadas na parte superior central da figura. Além disso, o nó em questão mantém uma estrutura **(ii)** $NeighR$, conjunto indicado por linhas pontilhadas na parte inferior central da figura, que possui informações sobre as leituras de todos os vizinhos espaciais. Tais informações envolvem a leitura individual de cada vizinho e a leitura agregada da vizinhança daquele vizinho, assim como a quantidade de nós cujas leituras que foram agregadas, para se determinar a importância desta agregação.

3.2. Sincronização da agregação de leituras

O DDFC define um componente de sincronização inspirado nos princípios biológicos de vagalumes [Tyrrell et al. 2006], denominado Agente Vagalume, que atende as questões de sincronização de agregação de leituras e estabelecimento de vizinhanças de leituras similares. O Agente Vagalume sincroniza localmente um valor que indica a agregação das leituras do agrupamento do nó em questão. Com este valor, os nós sabem quando

eles devem sair de seu agrupamento, em caso de leituras muito diferentes, e quando agrupamentos vizinhos devem ser unidos devido à leituras muito próximas, satisfazendo o *threshold* de similaridade de dados.

Inicialmente, cada nó da rede faz parte de um agrupamento diferente, e os diversos agrupamentos gradualmente são unidos, de acordo com o *threshold* de similaridade de dados. Após a convergência e formação estável inicial de agrupamentos, estes são mantidos dinamicamente através da sua união e fragmentação. Estas operações resultam da adição ou remoção de arestas, que representam a semelhança de dados entre pares de sensores, referentes à vizinhança de cada nó do agrupamento, como será visto adiante.

O Algoritmo 1 apresenta o funcionamento do Agente Vagalume do DDFC. Periodicamente, cada nó envia em *broadcast* uma mensagem *beacon*, análoga ao piscar de um vagalume, informando seu identificador $ADDR$, sua leitura atual, obtida através da função $getReading()$, a leitura média agregada dos nós com leituras similares em sua vizinhança, obtida através da função $getAverageReading()$ e a quantidade de vizinhos com leituras similares na vizinhança (l.1-5). Deve-se notar que o envio periódico destas mensagens insere sempre um tempo aleatório ínfimo para evitar transmissões simultâneas.

Algoritmo 1 Agente Vagalume

```
1:  procedimento BEACONTIMEREXPIRE
2:      Send(ADDR, getReading(), getAverageReading(), |SNeigh|)
3:      Wait(interval + rnd())
4:      BeaconTimerExpire()
5:  fim procedimento
6:
7:  procedimento RECEIVEBEACON(src, iR, aR, nR)
8:      NeighR[src] ← {iR, aR, nR}
9:      localAvg ← getAverageReading()
10:     se (|iR − localAvg| < CThresh) e (|getReading() − aR| < CThresh) então
11:         SNeigh ← SNeigh ⋃ {src}
12:     senão
13:         SNeigh ← SNeigh − {src}
14:     fim se
15: fim procedimento
16:
17: procedimento GETAVERAGEREADING
18:     accumulatedReading ← getReading()
19:     numberOfReadings ← 1
20:     para cada v ∈ SNeigh faça
21:         temp ← NeighR[v].aR * NeighR[v].nR
22:         accumulatedReading ← accumulatedReading + temp
23:         numberOfReadings ← numberOfReadings + NeighR[v].nR
24:     fim para cada
25:     retorna (accumulatedReading/numberOfReadings)
26: fim procedimento
```

A função $getAverageReading$ (l.17) calcula a média ponderada sincronizada da agregação de leituras na vizinhança local que satisfaz as relações de similaridade de acordo com o *threshold* desejado, ou seja, aqueles vizinhos que pertencem ao mesmo agrupamento do nó em questão. Considerando a leitura do nó atual (l.18-19), a média das leituras informadas pelos nós que fazem parte do mesmo agrupamento é computada

(l.20-24), considerando a leitura aR agregada de cada um destes nós e o número de leituras nR agregadas nele como um peso (l.21). Ao fim, obtém-se a média agregada de leituras similares na região do nó (l.25).

Ao receber um *beacon* (l.7), o nó saberá a sua origem src de envio, a leitura individual iR do nó origem, a leitura agregada aR média de sua vizinhança, e a quantidade de nós nR que foram considerados nesta agregação. Inicialmente, a estrutura $NeighR$ é atualizada (l.8) com esta informação, independente de relações de similaridade. A agregação média de leituras na região do nó atual (l.9) é considerada para verificar se as leituras do nó atual e do nó origem src satisfazem a similaridade de dados de acordo com o *threshold* de diferença de leituras $CThresh$ (l.10). A estrutura $SNeigh$ é atualizada, incluindo a origem src caso este *threshold* seja satisfeito, ou removendo o nó src, caso contrário.

A função de similaridade utilizada no Algoritmo 1 consiste de duas partes: **(i)** $|iR - localAvg| < CThresh$ e **(ii)** $|getReading() - aR| < CThresh$, que correspondem basicamente à mesma verificação de similaridade, porém com referências diferentes. A parte **(i)** verifica se a leitura iR recebida do nó vizinhos src satisfaz o *threshold* de diferença $CThresh$ com relação ao agrupamento do nó atual. Já a parte **(ii)** verifica se a leitura atual $getReading()$ do nó local satisfaz o *threshold* de diferença $CThresh$ com relação ao agrupamento do nó src. Estas relações são expressas pela Equação 1, onde X e Y representam respectivamente a leitura do nó atual e a leitura com a qual ela está sendo comparada, a fim de satisfazer a diferença $CThresh$.

$$\left| Y - \frac{X + \sum_{v \in SNeigh}(NeighR[v].aR * NeighR[v].nR)}{1 + \sum_{v \in SNeigh}(NeighR[v].nR)} \right| < CThresh \qquad (1)$$

A Figura 2 ilustra um exemplo de funcionamento do Agente Vagalume, mostrando a sincronização da agregação de leituras de cada agrupamento e consequentes relações de similaridade de leituras. As arestas pontilhadas indicam vizinhos puramente espaciais, enquanto as sólidas indicam vizinhos que satisfazem a relação de similaridade de dados. As caixas ao lado de cada nó correspondem à estrutura vista na Figura 1, indicando, de cima para baixo, a leitura individual daquele nó, a leitura agregada sincronizada sua e de seus vizinhos, e a quantidade de leituras agregadas. Cada instante T é separado por um envio de *beacon* de cada nó. No instante inicial $T1$, as leituras agregadas de cada nó correspondem à sua leitura inicial, pois ainda não houve nenhuma troca de mensagens.

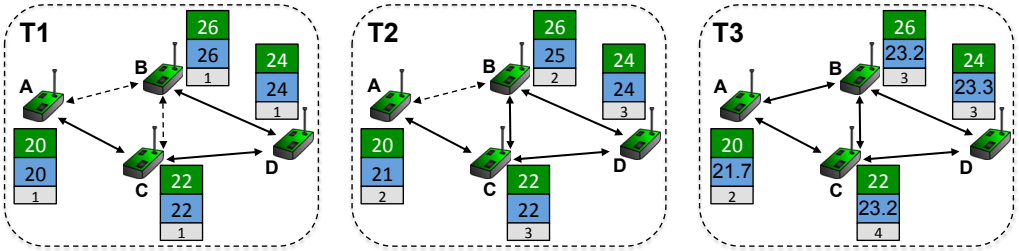


Figura 2. Funcionamento do agrupamento baseado em vagalumes.

Assim, considerando $CThresh = 3.0$, as arestas $((B, D), (D, C), (C, A))$ satisfazem a Equação 1 e estabelecem relações de similaridade no estado $T1$. Então, os nós atualizam suas leituras agregadas aR_{Tn} de acordo com as leituras do instante aR_{Tn-1} anterior, como elaborado no Algoritmo 1. No instante $T2$, $aR_{T2}(A) = \frac{20+1*22}{1+1}$, $aR_{T2}(B) =$

$\frac{26+1*24}{1+1}$, $aR_{T2}(C) = \frac{22+1*20+1*24}{1+1+1}$, $aR_{T2}(D) = \frac{24+1*22+1*26}{1+1+1}$. Neste instante, a aresta de similaridade (B, C) passa a existir. No instante $T3$, as leituras agregadas são atualizadas novamente, $aR_{T3}(A) = \frac{21+3*22}{1+3}$, $aR_{T3}(B) = \frac{25+3*24+3*22}{1+3+3}$, $aR_{T3}(C) = \frac{22+2*21+2*25+3*24}{1+2+2+3}$, $aR_{T3}(D) = \frac{24+2*25+3*22}{1+2+3}$. Neste instante, a aresta de similaridade (A, B) passa a existir.

Com o Agente Vagalume operando desta maneira, cada nó possuirá sua estrutura $SNeigh$ atualizada com a troca de *beacons*. Tal estrutura indica quais nós na vizinhança do nó atual são vistos como membros do mesmo agrupamento. Assim, como cada nó sabe quais vizinhos fazem parte do mesmo agrupamento, o agrupamento global de um nó corresponde ao conjunto formado pela união daquele nó com a união de cada um dos nós nas estruturas $SNeigh$, tal que essa operação é realizada recursivamente para cada nó da $SNeigh$. Indutivamente, se um nó A pertence ao agrupamento de um nó B e B pertence ao agrupamento de um nó C, então A também pertence ao agrupamento de C.

Contudo, esta visão global dos agrupamentos completos não é mantida localmente em cada um dos nós pois a coerência do estado reportado com a formação lógica real não poderia ser garantida. Tal operação causaria grande sobrecarga devido à quantidade de mensagens de atualização e manutenção que a dinamicidade do meio e das formações lógicas demandaria. Desta forma, embora os agrupamentos lógicos existam de maneira global, eles não são visíveis por completo em nível de nó. Entretanto, conhecendo as vizinhanças locais que pertencem aos seus agrupamentos, os nós podem organizar uma hierarquia distribuída baseada em líderes. Com estas estruturas definidas pelo Agente Vagalume, o Agente de Indexação, definido a seguir, opera respeitando a similaridade de dados, e garante que os nós consigam enviar mensagens aos líderes de seus agrupamentos.

3.3. Indexação local das rotas até líderes

O DDFC elege os líderes e estabelece rotas dos nós comuns até os líderes mais próximos através da componente denominada Agente de Indexação, considerando as relações de similaridade de leituras que o Agente Vagalume estabeleceu. O Agente de Indexação usa um esquema de pontuação baseado no sistema de regras proposto no KHOPCA [Brust et al. 2008], devido à sua flexibilidade e abordagem adaptativa. Através destas regras, cada nó atualiza sua pontuação auto atribuída de acordo com a pontuação de seus vizinhos que pertençam ao mesmo agrupamento - informação enviada em *piggyback* na mesma mensagem *beacon* utilizada pelo Agente Vagalume. É definido como parâmetro uma pontuação máxima $MaxK$ que determina também a distância máxima até um líder. Os nós que possuem pontuação equivalente à esta pontuação máxima $MaxK$ são determinados líderes, enquanto os demais nós utilizam sua pontuação como meio de determinar o próximo salto no roteamento até o líder mais próximo.

No início, todos os nós possuem uma pontuação individual $pts = 0$. Considerando $MaxK$ como sendo o parâmetro da pontuação máxima e a lista SN[1] de vizinhos pertencentes ao mesmo agrupamento, as regras utilizadas para a atualização dinâmica dos pontos são definidas de acordo com a Equação 2, baseada nas regras propostas pelo KHOPCA. A primeira condição das equações visa manter uma diferença máxima de 1 entre a pontuação dos nós adjacentes. A segunda regra define um nó como líder, maximizando seu pts para $MaxK$, caso seus vizinhos tenham pontuação mínima. A terceira regra visa diminuir a pontuação de um nó caso ele tenha pontuação maior que seus vizinhos

[1]Mesma lista $SNeigh$, porém abreviada por questões de espaço.

e não seja líder, a fim de manter a diferença máxima de 1 entre as pontuações adjacentes. Por fim, a quarta regra visa eliminar a existência de líderes, nós com $pts = MaxK$, adjacentes. Tais regras compõem o Sistema de Regras do Agente de Indexação.

$$pts(n) \leftarrow \begin{cases} max(pts(SN(n))) - 1, & \text{se } pts(m) > pts(n), \forall m \in SN(n), \\ MaxK, & \text{se } pts(m) = 0, \forall m \in SN(n), \\ pts(n) - 1, & \text{se } pts(n) \neq MaxK \text{ e } pts(n) > pts(m), \\ & \quad \forall m \in SN(n), \\ pts(n) - 1, & \text{se } pts(n) = MaxK \text{ e } \exists m \in SN(n) \text{ tal que} \\ & \quad pts(m) = MaxK \text{ e } \big((|SN(m)| > |SN(n)|) \\ & \quad \text{ou } (|SN(m)| = |SN(n)| \text{ e } m > n)\big). \end{cases} \quad (2)$$

Este sistema de regras, embora baseado no Jogo da Vida e nas regras propostas pelo protocolo KHOPCA [Brust et al. 2008], foi estendido para se adaptar melhor aos quesitos de dinamicidade do meio, possuindo melhor estabilidade. O Agente de Indexação dá mais prioridade no quesito pontuação àqueles nós que possuem mais vizinhos com leituras similares - prioridade expressa na quarta regra do Sistema de Regras. Desta forma, a estabilidade dos líderes é maior.

Estas regras são aplicadas periodicamente em cada nó, na mesma ordem que elas foram apresentadas, em uma sequência ordenada da primeira até a quarta. Entretanto, a cada período no qual elas são verificadas, apenas uma delas pode ser aplicada. Ou seja, se a primeira regra for verificada e aplicada, as demais não são sequer verificadas. Da mesma maneira, se uma regra não satisfizer as condições para que ela seja aplicada, são verificadas em sequência cada uma das sucessoras, parando a verificação das demais assim que a primeira delas for aplicada com sucesso. Tal operação é diferente do KHOPCA, que aplica mais de uma regra, em ordem e de maneira indeterminada em cada intervalo de tempo, produzindo resultados mais instáveis.

A Figura 3 ilustra como tais regras podem ser aplicadas a partir de uma topologia inicial, supondo um valor de $MaxK = 3$. As arestas sólidas entre cada par de nós indicam uma relação de similaridade de leituras entre eles e, desta maneira, eles se consideram vizinhos, de acordo com a estrutura $SNeigh$. No instante $T1$, todos os nós possuem a pontuação mínima $pts = 0$. No estado $T2$, dado que a verificação das regras não possui requisitos de sincronia, neste caso os nós B, C e D aplicam as regras primeiro, maximizando seus pontos através da regra 2. Como os nós A e E realizam a verificação depois, seus vizinhos já possuem $pts = MaxK$ e, desta forma, eles aplicam a regra 1. No instante $T3$, existem 3 nós adjacentes com $pts = MaxK$. Logo, os nós B e D aplicam a

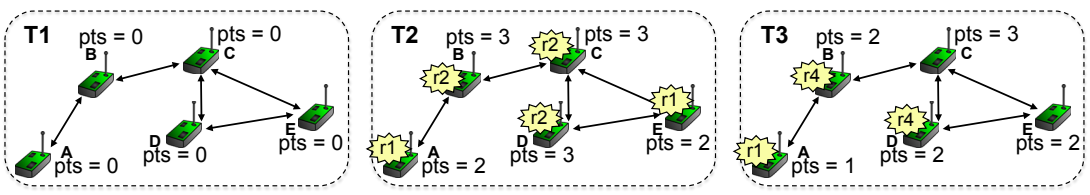


Figura 3. Aplicação das regras do Agente de Indexação.

regra 4, pois o nó C possui mais vizinhos similares. Novamente, o nó A aplica a regra 1, mantendo a diferença máxima de pontos entre nós adjacentes como 1.

A estrutura apresentada na figura é mantida dinamicamente com a variação das leituras e da topologia. Com tal estrutura, os nós que satisfazem $pts = MaxK$ são considerados os líderes. Os nós comuns podem rotear os seus dados para o seu líder mais próximo ao selecionar como próximo salto um nó pertencente à estrutura $SNeigh$, cuja pontuação seja maior do que a sua atual. Assim, como os líderes são aqueles com mais pontos e as regras estabelecem uma progressão de pontos na direção aos líderes, garante-se que ele é atingido no final do roteamento da mensagem de um nó comum.

4. Avaliação de Desempenho

Para a avaliação de desempenho do protocolo de agrupamento DDFC, ele foi implementado no simulador NS-3, versão 3.14.1. O cenário de avaliação cria uma situação realística de monitoramento de ambiente, a fim de se determinar a eficiência dos agrupamentos lógicos estabelecidos e verificar as relações de similaridade de dados e a qualidade das escolhas de líderes. Este cenário foi baseado nas leituras de úmidade coletadas pelo laboratório Intel, da universidade de Berkeley, disponibilizados em [Berkeley 2012]. Considerando um cenário urbano, assume-se que os nós não tem problemas com escassez de energia, sendo obtida de redes elétricas existentes, como de postes [Furlaneto et al. 2012].

O cenário é composto de 54 nós, que operaram por $1200s$. Como os dados de leituras obtidas eram de um ambiente pequeno, este foi aumentado em uma escala de $15x$, obtendo-se uma área retangular de $630m$ por $480m$, visto que no cenário original um raio de transmissão padrão seria capaz de cobrir a área toda, comprometendo os resultados obtidos. Nesta escala, foi definido um raio de transmissão de $100m$, possibilitando uma avaliação que ainda possui dados com relações de similaridade.

Três parâmetros de funcionamento do DDFC são variados nas simulações: **(i)** *CThresh*, que indica o a relação de similaridade entre os dados, **(ii)** *int*, que indica a duração do intervalo fixo de atuação do Controlador de *Beacons*, e **(iii)** *MaxK*, que indica a distância máxima até um nó líder, tal que a quantidade máxima de saltos é $MaxK + 1$. Além disso, é considerado tanto a operação do Agente de Indexação do protocolo **DDFC** com as suas regras propostas, baseadas no KHOPCA, quanto com as regras originais do KHOPCA, sendo este segundo caso de funcionamento denominado **DDFC-K**[2]. Note que o DDFC não é comparado diretamente com o KHOPCA, pois este não considera questões de similaridade de dados. Assim, o DDFC é completamente distinto do KHOPCA, fato evidente na componente principal Agente Vagalume, sendo apenas as regras do Agente de Indexação parcialmente baseadas nas do KHOPCA.

Considerando tais parâmetros, são definidas as métricas: **número de líderes**, **número de agrupamentos**, **número de nós solitários**, **duração dos líderes**, **amplitude média das leituras dos agrupamentos** e **inconsistência de rotas**. Estas métricas determinam o comportamento do protocolo com relação à variação das métricas, e o desempenho com relação à dinamicidade do protocolo para se adaptar às variações de leituras, ao *overhead* gerado e à qualidade das rotas até os líderes.

[2]Quando o KHOPCA for referenciado nos resultados, não se trata da sua implementação, mas sim do DDFC utilizando as regras originais do KHOPCA no seu Agente de Indexação.

O número de líderes, de agrupamentos e de nós solitários são avaliados não só por questões de desempenho, mas também a fim de se adequar ao propósito da aplicação. A duração dos líderes expressa quanto tempo um líder se manteve até ter sua pontuação reduzida, dado que durações médias maiores indicam que nós apropriados foram selecionados. Note que questões de energia são desconsideradas - a melhor aptidão destes nós como líder é independente de energia. A amplitude das leituras dos agrupamentos expressa a diferença média entre as maiores e menores leituras nos agrupamentos, e é importante para se determinar a corretude do comportamento da sincronização da agregação de leituras do Agente Vagalume. Por fim, a inconsistência de rotas corresponde à quantidade média de nós que não conseguem atingir seu líder dada a configuração da pontuação na rede em dado momento discreto, devendo ser minimizada.

Os resultados apresentados em seguida foram obtidos a partir de 35 simulações realizadas para cada combinação de parâmetros. Com isso, os gráficos apresentam os resultados com intervalos de confiança de 95%, indicados por barras verticais.

4.1. Agrupamentos formados

A Figura 4 apresenta gráficos que avaliam a influência do parâmetro $CThresh$ no número de líderes eleitos, de agrupamentos formados, e de nós solitários - isto é, cujo agrupamento é formado apenas por ele só. À esquerda, nota-se que quanto maior o $CThresh$, menor o número de líderes na rede. Ademais, o $MaxK$ gera a mesma influência, sendo mais evidente entre os patamares de $MaxK = 1$ e $MaxK = 2$. Isso acontece pois quanto maior for o parâmetro $CThresh$, menos agrupamentos existirão, como visto no gráfico central, pois nós com leituras mais distantes serão agrupados. O $MaxK$ atua de acordo com as regras propostas no Agente de Indexação, sendo que quanto maior for ele, menor é o número de líderes. Entretanto, nota-se que a diferença é acentuada apenas para $MaxK = 2$, quando $MaxK = 3$ não houve uma diferença significativa no número de líderes, indicando que mesmo havendo possibilidade de nós comuns utilizarem mais saltos para atingir um líder, são poucos os nós que de fato utilizam o número máximo de saltos. Por fim, o número de nós solitários tende a diminuir na medida que o parâmetro $CThresh$ aumenta, pois com um valor maior para este parâmetro, nós com leituras mais divergentes podem ser agrupados com mais facilidade.

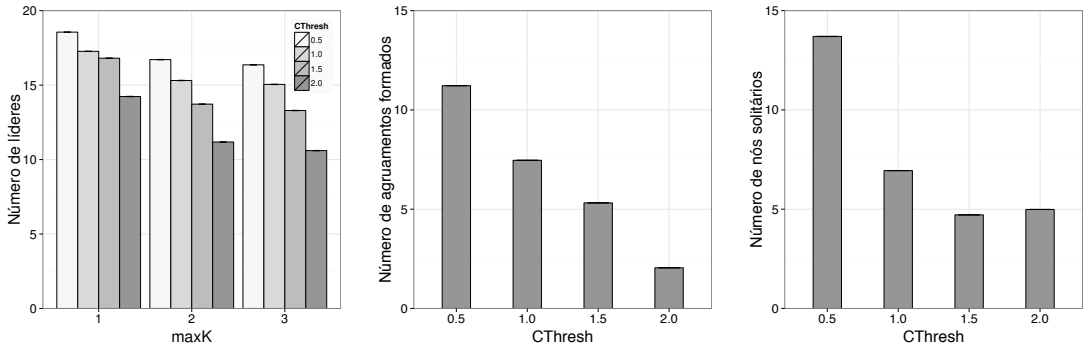


Figura 4. Relação do número de líderes, agrupamentos e nós solitários.

Nestes gráficos, o intervalo de confiança apresentado indica que em 95% das vezes, serão obtidos dados quase idênticos aos apresentados nos gráficos. Esta precisão se deve ao fato de o cenário ser estático, isto é, não possuir mudanças quanto à posição

dos nós e quanto às suas leituras. Entretanto, a operação do protocolo DDFC não é determinística, ou seja, depende de fatores de aleatoriedade. Por isso, tal intervalo de confiança mínimo indica que o protocolo opera de maneira controlada e estável.

4.2. Similaridade de leituras dos nós agrupados

Para determinar se o Agente Vagalume foi capaz de agrupar nós com leituras similares, a métrica amplitude é utilizada, correspondendo à diferença entre a menor e maior leituras dentro de um agrupamento. Os gráficos na Figura 5 mostram a amplitude dos agrupamentos com relação aos parâmetros avaliados. Percebe-se que ela aumenta proporcionalmente ao $CThresh$, mas se mantém sempre inferior à $2 * CThresh$. Isso mostra que o Agente Vagalume foi capaz de agrupar nós com leituras similares, pois dado um valor médio m, um agrupamento aceitaria novos nós no intervalo $[m - CThresh, m + CThresh]$, cuja amplitude é exatamente $2{*}CThresh$. Nota-se que o parâmetro $MaxK$ não exerce grande influência na amplitude. Embora ele influencie no número de líderes, como visto na Figura 4, o número de agrupamentos se mantém o mesmo, visto que ele depende apenas da relação de similaridade de leituras entre os nós e do parâmetro $CThresh$.

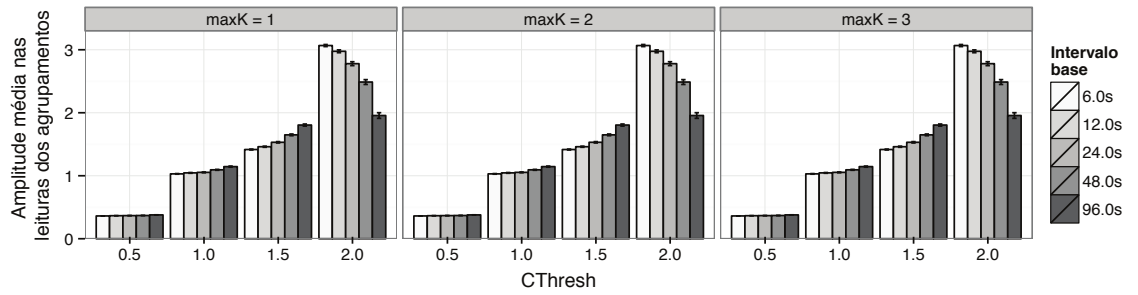


Figura 5. Similaridade de leituras dentro do mesmo agrupamento.

Por fim, quanto maior o parâmetro int, intervalo base da operação do Controlador de *Beacons*, maior é a amplitude das leituras nos agrupamentos. Isso ocorre pois com intervalos de operação muito grandes, os nós demoram mais para trocar *beacons*. Ademais, percebe-se que para $CThresh = 2.0$, o comportamento da variação de amplitude de acordo com o int é anômalo. Embora este comportamento não possa ser inteiramente justificável, isto acontece pois com $CThresh = 2.0$ a rede opera com apenas 2 agrupamentos, como visto na Figura 4. A fragmentação da rede em apenas dois agrupamentos lógicos é anômala devido ao tamanho imenso que eles atingem. Isso mostra que o parâmetro $CThresh$ deve ser ajustado de maneira sensível às leituras coletadas, sendo que $CThresh = 2.0$ é uma valoração inadequada para a operação no cenário descrito.

4.3. Duração dos líderes e inconsistência das rotas

A Figura 6 apresenta um conjunto de histogramas que avalia a relação entre o número de líderes e a sua duração, determinada em quantidades de turnos, dado que um turno representa um intervalo de tempo de $10s$. O conjunto de histogramas é apresentado em um quadro que varia horizontalmente o parâmetro int e verticalmente o parâmetro $CThresh$.

De imediato, nota-se que em todos os casos a maior concentração de nós se encontra na duração de 120 turnos, que neste caso corresponde ao tempo de vida inteiro da rede. O uso das regras modificadas no DDFC acarretou em uma duração de nós líderes

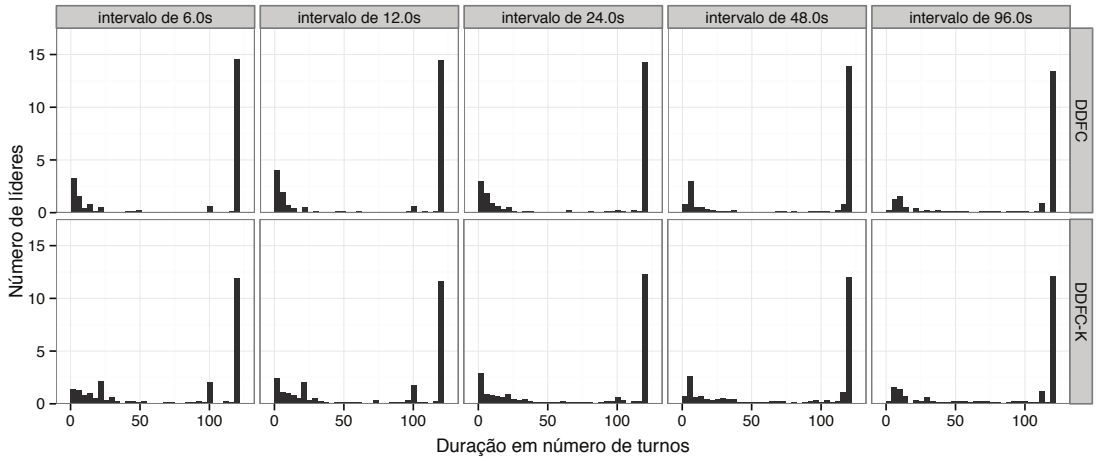


Figura 6. Duração dos líderes em número de turnos de $10s$.

ainda maior quando comparado ao uso das regras originais do KHOPCA no DDFC-K, que apresentou os nós um pouco mais distribuídos nas baixas durações. Esta maior duração no DDFC resulta do sistema de pontuação do Agente de Indexação, indicando que as regras utilizadas determinam nós capazes de se manter estáveis como líderes, mesmo com a dinamicidade das leituras. Isso se deve em especial à quarta regra utilizada pelo Agente de Indexação, que dá prioridade como líder àqueles nós que possuem maior vizinhança com leituras comuns.

Os gráficos na Figura 7 mostram o valor da inconsistência média acumulada das rotas nos cenários que obedecem os parâmetros indicados - isto é, a média de rotas inválidas no tempo total de simulação. Confirma-se que quanto maior o int, maior a quantidade de rotas inválidas e que o parâmetro $CThresh = 0.5$ é muito pequeno e, assim, inapropriado para este cenário. Este gráfico considera também o parâmetro $MaxK$. Percebe-se que, quanto maior o $MaxK$, maior a quantidade de rotas inválidas, devido ao fato das maiores distâncias possíveis entre nó comum e líder aumentarem. Quando comparado ao DDFC-K, as modificações utilizadas no DDFC diminuem o número de rotas inválidas na rede, devido à maior estabilidade dos nós líderes.

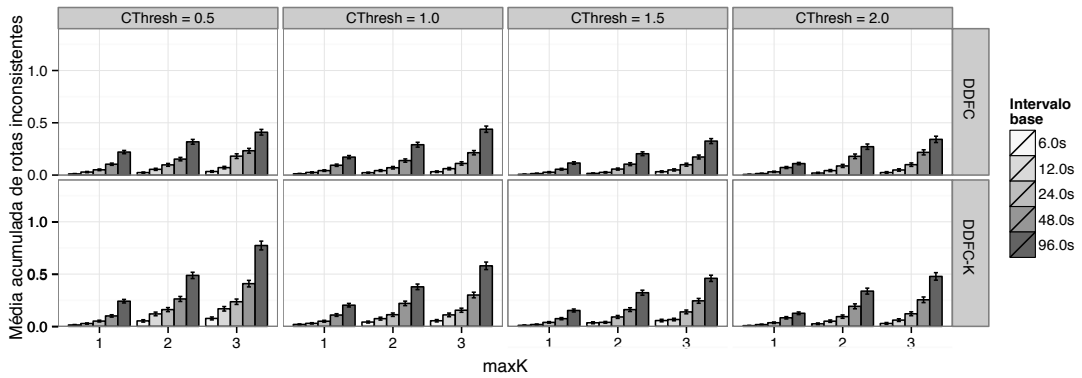


Figura 7. Inconsistência acumulada das rotas.

4.4. Análise crítica dos parâmetros

Com os resultados apresentados, pôde-se estabelecer diversas relações entre os parâmetros e as métricas avaliadas. O $CThresh$ influencia diretamente no número de agrupamentos formados, pois é ele que estabelece o critério de similaridade de leituras que deve ser satisfeito para que os nós se agrupem. Da mesma forma que um valor menor para o $CThresh$ aumenta o número de agrupamentos formados, o número de líderes eleitos também é afetado, visto que ele tem relação com o número de agrupamentos. Assim, com menos agrupamentos formados, os nós da rede normalmente ficam mais próximos dos líderes, utilizando menos saltos para alcançá-los.

O Parâmetro $MaxK$, assim como o $CThresh$, influencia no número de líderes estabelecidos na rede, mas de maneira mais branda. Ademais, embora o aumento neste parâmetro possibilite que existam nós mais distantes de líderes na rede, a maior parte dos nós ainda assim se encontra próximo de líderes, sendo que de $MaxK = 2$ para $MaxK = 3$ já não houve grande mudança. Por outro lado, notou-se um aumento considerável no número de rotas inconsistentes à medida que o $MaxK$ aumentou.

O intervalo de atuação int do Controlador de *Beacons* afeta diretamente a amplitude das leituras dos agrupamentos lógicos, sendo que intervalos menores diminuem a amplitude dos agrupamentos. Do mesmo modo, a inconsistência das rotas é inversamente proporcional ao parâmetro int. Esses comportamentos se devem ao número de *beacons* enviados, que aumenta para intervalos de atuação menores e, assim, possibilita uma maior frequência de atualização dos agrupamentos.

Mostrou-se que, independente da combinação de parâmetros, a amplitude dos agrupamentos sempre é inferior a $2 * CThresh$. Isso indica que a abordagem de sincronização *coletiva* de agregações de leituras nos nós foi bem sucedida. Através desta, os nós foram agrupados logicamente e dinamicamente, de acordo com a similaridade entre as suas leituras. Por fim, verificou-se que a mudança proposta nas regras originais do KHOPCA, utilizada pelo Agente de Indexação, foi capaz de estabelecer líderes mais estáveis, melhorando assim a consistência das rotas internas dos agrupamentos.

5. Conclusão

Agrupamentos de nós com leituras similares possibilitam melhor uso de técnicas de agregação, bem como detecção mais robusta de eventos anômalos. Baseando-se nisso e bioinspirado em vagalumes, o protocolo DDFC utiliza mensagens *beacon* periódicas para manter sincronizada a agregação das leituras dos nós de cada agrupamento. Com isso, definem-se de modo dinâmico nós vizinhos que satisfazem a similaridade de leituras desejável para o agrupamento, possibilitando a fragmentação e a união de agrupamentos lógicos. Considerando os vizinhos de leituras similares definidos pelo agente vagalume, um esquema de indexação mantém a indexação interna aos agrupamentos, estabelecendo rotas até os líderes mais próximos.

O DDFC foi avaliado com leituras reais, obtidas do laboratório Intel, de Berkeley. Simulações demonstram que o DDFC mantém os nós agrupados de maneira dinâmica através da agregação sincronizada das leituras dos agrupamentos, satisfazendo sempre o limiar de similaridade definido. Trabalhos futuros envolvem o controle adaptativo do intervalo entre as mensagens *beacon* enviadas e a comparação do protocolo DDFC com o SCCS, que também cria agrupamentos baseados na similaridade de dados.

Referências

Banerjee, J., Mitra, S. K., Ghosh, P., and Naskar, M. K. (2011). Memory based message efficient clustering (MMEC) for enhancement of lifetime in wireless sensor networks using a node deployment protocol. In *ICCCS '11: Proceedings of the 2011 International Conference on Communication, Computing & Security*. ACM.

Berkeley (2012). http://db.csail.mit.edu/labdata/labdata.html.

Brust, M. R., Frey, H., and Rothkugel, S. (2008). Dynamic multi-hop clustering for mobile hybrid wireless networks. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication - ICUIMC '08*, page 130, New York, New York, USA. ACM Press.

Dechene, D. J., Jardali, A. E., Luccini, M., and Sauer, A. (2007). A survey of clustering algorithms for wireless sensor networks. *Computer Communications*, pages 2826–2841.

Furlaneto, S., Santos, A., and Hara, C. (2012). An Efficient Data Acquisition Model for Urban Sensor Networks. *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 113–120.

Guo, B. and Li, Z. (2007). A dynamic-clustering reactive routing algorithm for wireless sensor networks. *Wireless Networks*, 15(4):423–430.

Islam, M., Abdullah, S., Wada, K., Uchida, J., and Chen, W. (2011). An efficient routing protocol on a Dynamic Cluster-based Sensor Network. In *Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, pages 161–165.

López, T. S., Ranasinghe, D. C., Harrison, M., and Mcfarlane, D. (2012). Adding sense to the Internet of Things. *Personal and Ubiquitous Computing*, 16(3).

Ma, H.-D. (2011). Internet of Things: Objectives and Scientific Challenges. *Journal of Computer Science and Technology*.

Murty, R. N., Mainland, G., Rose, I., Chowdhury, A. R., Gosain, A., Bers, J., and Welsh, M. (2008). CitySense: An Urban-Scale Wireless Sensor Network and Testbed. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 583–588.

Partridge, C. (2011). Realizing the future of wireless data communications. *Communications of the ACM*, 54(9).

Pham, N. D., Le, T. D., Park, K., and Choo, H. (2010). SCCS: Spatiotemporal clustering and compressing schemes for efficient data collection applications in WSNs. *International Journal of Communication Systems*, 23(11):1311–1333.

Rajkumar, R., Lee, I., Sha, L., and Stankovic, J. (2010). Cyber-physical systems: The next computing revolution. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 731–736. ACM.

Reis, I. A., Câmara, G., Assunção, R., and Monteiro, A. M. V. (2007). Data-aware clustering for geosensor networks data collection. *Anais XIII Simpósio Brasileiro de Sensoriamento Remoto*, pages 6059–6066.

Tyrrell, A., Auer, G., and Bettstetter, C. (2006). Fireflies as role models for synchronization in ad hoc networks. In *BIONETICS '06: Proceedings of the 1st international conference on Bio inspired models of network, information and computing systems*. ACM.

Villas, L., Guidoni, D., and Araujo, R. (2011). Explorando a correlacao espacial na coleta de dados em redes de sensores sem fio. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2011)*, 29:411–424.

Wu, X., Wang, P., Wang, W., and Shi, B. (2008). *Data-aware clustering hierarchy for wireless sensor networks*. Springer-Verlag.

Yoon, S. and Shahabi, C. (2007). The Clustered AGgregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks. *Transactions on Sensor Networks*, 3(1).

Roteamento e Agregação de Dados Usando *Sinks* em Alta Velocidade em Redes de Sensores Sem Fio

Leandro N. Balico[1], Horácio A.B.F. Oliveira[1], Eduardo F. Nakamura[12], Raimundo S. Barreto[1], e Antonio A.F. Loureiro [3]

[1]Instituto de Computação – Universidade Federal do Amazonas

[2]Centro de Analise, Pesquisa e Inovação Tecnológica – FUCAPI
Manaus – AM – Brasil

[3]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brasil

{balico,horacio,rbarreto}@icomp.ufam.edu.br

eduardo.nakamura@fucapi.br, loureiro@dcc.ufmg.br

Abstract. *In this work, we study the impact of data aggregation in WSNs with mobile sink nodes that can move at higher speeds. In these cases, propagated queries cannot be answered by using the sink's position when the query was sent, since the sink node will be elsewhere. Therefore, conventional data aggregation schemes may not be appropriate due to this high speed. Thus, we propose and evaluate the performance of three new algorithms for data routing and aggregation in WSNs when the sink is moving at a high speed. These algorithms explore the sink movement to create a routing graph composed by the union paths that intersect the sink's trajectory. At each hop, the sink speed and current network delays are used as metric to guide in-network data aggregation. Our results show clearly significant energy savings and efficient data delivery achieved by the proposed algorithms, as well as provide significant results about their behavior in different scenarios.*

Resumo. *Neste trabalho, avaliamos o impacto da agregação de dados em RSSFs utilizando nós sinks movendo-se em altas velocidades. Nestes casos, consultas propagadas na rede não podem ser respondidas utilizando a posição do nó sink quando a consulta foi enviada, uma vez que o sink estará em outro lugar na rede. Por esse motivo, esquemas convencionais de agregação de dados podem não ser apropriados devido à alta velocidade do nó sink. Portanto, neste trabalho, propomos e avaliamos o desempenho de três novos algoritmos para roteamento e agregação de dados usando sinks em alta velocidade em RSSFs. Estes algoritmos exploram o movimento do sink para criar um grafo de roteamento composto pela união de caminhos de entrega que interseccionam-se com a trajetória do sink. Em cada salto, a velocidade do sink e atrasos vigentes na rede são utilizados como métrica para guiar a agregação de dados durante o roteamento. Os resultados obtidos neste trabalho demonstram uma economia de energia significante bem como uma entrega de dados eficiente obtida pelos algoritmos propostos, assim como fornecem resultados relevantes sobre seu comportamento em diferentes cenários.*

1. Introdução

Redes de Sensores Sem Fio (RSSFs) [Akyildiz et al. 2002, Estrin et al. 2001] são tipicamente compostas por dispositivos computacionais autônomos (nós sensores) depositados em uma área de interesse. Esses sensores, de forma colaborativa, monitoram condições ambientais físicas e químicas em diferentes locais. Dados coletados pelos sensores são normalmente processados e enviados para o nó *sink* de forma distribuída, utilizando comunicação sem fio em múltiplos saltos.

Estudos recentes [Oliveira et al. 2010, Luo and Hubaux 2010, Kim et al. 2003, Ye et al. 2002] demonstraram o uso de *sinks* móveis em RSSFs como forma de prolongar o tempo de vida da rede. Neste contexto, decisões em relação à mobilidade do *sink* são tomadas visando equilibrar o consumo energético na rede [Faheem et al. 2009]. Nesses trabalhos, o nó *sink* tem em geral uma mobilidade baixa e o principal desafio para o roteamento é manter as tabelas de roteamento atualizadas e encaminhar os dados coletados em direção ao nó *sink* (como ilustrado na figura 1).

Cenários recentes de RSSFs, demonstram a necessidade de novos algoritmos para encaminhar os dados para *sinks* movendo-se em altas velocidades [Oliveira et al. 2010]. Nestes cenários, o nó *sink* pode ser um Veículo Aéreo Não Tripulado (VANT) ou até mesmo um avião. Entretanto, devido à alta velocidade do nó *sink*, os pacotes de resposta não podem ser enviados para a posição do *sink* quando ele enviou o pacote de consulta, uma vez que ele estará em outro lugar na rede (como mostra a figura 1).

Neste trabalho, propomos um novo algoritmo de Roteamento e Agregação de Dados para o envio de dados em direção a um *sink* em alta velocidade em RSSFs: HISPEAR (*HIgh SPEed Aggregation and Routing*). Com base neste algoritmo, propomos três diferentes variantes: HISPEAR *Shortest*, HISPEAR *Intercept* e HISPEAR *Hybrid*. A ideia principal dos algoritmos HISPEAR é calcular posições futuras do nó *sink* utilizando sua trajetória e efetuar roteamento e agregação de dados em direção a essas posições calculadas. O desempenho dos algoritmos propostos foi avaliado utilizando o simulador NS-2. Apresentamos um extenso conjunto de experimentos que demonstram a eficiência e significante economia de energia obtida pelos algoritmos propostos em diferentes cenários.

O restante deste trabalho é organizado como segue. A seção 2 descreve os trabalhos relacionados no contexto da mobilidade do *sink*. A seção 3 apresenta o algoritmo HISPEAR e suas variantes, enquanto que a seção 4 descreve sua avaliação de desempenho. Finalmente, a seção 5 apresenta as conclusões e sugestões de trabalhos futuros.

2. Trabalhos Relacionados

A primeira abordagem ao problema de entregar dados de nós sensores para um *sink* móvel em RSSF foi o algoritmo TTDD (*The Two-Tier Data Dissemination*) [Ye et al. 2002]. Nesse algoritmo, ao detectar um novo evento, o nó sensor constrói de maneira proativa uma estrutura de grade virtual que permite aos *sinks* móveis receberem dados através de consultas disseminadas de maneira controlada às células locais em que se encontram. Kim et al. [Kim et al. 2003] propuseram o algoritmo SEAD (*Scalable Energy-efficient Asynchronous Dissemination*) que utiliza a distância e a quantidade de tráfego entre os sensores para criar uma aproximação da Árvore de Steiner mínima para encaminhar dados e interconectar a redes aos *sinks* móveis sem disseminar a posição desses nós pela árvore.

Shim e Park [Shim and Park 2006] propuseram um algoritmo baseado em localizadores para *sinks* móveis. Esses nós localizadores são distribuídos uniformemente pelo campo de sensores e rastreiam a posição atual do *sink* móvel. Se um nó sensor tentar enviar dados para uma posição desatualizada do *sink*, esse nó pode obter uma posição mais atualizada utilizando os localizadores. Finalmente, Oliveira et al. [Oliveira et al. 2010] propuseram o algoritmo WHISPER (*Wireless High Speed Routing*) para cenários de RSSFs onde o *sink* movimenta-se em alta velocidade. Assim como o algoritmo HISPEAR proposto neste trabalho, o algoritmo WHISPER também encaminha os pacotes para posições futuras do nó *sink*. Entretanto, assim como todas as soluções citadas anteriormente, esses estudos não consideram a agregação de dados a fim de aperfeiçoar a eficiência energética de RSSFs com sinks movimentando-se em altas velocidades, a qual é a motivação principal deste trabalho.

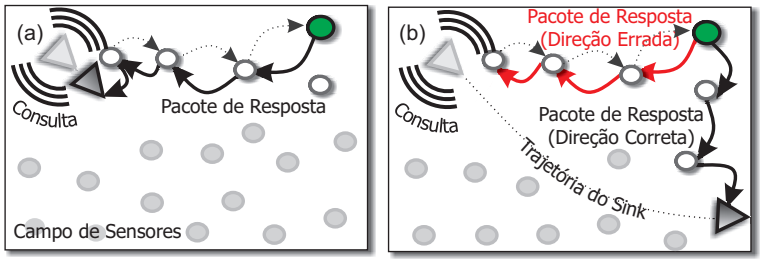


Figura 1. Consulta enviada por: (a) um *sink* em baixa velocidade; e (b) um *sink* em alta velocidade.

3. *High Speed Aggregation and Routing* – (HISPEAR) em RSSFs

Nesta seção, propomos um novo algoritmo para encaminhar dados para *sinks* em alta velocidade efetuando agregação de dados em cada salto da rede: o algoritmo HISPEAR (*High Speed Aggregation and Routing*). O algoritmo HISPEAR explora o movimento do *sink* para criar um grafo de roteamento composto pela união de caminhos de entrega que interseccionam-se com a trajetória do *sink*. Em cada salto, a velocidade de *sink* e atrasos vigentes na rede são utilizados como métrica para guiar a agregação de dados durante o roteamento. Este algoritmo calcula o melhor tempo disponível para agregar dados em cada salto da rede, a fim de entregar dados menos redundantes para uma posição mais atualizada, ou mesmo uma posição futura, do nó *sink*. Cada nó encaminha pacotes agregados para o nó vizinho mais próximo da nova posição calculada do *sink*. Dessa forma, é importante que cada nó conheça a sua própria posição, a posição dos seus vizinhos, a Trajetória e Deslocamento do *sink* bem como a Velocidade de Propagação dos Pacotes [Oliveira et al. 2010].

3.1. Definições Preliminares

Nesta seção, apresentaremos formalmente os conceitos utilizados neste trabalho.

Definição 1 (Redes de Sensores Sem Fio): nós definimos uma RSSF como um grafo Euclidiano $G = (V, E, r)$, onde $|V| = n$ é o número de nós e r é o raio de comunicação; $V = \{v_0, v_1, v_2, \ldots, v_n\}$, onde v_0 é o *sink* móvel e $\{v_1, v_2, \ldots, v_n\}$ o conjunto de nós sensores; $\langle i, j \rangle \in E$ se e somente se v_i alcança v_j, em outras palavras, v_i está dentro do raio de comunicação r de um nó v_j; e $\forall v_i \in V$, $(Xp_i, Yp_i, Zp_i) \in \mathbb{R}^3$ é a posição real do nó v_i; enquanto $(Xc_i, Yc_i, Zc_i) \in \mathbb{R}^3$ é a posição estimada do nó v_i (i.e., utilizando um sistema de localização).

Definição 2 (Alta Velocidade do Sink Móvel): neste trabalho, consideramos o nó *sink* como estando em alta velocidade caso não seja possível que o pacote de resposta de uma consulta enviada pelo *sink* alcance-o usando o caminho reverso em que a consulta percorreu na rede, devido a uma grande distância percorrida pelo *sink* desde a posição onde a consulta foi originada (ver figura 1).

Definição 3 (Algoritmo de Consulta de Dados): diferentemente de tradicionais esquemas de roteamento fim-a-fim que tentam manter rotas entre nós de origem e destino, este trabalho está focado em algoritmos de roteamento para consulta de dados [Boukerche and Nikoletseas 2004], objetivando dar suporte à agregação de dados *in-network*. Neste algoritmo, o nó *sink* envia uma consulta para a rede de sensores, como se esta fosse um sistema de banco de dados distribuído (e.g., *Sensor Databases* [Hong and Madden 2004]). Esta consulta é disseminada na rede por *flooding* e, nós sensores com dados relacionados montam um pacote de resposta que é enviado para o nó *sink*. Em cada salto, os nós sensores podem olhar o conteúdo dos pacotes e efetuar agregação de dados em múltiplos pacotes de entrada para encaminhar um único pacote agregado, de acordo com a função de agregação especificada na consulta. Neste trabalho, consideramos funções simples de agregação de dados (tais como *máximo, mínimo, média*). Trabalhos relacionados deste tipo incluem [Krishnamachari et al. 2002] and [Intanagonwiwat et al. 2003].

Definição 4 (Trajetória do Sink – T_s): esta trajetória pode ser uma linha, uma curva ou qualquer outra trajetória que possa ser expressa matematicamente. Por uma questão de simplicidade, e sem qualquer perda de generalidade, consideramos que, enquanto no interior do campo de sensores, o *sink* manterá uma trajetória em linha reta, que é comum para objetos em alta velocidade. Portanto, dado um ponto inicial (e.g., a posição do *sink* quando a consulta foi enviada) e uma direção, esta linha pode ser facilmente calculada como $y = \tan(\theta)(x - x_0) + y_0$, onde θ é o ângulo em relação ao eixo x e (x_0, y_0) é a posição inicial. Esta trajetória do *sink* é enviada junto com o pacote de consulta.

Definição 5 (Tempo de Espera Para Agregação de Dados – A_t): um nó sensor, ao receber um pacote de consulta, define uma linha perpendicular entre a sua posição (Xc_i, Yc_i) e a trajetória do *sink* pela equação $y = \frac{-1}{\tan(\theta)}(x - Xc_i) + Yc_i$. Dois pontos são calculados nesta linha para servir como referência para duas métricas. O primeiro ponto A_S, está localizado na intersecção da linha e a trajetória do *sink*, como mostra a figura 2(a). Para esse ponto, definimos a diferença entre o tempo para o nó *sink* atingir o ponto A_S e o tempo para um pacote do nó atual chegar a esse ponto como primeira métrica de tempo de espera para agregação de dados (A_t). Portanto, pela equação $A_t = dist(T_k, A_S) * v - \frac{dist((Xc_i, Yc_i), A_S)}{P_k}$ é estabelecida uma relação entre as velocidades do *sink* e pacote. O segundo ponto A_B, está localizado no ponto de intersecção da linha com a fronteira do campo de sensores, conforme ilustrado na figura 2(b). Para esse ponto, o tempo de espera para agregação de dados é definido como o tempo para uma consulta ser propagada a partir do nó atual e retornar como um pacote de resposta do nó mais distante na linha. Logo, pela equação $A_t = 2 \times \frac{dist((Xc_i, Yc_i), A_B)}{P_k}$ o nó atual estima o tempo necessário para os pacotes de resposta dos nós intermediários chegam à sua posição.

3.2. Algoritmo HISPEAR

O HISPEAR, mostrado e explicado no algoritmo 1, começa quando o nó *sink* entra no campo de sensores e envia um pacote de consulta (linhas 2-6). Cada nó que recebe a

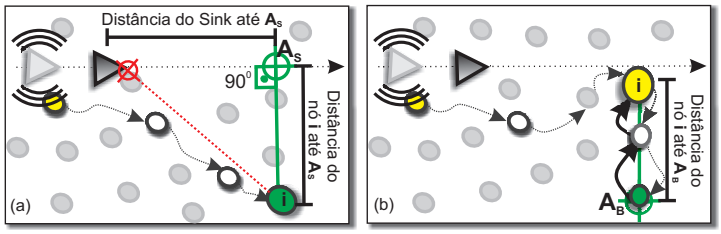


Figura 2. Tempo de Espera para Agregar Dados: (a) ponto de referência baseado nas velocidades do *sink* e pacotes; e (b) ponto de referência para os pacotes dos nós mais distantes.

consulta (diretamente do *sink* ou por meio de um nó intermediário) estima o atraso desse pacote, atualiza sua tabela de vizinhos e encaminha o pacote de consulta para os seus vizinhos (linhas 8–16). Em seguida, o tempo de espera para agregação de dados é calculado (linhas 17–21) e, um temporizador é então inicializado com o tempo escolhido (A_t , linha 22) para agregar os pacotes que chegam ao nó enquanto esse temporizador estiver ativo.

Algorithm 1 Algoritmo HISPEAR

▷ **Input:**
1: Nó *sink* 0 entra no campo de sensores e envia o pacote de consulta *query* com id id_0.
Action:
2: $id_0 + +; src_0 \leftarrow v_0; R_0 \leftarrow 0;$ {Id da consulta, origem e tempo de roteamento}
3: $T_0 \leftarrow traj(); S_0 \leftarrow speed();$ {Trajetória do *sink* e velocidade}
4: $Lp_0 \leftarrow (Xc_i, Yc_i);$ {Última posição do *sink*}
5: $cmd_0 \leftarrow 'SELECT\ MAX(temperature)\ FROM\ wsn';$ {Especificação da função de agregação de dados}
6: Envie $qry(src_0, id_0, cmd_0, T_0, S_0, R_0, Lp_i);$ {Dissemina a consulta}
▷ **Input:**
7: $msg_i \leftarrow qry(src_k, id_k, cmd_k, T_k, S_k, R_k, Lp_k); d_i = delay(msg_i);$
Action:
8: $R_i \leftarrow R_k + d_i;$ {Atualiza o tempo de roteamento}
9: $P_i \leftarrow dist((Xc_i, Yc_i), Tk)/R_i;$ {Velocidade de propagação da consulta}
10: **if** $k \neq 0$ **then** {SE: o pacote não veio diretamente do nó *sink* }
11: $Neig_i \leftarrow Neig_i \cup (k, Lp_k)$ {Atualiza a tabela de vizinhos}
12: **end if**
13: **if** $(src_k, id_k) \notin Fwd_i$ **then** {SE: o nó ainda não propagou a consulta }
14: $Fwd_i \leftarrow Fwd_i \cup (src_k, id_k)$ {Atualiza a tabela de encaminhamento}
15: $Lp_i \leftarrow (Xc_i, Yc_i)$ {Posição deste nó}
16: Envie $qry(src_k, id_k, cmd_k, T_k, S_k, R_i, Lp_i);$ {Encaminha a consulta}
17: $A_t \leftarrow dist(T_k, A_S) \times S_k - dist((Xc_i, Yc_i), A_S)/P_i;$ {Tempo para agregar os dados em relação ao *sink* }
18: $tmp \leftarrow 2 \times dist((Xc_i, Yc_i), A_B)/P_i;$ {Tempo para agregar os dados dos nós mais distantes}
19: **if** $tmp < A_t$ **then** {SE: Escolhe o menor tempo para agregar dados}
20: $A_t \leftarrow tmp;$
21: **end if**
22: $Timer_i.schedule(A_t)$ {Inicializa o temporizador para agregação de dados}
23: **end if**
24: **if** $data_i \leftarrow evaluate(cmd_k)$ **then** {SE: o nó tem dados para responder ao *sink*}
25: $src_i \leftarrow i;$ {Id de origem da resposta}
26: $Fwd_i \leftarrow Fwd_i \cup (ori_k, id_k);$ {Atualiza a tabela de encaminhamento}
27: $forward(src_i, id_k, data_i, T_k, S_k, R_i, P_i)$; {Chama o algoritmo de encaminhamento}
28: **end if**
▷ **Input:**
29: $msg_i \leftarrow reply(src_k, id_k, data_k, T_k, S_k, R_k, P_k); d_i = delay(msg_i);$
Action:
30: **if** $k = 0$ **then** {SE: este nó é o *sink*}
31: $store(data_k)$ {Recebe o pacote de resposta e o armazena}
32: **else**
33: **if** $(src_k, id_k) \notin Fwd_i$ **then** {SE: o este nó nunca encaminhou um pacote de resposta}
34: $Fwd_i \leftarrow Fwd_i \cup (src_k, id_k)$ {Atualiza a tabela de encaminhamento}
35: $R_i \leftarrow R_k + d_i$ {Atualiza o tempo de roteamento}
36: $forward(src_i, id_k, data_i, T_k, S_k, R_i, P_k)$; {Chama o algoritmo de encaminhamento}
37: **end if**
38: **end if**

Para encaminhar esses pacotes, o algoritmo HISPEAR chama a função de encaminhamento ($forward$, linhas 27 e 36), a qual irá escolher uma das três variantes do algoritmo. Neste trabalho, propomos três diferentes variantes do algoritmo HISPEAR: o algoritmo HISPEAR *Intercept* (seção 3.3), o algoritmo HISPEAR *Shortest* (seção 3.4) e o algoritmo HISPEAR *Hybrid* (seção 3.5). Nas próximas seções vamos explicar a operação destes três algoritmos propostos.

3.3. HISPEAR *Intercept*

O algoritmo HISPEAR *Intercept* calcula o primeiro ponto de intercepção entre a trajetória do nó *sink* e do pacote de resposta. Neste algoritmo, para encaminhar um pacote agregado ($data_{Ak}$), conforme descrito no algoritmo 2 (linha 2), primeiro é verificado se o pacote é capaz de interceptar o nó *sink* antes que ele saia do campo de sensores. Nesse caso, o pacote é transmitido imediatamente para o seu nó vizinho que está mais próximo da posição atual do nó *sink* (linhas 3–9). Caso contrário, o ponto de intercepção na trajetória do *sink* é calculado com base nas velocidades do *sink* e da velocidade de propagação da consulta [Oliveira et al. 2010] para encaminhar o pacote (linhas 11–15). Se o temporizador de agregação de dados ($Timer_k$) está ativo, os dados agregados são agendados para ser encaminhados para o nó vizinho que está mais próximo do ponto calculado (linhas 16-17), caso contrário, os dados são enviados imediatamente (linhas 18–21). Como mostra a figura 3(a), os caminhos criados por este algoritmo para encaminhar e agregar dados tendem a ser inclinados em relação à trajetória do nó *sink*.

Algorithm 2 Algoritmo de encaminhamento HISPEAR *Intercept*

▷ **Input:**
1: $forward(src_k, id_k, data_k, T_k, S_k, R_k, P_k)$;
Action:
2: $data_{Ak} = aggregate(data_{Ak}, data_k)$ {Efetua a agregação de dados }
3: **if** $S_k > P_k$ **then** {SE: se a velocidade do *sink* é maior que a do pacote}
4: $sinkPos.X = T_k.X + S_k \times \cos(T_k.\theta) \times R_k$;
5: $sinkPos.Y = T_k.Y + S_k \times \sin(T_k.\theta) \times R_k$;
6: $nextHope_k = closestNeigh(sinkPos.X, sinkPos.Y, sinkPos.Z)$; {Vizinho mais próximo do *sink*}
7: $Timer_k.cancel()$; {Cancela o temporizador}
8: Envie $reply(src_k, id_k, data_{Ak}, T_k, S_k, R_k, P_k)$ para $nextHop_k$ {Encaminhe o pacote imediatamente}
9: $data_{Ak} \leftarrow \emptyset$;
10: **else**
11: $timeIntercept = \sqrt{\frac{(T_k.Y - Y_{ci})^2}{P_k^2 - (S_k \times \sin(T_k.\theta))^2}}$
12: $sinkPos.X = T_k.X + S_k \times \cos(T_k.\theta) \times (R_k + timeIntercept)$;
13: $sinkPos.Y = T_k.Y + S_k \times \sin(T_k.\theta) \times (R_k + timeIntercept)$;
14: $sinkPos.Z \leftarrow T_k.Z$;
15: $nextHop_k = closestNeigh(sinkPos.X, sinkPos.Y, sinkPos.Z)$; {Ponto de interceptação}
16: **if** $Timer_k.isActive()$ **then** {SE: ainda há tempo para agregar dados}
17: $msg_k \leftarrow (src_k, id_k, data_{Ak}, T_k, S_k, R_k, P_k)$; {Atualiza o pacote agendado}
18: **else** {SENÃO: encaminhe o pacote agora}
19: $Timer_k.cancel()$; {Cancela o temporizador}
20: Envie $reply(src_k, id_k, data_{Ak}, T_k, S_k, R_k, P_k)$ para $nextHop_k$ {Encaminhe os pacotes agregados agora}
21: $data_A \leftarrow \emptyset$;
22: **end if**
23: **end if**

3.4. HISPEAR *Shortest*

O algoritmo *HISPEAR Shortest*, em vez de calcular o primeiro ponto de interceptação, calcula o ponto mais próximo de interceptação, como explicado no algoritmo 3 (linhas 2–7). Uma vez que o *sink* pode estar distante desse ponto mais próximo calculado, também é calculado um tempo de espera para que o *sink* esteja mais perto desse ponto ($timesink$,

linhas 8-10) para encaminhar os pacotes. Assim, antes de enviar o pacote de resposta, nós sensores utilizam esse tempo para efetuar agregação de dados enquanto esperam que o nó *sink* aproxime-se do ponto de entrega. Entretanto, também é verificado se o nó sink já cruzou o ponto mais próximo de interceptação ($nextTime$ negativo). Nesse caso, o pacote de resposta é enviado imediatamente utilizando o algoritmo HISPEAR *Intercept* (linhas 11–12).

Algorithm 3 Algoritmo de Encaminhamento HISPEAR *Shortest*

▷ **Input:**
1: $forward(src_k, id_k, data_k, T_k, S_k, R_k, P_k)$;.
Action:
2: $dX \leftarrow S_k \times \cos(T_k.\theta) \times R_k$;
3: $dY \leftarrow S_k \times \sin(T_k.\theta) \times R_k$;
4: $tan \leftarrow \frac{(Xc_i - T_k.X) \times dX + (Yc_i - T_k.Y) \times dY}{dX^2 + dY^2}$;
5: $sinkPos.X \leftarrow T_k.X + tan \times dX$
6: $sinkPos.Y \leftarrow T_k.X + tan \times dY$
7: $sinkPos.Z \leftarrow T_k.Z$;
8: $timePkt \leftarrow dist((Xc_i, Yc_i), sinkPos)/P_k$; {Tempo para o *sink* estar próximo}
9: $timeSink \leftarrow dist(sinkPos, T_k)/S_k$; {Tempo para o pacote alcançar o *sink*}
10: $nextTime \leftarrow timeSink - timePkt$;
11: **if** $nextTime < 0$ **then** {SE: o nó *sink* ultrapassou o ponto}
12: encaminhe o pacote imediatamente pelo algoritmo HISPEAR *Intercept*;
13: **else**
14: $nextHop_k = closestNeigh(sinkPos.X, sinkPos.Y, sinkPos.Z)$; {Ponto mais próximo}
15: $data_{Ak} = aggregate(data_A, data_k)$ {Efetua a agregação de dados}
16: $msg_k \leftarrow (src_k, id_k, data_{Ak}, T_k, S_k, R_k, P_k)$; {Atualiza o pacote agendado}
17: **if** $Timer_k.isNotActive()$ **then** {SE: o timer para agregação de dados não está ativo}
18: $Timer_A.schedule(nextTime)$ {Inicializa o temporizador para agregação de dados}
19: **else** {SENÃO: Atualiza o temporizador para agregação de dados }
20: $Timer_A.reschedule(nextTime)$ {Atualiza o temporizador para agregação de dados}
21: **end if**
22: **end if**

Um aspecto interessante deste algoritmo é que os nós sensores calculam aproximadamente o mesmo ponto mais próximo em um caminho de entrega. O tempo para o nó *sink* estar próximo do ponto mais próximo é recalculado em cada salto e utilizado para atualizar o temporizador da agregação de dados (linhas 17–20). Como ilustra a figura 3(b), a trajetória do pacote de resposta tende a ser uma linha perpendicular à trajetória do *sink*.

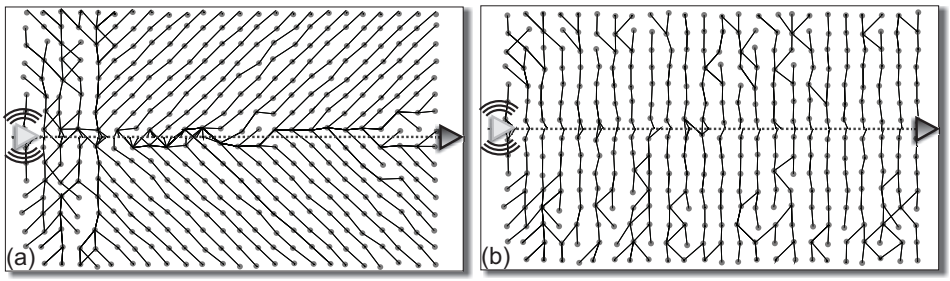


Figura 3. Grafos de roteamento HISPEAR: (a) HISPEAR *Intercept*; e (b) HISPEAR *Shortest*.

3.5. HISPEAR *Hybrid*

O algoritmo HISPEAR *Hybrid* foi projetado para explorar um número reduzido de transmissões, a partir de caminhos mais curtos de encaminhamento e efetuar agregação de dados mesmo quando a velocidade do *sink* é alta em comparação com a velocidade dos pacotes. O algoritmo proposto escolhe, a cada salto, a melhor opção entre o ponto de

menor distância (HISPEAR *Shortest*) e o primeiro ponto de intercepção (*HISPEAR Intercept*). Como mostrado no algoritmo 4, primeiro é calculado o ponto mais próximo de interceptação na trajetória do *sink* (linha 2), da mesma forma que é calculado no HISPEAR *Shortest* (algoritmo 3, linhas 2–7). O primeiro ponto de intercepção entre as trajetórias do nó *sink* e do pacote de resposta é também calculado (linha 3), assim como no HISPEAR *Intercept* (algoritmo 2, linhas 11–15).

Algorithm 4 Algoritmo de encaminhamento HISPEAR *Hybrid*

▷ **Input:**
1: $forward(src_k, id_k, data_k, T_k, S_k, R_k, P_k)$;
Action:
2: Calcula o ponto mais próximo de interceptação (HISPEAR *Shortest*);
3: Calcula o primeiro ponto de interceptação (HISPEAR *Intercept*);
4: **if** $timePkt < timeIntercept$ **then** { SE: o tempo do ponto mais próximo é melhor}
5: Encaminhe o pacote pelo algoritmo HISPEAR *Shortest*;
6: **else** { SENÃO: o tempo do primeiro ponto de interceptação é melhor}
7: Encaminhe o pacote pelo algoritmo HISPEAR *Intercept*;
8: **end if**

Com base nesses dois pontos calculados, o algoritmo HISPEAR *Hybrid* escolhe aquele que resulta no menor tempo para entrega de dados (linhas 4–8) e encaminha o pacote de resposta usando o algoritmo respectivo. No algoritmo HISPEAR *Hybrid*, a trajetória do pacote de resposta também tende a ser uma linha perpendicular à trajetória do *sink*, da mesma forma que no algoritmo HISPEAR *Shortest*, como ilustra a figura 3(b).

4. Avaliação de Desempenho

4.1. Metodologia

A avaliação de desempenho foi realizada através de simulações utilizando o simulador NS-2 (versão 2.34). Os parâmetros de simulação são baseados na plataforma MicaZ, os padrão de valores destes parâmetros são mostrados na Tabela 1. Em todos os resultados, as curvas representam valores médios, enquanto as barras de erro representam intervalos de confiança para 95 % de confiança a partir de 33 instâncias independentes (sementes aleatórias). Quanto à topologia da rede, assumimos que a implantação dos nós obedece a uma grade perturbada, na qual a localização de cada nó é perturbada por um erro aleatório Gaussiano de média zero. Portanto, os nós tendem a ocupar uniformemente o campo de sensores sem formar uma grade regular.

Parâmetro	Valor
Campo de sensores	728 m × 728 m
Número de nós	576
Densidade dos nós	0,001 nós/m^2
Raio de Comunicação	50 m
Número de Vizinhos	7.6 nós
Atraso em um salto	0.1 s
Erro não determinístico	30 µs
Erro de Localização	2 m
Altura do *sink*	40 m
Velocidade do *sink*	200 km/h
Trajetória do *sink*	linha

Tabela 1. Parâmetros de simulação.

Para simular imprecisões de cálculo de posição, perturbamos a posição dos nós utilizando uma distribuição de Gaussiana com média igual ao valor da posição real do nó e um desvio padrão de 2 m [Langendoen and Reijers 2003]. Para simular imprecisões na medição de atrasos, perturbamos o atraso médio em um desvio padrão de

30 µs [Maróti et al. 2004]. Finalmente, a trajetória do nó *sink* é definida como uma linha que passa pelo meio do campo de sensores por ser uma trajetória comum para aeronaves em alta.

Em todas as simulações, cada nó sensor gera um pacote de dados de eventos detectados. A economia de transmissões é a razão entre o número de transmissões de pacotes de dados que não foram necessárias devido à agregação de dados e o numero total de pacotes de dados que seriam transmitidos sem a utilização da agregação de dados. Como o cenário deste trabalho é focado em sinks em alta velocidade (como VANTs), é realístico afirmar que a coleta de dados não será frequente e uma grande quantidade de dados será detectada. Neste caso, optamos por analisar o pior cenário, onde todos os nós da rede têm dados coletados.

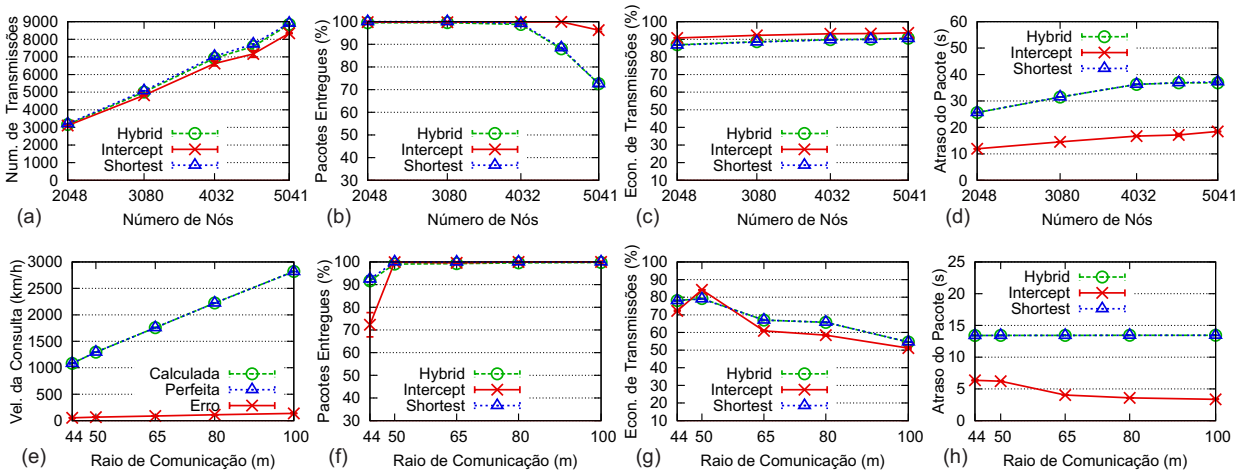


Figura 4. Escala da Rede e raio de comunicação.

4.2. O Impacto da Escala da Rede

A escalabilidade é avaliada através do aumento da quantidade de nós da rede de 2048 até 5041, mantendo mesma densidade. Assim, o campo de sensores é redimensionado de acordo com o número de nós. Quando se aumenta o tamanho da rede, a sobrecarga na rede aumenta na mesma proporção, uma vez que todos os nós sensores geram um pacote de dados para transmitir. A figura 4(a) compara o número de pacotes transmitidos enquanto aumenta o número de nós. Nesta avaliação, podemos perceber um aumento no número de pacotes transmitidos, onde o algoritmo HISPEAR *Intercept* tem a vantagem, especialmente para valores maiores 4032 nós. Para este cenário, os pacotes de resposta enviados pelos algoritmos HISPEAR *Shortest* e HISPEAR *Hybrid* têm os menores caminhos de comunicação, no entanto, o algoritmo HISPEAR *Intercept* tem um maior grau de agregação de dados resultando em menos pacotes transmitidos.

A figura 4(b) mostra que o HISPEAR *Intercept* é capaz de entregar mais que 95 % dos pacotes quando o número de nós aumenta para 5041. Entretanto, os algoritmos HISPEAR *Shortest* e HISPEAR *Hybrid* são altamente afetados pelo aumento da escala da rede, especialmente para valores maiores que 4032 nós, quando o *sink* deixa o campo de sensores antes de receber todos os pacotes de resposta destes algoritmos. Na figura 4(c), podemos notar que o algoritmo HISPEAR *Intercept* tem uma maior taxa de economia de transmissões enquanto não há diferença estatística entre os algoritmos HISPEAR *Shor-*

test e HISPEAR *Hybrid*. Relacionado com este resultado, a economia de transmissão alcançada por todas as técnicas pela agregação de dados contribui para reduzir a sobrecarga na rede, reduzindo em mais de 90% a quantidade de pacotes, contribuindo assim com a eficiência na entrega de dados e throughput. É importante destacar que que as vantagens da agregação de dados são proporcionais à quantidade de dados coletados, uma vez que diminuem proporcionalmente com a redução de eventos detectados.

Finalmente, a figura 4(d) mostra uma desvantagem dos algoritmos HISPEAR *Shortest* e HISPEAR *Hybrid*: um aumento no atraso dos pacotes. Esse comportamento era esperado, uma vez que o algoritmo HISPEAR *Shortest* espera que o nó *sink* esteja mais próximo do ponto de entrega, antes de enviar o pacote de resposta e, para este cenário, o algoritmo HISPEAR *Hybrid* foi projetado para ter o mesmo comportamento. Estes resultados, mostram claramente que o tempo de espera para que o nó *sink* esteja próximo do ponto de entrega introduz um atraso maior em relação ao tempo de espera necessário para agregar os dados coletados.

4.3. O Impacto do Raio de Comunicação

Para avaliar o impacto do raio de comunicação, aumentamos esse parâmetro de 44 m para 100 m. Ao aumentar o raio de comunicação, a velocidade dos pacotes também aumenta, conforme mostra a figura 4(e). Nesse caso, podemos notar que o algoritmo HISPEAR foi capaz de calcular com precisão esta velocidade, o que é muito importante para o funcionamento do algoritmo proposto. Depois de um raio de comunicação de 50 m, em torno de 100 % dos pacotes são entregues, como mostrado na figura 4(f). Considerando 44 m de raio de comunicação (e com o *sink* a uma altura de 40 m), podemos notar uma diminuição no número de pacotes entregues. Entretanto, os algoritmos HISPEAR *Shortest* e HISPEAR *Hybrid* ainda são capazes de entregar mais de 90 % dos pacotes, o que indica que estas variantes do algoritmo HISPEAR são mais confiáveis para menores diferenças entre a altura do *sink* e raio de comunicação dos nós.

Como ilustrado na figura 4(g), podemos notar uma diminuição na economia de transmissões quando o raio de comunicação aumenta. Esta diminuição ocorre devido ao o número de saltos para entregar os dados para o nó *sink* diminuir e, consequentemente, o número de oportunidades para efetuar agregação de dados diminui na mesma maneira. Além disso, um resultado interessante pode ser visto na figura 4(h): o atraso dos pacotes diminui no algoritmo HISPEAR *Intercept* enquanto em ambos os algoritmos HISPEAR *Shortest* e HISPEAR *Hybrid* o atraso dos pacotes permanece praticamente igual, já que os nós terão de esperar para que o nó *sink* esteja mais perto do ponto de entrega.

4.4. O Impacto do Atraso Médio de Um Salto

O atraso de um salto refere-se ao tempo de processamento exigido por nó sensor, antes de encaminhar um pacote (e.g., para calcular a posição do *sink*, o próximo salto, para agregar dados). Para avaliar o impacto desse atraso, variamos esse parâmetro de 0.1 s para 0.5 s. Como mostrado na figura 5(a), o aumento do atraso afeta o número de pacotes transmitidos, uma vez que os pacotes de resposta precisam percorrer uma distância maior para alcançar o nó *sink*. Os algoritmos HISPEAR *Shortest* e HISPEAR *Hybrid* tiveram um desempenho melhor, resultando em menores caminhos de comunicação. Como mostra a figura 5(b), quando o atraso é maior que 0.3 s, podemos notar que ambos os algoritmos HISPEAR *Intercept* e *HISPEAR Hybrid* são capazes de entregar mais pacotes, enquanto

o algoritmo HISPEAR *Shortest* é altamente afetado pelo aumento do atraso, uma vez que os pacotes de resposta de nós distantes não chegam a tempo ao nó *sink*, antes que ele saia do campo de sensores, de modo que o pacote é descartado.

Na figura 5(c), podemos ver uma desvantagem do algoritmo HISPEAR *Intercept* na economia de transmissões, uma vez que o atraso maior causa um aumento no número de pacotes de resposta enviados imediatamente para interceptar o nó *sink* sem serem agregados. Podemos notar também que economia de transmissões em ambos os algoritmos HISPEAR *Shortest* e HISPEAR *Hybrid* não é afetada pelo atraso nos pacotes. Como pode ser visto na figura 5(d), temos novamente outro interessante resultado para estes dois algoritmos em relação ao atraso introduzido, uma vez que o atraso nos pacotes permanece praticamente o mesmo. Esse comportamento também é explicado pelo tempo necessário para aguardar que o nó *sink* esteja mais perto do ponto de entrega

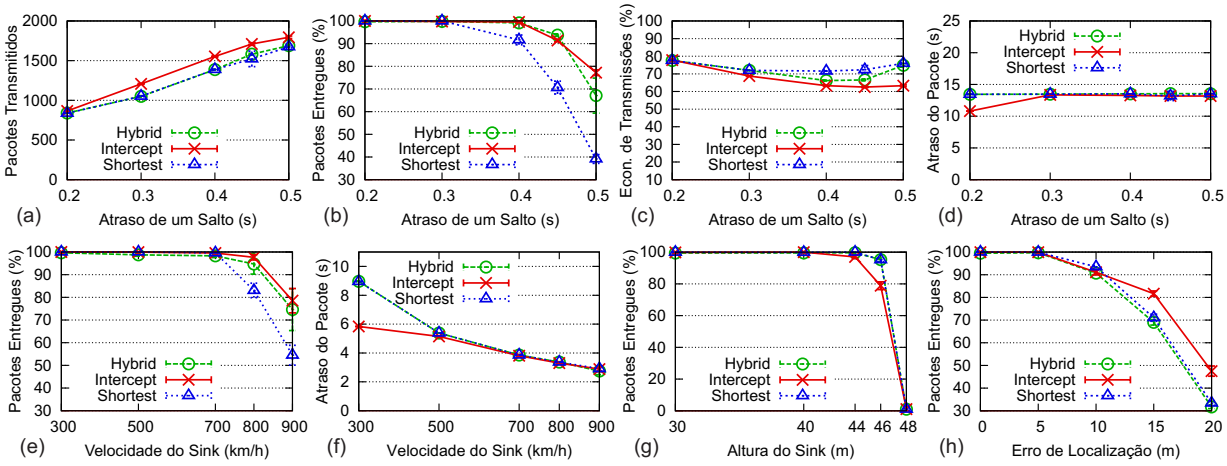


Figura 5. Atraso de um salto, velocidade do sink, altura do sink e erro de localização.

4.5. O Impacto da Velocidade e Altura do Nó *Sink*

Para avaliar o impacto da velocidade do nó *sink*, aumentamos esse parâmetro de 300 km/h (velocidade de um VANT) para 900 km/h (velocidade de um Boeing-737). Como pode-se ver na figura 5(e), em torno de 95 % dos pacotes são entregues até 700 km/h por todas as técnicas. Depois de 800 km/h, podemos notar uma diminuição no número de pacotes entregues devido ao nó *sink* deixar o campo de sensores antes de receber todos os pacotes de resposta. O algoritmo HISPEAR *Intercept* tem um melhor desempenho, enquanto o algoritmo HISPEAR *Shortest* é afetado pela velocidade do nó *sink*. Como esperado, o algoritmo HISPEAR *Hybrid* teve um comportamento semelhante ao algoritmo HISPEAR *Intercept* com uma pequena desvantagem. A figura 5(f) mostra que o atraso do pacote de resposta cai em todas as técnicas com uma vantagem para o algoritmo HISPEAR *Intercept* até 500 km/h. Esta queda no atraso dos pacotes também é explicada pelo aumento do número de pacotes enviados imediatamente em virtude do aumento da velocidade do nó *sink*.

Avaliamos o impacto da altura do nó *sink* aumentando esse parâmetro de 30 m para 48 m (nesse caso ficando com uma diferença de somente 2 m em relação ao raio de comunicação dos nós sensores). Como ilustra a figura 5(g), quando a altura do nó *sink* é superior a 44 m (apenas 6 m a menos do que o raio de comunicação dos nós) ocorre uma

queda acentuada no número de pacotes entregues, uma vez que a altura do nó *sink* fica muito próxima do limite do raio de comunicação dos nós sensores.

4.6. O Impacto do Erro de Localização

Para avaliar o impacto do erro de localização, aumentamos esse parâmetro de 0 m até 20 m. Como mostra a figura 5(h), o algoritmo HISPEAR algoritmo não é afetado de forma significativa por erros de localização, especialmente para grandes valores de erro, tais como erros até 10 m (o qual é 20% do raio de comunicação). Para esse valor, todas as técnicas são capazes de entregar mais de 90% dos pacotes.

5. Conclusão

Neste trabalho, propomos três novas técnicas de roteamento combinadas com agregação de dados, para um nó *sink* em alta velocidade, definidas como o algoritmo HISPEAR *(High Speed Aggregation and Routing)*. A ideia principal do algoritmo HISPEAR é calcular o melhor tempo disponível para agregar dados em cada salto, de maneira a entregar dados menos redundantes em direção a uma posição mais atualizada, ou mesmo uma posição futura do nó *sink*. Propomos três variantes do algoritmo HISPEAR: HISPEAR *Intercept*, HISPEAR *Shortest* e HISPEAR *Hybrid*.

O algoritmo HISPEAR *Intercept* obteve os melhores resultados em termos de escala da rede e velocidade do *sink* com um atraso menor nos pacotes. Embora não resultando nos menores caminhos de entrega de dados, ao responder as consultas para o *sink*, um elevado número de pacotes são agregados. Os algoritmos HISPEAR *Shortest* e HISPEAR *Hybrid* têm um atraso maior nos pacotes, porém ambos resultam nos menores caminhos de comunicação, com uma vantagem do algoritmo HISPEAR *Hybrid* para o número de pacotes agregados e entregues. Ainda, todas as técnicas foram capazes de alcançar significativa economia de energia e eficiente entrega de dados em diferentes cenários de RSSFs.

Os resultados são muito promissores, mas algumas limitações ainda precisam ser exploradas como trabalhos futuros. Primeiro, precisamos combinar nossa solução com um algoritmo de roteamento por superfície/face. Também pretendemos avaliar o desempenho dos algoritmos propostos utilizando algoritmos de localização reais, tais como [Boukerche et al. 2007]. Finalmente, iremos avaliar o custo computacional da solução proposta em relação modelos não lineares de trajetória do *sink* assim como o gasto energético de funções complexas de agregação de dados.

Agradecimentos

Este trabalho foi parcialmente suportado pelo CNPq, FAPESP e FAPEAM sob os processos 573963/2008-8, 08/57870-9 e 01135/2011.

Referências

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

Boukerche, A. and Nikoletseas, S. (2004). Protocols for data propagation in wireless sensor networks. pages 23–51. Plenum Press, New York, NY, USA.

Boukerche, A., Oliveira, H. A. B. F., Nakamura, E. F., and Loureiro, A. A. F. (2007). Localization systems for wireless sensor networks. *Wireless Communications, IEEE*, 14:6–12.

Estrin, D., Girod, L., Pottie, G., and Srivastava, M. (2001). Instrumenting the world with wireless sensor networks. In *ICASSP'01*, pages 2033–2036, Salt Lake City, Utah.

Faheem, Y., Boudjit, S., and Chen, K. (2009). Data dissemination strategies in mobile sink wireless sensor networks: A survey. In *2009 2nd IFIP, Wireless Days (WD)*, pages 1–6.

Hong, W. and Madden, S. (2004). Implementation and research issues in query processing for wireless sensor networks. In *ICDE '04*, page 876, Washington, DC, USA. IEEE.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., and Silva, F. (2003). Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11:2–16.

Kim, H. S., Abdelzaher, T. F., and Kwon, W. H. (2003). Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 193–204.

Krishnamachari, L., Estrin, D., and Wicker, S. (2002). The impact of data aggregation in wireless sensor networks. In *ICDCSW'02*, pages 575–578.

Langendoen, K. and Reijers, N. (2003). Distributed localization in wireless sensor networks: a quantitative comparison. volume 43, pages 499–518, New York, NY, USA. Elsevier North-Holland, Inc.

Luo, J. and Hubaux, J. P. (2010). Joint sink mobility and routing to maximize the lifetime of wireless sensor networks: The case of constrained mobility. *IEEE/ACM Transactions on Networking*, 18:871–884.

Maróti, M., Kusy, B., Simon, G., and Lédeczi, A. (2004). The flooding time synchronization protocol. In *SenSys'04*, pages 39–49, New York, NY, USA. ACM.

Oliveira, H. A. B. F., , Barreto, R. S., Fontao, A. L., Loureiro, A. A. F., and Nakamura, E. F. (2010). A novel greedy forward algorithm for routing data toward a high speed sink in wireless sensor networks. In *ICCCN'10*, pages 1–7.

Shim, G. and Park, D. (2006). Locators of mobile sinks forwireless sensor networks. In *ICPPW'06*, pages 159–164, Washington, DC, USA. IEEE.

Ye, F., Luo, H., Cheng, J., Lu, S., and Zhang, L. (2002). A two-tier data dissemination model for large-scale wireless sensor networks. In *MobiCom '02*, pages 148–159, New York, NY, USA. ACM.

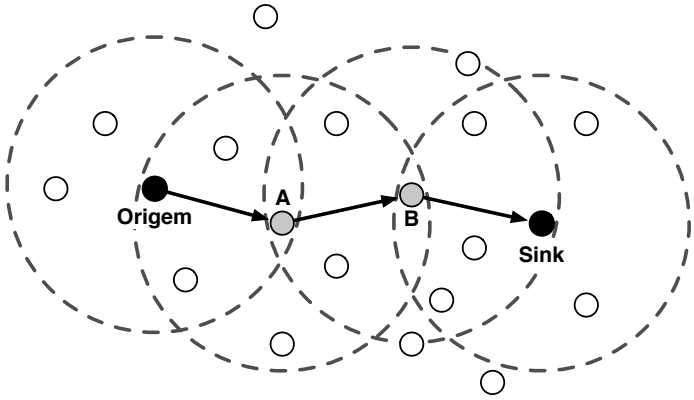
Roteamento e Agregação de Dados baseado no RSSI em Redes de Sensores Sem Fio

Moysés M. Lima[1,2], Horácio A. B. F. de Oliveira[1], Eduardo F. Nakamura[1,3] e Antônio A. F. Loureiro[4]

[1]Instituto de Computação – Universidade Federal do Amazonas
[2]Coordenação de Tecnologia da Informação – Instituto Nacional de Pesquisas da Amazônia
[3]Centro de Análise, Pesquisa e Inovação Tecnológica – FUCAPI
Manaus – Amazonas – Brasil
[4]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte – Minas Gerais – Brasil
{moyses.lima, horacio, nakamura}@icomp.ufam.edu.br, loureiro@dcc.ufmg.br

Abstract. *Geographic routing protocols have been considered as one of the most viable solution for routing in Wireless Sensor Networks (WSNs) due to their high scalability, dynamism, and high data delivery rate. These algorithms refer to nodes by position rather than address and use these coordinates to discover routes to the sink node. The main drawback of these algorithms is the need for position information of the nodes, which can be expensive in several ways. In this paper, we go further and propose the RADR (Roteamento e Agregação de Dados baseado no RSSI) algorithm, a new geographic-based routing algorithm that does not require position information while also adds data aggregation functionality to the network. Our algorithm takes advantage of the grater communication range of the sink node as well as the Received Signal Strength Indicator (RSSI) of the sensor nodes to configure routing paths and aggregation times back to the sink node. Our results indicate clearly that the proposed algorithm reduces the amount of redundant data and the number of transmissions in the network while maintaining all advantages of these kind of algorithms.*

Resumo. *Protocolos de roteamento geográficos têm sido vistos como uma das principais soluções de roteamento em Redes de Sensores Sem Fio (RSSFs) por serem escaláveis, dinâmicos e possuírem uma alta taxa de entrega de dados. Tais algoritmos referenciam a posição dos nós ao invés do endereço e utilizam essas coordenadas para descobrirem rotas para o nó sink. A principal desvantagem destes algoritmos é a necessidade de localização, que pode ser custoso em diversos aspectos. Neste trabalho, vamos além e propomos o algoritmo RADR (Roteamento e Agregação de Dados baseado no RSSI), um novo algoritmo de roteamento geográfico que não requer posição dos nós enquanto ainda provê a funcionalidade de agregação de dados na rede. Nosso algoritmo tira vantagem da possibilidade de maior alcance de comunicação do nó sink, bem como do indicador de potência do sinal recebido (RSSI) dos nós sensores para configurar rotas e tempos de agregação de dados em direção ao nó sink. Os resultados obtidos mostram claramente que o algoritmo proposto reduz a quantidade de dados redundantes e o número de transmissões na rede enquanto mantém todas as vantagens deste tipo de algoritmo.*



(a)

Figura 1. Algoritmo roteamento geográfico guloso. A cada passo, o pacote é repassado para o vizinho mais geograficamente próximo do nó *sink*. Baseado em [Park et al. 2010].

1. Introdução

Uma Rede de Sensores Sem Fio (RSSF) consiste em um conjunto de sensores, dispostos em um campo de sensoreamento em contato ou próximos a um evento ou fenômeno a ser sensoreado. Tais redes têm um grande impacto em aplicações que envolvem o monitoramento de condições ambientais tais como temperatura, luminosidade, movimento e presença de certos tipos de objetos. Nestas em na maioria das outras aplicações, os nós sensores possuem limitações de energia e largura de banda. Desta forma, técnicas de redução do consumo de energia e comunicação são necessárias em muitos cenários [Al-karaki and Kamal 2004].

Um dos objetivos principais no projeto de uma RSSF é prover a comunicação de dados entre os nós sensores e o nó *sink*. O desafio é prolongar a vida útil da rede mantendo uma qualidade na comunicação e na entrega dos dados [Akkaya and Younis 2005]. Um dos pontos chaves observados em diversos estudos é que os protocolos de roteamento em RSSFs irão diferir de acordo com a aplicação e a arquitetura da rede.

Recentemente, algoritmos que exploram informações geográficas, chamados Algoritmos de Roteamento Geográfico [Rao et al. 2003], têm sido propostos para o transporte de dados nas RSSFs por serem escaláveis, dinâmicos e possuírem uma alta taxa de entrega de dados. Estes algoritmos se baseiam na posição física dos nós da rede ao invés de seus endereços para criar rotas em direção ao nó *sink*. Dentre os algoritmos mais usados no roteamento geográfico está o algoritmo guloso de roteamento geográfico (*Greedy Geographic Routing*) [Xing et al. 2004]. Tal técnica realiza o roteamento dos pacotes através do encaminhamento do pacote ao nó vizinho mais geograficamente próximo do nó *sink* (Figura 1).

Um dos pontos fracos do roteamento geográfico é a necessidade de localização dos nós, que pode ser custoso e susceptível a grandes erros em RSSFs [Souza et al. 2010]. Além disso, tais algoritmos em geral não possuem agregação de dados, o que os tornam inviáveis para RSSFs em que grande parte dos nós possuem dados a serem enviados ao *sink* [Al-karaki and Kamal 2004]. Desta forma, neste trabalho propomos um novo algoritmo de roteamento geográfico com agregação de dados que não requer informações de posicionamento dos nós: O RADR (Roteamento e Agregação de Dados baseado no

RSSI). A ideia principal do algoritmo proposto é considerar a possibilidade de se equipar o nó *sink* com um dispositivo de comunicação de maior potência para alcançar todos os nós da rede em um único salto. Entretanto, a resposta/retorno dos nós sensores para o nó *sink* é realizada através de múltiplos saltos utilizando nossa abordagem de roteamento guloso que tira proveito do RSSI (*Received Signal Strength Indicator*) para calcular o próximo salto do roteamento em direção ao nó *sink* bem como para calcular o tempo de espera para possibilitar a agregação de dados.

O restante deste trabalho está organizado como segue. Na Seção 2, apresentamos os trabalhos relacionados bem como uma classificação dos algoritmos estudados. A Seção 3 descreve o algoritmo RADR, cuja avaliação de performance é mostrada na Seção 4. Na Seção 5, realizamos uma breve discussão sobre a aplicabilidade do algoritmo proposto e, finalmente, na Seção 6 apresentamos nossas conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Os protocolos de roteamento geográficos podem ser divididos em três categorias: (1) baseados em coordenadas geográficas; (2) baseados em coordenadas virtuais; e (3) livres de posicionamento.

Na primeira categoria, os protocolos requerem uma estimativa da localização global dos nós (latitude/longitude). Tal estimativa deve ser obtida (1a) equipando-se todos os nós com receptores GPS (*Global Positioning System*); (1b) ou através da execução de um algoritmo de localização; (1c) ou mesmo através do posicionamento manual dos nós [Boukerche et al. 2007]. Nesta categoria, o *Bounded Voronoi Greedy Forwarding (BVGF)* [Xing et al. 2004], é um algoritmo de localização que realiza decisões de encaminhamento guloso baseadas na localização dos vizinhos a um salto de distância. Neste algoritmo, uma rede é modelada através de um diagrama de Voronoi onde as regiões representam as localizações dos sensores. Cada nó conhece sua posição geográfica e mantém uma tabela com os endereços dos nós vizinhos e, para manter esta tabela, cada nós divulga periodicamente, uma mensagem de *broadcast* que inclui sua localização, bem como a localizações dos vértices das regiões de Voronoi. Já no *Geographic Random Forwarding (GeRaF)* [Zorzi and Rao 2003], quando um nó possui um pacote a ser enviado, este envia o dado através de uma mensagem de *broadcast*, contendo sua própria localização e a localização do destino pretendido para só então utilizar o método de transmissão pelo menor esforço. No algoritmo *Energy Aware Greedy Routing (EAGR)* [Haider et al. 2007], cada nó conhece sua localização geográfica e a de seus vizinhos, seu nível de energia e de seus vizinhos e executa uma decisão local para a escolha do próximo salto não somente baseado na localização dos nós, mas também nos seus níveis de energia, de modo a aumentar o tempo de vida da rede como um todo.

Na segunda categoria estão os algoritmos baseados em coordenadas virtuais. Esses algoritmos tentam criar um outro tipo de sistema de coordenadas, que não está relacionado ao sistema de posicionamento global, onde a informação de localização não está disponível [Rao et al. 2003]. Neste cenário, o *Greedy Minimum energy consumption Forwarding Protocol (GMFP)* [Panigrahi et al. 2009] foi concebido com o objetivo de aumentar o tempo de vida da rede, através da escolha ótima de um nó elegível em cada salto, utilizando encaminhamento geográfico guloso.

O algoritmo *Greedy Forwarding with Virtual Position (GF-ViP)* [You et al. 2009,

Takagi and Kleinrock 1984], utiliza o esquema de encaminhamento guloso baseado no progresso do encaminhamento em direção ao *sink*. O algoritmo foi proposto com dois complementos, o *Virtual Multi-Level Position (MVP)* e o *Greedy Forwarding with Virtual Hierarchical Position (HVP)*. Cada algoritmo possui duas variações que utilizam algoritmos Gulosos baseado em progresso *Most Forwarding Within Radius (MFR)*. A principal vantagem nestas abordagens é a aplicação do encaminhamento guloso durante todo o processo de roteamento, gerando resultados altamente eficientes na geração de rotas. Ainda nesta categoria, o *Greedy Embedding Spring Coordinates (GSpring)* [Leong et al. 2007] gera coordenadas virtuais úteis e viáveis, com o objetivo de gerar rotas eficientes e detectar buracos de cobertura. A abordagem usa um algoritmo de detecção de perímetro para identificar nós no perímetro da rede e usa essa informação para atribuir coordenadas iniciais.

Finalmente, na terceira categoria, os nós não precisam conhecer suas informações de localização nem as de seus vizinhos mas utilizam alguma outra informação para indicar a direção que o pacote deve seguir. Neste cotexto, o *Greedy Forward with Received Signal Strength Indicator (GF-RSSI)* [Pham et al. 2006] utiliza a potência do RSSI recebido como um filtro usado para gerar rotas. Já no *Received Signal Strength Routing (RSSR)* [Boukerche et al. 2008], algoritmo que serviu de base para o proposto no presente trabalho, o *sink* também é equipado com um dispositivo de comunicação de maior potência e seu pacote de consulta atinge todos os nós da rede em um único salto. O pacote de resposta dos nós sensores é encaminhado para o vizinho que recebeu a mensagem do *sink* com a maior potência que, na teoria, é o vizinho mais próximo do *sink*.

Nossa proposta difere das duas primeiras categorias, pois nossa abordagem não necessita de informações sobre localização dos nós. Assim como o algoritmo *RSSR*, a ideia principal do algoritmo proposto no presente trabalho é aproveitar a grande capacidade do nó *sink* e equipá-lo com um dispositivo de comunicação de maior potência. Entretanto, nossa abordagem se diferencia desta última por implementar um esquema de agregação de dados durante o envio das mensagens. Atualmente não existem soluções que explorem essa combinação de capacidades. Um resumo dos artigos citados é mostrado na Tabela 1.

3. RADR - Roteamento e Agregação de Dados baseado no RSSI

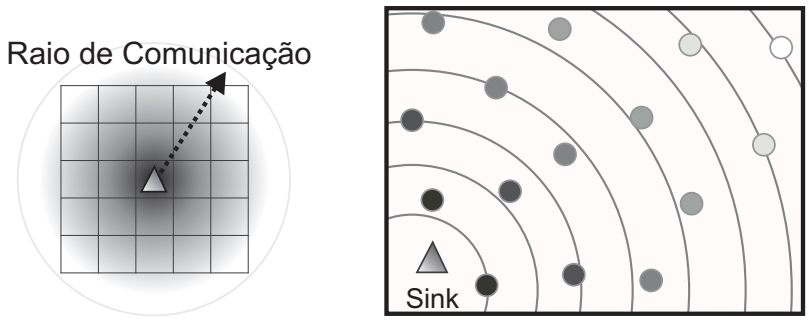
Neste artigo, está sendo proposto um novo algoritmo de roteamento geográfico com agregação de dados para redes de sensores sem fio que não requer informações de posicionamento dos nós: o RADR (Roteamento e Agregação de Dados baseado no RSSI). A proposta do algoritmo é possuir todas as vantagens de um algoritmo de roteamento geográfico tais como robustez, escalabilidade, dinamicidade e entrega confiável dos dados, mas sem precisar das informações de posicionamento dos nós ao mesmo tempo que provê o recurso de agregação de dados na rede.

A presente proposta considera a possibilidade de se equipar o nó *sink* com um dispositivo de comunicação de maior potência. Desta forma, em um único salto, o nó *sink* pode enviar uma consulta para toda a rede. A ideia desta arquitetura de comunicação surgiu a partir do projeto ATTO (*Amazonian Tall Tower Observatory*), que já está implementado na Amazônia e consiste em uma torre no meio da floresta com uma estrutura física de 320 m de altura e alta capacidade de comunicação. O foco do projeto é mapear

Tabela 1. Comparação dos algoritmos de roteamento geográficos.

	A	B	C	D	E
BVGF [Xing et al. 2004], [Xing 2006]		√			
EAGR [Haider et al. 2007]		√		√	
GeRaF [Zorzi and Rao 2003]		√			
GF-RSSI [Pham et al. 2006]	√	√	√		
GF-ViP [You et al. 2009]	√				
GMFP [Panigrahi et al. 2009]	√			√	
GSpring [Kermarrec and Tan 2010]	√				
RSSR [Boukerche et al. 2008]	√	√	√		
RADR	√	√	√	√	√

A) Não requer coordenadas globais.
B) Não requer coordenadas virtuais.
C) Utiliza RSSI como métrica
D) Ciente de energia.
E) Realiza agregação de dados.



(a) (b)

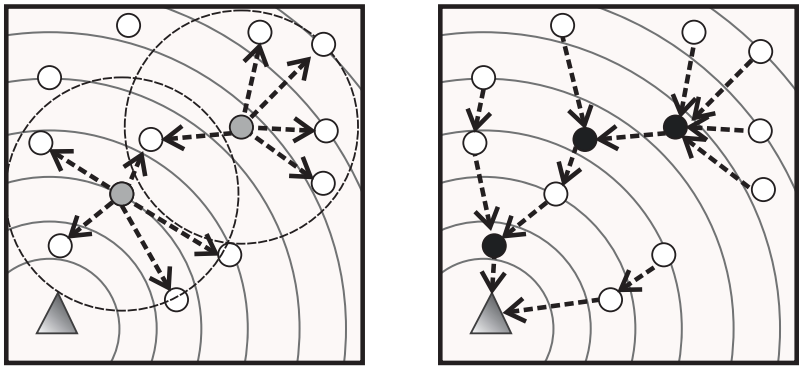
Figura 2. Potência do sinal decrescendo à medida que a mensagem de consulta do *sink* se propaga na rede. Nós sensores localizados mais distantes do *sink* receberão a mensagem com RSSI menor que os nós mais próximos. Baseado em [Boukerche et al. 2008].

eventos na região e fornecer medidas confiáveis de fontes e sumidouros de gases de efeito estufa como CO_2, CH_4 e N_2O [Tollefson 2010].

Na arquitetura proposta, a mesma mensagem de consulta do nó *sink* alcançará os nós da rede com diferentes potências (RSSIs), sendo que os nós mais distantes possuirão um sinal mais baixo, ao contrário dos nós mais próximos, que irão receber a mensagem com um sinal mais forte, conforme ilustrado nas Figuras 2(a) e 2(b).

Neste trabalho, o RSSI foi utilizado para estimar a distância entre os nós regulares e o *sink*. A aferição é baseada na potência do sinal recebido, de forma que nós mais distantes do *sink* receberam menor valor de RSSI (devido à propagação do sinal). A confiabilidade do RSSI, utilizado como métrica, é discutida em [Holland et al. 2006] e [Jacinto 2012].

O algoritmo proposto (definido no Algoritmo 1) inicia quando o nó *sink* envia uma consulta para toda a rede (linha 9). Os nós sensores, ao receberem essa consulta, estimam



(a) (b)

Figura 3. Exemplo do funcionamento do algoritmo RADR.

suas distâncias ao nó *sink* usando a técnica de RSSI (linhas 10 e 11). Em seguida, cada nó irá enviar a seus vizinhos um pacote de anúncio, que contém a sua distância estimada para o *sink* (linha 12). Ao mesmo tempo, os nós receberão diversos pacotes deste tipo, um de cada vizinho, e salvarão tais dados de distâncias nas suas tabelas de roteamento (linha 18 e 19, ilustrado na Figura 3(a)). Em seguida, cada nó irá verificar se possui dados a serem retornados à consulta do *sink*. Se tiver, tais dados serão adicionados à lista de dados agregados (linhas 13-15). O próximo passo do algoritmo é calcular um tempo de espera para agregação dos dados antes de enviar a resposta ao *sink*. Um temporizador é iniciado com o tempo de espera calculado (linhas 16 e 17). O tempo para o envio dos pacotes é definido de acordo com a potência do RSSI recebido por cada nó, ou seja, quanto mais baixo o sinal do RSSI, menos tempo o nó terá que esperar para enviar seus pacotes em direção ao *sink*.

Quando o temporizador de um nó expira (linha 20), ele prepara um pacote com todos os dados agregados e envia para o vizinho mais próximo do nó *sink* (linhas 20-23). Este pacote será recebido pelo vizinho escolhido (linha 24) que, provavelmente, não terá seu temporizador expirado ainda, por estar mais próximo do *sink*. Este último nó irá simplesmente agregar os dados recebidos em sua lista de dados agregados e continuar esperando que seu temporizador expire para enviar esses dados adiante (linha 26). Finalmente, quando o *sink* recebe algum pacote de resposta, ele envia os dados recebidos de volta à central de monitoramento (linhas 27-29).

Como este procedimento é executado em toda a rede, em vários casos nós vizinhos poderão escolher o mesmo nó de encaminhamento, ou seja, o vizinho eleito como mais próximo do *sink*. Desta forma, o nó de encaminhamento irá receber pacotes de vizinhos distintos, agregar as informações recebidas e repassar apenas um único pacote agregado ao próximo nó de encaminhamento. Este procedimento é repetido até que o pacote atinja o *sink*, conforme ilustrado na Figura 3(b). Desta forma, toda a rede pode responder a mensagem enviada pelo *sink* usando o esquema de comunicação multi-saltos e com agregação de dados.

Com o objetivo de visualizar e comparar as rotas geradas pelos algoritmos, a Figura 4(a) apresenta um gráfico de rotas gerado pela árvore inversa, gerada pelo algoritmo *Flooding+Agregação*, enquanto a Figura 4(b), ilustra o gráfico de rotas gerado pelo algoritmo *RADR*. Os gráficos foram gerados a partir de uma simulação realizada e

Algoritmo 1 Algoritmo do RADR.

Variáveis:
1: n_i; {Id do nó atual. Para o nó *sink*, $n_i = 0$}
2: $tabelaVizinhos_i = \emptyset$; {Distâncias dos vizinhos ao *sink*}
3: $dadosAgregados_i = \emptyset$; {Dados agregados para serem enviados}
4: $dadosConsulta_i$; {Dados obtidos a partir da consulta do *sink*}
5: $tempoEnvio_i$; {Tempo de espera para o envio da resposta}
6: $temporizador_i$; {Temporizador de espera para envio da resposta}
7: $proximoSalto_i$; {Próximo salto para enviar os dados agregados}

Evento:
8: *Sink* recebe da central de monitoramento uma requisição: $requisicao(consultaId_k, consulta_k)$;

Ação:
9: Envia $consulta(consultaId_k, consulta_k)$; {*Sink* envia a consulta para todos os nós}

Evento:
10: $msg_i = consulta(consultaId_k, consulta_k)$; tal que {Nó recebe consulta do *sink*}
11: $dist_i = RSSI(msg_i)$; {Calcula distância ao *sink* com base no RSSI}
Ação:
12: Envia $anuncio(n_i, dist_i)\ \forall\ n_j \in vizinhos_i$. {Nó envia pacote de anúncio com sua distância}
13: **Se** $dadosConsulta_i = executaConsulta(consulta_k)$ **Então** {Tenho dados para responder ao *sink*?}
14: $dadosAgregados_i := dadosAgregados_i \cup dadosConsulta_i$; {Atualiza dados agregados}
15: **Fim Se**
16: $tempoEnvio_i = estimaTempoEnvio(dist_i)$; {Configura o tempo de espera para agregação}
17: $temporizador_i.inicia(tempoEnvio_i)$; {Inicia o temporizador de espera}

Evento:
18: $msg_i = anuncio(vizinho_k, dist_k)$; {Recebido pacote de anúncio}
Ação:
19: $tabelaVizinhos_i = tabelaVizinhos_i \cup (vizinho_k, dist_k)$; {Adiciona vizinho na lista}

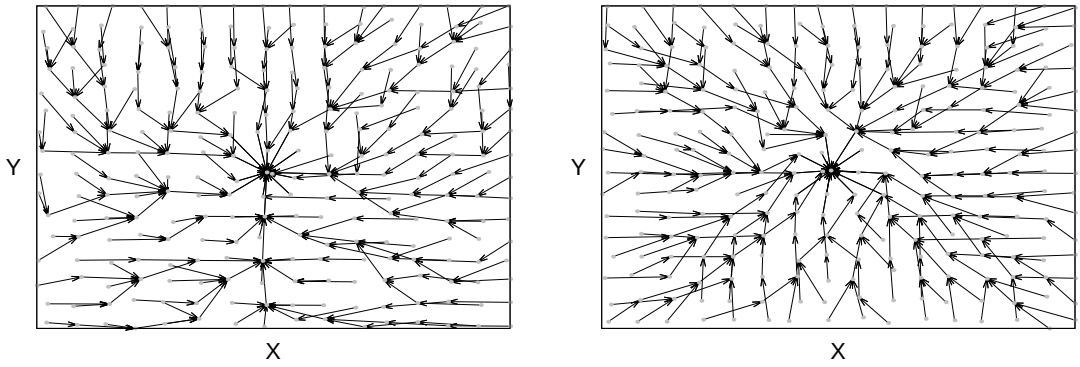
Evento:
20: $temporizador_i$ expirado; {Acabou o tempo de agregação. Vamos enviar a resposta}
Ação:
21: $proximoSalto_i = vizinhoMaisPertoDoSink()$; {Próximo salto será o vizinho mais perto do *sink*}
22: Envia $resposta(dadosAgregados_i)$ para $proximoSalto_i$; {Envia dados agregados}
23: $dadosAgregados_i = \emptyset$; {Limpa a lista de dados agregados}

Evento:
24: $msg_i = resposta(dadosAgregados_k)$; {Recebi um pacote com dados agregados de um vizinho}
Ação:
25: **Se** $n_i \neq 0$ **Então** {Sou um nó normal?}
26: $dadosAgregados_i := dadosAgregados_i \cup dadosAgregados_k$; {Atualiza meus dados agregados}
27: **Senão** {Sou o nó *sink*}
28: Envia os dados recebidos para a central de monitoramento.
29: **Fim Se**

é possível ver claramente os caminhos de roteamento guloso formados pela abordagem proposta usando apenas as informações de RSSI.

4. Avaliação de Performance

Na presente seção, avaliamos o algoritmo RADR proposto e comparamos sua performance com dois outros algoritmos: o *RSSR Selection* [Boukerche et al. 2008] e o *Flooding+Agregação*. O *RSSR Selection* serviu de base para a construção da nossa abordagem, mas não implementa agregação de dados. Já o *Flooding+Agregação* é um algo-



(a) Flooding+Agregação (b) RADR

Figura 4. Fluxo de pacotes encaminhados ao *sink*

ritmo básico de agregação de dados que usa o *flooding* clássico para divulgar a consulta do *sink* em múltiplos saltos e o retorno dos nós é feito usando a árvore inversa do *flooding* e realizando agregação de dados nos nós que possuem filhos.

A ideia de se utilizar esses dois algoritmos como base de comparação é que o primeiro permitirá avaliar o avanço do nosso novo algoritmo ao se adicionar a nova técnica de agregação de dados, enquanto que o segundo permitirá avaliar o ganho do algoritmo proposto em relação a outro que já faz agregação. O foco de nossa avaliação está na aplicação da nova técnica de encaminhamento guloso bem como na técnica de agregação de dados, que é baseada na métrica para o cálculo do tempo de espera para agregação. Espera-se também entender o funcionamento do algoritmo proposto nos mais diversos cenários.

4.1. Metodologia

A avaliação de performance foi realizada através de simulações usando o simulador Sinalgo [ETH-Zurich 2012], que provê um ambiente completo para simulação de algoritmos distribuídos. As simulações foram realizadas em um campo de sensoreamento de $130\,\text{m} \times 130\,\text{m}$. O posicionamento dos nós foi feito com base em uma grade perturbada de forma a evitar buracos na rede e mantendo a densidade de $0.03\,\text{nós/m}^2$. Os parâmetros de simulação foram baseados no sensor MicaZ e os valores utilizados são mostrados na Tabela 2.

Tabela 2. Cenário padrão de simulação.

Parâmetro	Valor padrão
Área monitorada	$130\,\text{m} \times 130\,\text{m}^2$
Quantidade de nós	512 nós (grade perturbada)
Densidade	$0.03\,\text{nós/m}^2$
Alcance da comunicação	40 m
Imprecisão do RSSI a curta distância	5 % da dist. real
Posição do Sink	Centro da área monitorada

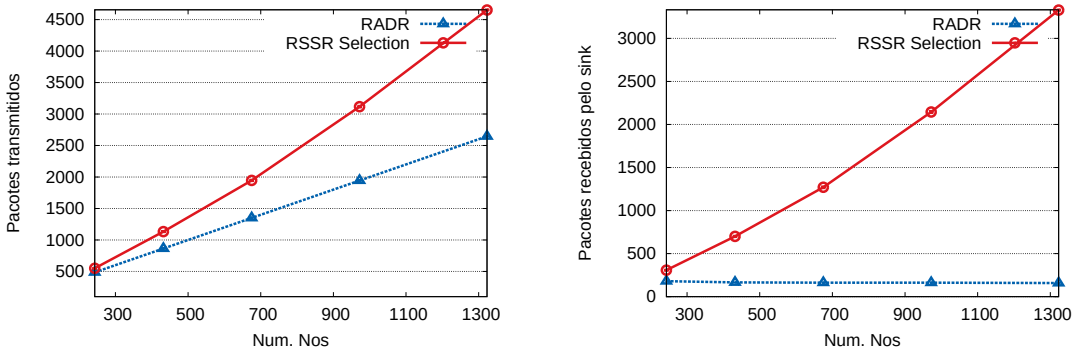


Figura 5. Resultados obtidos ao se avaliar o impacto da escalabilidade da rede.

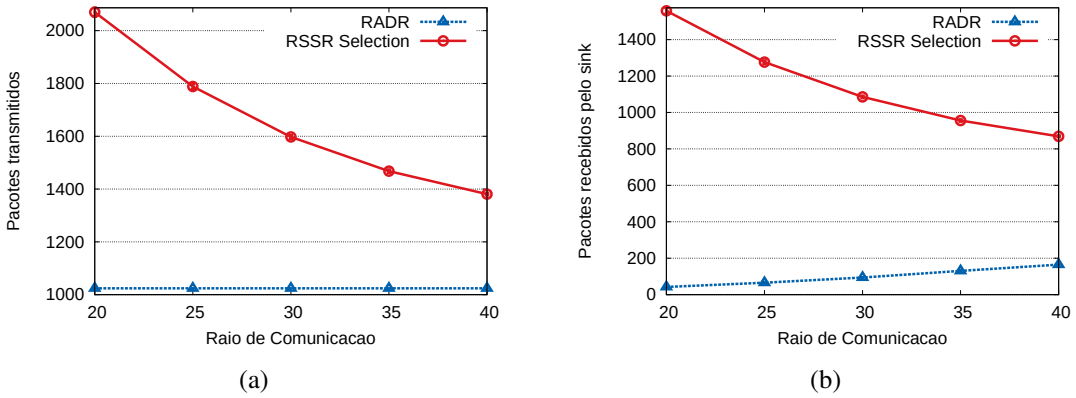
4.2. Impacto da Escalabilidade da Rede

A escalabilidade de rede foi avaliada através do aumento do número de nós na rede de 243 para 1343 nós, aumentando-se o tamanho da área monitorada de modo a manter a densidade constante. Em todos os casos, a taxa de entrega dos dados ao *sink* ficou acima de 95% e, portanto, não mostraremos o gráfico. Entretanto, pudemos confirmar que a solução proposta mantém a boa confiabilidade obtida pelos outros algoritmos estudados. Já o gráfico da Figura 5(a) compara a quantidade de pacotes transmitidos na rede. Pode-se perceber que o número de transmissões do algoritmo proposto cresce muito mais lentamente do que o *RSSR* ao se aumentar o número de nós na rede, o que mostra que a nova técnica de agregação de dados proposta pelo RADR é capaz de gerar grandes reduções no consumo de energia.

O gráfico da Figura 5(b) mostra a quantidade de pacotes recebidos pelo nó *sink*. Neste resultado, fica evidente que o número de pacotes recebidos pelo *sink* permanece praticamente constante mesmo ao aumentar bastante o número de nós, mostrando novamente a viabilidade da técnica proposta de agregação de dados. Neste contexto, o número de pacotes entregues se limita ao número de vizinhos do nó *sink* que possuem o mesmo raio de comunicação. Nestes e nos resultados seguintes, até que seja especificado o contrário, o algoritmo de agregação de dados *Flooding+Agregação* obteve os mesmos resultados do RADR, confirmando o bom funcionamento deste último e, portanto, tais resultados não foram colocados nos gráficos. Mais para o final da avaliação, mostraremos em quais aspectos o nosso algoritmo se destaca em relação ao *Flooding+Agregação*.

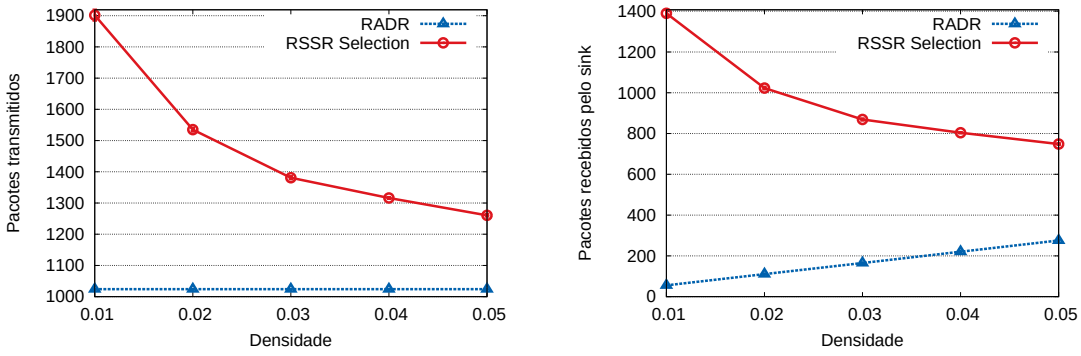
4.3. Impacto do Raio de Comunicação

Para avaliarmos o impacto do raio de comunicação dos nós em nosso algoritmo, começamos com um raio relativamente pequeno, de 20 m, e o aumentamos para 40 m. Raios de comunicação pequenos reduzem o número de nós vizinhos, o que consequentemente reduz o número de possibilidades para o próximo salto [Boukerche et al. 2008]. Na Figura 6(a), é possível observar que o RADR consegue manter o número de pacotes transmitidos na rede bem baixo, mesmo quando há baixa densidade de nós, o que poderia dificultar a agregação de dados. Podemos observar ainda a grande economia no número de transmissões em relação ao *RSSR* que, ao se aumentar o raio de comunicação, transmite menos pacotes, uma vez que os pacotes serão entregues diretamente ao *sink*.



(a) (b)

Figura 6. Resultados obtidos ao se avaliar o impacto do raio de comunicação dos nós.



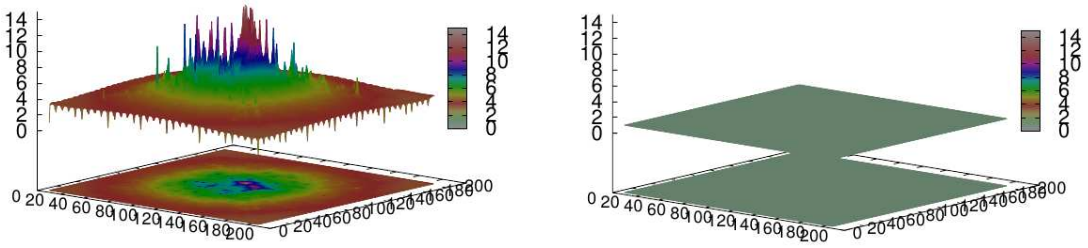
(a) (b)

Figura 7. Resultados obtidos ao se avaliar o impacto da densidade da rede.

Já no gráfico da Figura 6(b), é possível perceber que mesmo quando aumentamos o raio de comunicação entre os nós, o número de pacotes recebidos pelo *sink* ainda é expressivamente menor em nosso algoritmo que, neste caso, tem o número de vizinhos diretos do *sink* aumentado e, portanto, maior possibilidade de envio direto dos pacotes (mas com menos pacotes agregados).

4.4. Impacto da Densidade da Rede

O impacto da densidade da rede em nosso algoritmo foi avaliado começando com uma densidade relativamente baixa de 0.01 nós/m^2 e aumentando tal densidade até 0.05 nós/m^2. Em densidades baixas, um nó terá poucos vizinhos, dificultando a agregação, enquanto que ao se aumentar a densidade, o número de vizinhos de cada nó aumentará. O gráfico da Figura 7(a) mostra a quantidade de pacotes transmitidos na rede ao se aumentar a densidade. Neste gráfico, é possível observar que mesmo a densidades baixas o nosso algoritmo continua funcionando e que ele é capaz de funcionar de forma bem mais eficiente que o *RSSR Selection*. Neste contexto, o número de pacotes recebidos pelo *sink* também foi avaliado. Na Figura 7(b), é possível observar o aumento do número de pacotes chegando no *sink* com uma agregação menor, uma vez que o *sink* terá mais vizinhos que enviarão os pacotes diretamente para ele.



(a) Algoritmo RSSR Selection. (b) Algoritmo RADR.

Figura 8. Quantidade de pacotes transmitidos em relação à posição física dos nós.

4.5. Tempo de Vida da Rede e o Problema de Esgotamento de Energia dos Nós Próximos ao *Sink*

O gráfico da Figura 8(a) mostra a quantidade de pacotes transmitidos por um nó em relação à sua posição física após executar o *RSSR Selection*. Como pode-se observar, os nós mais próximos do *sink* tendem a enviar muitos pacotes, uma vez que estes devem enviar não só os seus dados para o *sink* mas também retransmitir os pacotes dos nós mais distantes. Tal comportamento resulta no esgotamento mais rápido da energia dos nós mais próximos do *sink*, chegando a um momento em que o *sink* não terá mais nenhum vizinho. Por outro lado, os nós mais distantes ainda estarão com uma grande quantidade de energia mas não poderão enviar seus dados, uma vez que não haverá nós para retransmiti-los para o *sink*.

Já o gráfico da Figura 8(b) mostra o comportamento do RADR, proposto neste trabalho. Como pode-se observar, devido à agregação de dados, os nós da rede como um todo trocam quase a mesma quantidade de pacotes, de modo que quando a energia de um nó estiver acabando em virtude das trocas de pacotes, é provável que toda a rede deva estar no mesmo nível de energia. Tal característica resolve o problema do esgotamento de energia dos nós próximos ao *sink* e ainda resulta na otimização do tempo de vida da rede.

4.6. Acertos na Escolha do Próximo Salto

Até o momento, nos resultados apresentados, o algoritmo proposto obteve o mesmo desempenho que o algoritmo básico de agregação de dados *Flooding+Agregação*, de forma que este último não foi mostrado nos gráficos. Tal característica é um ponto positivo para a nossa abordagem, uma vez que o *flooding* é um dos algoritmos mais confiáveis em RSSFs. Entretanto, apesar de sua confiabilidade, o *flooding* não garante os menores nem os melhores caminhos na rede e muito menos a criação de árvores de escoamento de qualidade. O RADR, por outro lado, é capaz de gerar árvores de escoamento muito mais próximas da ótima, uma vez que este se baseia em distâncias ao *sink*.

Para avaliarmos esse aspecto, os gráficos das Figuras 9(a) e 9(b) mostram a quantidade de acertos na escolha do próximo salto de um pacote. Um acerto se dá quando um pacote sai de um nó em direção ao seu vizinho que está fisicamente mais próximo do nó *sink*, enquanto que um erro se dá quando este pacote foi enviado para um vizinho que não é o mais próximo do *sink*, ou seja, ele poderia ter sido enviado para um vizinho melhor. Quanto maior a taxa de acerto, melhor será a agregação e o caminho dos pacotes em direção ao *sink*.

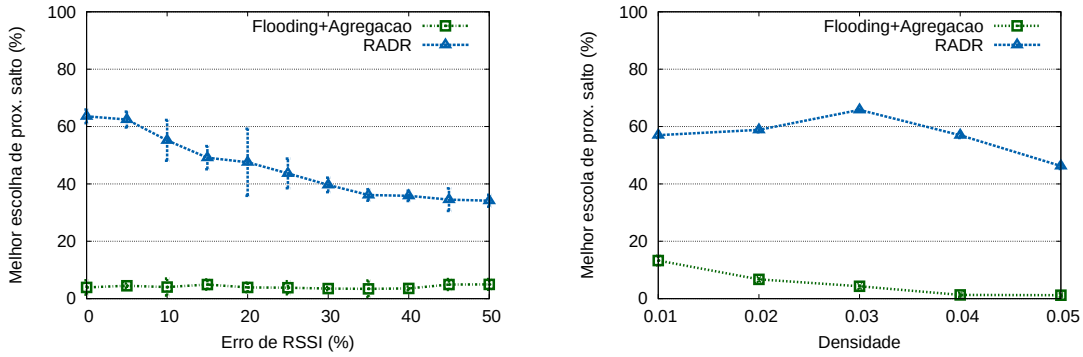


Figura 9. Resultados da taxa de acertos na escolha dos próximos saltos dos pacotes.

No gráfico da Figura 9(a) é possível observar o comportamento da escolha do próximo vizinho ao se aumentar o erro de RSSI, uma vez que este é um dos fatores mais diretamente relacionados à escolha do vizinho em nossa abordagem. Como pode-se observar, a taxa de acerto obtida pelo RADR é muito superior à do *Flooding+Agregação*, mostrando que nossa abordagem possui todas as vantagens deste último, mas obtendo uma árvore de agregação muito mais eficiente. É possível observar ainda a baixa qualidade da árvore obtida pelo *Flooding+Agregação*, o que já era esperado. Finalmente, vê-se que a qualidade do RADR cai ao se aumentar o erro de RSSI mas, em todos os casos, ficando bem acima do *Flooding+Agregação*.

Já o gráfico da Figura 9(b) mostra o comportamento da mesma taxa ao se aumentar a densidade da rede. É possível observar que nos dois algoritmos, a taxa de acerto é maior a baixas densidades, uma vez que há uma quantidade menor de vizinhos a serem escolhidos.

5. Aplicabilidade da Solução Proposta

Neste trabalho consideramos um nó *sink* equipado com um dispositivo de comunicação potente, de forma que em um único salto seja possível alcançar todos os nós regulares na rede. Nossa abordagem é aplicável em diversos cenários, dentre eles, o projeto ATTO (*Amazonian Tall Tower Observatory*), para o qual foi proposto. Tal projeto consiste em uma torre no meio da floresta amazônica, com uma estrutura física de 320 metros de altura, somente comparável à torre *Eiffel*, cujo foco é mapear eventos da região e fornecer medidas confiáveis de fontes e sumidouros de gases de efeito estufa como CO_2, CH_4 e N_2O [Tollefson 2010]. A torre, a primeira desta natureza no cenário amazônico, está instalada na Reserva de Desenvolvimento Sustentável do Uatumã, no interior do Amazonas. Neste contexto, o algoritmo RADR é proposto como uma solução de encaminhamento guloso utilizando agregação de dados para os nós sensores em volta da torre, que servirá como *sink*. Ao se aproveitar a comunicação de alto alcance do *sink*, nossa solução evita a dependência de um *flooding* inicial, que tem comportamento ruim em ambientes com baixa qualidade de comunicação, como é o caso na região amazônica.

6. Conclusão

Neste trabalho, propomos um novo protocolo de roteamento geográfico com agregação de dados para RSSFs: o RADR (Roteamento e Agregação de Dados baseado no RSSI). Tal

protocolo parte do princípio de que o nó *sink* possui uma capacidade alta de comunicação de modo a mandar uma única mensagem alcançando todos os nós da rede. Com base apenas no RSSI da mensagem do *sink* chegando em cada um dos nós, propomos um novo e inovador esquema de roteamento e agregação de dados. Nosso algoritmo de agregação de dados utiliza um temporizador para esperar pelos pacotes a serem agregados que também é feito com base na mesma informação de RSSI.

Uma série de experimentos foram realizados para avaliar o protocolo proposto e compará-lo com dois outros algoritmos da literatura: o *RSSR Selection* e o *Flooding+Agregação*. O primeiro utiliza um esquema de encaminhamento baseado no RSSI semelhante ao nosso, mas não faz agregação de dados. Enquanto que o segundo faz agregação de dados mas é baseado na árvore de escoamento obtida a partir de um *flooding*. Os resultados obtidos mostram claramente os benefícios introduzidos pelo esquema de agregação de dados proposto bem como a qualidade superior da árvore de agregação obtida em relação à obtida por um *flooding* clássico.

Apesar de obtermos resultados satisfatórios com vantagens importantes, algumas limitações devem ser exploradas em trabalhos futuros, como, por exemplo, a combinação de nossa solução com algoritmos de reconhecimento de perímetro e desvio de buracos que sejam também baseadas apenas nas informações disponíveis de RSSI, algo ainda não explorado na literatura.

Referências

Akkaya, K. and Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3:325–349.

Al-karaki, J. N. and Kamal, A. E. (2004). Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications*, 11:6–28.

Boukerche, A., Oliveira, H., Nakamura, E., and Loureiro, A. (2007). Localization systems for wireless sensor networks. *Wireless Communications, IEEE*, 14(6):6 –12.

Boukerche, A., Oliveira, H., Nakamura, E., and Loureiro, A. (2008). A novel location-free greedy forward algorithm for wireless sensor networks. In *Communications, 2008. ICC '08. IEEE International Conference on*.

ETH-Zurich, D. C. G. a. (2012). Sinalgo - simulator for network algorithms. In *http://dcg.ethz.ch/projects/sinalgo/*.

Haider, R., Javed, M., and Khattak, N. (2007). Eagr: Energy aware greedy routing in sensor networks. In *Future Generation Communication and Networking (FGCN 2007)*, volume 2, page 344349.

Holland, M., Aures, R., and Heinzelman, W. (2006). Experimental investigation of radio performance in wireless sensor networks. In *Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on*, pages 140 –150.

Jacinto, R. M. P. (2012). Modelação da Propagação numa Rede de Sensores sem Fios. Master's thesis, Universidade Nova de Lisboa.

Kermarrec, A.-M. and Tan, G. (2010). Greedy geographic routing in large-scale sensor networks: a minimum network decomposition approach. In *Proceedings of the ele-*

venth ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '10, pages 161–170, New York, NY, USA. ACM.

Leong, B., Liskov, B., and Morris, R. (2007). Greedy virtual coordinates for geographic routing. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 71 –80.

Panigrahi, B., De, S., and Sun Luk, J.-D. (2009). A greedy minimum energy consumption forwarding protocol for wireless sensor networks. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1 –6.

Park, E., Bae, D., and Choo, H. (2010). Energy efficient geographic routing for prolonging network lifetime in wireless sensor networks. In *Proceedings of the 2010 International Conference on Computational Science and Its Applications*, ICCSA '10, pages 285–288, Washington, DC, USA. IEEE Computer Society.

Pham, N., Youn, J., and Won, C. (2006). A comparison of wireless sensor network routing protocols on an experimental testbed. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 2, pages 276 –281.

Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., and Stoica, I. (2003). Geographic routing without location information. In *MOBICOM'03*, pages 96–108. ACM Press.

Souza, E. L. d., Nakamura, E. F., and Oliveira, H. A. B. F. d. (2010). Uma abordagem de fusao de dados em redes de sensores para reduzir o impacto de erros de localizacao em algoritmos de rastreamento. In *28o Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos (SBRC'2010)*, pages 291–304.

Takagi, H. and Kleinrock, L. (1984). Optimal transmission ranges for randomly distributed packet radio terminals. *Communications, IEEE Transactions on*, 32(3):246 – 257.

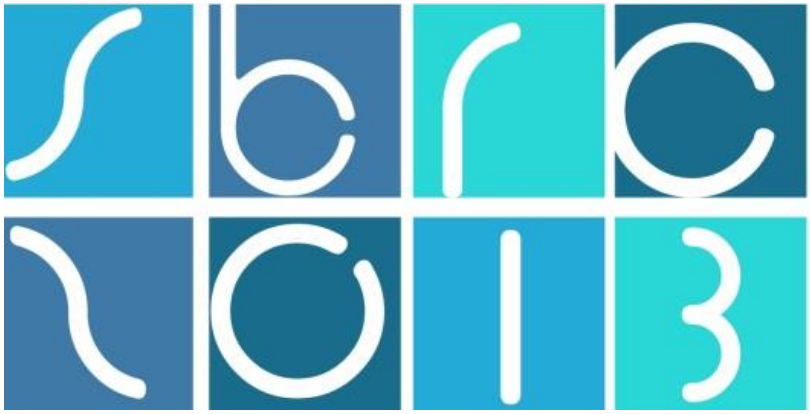
Tollefson, J. (2010). A towering experiment an ambitious project to track greenhouse gases from a perch high above the amazon forest will provide crucial data - but only if scientists can get it built. *Nature*, 467(386).

Xing, G. (2006). *Unified power management in wireless sensor networks*. PhD thesis, St. Louis, MO, USA. AAI3238702.

Xing, G., Lu, C., Pless, R., and Huang, Q. (2004). On greedy geographic routing algorithms in sensing-covered networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '04, pages 31–42, New York, NY, USA. ACM.

You, J., Lieckfeldt, D., Han, Q., Salzmann, J., and Timmermann, D. (2009). Look-ahead geographic routing for sensor networks. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1 –6.

Zorzi, M. and Rao, R. (2003). Geographic random forwarding (geraf) for ad hoc and sensor networks: multihop performance. *Mobile Computing, IEEE Transactions on*, 2(4):337 – 348.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 14

Computação nas Nuvens: Alocação de Recursos

Planejamento de Capacidade a Longo Prazo Dirigido por Métricas de Negócio para Aplicações SaaS

David Candeia[1], Raquel Lopes[2], Ricardo Araújo Santos[2]

[1] Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campina Grande - PB – Brasil

[2]Universidade Federal de Campina Grande
Laboratório de Sistemas Distribuídos (LSD) – Campina Grande - PB – Brasil

david.maia@ifpb.edu.br, raquel@dsc.ufcg.edu.br, ricardo@lsd.ufcg.edu.br

Resumo. *O planejamento de capacidade tem feito parte da cultura do departamento de TI das empresas ao longo dos anos. No contexto da Computação na Nuvem, provedores de SaaS que compõem sua infraestrutura de TI com recursos adquiridos em provedores de IaaS por um lado economizam ao adquirir instâncias no mercado de reserva, mas por outro precisam prever a longo prazo a quantidade de instâncias necessárias. Este trabalho investiga a importância do planejamento de capacidade neste contexto e como heurísticas simples dirigidas por métricas de negócio impactam no lucro dos provedores de SaaS. Experimentos de simulação foram realizados usando cargas de trabalho sintéticas de comércio eletrônico combinadas a um modelo de utilidade baseado em provedores de SaaS atuais. Os resultados desta avaliação mostram que as heurísticas propostas aumentam, em média, o lucro do provedor de SaaS em* $3,5\%$*, havendo espaços para melhorias. Com isso, o estudo aponta indícios de que o planejamento de capacidade é uma atividade importante neste contexto, contribuindo para um aumento do lucro de provedores de SaaS.*

Abstract. *Capacity Planning is an important activity which has been part of IT department lifecycle through years. In Cloud Computing context, SaaS providers whose IT infrastructure is composed by resources from a IaaS provider can save money by acquiring instances in the reservation market with the additional problem of anticipating how many instances will be needed in a long term. This work investigates the importance of capacity planning in such context and how simple business-driven heuristics can affect SaaS providers profit.Simulation experiments were carried out using sinthetic e-commerce workloads combined to a utility model based on actual SaaS providers.The results show that proposed heuristics can increase SaaS provider profit in* 3.5% *on average, with room for improovements. Therefore, this work provides evidence that capacity planning is an important activity in the cloud computing context and can affect directly SaaS providers profit.*

1. Introdução

Planejamento de capacidade é uma atividade importante na gerência de recursos de provedores de aplicações. Tradicionalmente, uma estimativa da demanda futura da aplicação é utilizada para definir a quantidade de recursos necessários para prover a aplicação em

um certo nível de qualidade de serviço (QoS, do inglês *Quality of Service*) ou para atender certos aspectos de negócio. Considerando a hospedagem de aplicações Web tradicionais, o planejamento de capacidade tipicamente envolve superprovisionamento estático dado que a compra, instalação e configuração de recursos físicos são lentas comparadas às variações na demanda de tais aplicações, que costumam ocorrer na ordem de minutos.

O crescimento do mercado de Computação na Nuvem modificou a maneira de executar o planejamento de capacidade. Serviços providos na nuvem são normalmente divididos em três categorias de acordo com o que está sendo oferecido: (i) **infraestrutura como serviço (IaaS, do inglês *Infrastructure as a Service*)**; (ii) **plataforma como serviço (PaaS, do inglês *Platform as a Service*)**; (iii) **software como serviço (SaaS, do inglês *Software as a Service*)**. Provedores de SaaS podem executar suas aplicações em plataformas oferecidas por provedores de PaaS [Buyya et al. 2011], no entanto, consideramos neste trabalho o caso em que o provedor de SaaS monta sua infraestrutura de TI usando instâncias compradas a provedores de IaaS [Namjoshi and Gupte 2009], obtendo, assim, mais flexibilidade e controle sobre a gerência de sua infraestrutura.

Provedores de IaaS atualmente oferecem instâncias em diferentes mercados, com diferentes modelos de cobrança e QoS. Consideramos dois deles aqui: (i) o mercado **sob demanda**, no qual instâncias disponíveis no provedor podem ser adquiridas a qualquer momento mediante o pagamento de uma taxa de uso a cada intervalo de tempo menor ou igual ao intervalo mínimo de cobrança (tipicamente uma hora); e (ii) o mercado de **reserva**, no qual clientes reservam instâncias para longos intervalos futuros (geralmente maiores que um ano) mediante o pagamento de uma taxa única de reserva e de uma taxa de uso com desconto (comparada à taxa de uso do mercado sob demanda) quando as instâncias são usadas. O provedor de IaaS assegura que as instâncias reservadas estarão disponíveis sempre que o usuário desejar usá-las dentro do intervalo de reserva. Já no mercado sob demanda não há a garantia de que o usuário sempre vai ter o seu pedido de instâncias completamente atendido.

Sistemas de planejamento de capacidade podem, então, explorar os benefícios de tais mercados adquirindo instâncias mais baratas com maior disponibilidade no mercado de reserva, bem como obtendo instâncias no mercado sob demanda para lidar com variações bruscas na demanda da aplicação que não possam ser supridas pelas instâncias reservadas. Ou seja, o mercado sob demanda pode ser considerado para aquisição de instâncias adicionais de modo a prover a aplicação atendendo aos requisitos de QoS estabelecidos. Este artigo concentra-se na gerência de capacidade a longo prazo avaliando heurísticas que definem *quantas instâncias de máquinas virtuais devem ser adquiridas no mercado de reserva considerando estimativas da demanda futura*.

Como contribuições deste trabalho, primeiramente definimos um modelo de negócio para provedores de SaaS que captura a receita obtida dos clientes e os custos de adquirir instâncias nos provedores de IaaS. Este modelo considera a estratégia de *pay-as-you-go*, sendo particularmente diferente dos modelos para as aplicações Web tradicionais avaliadas no passado que tipicamente consideram receitas oriundas de um pagamento único e de atualização de versões. Propomos, também, duas heurísticas de planejamento de capacidade que determinam quantas instâncias adquirir no mercado de reserva buscando maximizar o lucro do provedor de SaaS. Por último, avaliamos as heurísticas propostas através de simulação e as comparamos com outras estratégias. Os resultados reafir-

mam a necessidade do planejamento de capacidade no cenário de Computação na Nuvem, além de demonstrar que as técnicas simples que foram propostas podem aumentar o lucro do provedor em torno de $3,5\%$, mesmo com uma predição imperfeita da demanda.

2. Trabalhos Relacionados

O termo planejamento de capacidade é comumente usado em dois contextos diferentes: a curto e a longo prazo. O planejamento a curto prazo acontece em tempo de execução, com um Sistema de Provisão Dinâmica de Recursos (DPS, do inglês *Dynamic Provisioning System*) coordenando o uso/alocação/liberação de recursos de acordo com a previsão da carga de trabalho da aplicação nas próximas horas [Urgaonkar et al. 2008, Lee et al. 2010, Sharma et al. 2011]. O planejamento a longo prazo lida com previsões de longos intervalos futuros (próximo ano, por exemplo) antecipando a quantidade de recursos que deverá ser disponibilizada para a aplicação. Considerando centros de dados tradicionais, o planejamento a longo prazo resulta na aquisição de recursos físicos [Marques et al. 2006, Sauve et al. 2006]. Entretanto, neste trabalho, assume-se que o planejamento a longo prazo resulta na reserva de instâncias em um provedor público de IaaS. Apresentamos a revisão de trabalhos de planejamento a longo e a curto prazo, uma vez que ambos os contextos podem ser considerados na elaboração de estratégias de planejamento de capacidade.

Planejamento de Capacidade baseado em métricas técnicas. Heurísticas neste grupo têm como objetivo manter certos níveis de QoS relacionados à disponibilidade da infraestrutura [Janakiraman et al. 2003], ao tempo de resposta da aplicação [Cherkasova et al. 2004] ou, ainda, a uma combinação de tais métricas [Menascé et al. 2001]. A melhor configuração da infraestrutura, que respeite os limites mínimos das métricas, será implantada.

Planejamento de Capacidade baseado em métricas de negócio. Usar apenas métricas técnicas para planejar uma infraestrutura pode gerar configurações economicamente inviáveis dado que custos e receitas não são considerados na tomada de decisão. A Gerência de TI Orientada a Negócios (BDIM, do inglês *Business-driven IT Management*) [Moura et al. 2008] tem por objetivo combinar métricas técnicas e de negócio na tomada de decisão realizada na gerência. Modelos de avaliação foram desenvolvidos considerando o custo de infraestruturas [Stage et al. 2009, Wu et al. 2011, Sharma et al. 2011], as perdas ocasionadas por negar serviço aos usuários [Marques et al. 2006, Sauve et al. 2006] e o lucro do negócio [Lopes et al. 2010, Maciel et al. 2011, Ardagna et al. 2011].

Nosso trabalho se assemelha a outros trabalhos de planejamento de capacidade dirigido a negócios uma vez que consideramos aspectos de negócio no planejamento. Entretanto, três pontos principais destacam nosso trabalho dos demais encontrados em nossa pesquisa: (i) o modelo de negócio considerado neste trabalho é baseado em provedores de SaaS; (ii) não é de nosso conhecimento a existência de trabalhos que exploram o planejamento de capacidade de provedor de SaaS usando instâncias adquiridas em diferentes mercados de um provedor de IaaS; e (iii) as heurísticas propostas adaptaram conceitos de utilização de recursos e Teoria das Filas ao cenário investigado.

3. Modelo de Utilidade

Diversos métodos podem ser utilizados para combinar métricas técnicas e métricas de negócio. Neste trabalho, usamos o conceito da microeconomia denominado de função de utilidade [Wilkes 2008], que busca capturar as preferências que guiam o comportamento de agentes. Mapeamos, então, o lucro do provedor de SaaS em uma função de utilidade e a usamos para guiar as heurísticas de planejamento. O modelo de utilidade proposto define o lucro do provedor de SaaS a partir de dois componentes: (i) da receita obtida com a oferta da aplicação para os clientes; e (ii) do custo da aquisição de instâncias no provedor de IaaS . Este modelo de utilidade considera modelos de receitas e de custos em uso atualmente por provedores de SaaS[1] e IaaS[2].

Receita: Um provedor de SaaS oferece uma aplicação A para um conjunto de clientes $U = \{u_1, u_2, \ldots, u_{|U|}\}$, cada um com requisitos de demanda particulares. Para capturar estas diferenças, o provedor oferece um portfólio de planos de assinatura $P = \{p_1, p_2, \ldots, p_{|P|}\}$ para um intervalo de tempo $D = [n^b, n^e]$ composto por $n^e - n^b$ períodos, onde $n^e \geq n^b$. Em cada período, um cliente $u \in U$ assinante de um plano $p \in P$ realiza um pagamento ao provedor de SaaS. Dessa forma, a receita total do provedor de SaaS no intervalo D é dada por: $\iota(D) = \sum_{n=n^b}^{n^e} i(n)$, onde $i(n)$ é a receita total obtida pelo provedor de SaaS no período n dentro do intervalo D. A receita do provedor de SaaS em cada período n é proveniente do conjunto de pagamentos feitos por cada cliente de SaaS $u \in U$ no período n (Equação 1). A receita $i_k(n)$, referente ao cliente u_k no período n, é composta por um valor fixo pago pelos clientes e por um valor adicional cobrado pelo uso de recursos extras quando o limite de uso no período, estabelecido no plano contratado, for ultrapassado.

$$i(n) = \sum_{k=1}^{k=|U|} i_k(n) \quad | \quad \forall 1 \leq k \leq |U|, u_k \in U \tag{1}$$

Custo: O custo total para o provedor de SaaS durante o intervalo D é calculado somando o custo total $c(n)$ em cada período n, ou seja, $\alpha(D) = \sum_{n=n^b}^{n^e} c(n)$. O custo total $c(n)$ para o provedor de SaaS em cada período n, mostrado na Equação 2, é uma combinação de três componentes: (i) do custo $ca(n)$ de uso das instâncias adquiridas no provedor de IaaS; (ii) do custo $cv(n)$ de reserva de instâncias no provedor de IaaS; e (iii) das penalidades $p(n)$ pagas aos clientes do provedor de SaaS sempre que a aplicação violar o Acordo de Nível de Serviço (SLA, do inglês *Service Level Agreement*) definido ao contratar o plano p_j .

$$c(n) = ca(n) + cv(n) + p(n) \tag{2}$$

Dados os modelos de receita e custo, a utilidade do provedor de SaaS durante o intervalo de tempo $D = [n^b, n^e]$, onde $n^e \geq n^b$, é dada pela Equação 3.

$$\upsilon(D) = \iota(D) - \alpha(D) \tag{3}$$

[1]BigCommerce, SalesForce e SurveyMonkey

[2]Amazon EC2 e Rackspace.

O modelo proposto[3] foi usado para: (i) estimar a utilidade de um plano de reserva e guiar as heurísticas de planejamento propostas; (ii) calcular a utilidade obtida por um provedor de SaaS ao implantar um plano de reserva para oferecer a aplicação.

4. Heurísticas de Planejamento de Capacidade

Detalhamos abaixo as duas heurísticas dirigidas por métricas de negócio propostas. Ambas tem como entrada uma predição da carga de trabalho futura (obtida através de dados históricos, por exemplo) contendo dados sobre um intervalo de tempo D com mesma duração do intervalo para o qual se está planejando a capacidade. Ambas produzem um plano de reserva com uma descrição do tipo e quantidade das instâncias a serem reservadas no provedor de IaaS, tal que este plano maximize a utilidade do provedor SaaS. A primeira heurística aplica conceitos de utilização de recursos enquanto a outra baseia-se em Teoria das Filas, ambas adaptadas ao contexto de Computação na Nuvem.

4.1. Heurística baseada em Utilização - UT

Esta heurística possui duas etapas: (i) simulação da execução da carga de trabalho prevista; e (ii) avaliação da simulação para definir um plano de reserva.

A primeira etapa tem como entrada a previsão da carga de trabalho composta pelo tempo de chegada e demanda de processamento, em uma instância base, de cada requisição feita pelos usuários. A heurística UT simula o processamento desta carga prevista usando instâncias adquiridas periodicamente (por exemplo, a cada hora) por um Sistema de Provisão Dinâmica de Recursos (DPS) no mercado sob demanda do provedor de IaaS. Tal simulação assume que o DPS adquire tipos e quantidades adequados de instâncias para atender ao SLA estabelecido.

Na segunda etapa (algoritmo na Figura 1), dados os componentes do custo ca e cv da Equação 2, UT calcula, para cada tipo de instância ofertado pelo provedor de IaaS, qual a menor quantidade de tempo ($limiar_t$) em que uma instância reservada deve ser utilizada para se tornar mais barata que uma instância sob demanda (linha 3). Isto é possível devido ao desconto na taxa de uso de instâncias reservadas em comparação à taxa de uso de instâncias sob demanda, compensando a taxa de reserva paga inicialmente. Por exemplo, de acordo com as taxas praticadas pela Amazon em 2011, este consumo mínimo é de $4136,364$ horas para um intervalo D de um ano. Em seguida, a heurística UT avalia, para cada tipo t, a quantidade de horas de uso de cada instância adquirida na primeira etapa (linhas 4 a 9). Esta contabilidade considera que sempre que uma instância for utilizada em D será sempre a mesma instância. Se duas instâncias forem utilizadas serão sempre as mesmas duas instâncias, sendo uma além da instância anteriormente citada.

Uma vez calculados os limiares de consumo mínimo para cada tipo de instância, a heurística seleciona a maior quantidade n de instâncias que foram usadas em paralelo por um intervalo de tempo maior que o limiar (linha 10) e calcula a quantidade de instâncias a reservar para cada tipo t como a divisão entre a quantidade total de horas (multiplicando o total n de instâncias usadas pela quantidade de tempo $uso_t[n]$ que estiveram usadas em paralelo) e o limiar de consumo mínimo para o tipo t (linha 11). É importante destacar que a heurística UT é dependente do DPS sendo usado, dado que ela considera apenas

[3]Uma versão mais detalhada do modelo de utilidade aqui apresentado pode ser obtida em `http://www.lsd.ufcg.edu.br/~davidcmm/utilityModel`.

os tipos de instâncias adquiridos na primeira etapa, ou seja, tipos não usados na primeira etapa e que poderiam trazer uma melhoria na utilidade não são verificados. A heurística também não faz equivalência entre o consumo de várias instâncias menores no consumo de uma instância com mais recursos.

Dessa forma, os planos de reserva produzidos pela heurística UT tem por objetivo encontrar a quantidade de instâncias de cada tipo t consumidas por um período de tempo maior ou igual ao $limiar_t$, sendo, assim, mais baratas que às instâncias adquiridas no mercado sob demanda do provedor de IaaS.

1: **procedure** PLAN_UT_ETAPA_2 ($consumo_{tipo_1}$, ..., $consumo_{tipo_n}$) ▷ $consumo_{tipo_n}$ é um conjunto de tuplas $\langle hora, quantidade \rangle$ indicando a quantidade de instâncias de $tipo_n$ consumidas em cada hora simulada na etapa 1
2: **for all** t in $tipo_1, tipo_2, \ldots, tipo_n$ **do**
3: Calcula $limiar_t$ baseado nas taxas do tipo t
4: **for all** i in 1, ..., maxInt **do**
5: $uso_t[i] = 0$
6: **end for**
7: **for all** $\langle hora, quantidade \rangle$ in $consumo_t$ **do**
8: $uso_t[quantidade] += 1$
9: **end for**
10: Select n tal que n é o maior índice de uso_t onde $uso_t[n] \geq limiar_t$
11: Reserva $\lceil \frac{uso_t[n] \times n}{limiar_t} \rceil$
12: **end for**
13: **end procedure**

Figura 1. Etapa 2 da Heurística UT

1: **procedure** PLAN_RF ($cargaPrevista$, ρ) ▷ $cargaPrevista$ representa um resumo da carga para um intervalo D
2: $T = \sum_{m=1}^{total\_de\_horas} \bar{S}_m * \bar{\lambda}_m$
3: Calcula $limiar_t$ baseado nas taxas do tipo t
4: **for all** t in tipoInstâncias **do**
5: $MAX_t = \lfloor T / limiar_t \rfloor$
6: **end for**
7: $planosPossíveis \leftarrow$ monta todos os planos de reserva com quantidades de instâncias variando de 0 a MAX_t
8: **for all** plano in planosPossíveis **do**
9: utilidade[plano] $\leftarrow 0$
10: **for all** hora **m** in cargaPrevista **do**
11: $resReq \leftarrow$ calcula quantidade de requisições processadas por instâncias reservadas (utilização limitada a ρ)
12: $onDemReq \leftarrow$ calcula quantidade de requisições processadas por instâncias sob demanda
13: $nãoProcessadas+ =$ calcula quantidade de requisições não processadas
14: $violadas+ =$ calcula quantidade de requisições que violaram o SLA
15: $horasReservadas+ = \lceil resReq * \bar{S}_m \rceil$
16: $horasSobDemanda+ = \lceil onDemReq * \bar{S}_m \rceil$
17: **end for**
18: $utilidade[plano] = estimaReceita() - estimaCusto(horasReservadas, horasSobDemanda) - estimaPenalidade(nãoProcessadas, violadas)$
19: **end for**
20: **return** plano que maximiza utilidade[plano]
21: **end procedure**

Figura 2. Heurística RF

4.2. Heurísticas baseada em Teoria das Filas - RF

A heurística RF elabora um plano de reserva baseado em conceitos da Teoria das Filas (algoritmo da Figura 2). A heurística recebe como entrada um valor alvo de utilização média (ρ) para as instâncias adquiridas no mercado de reserva e um sumário estatístico para cada hora da carga prevista para o intervalo D composto por: taxa média de chegada de requisições ($\bar{\lambda}$), tempo médio de serviço das requisições ($\bar{S}$), número médio de usuários e tempo médio de espera do usuário. A heurística RF estima, então, a carga total T para o período D, em horas de CPU, baseado nestes dados (linha 2) e calcula para cada tipo de instância oferecido pelo provedor de IaaS o mínimo de tempo durante o qual uma instância reservada deve ser utilizada ($limiar_t$), de maneira similar ao que foi realizado pela heurística UT descrita na Seção 4.1.

Para cada tipo t, a heurística calcula a quantidade MAX_t de instâncias do tipo t que sozinhas seriam suficientes para executar a carga total T, considerando o valor de $limiar_t$ (linhas 4 a 6). A partir dos valores de MAX_t, a heurística RF elabora o conjunto

de planos de reserva com todas as combinações possíveis de quantidades de instâncias tal que para cada tipo t se tenha uma quantidade de instâncias entre 0 e MAX_t (linha 7).

Para efetuar o cálculo da utilidade de cada possível plano de reserva, a heurística RF precisa estimar para cada hora do intervalo de tempo D, quantas instâncias serão necessárias para executar a demanda sem violar o SLA (linhas 10 a 17). Para executar a demanda RF utiliza instâncias adquiridas no mercado de reserva e instâncias adquiridas no mercado sob demanda. Cada instância reservada, pertencente ao plano de reserva sob avaliação, tem seu uso limitado ao valor alvo de utilização média ρ. Caso as instâncias reservadas, com utilização máxima ρ, não sejam suficientes para executar a demanda passa-se a utilizar instâncias sob demanda para executar o restante da demanda. No entanto, há ainda a possibilidade de que algumas requisições não possam ser atendidas devido ao risco de negação do pedido de compra de instâncias no mercado sob demanda. O não atendimento de algumas requisições pode resultar no pagamento de penalidades de acordo com o modelo de utilidade aplicado (linha 18). O plano de reserva escolhido (linha 20) será aquele que levar à maior utilidade obtida de acordo com a Equação 3.

Esta heurística é mais flexível por considerar várias combinações de diferentes tipos de instâncias, além de ter um valor alvo de utilização média ρ configurável, permitindo que a abordagem de reserva seja mais ou menos conservadora de acordo com o valor deste limiar (quanto menor, mais máquinas serão reservadas).

5. Avaliação

5.1. Modelo de Simulação

As heurísticas propostas na Seção 4 foram avaliadas com experimentos de simulação. Optamos por não usar simuladores existentes (como o CloudSim[4]) dada a dificuldade de implantar o modelo de utilidade considerado, uma vez que tratam de detalhes que não são o foco deste trabalho (e.g. modelos de consumo de energia e de alocação de máquinas virtuais). Implementamos uma extensão do *framework* SaaSim [Santos 2012][5] considerando técnicas de Verificação & Validação propostas em [Sargent 2005]. O modelo implementado[6] considera um provedor de SaaS, que oferece uma mesma aplicação para diferentes clientes, cuja infraestrutura é composta por instâncias adquiridas em um provedor de IaaS. A simulação foi dividida em duas fases: (i) planejamento de capacidade, com a execução da heurística para produzir um plano de reserva; e (ii) implantação do plano de reserva, com a execução de uma carga de trabalho sintética em uma infraestrutura baseada no plano de reserva produzido no planejamento de capacidade.

Na primeira fase, a heurística é usada para produzir o plano de reserva a partir de uma previsão da carga de trabalho para o intervalo D futuro. Consideramos impossível que o preditor, ao estimar a carga futura, produza uma estimativa perfeita para o intervalo D e para modelar a precisão da previsão usada, consideramos um erro relacionado à quantidade de clientes de SaaS submetendo requisições à aplicação. Sendo assim, um erro positivo superestima o número de clientes que compõem a carga do sistema, enquanto um erro negativo produz uma previsão subestimada dessa quantidade. Por exemplo, um erro

[4]http://www.cloudbus.org/cloudsim/

[5]Disponível em `http://github.com/ricardoas/saasim`

[6]Disponível em `http://code.google.com/p/saasim-david`

de predição de 10% indica que se a carga futura será gerada por 100 clientes de SaaS, a previsão estima uma carga com 110 clientes. Um erro de predição de -10% indica que se a carga futura será gerada por 100 clientes, a carga prevista será composta por 90 clientes.

A segunda fase tem por objetivo avaliar o plano de reserva produzido. Nesta etapa, simulamos a execução de uma carga de trabalho sintética em uma infraestrutura composta segundo o plano de reserva e avaliamos a utilidade do provedor de SaaS face aos diferentes erros de predição da carga de trabalho. É importante lembrar que a carga de trabalho de cada cliente de SaaS é composta pelo agregado de requisições dos usuários finais que acessam a aplicação adquirida pelo cliente de SaaS.

Consideramos que as requisições são distribuídas para execução na infraestrutura por um balanceador de carga que segue uma política de *round-robin* e que distribui as requisições nas instâncias proporcionalmente ao número de núcleos de CPU de cada uma. Cada instância executa as requisições segundo um modelo de processamento bem consolidado [Menasce et al. 2004]. Cada instância atende paralelamente um número máximo de m requisições controladas por um conjunto de m fichas que representam os processos/*threads* disponíveis (Figura 3). Uma requisição ao chegar à instância adquire uma ficha e entra na fila de processamento. Caso não existam fichas disponíveis, a requisição é direcionada para espera em uma fila de **backlog** de política **first-come, first-served** até que uma ficha se torne disponível. Por fim, requisições que chegam a uma instância cujo **backlog** está cheio são descartadas.

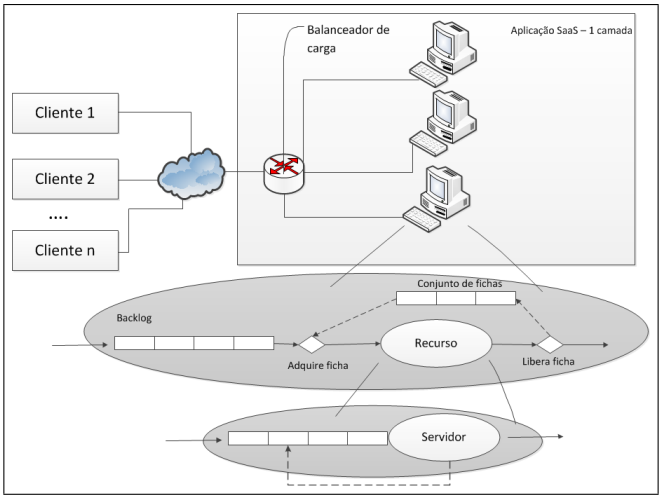


Figura 3. Modelo do Sistema: visão geral sobre filas e processamento de requisições

A fila de processamento é modelada com uma política de **time sharing**[7]. Além da demanda por processamento, cada requisição tem uma demanda de transferência de dados. Esta demanda é atendida pelo provedor de IaaS independentemente da escolha e negociação de instâncias. Cada cliente de SaaS tem, ainda, uma demanda por armazenamento relacionada à hospedagem da aplicação e aos registros de usuários. Estas duas demandas são sempre atendidas e calculamos seu custo conforme modelo apresentado na

[7]Uma requisição pode utilizar a CPU por um intervalo Δ, tipicamente muito pequeno, e em seguida a CPU é alocada para outra requisição que esteja esperando para utilizar a CPU. Desta forma, todas as requisições progridem simultaneamente e os atrasos relacionados às condições de contenção são capturados.

Seção 3.

Dada a demanda tipicamente variável das cargas de trabalho de aplicações Web [Crovella and Bestavros 1996], o controle de quantas instâncias estão sendo usadas no momento para compor a infraestrutura é feita por um DPS. Neste trabalho, usamos um DPS não realista que, a partir do conhecimento da carga futura, decide a quantidade de instâncias necessárias. Apesar desta simplificação não ser realista, ela é importante para focarmos a avaliação na qualidade do plano de reservas produzido pelas heurísticas de planejamento.

5.2. Instância do Modelo de Simulação

O planejamento fatorial completo realizado apontou o número de clientes de SaaS e o erro de predição da carga de trabalho como os fatores mais significativos. Os experimentos foram realizados de modo a explorar diversas combinações destes fatores enquanto os outros receberam níveis fixos. Sabemos que nossa análise não é exaustiva e que diferentes níveis poderiam ter sido utilizados para estes fatores fixos, porém confiamos que nossa abordagem foi suficiente para avaliar as tendências das heurísticas propostas e obter ideias de trabalhos futuros.

Para instanciar o modelo de utilidade da Seção 3 usamos dados de provedores bem conhecidos de SaaS e IaaS. Para o modelo de receita, três dos planos oferecidos em 2011 pelo provedor de SaaS **BigCommerce** foram tomados como modelo: *Bronze*, *Gold* e *Diamond*. O **BigCommerce** tarifa seus clientes mensalmente (logo, n é igual a 1 mês). Além disso, uma margem de contribuição de 30% foi escolhida para cada plano de acordo com o que é praticado no mercado[8].

Quanto ao modelo de custo, o provedor de IaaS simulado é inspirado no serviço provido pela **Amazon EC2** em 2011. Três tipos de instâncias foram considerados: *small* (1 núcleo virtual), *large* (2 núcleos virtuais) e *xlarge* (4 núcleos virtuais). A única diferença considerada entre cada tipo de instância é o total de núcleos virtuais. Após uma instância ser requisitada ao provedor de IaaS existe um período, fixado aqui como 5 minutos [Wu et al. 2011], para que a instância e a aplicação iniciem. Por último, modelamos o risco de que um pedido por instâncias no mercado sob demanda seja negado como 10%. Esse valor é intimamente ligado à imagem do provedor de IaaS no mercado e acreditamos que o valor usado seja razoável.

Nós simulamos o planejamento de capacidade e a execução da carga de trabalho para um período D de 1 ano. Foram usados 50 e 100 clientes de SaaS distribuídos uniformemente entre os planos ofertados pelo provedor de SaaS. Os erros de predição da carga de trabalho foram 40% e -40%. Para cada combinação dos valores, foram executadas 70 cargas sintéticas de trabalho diferentes de modo a calcularmos intervalos com 95% de confiança.

De acordo com Arlitt et al. [Arlitt et al. 2001], um dia de carga possui um período de pico entre 9:00 e 21:00, e as semanas possuem dias com mais e menos carga que dias típicos. Um pico de carga corresponde a 2 vezes a média de requisições, enquanto períodos mais suaves apresentam 50% da média de requisições. Estas invariantes foram combinadas com os limites de uso, os preços e as margens de contribuição dos planos

[8] `http://biz.yahoo.com/p/sum_qpmd.html`

de SaaS para calcular as taxas de chegada de requisições diárias para cada plano considerado (Tabela 1). Além disso, alguns eventos especiais (e.g. Dia das Mães) causam picos de carga em relação às semanas típicas [Arlitt et al. 2001]. As cargas utilizadas nas simulações consideram estas invariantes e foram geradas pelo gerador de cargas GEIST [Kant et al. 2001], enquanto as previsões das cargas foram derivadas destas cargas. O GEIST gera uma carga considerando uma distribuição de Poisson como a distribuição marginal do processo de chegada e, em seguida, adiciona as propriedades de multifractal e autossimilaridade presentes neste tipo de carga de trabalho.

	Plano de SaaS		
	Bronze	Gold	Diamond
Dia típico	$0,058$ req/s	$0,176$ req/s	$0,650$ req/s
Dia de pico	$0,117$ req/s	$0,350$ req/s	$1,300$ req/s
Dia suave	$0,029$ req/s	$0,090$ req/s	$0,325$ req/s

Tabela 1. Taxas de chegada (requisições/segundo) para uma semana típica

5.3. Apresentação e Análise de Resultados

As heurísticas UT e RF foram avaliadas frente a três estratégias de referência: (i) uma estratégia que usa apenas instâncias do mercado sob demanda - denominada de ON; (ii) uma estratégia que considera o pico da demanda de instâncias e reserva 20% deste pico fazendo uso de instâncias *large* - denominada de SUPER[9]; e (iii) uma estratégia ótima que, conhecendo o funcionamento futuro do DPS, testa planos de reserva contendo da menor até a maior quantidade de instâncias utilizadas pelo DPS e escolhe o plano com maior utilidade estimada - denominada de OP.

Nossa análise considera a utilidade do provedor de SaaS e o **ganho** de utilidade, em porcentagem, de cada heurística em relação à estratégia ON segundo a Equação 4.

$$ganho(\upsilon_A(D), \upsilon_{ON}(D)) = 100 * \frac{(\upsilon_A(D) - \upsilon_{ON}(D))}{|\upsilon_{ON}(D)|} \quad (4)$$

Primeiramente, verificamos a viabilidade do planejamento realizado pelas heurísticas propostas. A hipótese nula $\upsilon_{SUPER}(D) = \upsilon_{ON}(D) = \upsilon_{UT}(D) = \upsilon_{RF}(D)$ foi rejeitada em cenários com 100 clientes de SaaS pela análise de variância (ANOVA) realizada. Realizamos uma análise *post-hoc* para avaliar se alguma heurística obteve utilidades similares às utilidades de ON e concluímos que $\upsilon_{UT}(D), \upsilon_{RF}(D) > \upsilon_{ON}(D) > \upsilon_{SUPER}(D)$. Logo, as heurísticas propostas apresentam ganhos diferentes entre si, e diferentes de zero, ou seja, aumentam a utilidade do provedor de SaaS em relação à utilidade de ON. A avaliação estatística dos cenários com 50 clientes conduz às mesmas conclusões.

O segundo passo da análise *post-hoc* buscou quantificar os **ganhos** das heurísticas avaliadas analisando os cenários de 50 e 100 clientes de SaaS. A Figura 4 apresenta os **ganhos** obtidos com cada heurística/estratégia para erros de predição de -40% e 40%.

A estratégia SUPER reserva o maior número de núcleos virtuais usando instâncias do tipo *large*, enquanto as outras heurísticas usam instâncias do tipo *small*, o que a leva

[9]A estratégia SUPER reserva 20% do pico da demanda por instâncias dado que 20% é uma utilização esperada para uma infraestrutura planejada para o pico da carga de trabalho [Armbrust et al. 2009].

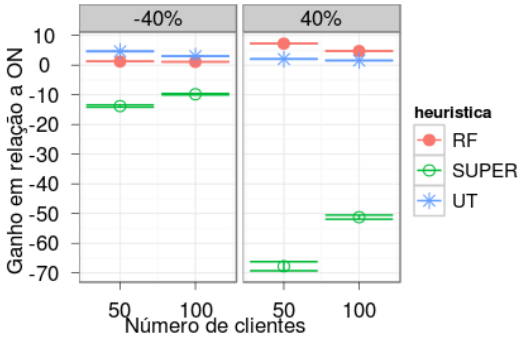


Figura 4. Ganhos de utilidade: erros de predição de -40% e 40%

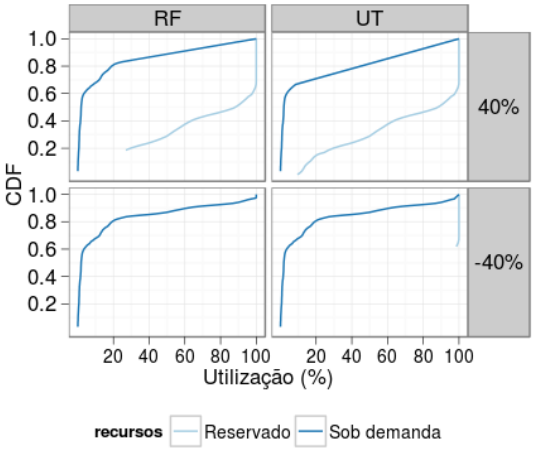


Figura 5. CDF da taxa de utilização das instâncias

a obter os maiores custos de infraestrutura. Este aumento no custo se deve ao aumento na ociosidade da infraestrutura uma vez que se precisa, na verdade, de instâncias menores. Como consequência, SUPER obteve, em todos os cenários, utilidades inferiores às utilidades obtidas por ON, logo é melhor não realizar um planejamento do que superprovisionar a infraestrutura com a estratégia SUPER. As heurísticas UT e RF apresentaram **ganhos** médios de $1,5\%$ a $7,3\%$, indicando que bons planos podem aumentar o lucro do provedor de SaaS. Considerando o conhecimento total da carga de trabalho futura, a estratégia OP apresenta **ganhos** da ordem de 11% a 16% [10]. Apesar dos valores encontrados para os **ganhos** serem baixos, o ganho monetário para o provedor de SaaS é mais significativo quão maiores forem a receita e a quantidade de clientes.

O próximo passo da análise consistiu em avaliar o comportamento de cada heurística. As heurísticas RF e UT reservam menos instâncias que SUPER e, além disso, usam apenas instâncias *small*. A Figura 5 mostra que as instâncias reservadas por UT e RF apresentam altas taxas de utilização durante o intervalo D (i.e. taxas de utilização superiores ao limiar de $47,8\%$ da Amazon), com as instâncias de RF apresentando taxas de 100% para erros de predição de -40%. Esta alta utilização conduz a uma redução de custos em relação aos custos de ON e SUPER. Em cenários com superestimativa da carga, UT e RF reservam algumas instâncias que apresentam baixa utilização e um custo superior ao custo de instâncias sob demanda. Como consequência, o custo total da infraestrutura é aumentado e os **ganhos** das heurísticas reduzidos.

Avaliando os planos de reserva de RF e UT observa-se que RF sempre reserva menos instâncias que UT. Isto acontece porque a heurística RF busca atingir 75%[11] de taxa de utilização para suas instâncias reservadas, ao passo que UT busca uma taxa de $47,8\%$ de utilização, aumentando, assim, o número de instâncias reservadas. Como con-

[10]Os intervalos de 95% de confiança para a média dos ganhos com 50 e 100 clientes de SaaS foram, respectivamente, $(15,748\%; 16,026\%)$ e $(11,527\%; 11,624\%)$. O erro de predição da carga é zero para a estratégia OP.

[11]Este valor se baseia no limiar de utilização a partir do qual o tempo de resposta passa a apresentar crescimento exponencial.

sequência destas escolhas, em cenários de subestimativa da carga, UT utiliza uma maior quantidade de instâncias mais baratas, aumentando sua utilidade. Por outro lado, em cenários de superestimativa da carga, UT utiliza uma maior quantidade de instâncias mais caras, obtendo, assim, utilidades inferiores às utilidades de RF.

Para considerar a dificuldade de predição da carga de trabalho baseada em histórico e em tendências de mercado, realizamos uma análise de sensibilidade do erro de predição da carga. Esta análise tentou refletir a possibilidade de uso de preditores que resultam em diferentes erros de predição. Os erros de predição considerados foram: -40%, -20%, -10%, 0%, 10%, 20% e 40%.

A análise demonstrou que UT e RF apresentam comportamentos diferenciados para erros de superestimativa e de subestimativa da carga. Para cenários de superestimativa da carga, a natureza mais conservadora de RF (reservas menores) fez com que a mesma obtivesse melhores **ganhos** que UT. Para cenários de subestimativa da carga, UT apresentou os melhores resultados. É importante destacar que erros de predição menores contribuem para melhores **ganhos** de ambas as heurísticas. Quando o erro de predição foi zero, os intervalos de confiança para a média dos **ganhos** obtidos por UT e RF para 100 clientes de SaaS foram de $(5,21\%; 5,26\%)$ e $(5,15\%; 5,18\%)$, respectivamente. A estratégia OP para 100 clientes obtém um intervalo de confiança para a média dos **ganhos** de $(11,527\%; 11,624\%)$ sugerindo que, apesar das heurísticas propostas terem aumentado a utilidade do provedor de SaaS, existe espaço para melhorias.

6. Conclusões e Trabalhos Futuros

O foco deste trabalho é demonstrar que o planejamento de capacidade não deve ser negligenciado no contexto de Computação na Nuvem. Para isso, um modelo de utilidade que captura aspectos de negócio relacionados à oferta de SaaS foi elaborado e utilizado para guiar a reserva de instâncias em um provedor de IaaS realizada por duas heurísticas propostas, UT e RF. As heurísticas propostas foram avaliadas em um cenário de planejamento de capacidade para um intervalo de um ano através de experimentos de simulação, usando cargas de trabalho sintéticas de comércio eletrônico e comparadas tanto com uma estratégia de superprovisionamento (SUPER) como com uma estratégia que não reserva instâncias (ON). Nossos resultados mostram que RF conduz a um aumento médio de $3,77\%$ no lucro do provedor de SaaS, enquanto UT conduz a um aumento médio de $3,19\%$. A estratégia SUPER obteve os piores lucros devido às más escolhas de tipo e quantidade de instâncias reservadas. Conclui-se que, para os cenários observados, não se deve superprovisionar a infraestrutura (usar a heurística SUPER) já que heurísticas simples de planejamento melhoram o lucro do provedor de SaaS.

A análise de sensibilidade mostrou que a qualidade da predição da carga de trabalho influencia os resultados das heurísticas. Grandes provedores de SaaS tendem a possuir grandes quantidades de dados históricos e a investir em boas técnicas de predição de carga, obtendo erros pequenos e explorando melhor as heurísticas propostas. Todavia, pequenos provedores podem não ter acesso a estas possibilidades, obtendo erros maiores na predição e, assim, não conseguindo explorar ao máximo as heurísticas propostas.

As simplificações consideradas conduzem a algumas ameaças de validade que podem ser investigadas em trabalhos futuros. Quanto à *validade externa*, usamos cargas de trabalho artificiais de comércio eletrônico criadas pelo GEIST, um gerador antigo, dado

que não encontramos um gerador baseado em estudos recentes. Seria interessante usar cargas reais coletadas de aplicações. Apesar disso, alimentamos o modelo de utilidade com informações de provedores de IaaS e SaaS reais e, por isso, nossos resultados são importantes para traçar uma visão geral da atuação das heurísticas no cenário considerado. Quanto à *validade de construção*, a aplicação foi modelada com uma única camada e sessões de usuários não foram consideradas, o que pode ser melhorado com o uso de um modelo mais apurado do tipo de aplicação investigado. Por fim, consideramos importante investigar o impacto da negação de serviço no desempenho das heurísticas de planejamento de capacidade e pretendemos implementar melhorias nas heurísticas propostas com base no que pode ser aprendido com o estudo da estratégia OP.

Referências

Ardagna, D., Panicucci, B., and Passacantando, M. (2011). A game theoretic formulation of the service provisioning problem in cloud systems. In *Proceedings of the 20th international conference on World wide web*, pages 177–186. ACM.

Arlitt, M., Krishnamurthy, D., and Rolia, J. (2001). Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology*, 1(1):44–69.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the Clouds : A Berkeley View of Cloud Computing Cloud Computing : An Old Idea Whose Time Has (Finally) Come. *Computing*, pages 07–013.

Buyya, R., Broberg, J., and Goscinski, A. (2011). *Cloud computing: Principle and Paradigms*. Wiley Online Library.

Cherkasova, L., Tang, W., and Singhal, S. (2004). An sla-oriented capacity planning tool for streaming media services. In *Dependable Systems and Networks, 2004 International Conference on*, pages 743–752. IEEE.

Crovella, M. and Bestavros, A. (1996). Self-similarity in world wide web traffic: evidence and possible causes. In *ACM SIGMETRICS Performance Evaluation Review*, volume 24, pages 160–169. ACM.

Janakiraman, G., Santos, J., and Turner, Y. (2003). Automated multi-tier system design for service availability. In *Proceedings of the First Workshop on Design of Self-Managing Systems*. Citeseer.

Kant, K., Tewari, V., and Iyer, R. (2001). Geist: Generator of ecommerce and internet server traffic. In *Proc. of Int. Symposium on Performance Analysis of Systems and Software*.

Lee, Y., Wang, C., Zomaya, A., and Zhou, B. (2010). Profit-driven service request scheduling in clouds. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 15–24. IEEE Computer Society.

Lopes, R., Brasileiro, F., and Maciel, P. (2010). Business-driven capacity planning of a cloud-based it infrastructure for the execution of web applications. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE.

Maciel, P., Brasileiro, F., Santos, R., Maia, D., Lopes, R., Aquino de Carvalho, M., Costa Ribeiro, R., Andrade, N., and Mowbray, M. (2011). Business-driven short-term management of a hybrid it infrastructure. *Journal of Parallel and Distributed Computing*.

Marques, F., Sauvé, J., and Moura, A. (2006). Business-oriented capacity planning of it infrastructure to handle load surges. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 1–4. IEEE.

Menasce, D., Almeida, V., Dowdy, L., and Dowdy, L. (2004). *Performance by design: computer capacity planning by example*. Prentice Hall.

Menascé, D., Barbará, D., and Dodge, R. (2001). Preserving qos of e-commerce sites through self-tuning: A performance model approach. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 224–234. ACM.

Moura, A., Sauve, J., and Bartolini, C. (2008). Business-driven it management-upping the ante of it: exploring the linkage between it and business to improve both it and business results. *Communications Magazine, IEEE*, 46(10):148–153.

Namjoshi, J. and Gupte, A. (2009). Service Oriented Architecture for Cloud Based Travel Reservation Software as a Service. *2009 IEEE International Conference on Cloud Computing*, pages 147–150.

Santos, R. A. (2012). Saasim - um framework para simulação de software as a service. Master's thesis, Federal University of Campina Grande.

Sargent, R. (2005). Verification and validation of simulation models. In *Proceedings of the 37th conference on Winter simulation*, pages 130–143. Winter Simulation Conference.

Sauve, J., Marques, F., Moura, A., Sampaio, M., Jornada, J., and Radziuk, E. (2006). Optimal design of e-commerce site infrastructure from a business perspective. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 8, pages 178c–178c. IEEE.

Sharma, U., Shenoy, P., Sahu, S., and Shaikh, A. (2011). A cost-aware elasticity provisioning system for the cloud. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 559–570. IEEE.

Stage, A., Setzer, T., and Bichler, M. (2009). Automated capacity management and selection of infrastructure-as-a-service providers. In *Integrated Network Management-Workshops, 2009. IM'09. IFIP/IEEE International Symposium on*, pages 20–23. IEEE.

Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., and Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1.

Wilkes, J. (2008). Utility functions, prices, and negotiation. *Market-Oriented Grid and Utility Computing*, pages 67–88.

Wu, L., Garg, S., and Buyya, R. (2011). Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 195–204. IEEE.

Um Arcabouço Para Provisionamento Automático de Recursos em Provedores de IaaS Independente do Tipo de Aplicação

Fábio Morais[1], Francisco Brasileiro[1], Raquel Lopes[1], Ricardo Araújo[1], Augusto Macedo[1], Wade Satterfield[2], Leandro Rosa[3]

[1] Universidade Federal de Campina Grande
Laboratório de Sistemas Distribuídos, Campina Grande – Brasil

[2]Hewlett-Packard – ESSN – Fort Collins, Colorado Area – USA

[3]Hewlett-Packard – ESSN – Brazil Lab, Porto Alegre – Brasil

{fabio,fubica,raquel,ricardo,augustoq}@lsd.ufcg.edu.br
{Wade.Satterfield,Leandro.Rosa}@hp.com

Abstract. *Infrastructure cost reductions can be achieved by taking advantage of the elasticity provided by IaaS deployments. However, state-of-the-practice auto-scaling solutions use simple reactive approaches, which can successfully reduce these costs, but are often not efficient at minimizing SLA violations. In this paper we propose a flexible and non-intrusive framework for auto-scaling services that follows both a reactive and a proactive approach. Our framework uses a configurable set of predictors of future service demands and a selection mechanism that periodically decides the best predictor to be used. We also propose a prediction correction method that aims at minimizing underestimation errors, which may reduce the number of SLA violations. Simulation experiments using production traces from HP customers were carried out. Results show costs savings of as much as* 37%*, while the probability of an SLA violation can be kept, on average, as small as* 0.008%*, and no larger than* 0.036%*.*

Resumo. *Custos de infraestrutura podem ser reduzidos por meio da elasticidade oferecida pelos provedores de IaaS. No entanto, soluções do mercado utilizam abordagens reativas, que reduzem esses custos, mas não são suficientes para evitar quebras de SLA. Nesse trabalho, propomos um arcabouço para provisionamento automático de recursos, flexível e não intrusivo, que emprega abordagens reativa e proativa, baseadas no uso de um conjunto configurável de preditores de demandas dos serviços, além de um mecanismo de seleção que decide periodicamente o melhor preditor a ser usado. Também propomos uma nova maneira de corrigir predições subestimadas, reduzindo por consequência o número de quebras de SLA. Nós avaliamos o arcabouço através de simulações que usam dados de produção de clientes da HP. Os resultados mostram que é possível obter uma economia de até* 37% *enquanto a probabilidade de quebra de SLA é mantida em média em* $0,008\%$ *e limitada superiormente a* $0,036\%$*.*

1. Introdução

O modelo de infraestrutura como um serviço (IaaS, do inglês *infrastructure-as-a-service*) oferecido pelo mercado de *Computação na Nuvem* vem sendo, cada vez mais, usado pelo

departamento de TI (Tecnologia da Informação) das empresas para aquisição de capacidade computacional. Isso é feito atualmente de duas formas: (i) através de infraestruturas privadas, reduzindo o custo da aquisição por meio da economia de escala de infraestruturas próprias; ou (ii) com o uso de infraestruturas públicas, nas quais a capacidade é adquirida sob demanda de provedores externos de IaaS, que normalmente empregam a tarifação do serviço baseada no modelo "pague conforme utilização" (do inglês *pay-as-you-go*), no qual clientes pagam apenas pelos recursos de fato utilizados.

Uma das principais vantagens oferecidas pelos provedores de IaaS para os clientes é a elasticidade[1] no provisionamento de capacidade a curto prazo. Uma estratégia ótima para executar serviços com demandas que variam no tempo através de recursos virtuais deve prover, em qualquer instante de tempo, a capacidade exata para manter o desempenho do serviços e satisfazer os acordos de nível de serviço (SLA, do inglês *service level agreement*). Tal abordagem gerencia automaticamente a capacidade, provendo mais recursos quando a demanda aumenta, e liberando recursos à medida que não são mais necessários.

Um aspecto importante de um mecanismo de provisionamento automático eficiente é a habilidade de antecipar mudanças na demanda do serviço. Monitoramento e predição são tarefas importantes que precisam ser cuidadosamente realizadas pelo mecanismo. No entanto, estimar a demanda é uma tarefa não trivial que tem recebido pouca atenção nas soluções de planejamento de capacidade a curto prazo, tando do mercado como da academia. Em geral, essas soluções atuam reagindo a mudanças na carga [Amazon 2012, RightScale 2012, Marshall et al. 2010, Vijayakumar et al. 2010, Calcavecchia et al. 2012], ou assumem a disponibilidade de um preditor sem de fato avaliar a influência de tais preditores na performance do mecanismo de provisionamento automático em si.

Além do mais, alguns trabalhos requerem a coleta de informações sobre o serviço em execução na infraestrutura (como tempo de resposta, tamanho da fila, taxa de chegada e demanda das requisições) [Urgaonkar et al. 2008, Sharma et al. 2011, Vijayakumar et al. 2010]. A dependência dessas informações pode inviabilizar esse tipo de solução de provisionamento, visto que, em alguns casos, essas são informações críticas que nem sempre podem ser compartilhadas, mesmo com os provedores de IaaS.

Neste artigo nós abordamos limitações existentes em trabalhos anteriores propondo um mecanismo proativo e reativo de provisionamento automático que: é *independente do tipo de aplicação* (do inglês *service-agnostic*) — necessita monitorar apenas os recursos da infraestrutura onde a aplicação está em execução, ou seja, o mecanismo é não intrusivo; e possui *configuração dinâmica e automática* em relação às atividades de predição e de monitoramento. Em particular, o mecanismo considera o uso de múltiplos preditores, dado que não existe um único preditor que é o melhor em estimar a carga para todas as aplicações [Jiang et al. 2012]. Um seletor avalia continuamente as opções de predição que estão disponíveis e criteriosamente escolhe o preditor para ser utilizado no futuro próximo, permitindo que o mecanismo se adapte a alterações nos padrões de carga. Além disso, propomos um novo método de correção de predição que visa evitar erros de subprovisionamento e diminuir o número de quebras de SLA.

[1] Propriedade que permite alterar dinamicamente a capacidade da infraestrutura.

2. Trabalhos Relacionados

Gerência de recursos em infraestruturas virtualizadas é um tema abordado por diversas pesquisas, que em geral levam em consideração uma ou mais das seguintes atividades:

- Planejamento de capacidade a longo prazo: decide quantos recursos são necessários para compor a infraestrutura no longo prazo (pelo menos um ano à frente) [Maciel-Jr. et al. 2011];
- Planejamento de capacidade a médio prazo: decide quantos recursos físicos devem ser ligados nas próximas horas ou dias, com o objetivo principal de reduzir custos de energia [Jiang et al. 2012];
- Planejamento de capacidade a curto prazo: decide quantas máquinas virtuais (VM, do inglês *virtual machine*) são necessárias para implantar uma dada aplicação, ou serviço, nos próximos minutos [Calcavecchia et al. 2012, Urgaonkar et al. 2008, Marshall et al. 2010, Sharma et al. 2011];
- Escalonamento de VM: decide onde criar as VMs que são necessárias, dentre os recursos físicos disponíveis, tentando maximizar a utilização de recursos e reduzir os custos, sem comprometer o desempenho [Shen et al. 2011, Almeida et al. 2010, Han et al. 2012].

Todas estas áreas estão fortemente relacionadas. A gerência de capacidade a curto prazo precisa instanciar VMs sobre os recursos físicos disponíveis, o que é realizado pelo gerente de escalonamento de VM. O gerente de escalonamento de VM utiliza os recursos físicos ligados de acordo com a decisão tomada pelo gerente de capacidade a médio prazo que, por sua vez, controla os recursos físicos disponíveis decorrentes da decisão tomada pelo gerente de capacidade a longo prazo. Este trabalho concentra-se no planejamento a curto prazo.

Ao analisar em profundidade soluções de planejamento de capacidade a curto prazo, podemos compará-las segundo dois aspectos: o modo de operação e o nível de intrusão. Alguns gerentes são proativos em seu modo de operação [Sharma et al. 2011], tentando estimar cargas futuras e aumentando ou diminuindo a capacidade dos serviços quando necessário. Outros, apenas reagem a mudanças na demanda [Marshall et al. 2010, Calcavecchia et al. 2012, Merino et al. 2010], e outros seguem uma abordagem híbrida [Urgaonkar et al. 2008], proativa e reativa, realizando predições de carga e reagindo, o mais rápido possível, às más decisões de planejamento de capacidade. O mecanismo proposto nesse artigo segue esta última abordagem.

Quanto ao aspecto da intrusão, alguns gerentes a curto prazo são dependentes de aplicação [Urgaonkar et al. 2008, Seung et al. 2011, Marshall et al. 2010] no sentido de que eles precisam de informações específicas sobre os serviços que estão sendo providos, como tempos de resposta, tamanhos de filas, taxas de chegada e demandas de requisições. Esta exigência não só levanta questões de privacidade, como também traz mais complexidade à estrutura de gestão, uma vez que o serviço deve ser instrumentado para ser monitorado corretamente.

Por outro lado, existem gerentes que requerem conhecimento sobre o que está sendo provisionado. Por exemplo, gerentes a curto prazo que necessitam apenas do monitoramento da carga das VMs, mas assumem que os serviços seguem a estrutura característica de processamento em lote (do inglês *batch*) [Calcavecchia et al. 2012]. Nós

propomos um mecanismo que é independente do tipo de aplicação e que não exige informações específicas sobre a mesma, nem possui o requisito de que o serviço seja de um determinado tipo. Nosso mecanismo necessita apenas do histórico de utilização da infraestrutura virtual, o que pode ser facilmente obtido no nível da VM.

Até onde sabemos, quando considerado o provisionamento horizontal, o mecanismo de provisionamento automático que propomos nesse artigo é o primeiro a ser totalmente independente do serviço que está sendo provido. Além disso, ele segue um modo de operação proativo e reativo, onde o comportamento proativo considera a utilização de múltiplos preditores, criteriosamente selecionados de tempos em tempos para prover estimativas das demandas futuras do serviço, permitindo ao provisionamento automático a capacidade de adaptação a mudanças no padrão das cargas.

3. Definição do Problema

Consideramos que o centro de processamento de dados que dá suporte ao modelo de IaaS é composto por um número de servidores que utilizam alguma tecnologia de virtualização. Esses servidores abrigam as VMs que executam os serviços dos clientes do provedor de IaaS. O centro de processamento de dados define um conjunto de tipos de instâncias que podem ser providas por esses servidores. Cada tipo de instância caracteriza uma VM em termos de capacidade de recursos (CPU, memória RAM, largura de banda, etc.).

Os serviços, ou aplicações, que executam na infraestrutura são compostos por uma ou mais camadas. Cada uma destas camadas pode apresentar cargas de trabalho que variam no tempo, o que significa que a quantidade de VMs necessárias para executar adequadamente uma camada do serviço pode variar com o tempo. Desta forma, cada camada do serviço é vista como um componente que deve ser provisionado independentemente. Assumimos que cargas de trabalho são escaláveis horizontalmente e que um serviço de balanceamento de carga é capaz de balancear adequadamente a demanda da camada em todas as VMs que foram alocadas para essa camada. Por uma questão de simplicidade, assume-se que cada camada do serviço está associada a um tipo de instância, ou seja, uma camada do serviço só pode ser executada num único tipo de instância, enquanto camadas diferentes podem ser executadas por tipos diferentes.

Cada camada do serviço está associada a um ou mais objetivos de nível de serviço (SLO, do inglês *service level objective*). Assumimos que existe um mapeamento que relaciona a satisfação dos SLOs com a utilização de recursos das VMs alocadas ao serviço, de tal forma que se a utilização do recurso é mantida abaixo de um valor alvo na maior parte do tempo, então os SLOs da camada do serviço são satisfeitos. É certo que, quanto mais próxima do alvo a utilização de recursos for mantida, menor é o número de máquinas virtuais necessárias e menor é o custo para executar esta camada do serviço com o nível de desempenho desejado.

Portanto, o objetivo do mecanismo de provisionamento automático é realizar o planejamento de capacidade a curto prazo para cada camada do serviço em execução no centro de processamento de dados, com o intuito de minimizar a quantidade de recursos ativos, além de satisfazer os SLOs das camadas.

4. Um Arcabouço Para Provisionamento Automático de Recursos

4.1. Arquitetura

A nossa solução baseia-se num laço de controle com retroalimentação (do inglês *feedback control loop*). A Figura 1 mostra a arquitetura conceitual. Adicionamos ao sistema um monitor responsável por coletar informações sobre a utilização dos recursos no nível de VM. Ou seja, coletar informações sobre a utilização de cada VM ativa. Assumimos que o monitor é capaz de monitorar um grupo restrito de recursos como consumo de CPU, memória, largura de banda, etc. Informações sobre diferentes recursos podem ser coletadas para cada camada do serviço com periodicidade configurável. A informação coletada é periodicamente comparada a valores de referência (por exemplo, nível desejado de utilização) definidos pelo administrador da infraestrutura e pelo cliente. As diferenças entre os valores coletados e os de referência, ou seja, os erros medidos, são repassados ao controlador, que atua no sistema para trazê-lo a um estado desejado, no qual tais erros se aproximem de zero. No nosso caso, o controlador atua no sistema iniciando novas VMs ou parando VMs em execução para manter a utilização dos recursos o mais próximo possível dos valores alvo associados a esses recursos.

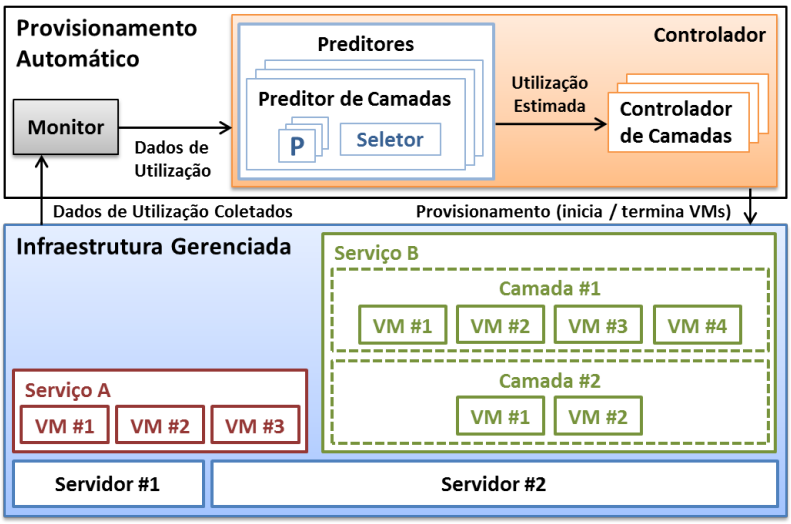


Figura 1. Arquitetura do Arcabouço para Provisionamento Automático

O controlador é composto por controladores de camada que executam periodicamente, em intervalos configuráveis, para realizar planejamento de capacidade a curto prazo de cada camada. Os controladores de camada operam de forma reativa e proativa. O comportamento reativo é implementado definindo ações associadas a cada tipo de recursos usado numa determinada camada. Essas ações são executadas toda vez que a utilização dos recursos atingir um dado limite. O comportamento proativo, por sua vez, é implementado da seguinte forma: para cada camada do serviço e tipo de recurso usado para controlar uma camada em particular, existe um conjunto de preditores para estimar a utilização futura do recurso para a camada. Esses preditores consomem a informação coletada sobre a utilização dos recursos e, baseados nessa informação, estimam a demanda futura da camada para o recurso específico. Esses preditores diferem apenas na estratégia de predição usada, ou seja, como o preditor usa a informação coletada para estimar a carga futura. Preditores se registram no monitor do sistema para receber as informações

específicas que necessitam. Controladores de camadas diferentes podem usar diferentes conjuntos de preditores.

Em cada instante de tempo e para cada tipo de recurso usado para controlar a capacidade de uma camada, apenas um dos preditores independentes é usado para estimar a utilização futura de um dado recurso. De tempos em tempos, um módulo seletor tenta descobrir qual o preditor mais apropriado para usar num futuro próximo, dentre o conjunto de preditores disponíveis para uma camada particular. Essa seleção é importante pois, como mostraremos mais à frente, não há um único preditor que seja eficiente para estimar a ampla variedade de padrões de utilização existentes na prática, tanto ao comparar camadas distintas, como para uma única camada, especialmente ao considerar janelas de tempo mais longas.

Cada controlador de camada calcula o número de VMs necessárias no próximo intervalo de controle para manter a camada em execução, baseado nos valores de referência de utilização e nas estimativas para futuros valores de demanda dos recursos usados pela camada. Se a quantidade necessária de VMs é maior do que a atualmente alocada para a execução da camada, VMs adicionais são iniciadas. Caso contrário, a ação a ser tomada depende do tipo de infraestrutura provida. Em um modelo público de IaaS, por exemplo, os provedores cobram geralmente um preço fixo por VM usada numa unidade de tempo pré-definida, por exemplo US$ 0.10 por hora. Nesse caso, o controlador deve somente desligar VMs que estejam próximas de completar a hora. Por outro lado, num modelo privado de IaaS, VMs devem ser desligadas o mais cedo possível. Obviamente, deve haver algum tipo de orquestração entre os controladores de camadas, tal que decisões tomadas por um controlador não afetem outras camadas do mesmo serviço e que pedidos simultâneos de criação de VMs sejam devidamente satisfeitos pela infraestrutura física. Evidentemente, modelos de orquestração mais sofisticados podem ser implementados de modo a melhorar a eficiência do sistema. Isso, no entanto, está fora do escopo desse trabalho.

Diferente de outras abordagens, nossa solução realiza planejamento de capacidade baseado somente em informações históricas da infraestrutura usada pelas camadas do serviço e não requer nenhuma informação sobre o serviço em si, além do mapeamento entre as configurações particulares de VMs e as camadas do serviço. Nesse sentido, a solução é independente do serviço em execução na infraestrutura, além de prover mais flexibilidade na forma com a qual as predições são feitas, dado que é possível incorporar quantos preditores se queira, por camada, sem a necessidade de alterar o funcionamento do seletor.

4.2. Escolhendo Dinamicamente Entre Múltiplos Preditores

4.2.1. Modelo de Simulação

A decisão de usar múltiplos preditores selecionados dinamicamente ao longo do tempo baseia-se em evidências empíricas de que é melhor do que usar um único preditor, mesmo que este preditor combine características de diversos outros, como quando é utilizada uma abordagem de agregação [Jiang et al. 2012]. O impacto dessa decisão não inviabiliza o uso do arcabouço, visto que experimentos usando a linguagem R [Chambers 2012] mostraram que em 50% dos casos o tempo de predição é inferior a $0,15$ segundos e em 99% dos casos não é maior que $13,5$ segundos.

Implementamos um modelo de simulação do arcabouço apresentado na subseção anterior também usando a linguagem R [2]. O simulador é alimentado com arquivos de dados de utilização de aplicações, em produção, de clientes da HP. Consideramos que cada arquivo de dados corresponde a uma camada diferente do serviço. Visto que cada controlador de camada é independente e que o planejamento de capacidade do provedor IaaS não é nosso foco, cada controlador de camada é simulado separadamente.

Os arquivos de dados provêm itens coletados a cada 5 minutos. Cada item é um par (u, c), onde u é a utilização média de CPU no intervalo de 5 minutos e c é o número de núcleos de CPU usados. Um novo par lido é usado para computar a utilização real das VMs no sistema simulado para os próximos 5 minutos. Consideramos VMs com um único núcleo de CPU. Seja t o instante de tempo no qual um novo par (u, c) é lido e a o número de VMs alocadas para executar a camada do serviço no experimento de simulação, a utilização real de cada VM do instante de tempo t até $t + 5$ minutos é computada como o mínimo entre 100% e $u \cdot c/a$. Além disso, se $u \cdot c/a > 1$, então o excesso da demanda que não pôde ser alocado no tempo t (dado por $1 - u \cdot c/a$) é adicionado à demanda do próximo intervalo de tempo, que ocorre em $t + 5$ minutos.

A decisão de adicionar ou remover VMs pode ser tomada proativamente a cada 5 minutos ou reativamente toda vez que uma determinada condição acontecer. Por exemplo, a utilização de CPU está acima de um dado limite. No entanto, dado que é necessário um tempo até que o provisionamento das novas VMs tenha efeito, no modelo de simulação consideramos que se um controlador de camada decidir no tempo t que uma nova VM deve ser alocada, tal VM estará de fato operacional apenas no tempo $t + 5$. Dessa forma, no caso proativo, os preditores no controlador de camada usam informações históricas coletadas até o tempo t para estimar a demanda no tempo $t + 5$ minutos e usam essa estimativa para decidir, no tempo t, quantas VMs devem estar executando no tempo $t + 5$ minutos. Portanto, qualquer quantidade extra de VMs necessária no tempo $t + 5$ minutos é requisitada no tempo t. Foram utilizados intervalos de 5 minutos em conformidade com os dados de utilização usados para avaliar o arcabouço. No entanto, esse intervalo é configurável e, idealmente, pode ser um valor representativo do tempo necessário para tornar uma nova VM operacional.

Nos experimentos de simulação reportados nesse artigo, consideramos apenas o modelo público de IaaS e um modelo de tarifação que cobra por hora pelo uso de VMs. Ou seja, VMs são terminadas somente em horas completas, como discutido na Subseção 4.1.

4.2.2. Camadas diferentes necessitam de diferentes preditores

Para esse estudo inicial, cada controlador de camada tem um único preditor, o que torna o mecanismo de seleção trivial. O controlador trabalha para manter a utilização de CPU abaixo do limite de 70%. Avaliamos 5 preditores populares, baseados em: autocorrelação (AC), regressão linear (LR), autorregressão (AR), autorregressão com média móvel integrada (ARIMA) e no valor da medição anterior (LW, do inglês *last window*). Também avaliamos um sexto preditor que usa os outros 5 preditores em conjunto (EN), como proposto por Jiang et al. [Jiang et al. 2012].

[2]O código-fonte encontra-se disponível no endereço http://www.lsd.ufcg.edu.br/∼fabio/autoflex.

Usamos dois conjuntos de dados de utilização de produção, um com 265 arquivos com um mês de duração e outro com 33 arquivos com média de 8 meses de duração. A função distribuição acumulada das utilizações de CPU desses dados é respectivamente mostrada nas Figuras 2(a) e 2(b). Nota-se que os dados apresentam uma ampla variedade de distribuições de utilização em ambos os casos. Isso também acontece para a função distribuição acumulada (FDA) para as variações de utilização de CPU, definidas como a diferença entre as utilizações no tempo t e no tempo $t + 5$ minutos. Dessa forma, consideramos que os dados utilizados no estudo são representativos.

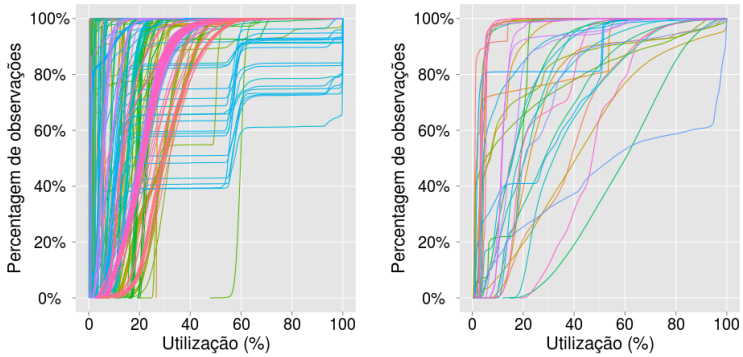


Figura 2. FDA da utilização de CPU para os dois conjuntos de dados de utilização, com duração de um mês (a) e com duração média de 8 meses (b).

A partir da simulação de cada um dos 265 arquivos, com um mês de duração, medimos o número de intervalos de 5 minutos em que a utilização de CPU foi de 100% e havia demanda que não pôde ser atendida, e foi, portanto, deslocada para o próximo intervalo. Isso representa o número de intervalos de tempo onde a capacidade provida não foi suficiente para suprir a demanda do serviço[3]. Chamamos esses eventos de *violações* de SLO do serviço. Medimos também a economia conseguida quando comparamos o provisionamento automático com uma abordagem que realiza provisionamento perfeito, ou seja, uma abordagem que conhece a demanda mais alta no período simulado e estaticamente aloca recursos tal que a utilização seja sempre menor que um dado limite (70% em nossos experimentos). Essas métricas foram utilizadas independentemente para comparar o desempenho dos vários preditores usados. Na Figura 3 apresentamos, para cada preditor, a percentagem dos dados de utilização para os quais um dado preditor foi avaliado como o melhor[4], considerando separadamente ambas as métricas.

Como pode ser visto, quando consideramos o número de violações de SLO, o preditor com melhor desempenho responde apenas por 44% dos casos, enquanto esse número aumenta para 64% quando usamos a métrica de custo para ordenar os preditores. Dessa forma, é possível perceber que utilizar múltiplos preditores tem um impacto direto no desempenho alcançado com o mecanismo.

[3]Ressaltamos que só foi possível computar essa métrica por se tratar de um experimento simulado. Em um ambiente real, não há como medir a demanda real quando a utilização está em 100%.

[4]Lembramos que a soma das partes no gráfico pode ser maior que 100% dado que, para alguns dos dados, vários preditores foram igualmente bons.

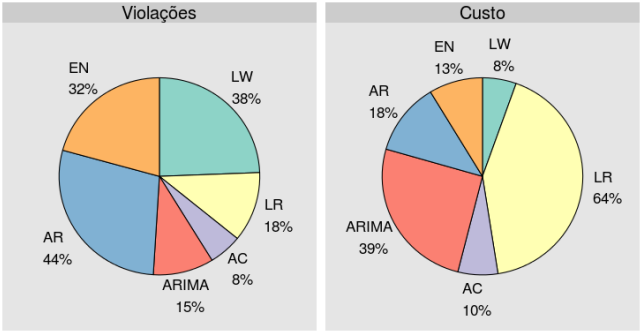


Figura 3. Percentagem de vezes em que um dado preditor é avaliado como o melhor, considerando os 265 diferentes arquivos de dados de utilização

4.2.3. Uma Camada Requer Diferentes Preditores ao Longo do Tempo

Avaliamos também como a qualidade de um preditor varia ao longo do tempo ao predizer demandas de um mesmo serviço. Para tal, alimentamos as simulações com os 33 arquivos de dados de produção que, em média, possuem aproximadamente 8 meses de duração.

Nesse caso, o conjunto de preditores incluem todos os seis preditores usados no experimento anterior. Um novo preditor é selecionado no início de todo período de 24 horas. Um seletor perfeito é usado para escolher o preditor mais apropriado. Isso foi possível alimentando os preditores com dados de utilização das duas semanas anteriores e usando a utilização do próximo dia para medir o desempenho de cada preditor[5]. O seletor então escolhe o melhor preditor de acordo com a métrica definida. As Figuras 4(a) e 4(b) apresentam, respectivamente, as funções distribuição acumulada do número de diferentes preditores escolhidos por arquivo de dados e do número de vezes que o preditor a ser usado é trocado pelo seletor ao longo do tempo, para os 33 arquivos simulados.

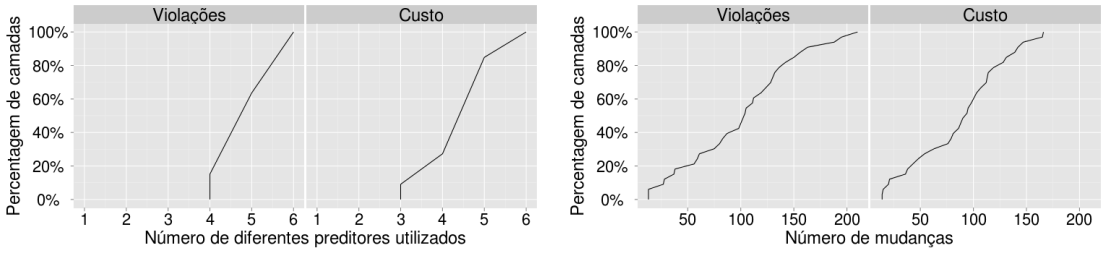


Figura 4. Número de diferentes preditores escolhidos pelo seletor (a) e a quantidade de vezes que o seletor troca de preditor (b).

Podemos ver que no mínimo 3 preditores diferentes são sempre utilizados e que o número de vezes que o seletor opta por trocar o preditor utilizado varia de dezenas a centenas de vezes durante o período de 8 meses. Isso indica que o desempenho do mecanismo pode ser melhorado se os preditores forem dinamicamente selecionados para uso à medida que o tempo avança e o padrão de utilização muda.

[5]Note que esse seletor não é implementável dado que ele requer conhecimento perfeito sobre utilização futura; ainda assim, ele é útil para o propósito de avaliação que realizamos nesse experimento.

5. Instanciação e Avaliação do Mecanismo de Provisionamento Automático

Nessa seção avaliamos uma implementação do arcabouço de uso geral proposto na seção anterior.

A fim de definir com precisão como o controlador irá operar, a instanciação do arcabouço requer as seguintes ações: (i) definir os limites e as ações associadas ao comportamento reativo do controlador; (ii) prover implementações para o conjunto de preditores a serem utilizados por cada controlador de camada; (iii) prover a implementação de um mecanismo de seleção para dinamicamente escolher o preditor a ser usado em cada instante de tempo; (iv) definir os valores apropriados para os parâmetros do arcabouço, tais como a periodicidade de execução dos controladores de camada e dos seletores e a frequência com que as informações monitoradas são coletadas.

A seguir, propomos uma maneira de implementar o seletor, preditores que tentam reduzir os erros que conduzem ao subprovisionamento e ações reativas simples. Desta forma, instanciamos o arcabouço de provisionamento automático visando reduzir substancialmente o número de violações, mas sem comprometer demasiadamente a economia de recursos que pode ser alcançada através da abordagem de provisionamento automático.

5.1. Como Selecionar o Preditor Apropriado?

Nossa implementação do modulo de seleção de preditor usa dados históricos de utilização para avaliar qual dentre os preditores disponíveis teria o melhor desempenho, com base no passado recente. Para isso, nós utilizamos as últimas três semanas de histórico de utilização. As duas semanas mais antigas são utilizadas para alimentar os preditores, enquanto que a semana mais recente de dados é utilizada como base para avaliar o desempenho dos preditores. O preditor que proporciona o melhor desempenho em termos do número de violações (e de custos, para o caso de empates) é selecionado para ser utilizado até que uma nova seleção seja realizada. Nos experimentos reportados a seguir uma nova seleção é realizada a cada dia. Outros valores de periodicidade de seleção foram experimentados, no entanto, usando o teste estatístico de Kruskal-Wallis [Kruskal and Wallis 1952], não foram constatadas diferenças significativas nos resultados obtidos para o número de violações de SLO.

Para avaliar o desempenho deste seletor, executamos simulações usando as mesmas configurações descritas da Subseção 4.2.3. Além do seletor apresentado anteriormente, consideramos para comparação dois outros seletores: o seletor perfeito, não realista, descrito na Subseção 4.2.3 e um seletor randômico, que aleatoriamente seleciona o preditor para ser usado. Estes podem ser vistos como limites superior e inferior, respectivamente, sobre o desempenho que pode ser obtido pelo nosso seletor. Na Figura 5 apresentamos o diagrama de caixa do número de violações ao simular as 33 cargas de trabalho de longa duração. Como pode-se ver, o método proposto é ligeiramente pior que o perfeito e muito melhor que a seleção randômica. Para todos os casos, a redução de custos quando comparada com um sistema superprovisionado é em torno de 65%.

5.2. Melhorando Predições Para Evitar Subestimativas

Verificamos que mesmo se o seletor perfeito for utilizado com o conjunto padrão de preditores considerado nesse artigo, o número de violações ainda pode ser muito elevado.

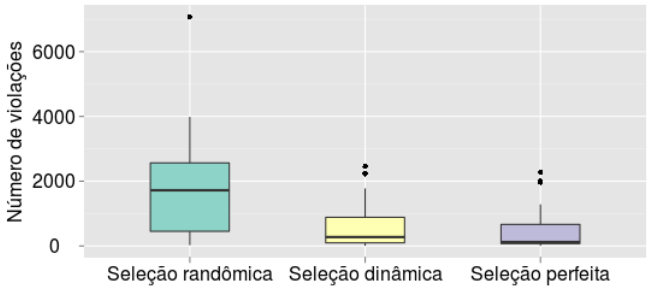


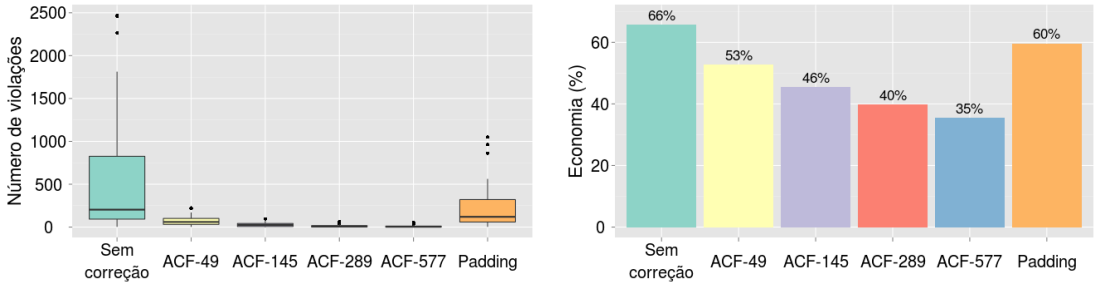
Figura 5. Avaliação do método de seleção dinâmica, de acordo com o número de violações, em comparação com a seleção randômica e a seleção perfeita.

Violações de SLO podem levar a quebras de SLA, que em geral geram custos elevados. Assim, em alguns casos, é importante reduzir, tanto quanto possível, o número de tais eventos, mesmo que isso implique em um aumento no custo final de provisionamento. Uma maneira de alcançar esse objetivo no provisionamento automático é através da utilização de preditores mais conservadores. Essencialmente, esses são preditores que tentam corrigir suas predições com base nos erros cometidos anteriormente. Em particular, considerando-se os erros que conduzem à subestimativa da capacidade necessária e consequentemente à ocorrência de violações [Gong et al. 2010].

Aqui propõe-se uma abordagem para corrigir predições, de modo a reduzir o número de subestimativas. O processo de correção calcula o histórico de erros de predição e tenta correlacionar a sequência recente de erros de predição com sequências de erros de predições no passado, através de uma função de autocorrelação (ACF, do inglês *autocorrelation function*). O ponto do passado em que a correlação é máxima é utilizado como o ponto central de uma janela de valores de erro. Então, o maior erro negativo dentro desta janela é utilizado para corrigir a predição seguinte. Tanto o tamanho da sequência, como o tamanho da janela são parâmetros do mecanismo. Utilizamos um tamanho de sequência de 2 semanas e valores para o tamanho da janela que variam de 4 horas a 2 dias.

A Figura 6(a) mostra o diagrama de caixa do número de violações em simulações alimentadas com as cargas de trabalho de longa duração, utilizando uma configuração semelhante ao experimento anterior, mas incluindo, além dos seis preditores originais, versões de todos os preditores com o método de correção proposto acima e utilizando diferentes tamanhos de janela (referenciados na figura como "ACF-x", onde x é o tamanho da janela em intervalos de 5 minutos). Mostramos também o desempenho alcançado comparado ao método de correção proposto por Gong et al. [Gong et al. 2010] (referenciado como "padding"). Embora este método tenha sido proposto em um contexto diferente, ele é utilizado aqui para comparação, pois não encontramos outro método usado no mesmo contexto do nosso.

Pode-se observar que todos os métodos de correção são capazes de reduzir substancialmente o número de violações, sendo o método "ACF-x" mais eficiente para a configuração que nós consideramos. Conforme esperado, o número de violações do método "ACF-x" melhora quando o valor de x aumenta. A Figura 6(b) mostra a redução do custo correspondente para o mesmo experimento. É possível observar que a redução no número de violações ocorre às custas de um aumento no custo final de provisionamento.



(a) (b)

Figura 6. Comparação dos dois métodos de correção de predição com relação ao número de violações (a) e à economia de recursos (b).

5.3. Agrupando Todos os Mecanismos

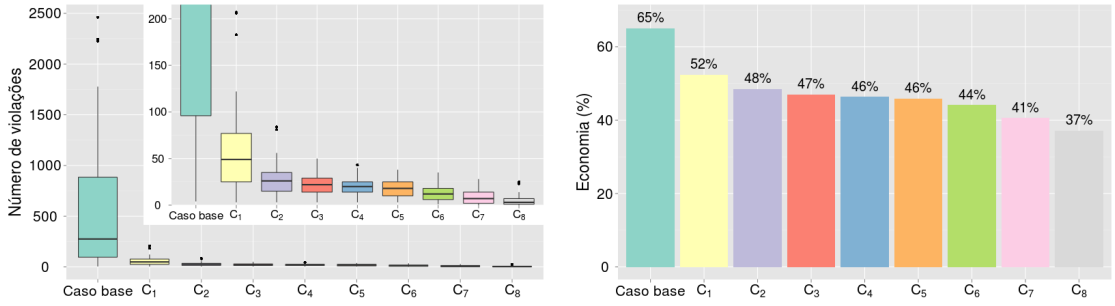
É possível ver a correção de predições como um novo tipo de preditor. Ao fazer isso, consideramos as correções no arcabouço proposto, simplesmente incluindo no conjunto de preditores disponíveis alguns preditores que implementam procedimentos de correção. Logo, resultados diferentes podem ser obtidos quando se é mais ou menos conservador na configuração da instância do arcabouço de provisionamento automático.

Em seguida analisamos a relação de compromisso entre a redução do número de violações e a redução do custo de provisionamento envolvida na configuração de uma instância do arcabouço proposto. Executamos experimentos de simulação que realizam provisionamento automático proativo utilizando o seletor apresentado na Seção 5.1 e consideram diferentes conjuntos de preditores. Seja P qualquer um dos preditores no conjunto $\{$AC, LR, AR, ARIMA, LW, EN$\}$, e P-x um preditor que utiliza o preditor P e realiza correções através do método "ACF-x" com uma janela de tamanho x, utilizamos os seguintes conjuntos de preditores: C_1=$\{$P, P-49$\}$, C_2=$\{$P, P-49, P-145$\}$, C_3=$\{$P, P-49, P-145, P-289$\}$, C_4=$\{$P, P-49, P-145, P-289, P-577$\}$, C_5=$\{$P-49, P-145, P-289, P-577$\}$, C_6=$\{$P-145, P-289, P-577$\}$, C_7=$\{$P-289, P-577$\}$, C_8=$\{$P-577$\}$. Em todos os casos, o provisionamento automático reativo é disparado sempre que uma violação ocorre. Neste caso, o preditor que realiza as maiores correções é utilizado até que o período corrente de 24 horas termine e uma nova seleção seja realizada pelo mecanismo de provisionamento automático proativo.

A Figura 7 apresenta o desempenho de diferentes configurações com relação ao número de violações e à redução de custo. Nós também apresentamos o desempenho da configuração que não realiza correção de erros de predição, para servir como caso base para comparação.

Como pode ser visto, as configurações que contêm P-577 são aquelas que proporcionam as maiores reduções no número de violações. Em média, quando comparado com o caso base, elas produzem um número de violações que é entre 31 e 122 vezes menor. Isto vem com uma redução na economia média dos custos alcançados que varia de 29% a 43%, quando comparado com a redução na economia média dos custos obtidos no cenário do caso base, mas que ainda rende substanciais economias de custo, variando de 37% a 46%. Por outro lado, as outras configurações apresentam menores reduções do número de violações, variando de 10 a 27 vezes, com reduções na economia média dos custos

alcançados de não mais que 28%.



(a) (b)

Figura 7. Comparação de diferentes configurações com relação ao número de violações (a) e à redução de custos (b).

6. Conclusão e Trabalhos Futuros

Propusemos neste artigo um novo arcabouço para a implementação de mecanismos de provisionamento automático que seguem abordagens tanto reativas quando proativas, e que é independente do tipo de aplicação. A flexibilidade e a eficiência do arcabouço foram demonstradas por meio de experimentos de simulação conduzidos com o uso de dados de utilização de aplicações de clientes da HP. Os resultados destes experimentos mostram que uma configuração adequada do arcabouço proposto é capaz de reduzir a probabilidade média de ocorrência de violações para algo um pouco inferior a $0,008\%$, e gerando uma economia média de custos de 37%, quando comparado a um sistema super-provisionado. Além disto, para esse caso a probabilidade de ocorrer violações é sempre mantida abaixo de $0,036\%$. Economias mais substanciais podem ser alcançadas com um pequeno aumento na probabilidade de ocorrência de violações.

Como trabalho futuro, estamos avaliando como o desempenho do mecanismo de provisionamento automático pode ser melhorado através da combinação de dados de utilização de vários recursos diferentes. Além disso, estamos implementando o arcabouço proposto para avaliar seu desempenho em um ambiente real.

Agradecimentos

Essa pesquisa foi realizada em cooperação com Hewlett-Packard Brasil Ltda usando incentivos da Lei de Informática (Lei N° 8.248 de 1991). Francisco Brasileiro é pesquisador do CNPq (processo 305858/2010-6).

Referências

Almeida, J., Almeida, V., Ardagna, D., Cunha, I., Francalanci, C., and Trubian, M. (2010). Joint admission control and resource allocation in virtualized servers. *J. Parallel Distrib. Comput.*, 70(4):344–362.

Amazon (2012). Amazon auto scaling. `http://aws.amazon.com/autoscaling/`. Online; novembro, 2012.

Calcavecchia, N., Caprarescu, B., Di Nitto, E., Dubois, D., and Petcu, D. (2012). DEPAS: a decentralized probabilistic algorithm for auto-scaling. *Computing*, 94:701–730.

Chambers, J. (2012). The R project for statistical computing. http://www.r-project.org/. Accessed: 19/11/2012.

Gong, Z., Gu, X., and Wilkes, J. (2010). PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 9–16. IEEE.

Han, R., Guo, L., Ghanem, M. M., and Guo, Y. (2012). Lightweight resource scaling for cloud applications. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 644–651, Washington, DC, USA.

Jiang, Y., Perng, C.-s., Li, T., and Chang, R. (2012). Self-adaptive cloud capacity planning. In *Proceedings of the 2012 IEEE Ninth International Conference on Services Computing*, pages 73–80, Washington, DC, USA.

Kruskal, W. and Wallis, W. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621.

Maciel-Jr., P. D., Brasileiro, F. V., Lopes, R. V., Carvalho, M., and Mowbray, M. (2011). Evaluating the impact of planning long-term contracts on the management of a hybrid IT infrastructure. In *Integrated Network Management*, pages 89–96.

Marshall, P., Keahey, K., and Freeman, T. (2010). Elastic site: Using clouds to elastically extend site resources. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 43–52, Washington, DC, USA.

Merino, L. R., Vaquero, L. M., Gil, V., Galán, F., Fontán, J., Montero, R. S., and Llorente, I. M. (2010). From infrastructure delivery to service management in clouds. *Future Generation Computer Systems*, 26(8):1226 – 1240.

RightScale (2012). Right scale cloud management. http://www.rightscale.com. Online; novembro, 2012.

Seung, Y., Lam, T., Li, L. E., and Woo, T. (2011). Cloudflex: Seamless scaling of enterprise applications into the cloud. In *INFOCOM*, pages 211–215.

Sharma, U., Shenoy, P., Sahu, S., and Shaikh, A. (2011). A cost-aware elasticity provisioning system for the cloud. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, pages 559–570, Washington, DC, USA.

Shen, Z., Subbiah, S., Gu, X., and Wilkes, J. (2011). CloudScale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, pages 5:1–5:14, New York, NY, USA.

Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., and Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. *ACM Trans. Auton. Adapt. Syst.*, 3(1):1:1–1:39.

Vijayakumar, S., Zhu, Q., and Agrawal, G. (2010). Dynamic resource provisioning for data streaming applications in a cloud environment. In *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 441–448, Washington, DC, USA.

Diagnóstico do Provisionamento de Recursos para Máquinas Virtuais em Nuvens IaaS

Ricardo J. Pfitscher[1], **Maurício A. Pillon**[1], **Rafael R. Obelheiro**[1]

[1]PPGCA - Programa de Pós-Graduação em Computação Aplicada – Universidade do Estado de Santa Catarina (UDESC/Joinville)

ricardo.pfitscher@gmail.com,{mpillon,rro}@joinville.udesc.br

Abstract. *Infrastructure-as-a-service (IaaS) clouds enable customers to allocate computing resources in a flexible manner to satisfy their needs, and pay only for the allocated resources. One of the challenges for IaaS customers is the correct provisioning of their resources. Many users end up underprovisioning, hurting application performance, or overprovisioning, paying for resources that are not really necessary. Our work uses monitoring to enable a cloud customer to determine if the resources available to his virtual machines are correctly provisioned, or are under-/overprovisioned. Our focus is on processor and network resources, which can be easily provisioned in current virtualization environments. Experimental results with the Xen platform demonstrate the effectiveness of the proposed approach.*

Resumo. *Nuvens infrastructure-as-a-service (IaaS) permitem que clientes aloquem recursos computacionais de forma flexível para atender suas necessidades, e paguem apenas pelos recursos alocados. Um dos desafios para clientes IaaS é o provisionamento adequado de seus recursos. Muitos usuários incorrem no subprovisionamento, prejudicando o desempenho de suas aplicações, ou no superprovisionamento, pagando por recursos que não são realmente necessários. Este trabalho usa monitoração para permitir que um cliente de nuvem determine se os recursos disponíveis para suas máquinas virtuais estão provisionados corretamente, ou se estão sub- ou superprovisionados. O foco está nos recursos de processador e rede, que podem ser facilmente reservados nos ambientes de virtualização atuais. A eficácia da abordagem proposta é demonstrada por resultados experimentais na plataforma Xen.*

1. Introdução

Um dos desafios da computação em nuvem envolve o provisionamento adequado dos recursos computacionais necessários para atender às demandas dos clientes-usuários. As nuvens trazem como grandes atrativos a elasticidade de recursos e a tarifação por recursos alocados (*pay-per-use*), transferindo os custos fixos de infraestrutura computacional do cliente para um provedor [Armbrust et al. 2010]. Provedores de *infrastructure-as-a-service* (IaaS) oferecem como recursos máquinas virtuais (processador, memória, disco) e largura de banda de rede, que podem ser alocados conforme a necessidade do cliente [Suleiman et al. 2012].

Em uma solução IaaS, usuário e provedor possuem o mesmo objetivo geral, que é o de minimizar custos, mas diferem no modo de atingir esse objetivo. O usuário obtém

um custo mais baixo alocando o mínimo de recursos suficientes para obter um desempenho aceitável. De outro lado, o provedor está interessado em maximizar a ocupação dos seus recursos físicos desde que isso não comprometa a qualidade dos serviços oferecidos. Quando a demanda por recursos é variável, e não constante, o conflito entre essas estratégias se torna evidente: se um recurso físico estiver com toda a sua capacidade alocada para clientes, ele será incapaz de acomodar picos de carga sem prejudicar o desempenho das máquinas virtuais. Em contrapartida, a existência de capacidade ociosa (reservada mas não usada) vai de encontro ao interesse do provedor.

Geralmente, o equilíbrio entre os interesses de clientes e provedores é estabelecido por acordos de nível de serviço (SLAs – *Service Level Agreements*), que definem métricas (SLIs – *Service Level Indicators*) e respectivos valores-objetivo (SLOs – *Service Level Objectives*) para que o desempenho dos recursos alocados seja considerado satisfatório [Sauvé et al. 2005]. Muitos trabalhos de pesquisa que propõem estratégias para melhorar a alocação de recursos em ambientes IaaS, tais como [Buyya et al. 2011, Beloglazov e Buyya 2010, Kundu et al. 2012], tomam a existência de SLAs como uma premissa. Essa premissa embute um outro pressuposto, o de que o cliente é capaz de identificar corretamente as suas necessidades, e expressá-las em termos de SLIs e SLOs. Na prática, porém, fixar SLOs que garantam o desempenho desejado, a um custo aceitável, para as aplicações que serão executadas na nuvem não é trivial, haja vista a falta de diretrizes confiáveis para a especificação dos parâmetros. Logo, muitos usuários acabam sub- ou superdimensionando os seus objetivos, e consequentemente os seus recursos.

Quando recursos são subprovisionados, o usuário é capaz de perceber que o desempenho está aquém do esperado, mas muitas vezes não consegue dizer quais recursos precisam de mais capacidade. Por outro lado, o efeito do superprovisionamento de recursos é percebido menos pelo desempenho e mais pelo custo financeiro. A ausência de parâmetros de comparação da relação custo $\times$ aplicação (que permitam constatar que aplicações similares conseguem obter desempenho satisfatório a um custo mais baixo) e a dificuldade em conhecer a capacidade suficiente para cada recurso (quais SLOs podem ser reduzidos?) tornam ainda mais complexa a questão.

Nesse sentido, o objetivo deste trabalho é fornecer ao cliente um diagnóstico sobre o provisionamento dos recursos para máquinas virtuais em nuvens IaaS. A ideia é monitorar métricas de desempenho específicas de cada recurso e identificar, em tempo de execução, se a alocação de recursos para a máquina virtual está adequada, subprovisionada ou superprovisionada. Esse diagnóstico serve a variados propósitos. Por exemplo, o cliente pode usá-lo para ajustar a capacidade de suas máquinas virtuais à demanda, seja redefinindo a alocação de recursos para as MVs existentes ou aumentando/diminuindo o número de MVs instanciadas. Em serviços regidos por SLA, o diagnóstico pode ser usado para refinar os SLOs definidos em contrato. O diagnóstico também pode ser usado como entrada para um mecanismo automatizado de gerência da elasticidade característica das nuvens, permitindo que recursos sejam alocados ou liberados conforme a necessidade.

Os recursos considerados neste trabalho são processador e rede, que podem facilmente ser reservados nos monitores de máquinas virtuais atuais e também podem ser objeto de SLA em alguns provedores IaaS [Suleiman et al. 2012]. As métricas de desempenho são todas coletadas dentro das máquinas virtuais, sendo portanto diretamente acessíveis ao usuário, sem a necessidade de intermediação do provedor. Além disso, elas

refletem o desempenho efetivamente verificado nas máquinas virtuais, considerando a influência tanto da capacidade do *hardware* subjacente como do compartilhamento desse *hardware* com outras MVs. As avaliações experimentais são conduzidas com Linux na plataforma Xen,[1] um dos ambientes de virtualização mais usados por provedores IaaS, sendo a principal solução de código aberto desse mercado [Butler 2012].

A principal contribuição do trabalho é a proposta de uma abordagem que permita a clientes IaaS obter um diagnóstico da adequação dos recursos alocados a suas máquinas virtuais em relação à demanda existente. Uma segunda contribuição consiste na introdução do uso do tamanho da fila do adaptador de rede como métrica auxiliar para determinar se a largura de banda disponível a um sistema é ou não suficiente.

O restante do artigo está organizado da seguinte forma. Na seção 2 são descritas as abordagens para diagnosticar o provisionamento de processador e rede, que são validadas pelos resultados experimentais mostrados na seção 3. A seção 4 discute trabalhos relacionados, e a seção 5 traz as considerações finais.

2. Diagnóstico do Provisionamento de Recursos

Para diagnosticar o provisionamento de processador e rede, é necessário definir métricas que indiquem se esse provisionamento está adequado, sub ou superprovisionado. Como o foco está no cliente, são propostas métricas que podem ser obtidas dentro de uma MV.

2.1. Processador

Em ambientes virtualizados, o monitor de máquinas virtuais (*hypervisor* no Xen) disponibiliza unidades de processamento virtuais (VCPUs) para cada domínio virtual (ou MV). Mesmo quando várias VCPUs são multiplexadas sobre o mesmo processador físico, cada VCPU é vista pela MV como um processador real. O *hypervisor* então escalona as VCPUs nas CPUs físicas de acordo com o algoritmo de escalonamento do MMV [Wood et al. 2008]. Sendo assim, a monitoração de processador realizada pelo sistema operacional (SO) da MV deve obter valores relacionados à ocupação de sua VCPU, da mesma forma que uma máquina não virtualizada monitora a sua CPU física.

Para diagnosticar a adequação do provisionamento de processador a máquinas virtuais em nuvem, serão avaliadas duas métricas: a utilização de processador (U_{cpu}) e o percentual de "roubo" (*%Steal*). A utilização U_{cpu} representa quanto tempo o processador estava executando tarefas em um espaço de tempo. No Linux, a contabilização do processador subdivide a utilização em vários componentes; por exemplo, *%User* representa o percentual de tempo em que o recurso foi utilizado por processos de usuário, e *%System* o tempo usado pelo sistema operacional [Hoch 2010]. Para simplificar, é possível usar o percentual de ociosidade (*%Idle*) para obter de forma indireta a utilização do processador. Sendo assim, a utilização de processador é dada por $U_{cpu} = 100\% - \%Idle$, sendo sua média em um período denotada por $\overline{U_{cpu}}$.

O tempo de roubo (*steal time*) é uma métrica conhecida em ambientes de *mainframe*, e que representa o percentual de tempo em que uma máquina virtual deseja executar (ou seja, possui processos em estado de pronto) mas sua VCPU não está alocada em uma CPU física, devido à contenção de processador entre MVs [van Riel 2006]. Essa

[1] `http://www.xen.org/`

métrica pode auxiliar a identificar a ausência de recursos na máquina física, de forma complementar à utilização de processador: quando *%Steal* é maior do que zero, a máquina virtual teve necessidades de execução em processador que não foram supridas, ou seja, existem máquinas concorrentes disputando o recurso.

A taxa de ocupação de processador tende a variar de acordo com a execução das aplicações no período de tempo. Logo, deve-se definir limiares para caracterizar os três níveis relacionados ao provisionamento do recurso. Entretanto, não há um consenso relacionado aos valores admitidos nos limiares; por exemplo, [Hoch 2010] aponta que valores entre 0–5% para *%Idle* indicam saturação, enquanto [Padala et al. 2007] adota 80% para utilização (20% de *%Idle*) como limite de saturação, com a justificativa de manter uma margem de segurança para acomodar o comportamento variável das aplicações. Considerando que a ocupação de CPU pode alternar entre picos de utilização, define-se o recurso como subprovisionado quando $\overline{U_{cpu}}$ mantiver-se menor ou igual a 80%.

Contudo, apesar de serem propostos valores para saturação do recurso, não há regras práticas para apontar a sua subutilização. Considerando que o valor apropriado para utilização de processador para uma MV é o equilíbrio entre o máximo e mínimo disponível, espera-se que o diagnóstico de provisionamento superdimensionado ou adequado considere a proximidade ao equílibrio (50%), permitindo que a capacidade provisionada suporte oscilações positivas e negativas de carga a um custo aceitável. Sendo assim, cenários onde a média de utilização de processador estiver entre 50% e 80% serão diagnosticados como adequados, pois não há carga suficiente para saturar o recurso e a redução da capacidade pode afetar o desempenho. Nos cenários com utilização inferior a 50%, a média de utilização pode não expressar corretamente um provisionamento superdimensionado, pois a carga pode ter $\overline{U_{cpu}}$ baixa mas apresentar picos de utilização que impeçam uma redução no recurso alocado (caso de C3 na Fig. 1(a)).

Para lidar com esse caso, avalia-se a variabilidade da utilização nas situações em que $\overline{U_{cpu}}$ é menor que 50%, verificando qual a fração de pontos que atingem o pico de utilização do recurso no intervalo monitorado. Seja U^i_{cpu} um ponto de medição de U_{cpu}, θ o limiar de saturação de CPU (i.e., $U^i_{cpu} \geq \theta$ indica CPU saturada), N o número de pontos no intervalo e N' o número de pontos com CPU saturada ($U^i_{cpu} \geq \theta$). Quando $N'/N > \omega$, onde ω é o limiar para o percentual de pontos em saturação, a MV utiliza toda a capacidade de CPU com frequência, e o provisionamento é adequado; caso contrário, ele é superdimensionado. O diagnóstico de provisionamento de processador é expresso pelas equações (1)–(7). Com base em observações experimentais, os limiares adotados foram $\theta = 95\%$ e $\omega = 20\%$, mas esses valores podem ser ajustados.

$$u_i = \begin{cases} 1, & \text{se } U^i_{cpu} \geq \theta \\ 0, & \text{se } U^i_{cpu} < \theta \end{cases} \quad (1) \qquad N' = \sum_{i=0}^{N} u_i \quad (2) \qquad \varphi = \frac{N'}{N} \quad (3)$$

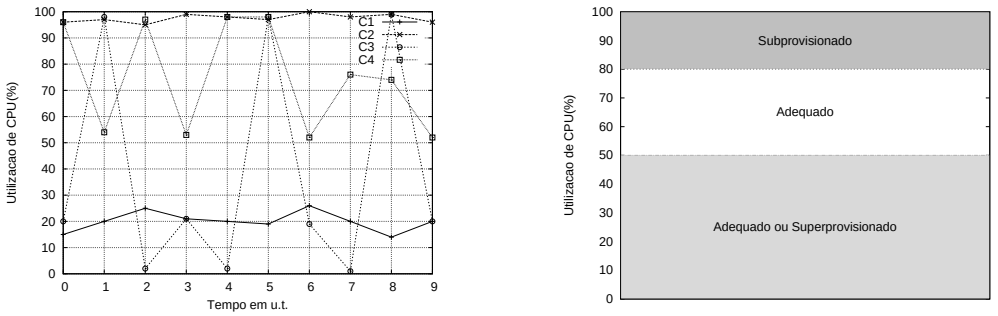
$$\overline{U_{cpu}} \geq 80\% \Rightarrow \text{Subprovisionado} \quad (4)$$

$$50\% \leq \overline{U_{cpu}} < 80\% \Rightarrow \text{Adequado} \quad (5)$$

$$\overline{U_{cpu}} < 50\% \wedge \varphi > \omega \Rightarrow \text{Adequado} \quad (6)$$

$$\overline{U_{cpu}} < 50\% \wedge \varphi \leq \omega \Rightarrow \text{Superprovisionado} \quad (7)$$

Para exemplificar, seja um cenário hipotético com 10 medições de U^i_{cpu} associadas a quatro cargas de trabalho distintas (C1–C4), mostrado na Fig. 1(a). Observa-se que, para C1, a média de U_{cpu} no período é de 20%, logo o processador está superprovisionado. Para C2, tem-se $\overline{U_{cpu}}$ próxima a 100%, ou seja, processador subprovisionado. A carga de trabalho C3 apresenta média de U_{cpu} igual a 38%, com quatro pontos onde a utilização é próximo ao topo ($\varphi = 40\%$); assim, o provisionamento é adequado, pois o recurso alocado suporta a variação entre picos de utilização e ociosidade. A carga C4 apresenta média de U_{cpu} igual a 75%, e assim o provisionamento é considerado adequado.



(a) Cenário hipotético (b) Limiares

Figura 1. Utilização de CPU em um cenário hipotético 1(a) e Limiares para diagnóstico de provisionamento 1(b). C1 está superprovisionado, C2 está subprovisionado, C3 e C4 têm provisionamento adequado.

A Fig. 1(b) resume os possíveis diagnósticos do provisionamento de processador com base em medições de U_{cpu}. No caso de utilização média situada na região superior ($\overline{U_{cpu}} \geq 80\%$), tem-se subprovisionamento. Para utilização na região intermediária ($50\% \leq \overline{U_{cpu}} < 80\%$), o provisionamento é adequado. Caso a utilização média corresponda à região inferior ($\overline{U_{cpu}} < 50\%$), o recurso pode ser adequado (quando $\varphi > \omega$) ou superprovisionado (quando $\varphi \leq \omega$).

2.2. Rede

Em um ambiente de virtualização, as máquinas virtuais compartilham a largura de banda das interfaces de rede do *hardware* subjacente. Tipicamente, o MMV permite que seja reservada uma fração da largura de banda disponível para cada MV; caso não seja feita uma alocação fixa, ocorre a multiplexação estatística do recurso, com cada MV usando a largura de banda disponível durante um tempo, conforme sua necessidade. Internamente, cada MV possui uma interface de rede virtual, que o seu SO enxerga como uma interface de rede física. O MMV é responsável por transmitir pela interface física os quadros que as MVs enviam por suas interfaces virtuais, e vice-versa [Apparao et al. 2006].

A métrica básica para verificar a adequação da capacidade de rede alocada a uma máquina virtual é a largura de banda. Considera-se apenas a largura de banda de transmissão, pois é a única que pode ser analisada com mais profundidade apenas com dados internos à MV; para a largura de banda de recepção, a análise possível é se a banda consumida se aproxima da banda alocada, não sendo possível inferir se há demanda além da alocação. Do mesmo modo que a utilização de processador, a largura de banda consumida pode variar bastante ao longo do tempo, o que faz com que a média das observações

seja insuficiente para caracterizar esse consumo. [Pras et al. 2009] sugere que se utilize a variância entre os pontos medidos. No nosso trabalho, a variabilidade do consumo de banda é mensurada pelo coeficiente de variação (CV), que é a razão entre o desvio padrão e a média dos pontos medidos: se o CV for de até 5%, considera-se comportamento de consumo constante. Em relação à variância, essa métrica tem a vantagem de ser relativa, e não absoluta. O limiar de 5% foi estabelecido com base em observações experimentais.

Todavia, apenas a análise da largura de banda não permite obter um diagnóstico preciso da adequação do provisionamento da rede. Em particular, quando a utilização de largura de banda se aproxima do máximo disponível, é necessário avaliar se há uma real necessidade de banda adicional (ou seja, se a capacidade disponível está sendo um fator limitante para a MV) ou se a MV está apenas explorando ao máximo a banda disponível, sem precisar de banda extra. Protocolos de rede que implementam controle de congestionamento, como o TCP, adaptam suas taxas de transmissão às condições da rede, tentando encontrar a largura de banda disponível na rede, que é limitada pela menor largura de banda ao longo do caminho de transmissão [Jacobson 1988]. Para fins de diagnóstico da adequação dos recursos, deve-se identificar quando essa limitação está na largura de banda local (seja física ou virtual), ou se ela está ocorrendo em algum ponto no interior da rede. No primeiro caso (que é o único em que se pode tomar alguma providência local), o MMV não consegue transmitir os quadros das máquinas virtuais com a rapidez necessária, e ocorre formação de fila na interface de rede da MV. No segundo caso, não há formação de fila na interface local. Portanto, o comprimento da fila da interface de rede da MV pode ser usado para verificar a adequação da alocação de recursos quando a largura de banda consumida se aproxima do limite.

Com base nas duas métricas propostas, coeficiente de variação do consumo de banda e fila na interface de rede, definem-se três categorias de provisionamento (subprovisionado, adequado e superprovisionado). Quando existe fila na interface de rede da MV e o consumo de largura de banda é constante, diagnostica-se o recurso como subprovisionado. Quando a banda consumida é constante e não há enfileiramento, o recurso está provisionado de forma adequada. O superprovisionamento ocorre quando o consumo de banda é variável, sem enfileiramento na interface de rede da MV. Essas categorias são descritas pelas Eqs. (8)–(10), onde CV_{lb} é o coeficiente de variação da largura de banda consumida e $\overline{fila}$ é a quantidade média de pacotes na fila da interface.

$$CV_{lb} \leq 5\% \wedge \overline{fila} > 0 \;\Rightarrow\; \text{Subprovisionado} \tag{8}$$

$$CV_{lb} \leq 5\% \wedge \overline{fila} = 0 \;\Rightarrow\; \text{Adequado} \tag{9}$$

$$CV_{lb} > 5\% \wedge \overline{fila} = 0 \;\Rightarrow\; \text{Superprovisionado} \tag{10}$$

3. Análise Experimental

A fim de verificar as propostas definidas na seção 2 para diagnóstico do provisionamento de recursos em nuvem, fez-se uma avaliação experimental. Os recursos foram monitorados a partir da execução de diferentes cargas de trabalho que representam cenários variados de utilização dos recursos.

3.1. Ambiente de Testes

O ambiente de testes é composto por dois computadores com *hardware* idênticos: processador AMD Phenom II X4 B93 2,8 GHz (*quad-core*), 4 GB de memória RAM e disco

500 GB SATA (7200 rpm). A conexão entre as máquinas é realizada por uma rede local de 100 Mbps. As máquinas virtuais criadas dispunham de uma VCPU, 256 MB de memória e 4 GB de espaço em disco montado sobre uma partição LVM.

Em uma das máquinas, denominada cliente, foi instalado o sistema operacional Ubuntu Linux 9.10 com *kernel* 2.6.31-23. Na outra, denominada servidor, foi instalado o sistema operacional Debian Squeeze 64 bits, com *kernel* 2.6.32-5, e sobre este instalou-se o *hypervisor* Xen na versão 4.0.1. As máquinas virtuais utilizam sistema operacional paravirtualizado Debian Squeeze 64 bits, com *kernel* 2.6.32-5.

Para a avaliação de processador, a ferramenta *stress*[2] foi usada para geração de carga. A monitoração foi efetuada nas próprias MVs com a ferramenta *sar*, do pacote *sysstat*[3]. Para os experimentos envolvendo a rede, as máquinas virtuais no servidor possuíam um servidor *web* Apache 2.2.16, e o cliente realizava requisições com a ferramenta de *benchmark httperf*[4], versão 0.9.0-1. Para monitorar o consumo de largura de banda, foram extraídas estatísticas de `/proc/net/dev`, que discriminam a banda consumida por interface. Para monitorar a fila da interface de rede, utilizou-se a ferramenta de configuração e gerenciamento de interface de rede *tc*[5], que fornece, entre outras estatísticas, a quantidade de pacotes na fila da interface.

As métricas foram observadas a cada 1 s, o que oferece uma boa resolução para acompanhar a variabilidade de comportamento das MVs; além disso, as ferramentas *sar* e *stress* têm granularidade mínima de 1 s. Como o *overhead* de monitoração das métricas escolhidas é baixo (*sar* consumiu 0,06% de CPU nos testes), esse mesmo intervalo pode ser usado em ambientes de produção. O diagnóstico foi realizado com base em um período de 60 s, o que é suficiente para nossos propósitos. Em produção, esse período de diagnóstico deve ser compatível com o período de tarifação do provedor IaaS, que tipicamente é de 1 h [Suleiman et al. 2012], possibilitando que se analise o comportamento com base em um número grande de amostras, reduzindo a influência de observações discrepantes. Foram cinco repetições para cada cenário, com CV $\leq 5\%$ em todos os casos.

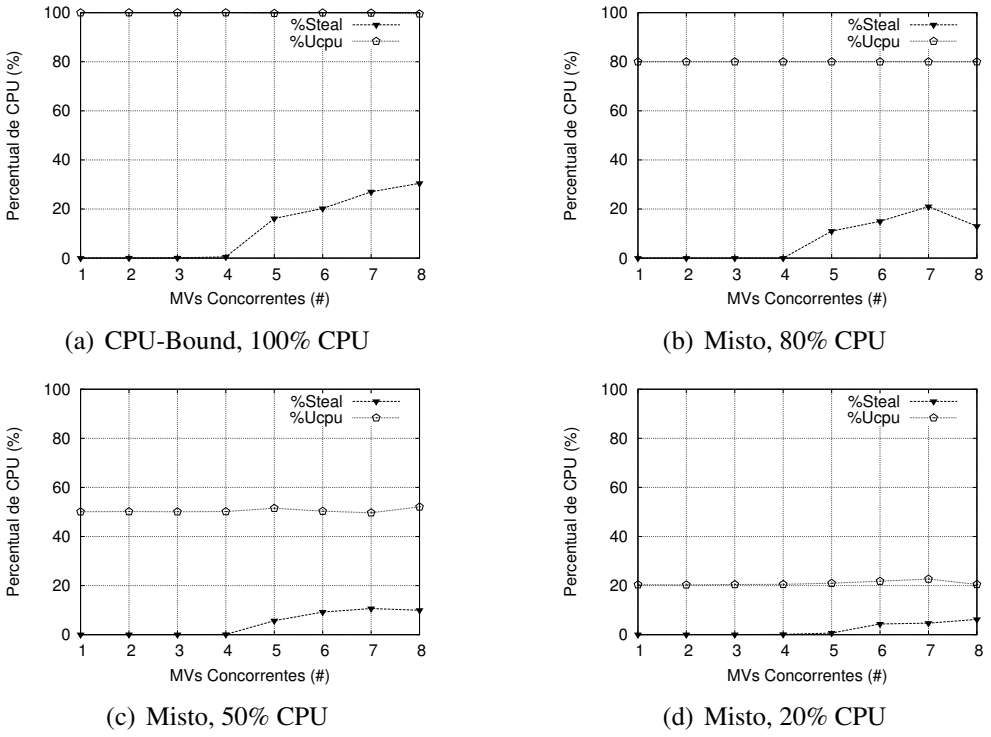
3.2. Diagnóstico do Provisionamento de Processador

Para avaliar o provisionamento de processador de acordo com o proposto na seção 2.1, foram definidos três cenários de carga sobre o recurso: CPU-Bound, Misto e IO-Bound. No cenário CPU-Bound, a ferramenta *stress* executa carga intensiva de processador sobre cada MV. No cenário IO-Bound, os processos criados por *stress* executam essencialmente operações em disco. No cenário Misto, os processos alternam entre execuções no processador e requisições a disco. O tempo total de execução da ferramenta *stress* foi dividido em intervalos de 10 segundos, sendo que os tempos para execução em processador e disco definidos são 2, 5 e 8 s. Portanto, são avaliados três cenários Mistos: carga de processador maior que a de disco (8 s processador/2 s disco), cargas equivalentes (5 s processador/5 s disco), e carga de disco maior que a de processador (2 s processador/8 s disco).

Em todos os cenários (CPU-Bound, Misto e IO-Bound) variou-se o número de máquinas virtuais com execução da carga e monitorou-se as métricas *%Steal* e *%Idle* (de

[2] `http://weather.ou.edu/~apw/projects/stress/`
[3] `http://sebastien.godard.pagesperso-orange.fr/`
[4] `http://www.hpl.hp.com/research/linux/httperf/`
[5] `http://lartc.org/manpages/tc.txt`

onde deriva-se U_{cpu}). Os gráficos da Fig. (2) mostram a média dos pontos observados nos 60 s de monitoração.



(a) CPU-Bound, 100% CPU

(b) Misto, 80% CPU

(c) Misto, 50% CPU

(d) Misto, 20% CPU

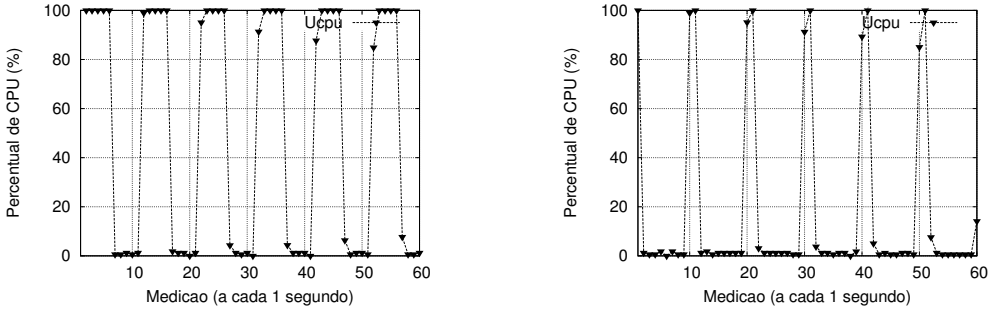
Figura 2. Média de U_{cpu} e *%Steal* (período de 60 s, 1–8 MVs concorrentes)

Ao observar o gráfico da Fig. 2(a), é possível perceber que, na execução de cargas CPU-Bound, o processador físico se torna saturado a partir de cinco máquinas virtuais, pois possui quatro *cores*. Isso pode ser verificado pelo surgimento de *%Steal* na monitoração do consumo de processador das MVs. Como $\overline{U_{cpu}}$ é igual a 100%, o processador está subprovisionado. Os dados relacionados à carga IO-Bound (cujo gráfico foi omitido por questão de espaço) permitem concluir que para todos os pontos o percentual de ociosidade manteve-se em 100%, o que indica superprovisionamento.

Na execução de carga mista com maioria do tempo em processador (Fig. 2(b)), a média de U_{cpu} se mantém em 80% para qualquer número de máquinas virtuais, o que indica subprovisionamento. Nos casos de 50% e 20% de utilização de CPU (Figs. 2(c) e 2(d)), o diagnóstico sobre o provisionamento não pode ser obtido pela simples avaliação da média geral, pois a utilização de CPU pode não ser constante. Assim, gerou-se, para estes dois casos, os gráficos das Figs. 3(a) e 3(b), onde todos os pontos medidos para uma MV são apresentados.

O cenário de carga mista equilibrada (Fig. 3(a)), com $\overline{U_{cpu}} = 49{,}95\%$, apresenta $N' = 26$ (Eq. (2)) e $\varphi = 43\%$ (Eq. (3)); adotando $\omega = 20\%$ na Eq. (6), o provisionamento é adequado. No cenário com maior carga de disco (Fig. 3(b)), $\overline{U_{cpu}} = 20\%$, $N' = 8$ e $\varphi = 13\%$, e o diagnóstico é de superprovisionamento.

A avaliação sobre *%Steal* nos cenários da Fig. 2 permite concluir que, a partir de cinco máquinas virtuais em execução simultânea, passa a existir contenção de processa-



(a) Misto, 50% CPU (b) Misto, 20% CPU

Figura 3. Utilização de CPU no período de 60 s com 1 MV em execução

dor, e ainda que o crescimento de *%Steal* está associado ao número de MVs concorrentes e à carga executada pelas MVs. Portanto, nos cenários diagnosticados como subprovisionados em que $\%Steal > 0$, o usuário tem ciência de que, para resolver o problema, não adianta alocar mais VCPU no mesmo *hardware*, sendo necessário migrar a MV para um servidor menos carregado ou instanciar novas máquinas virtuais.

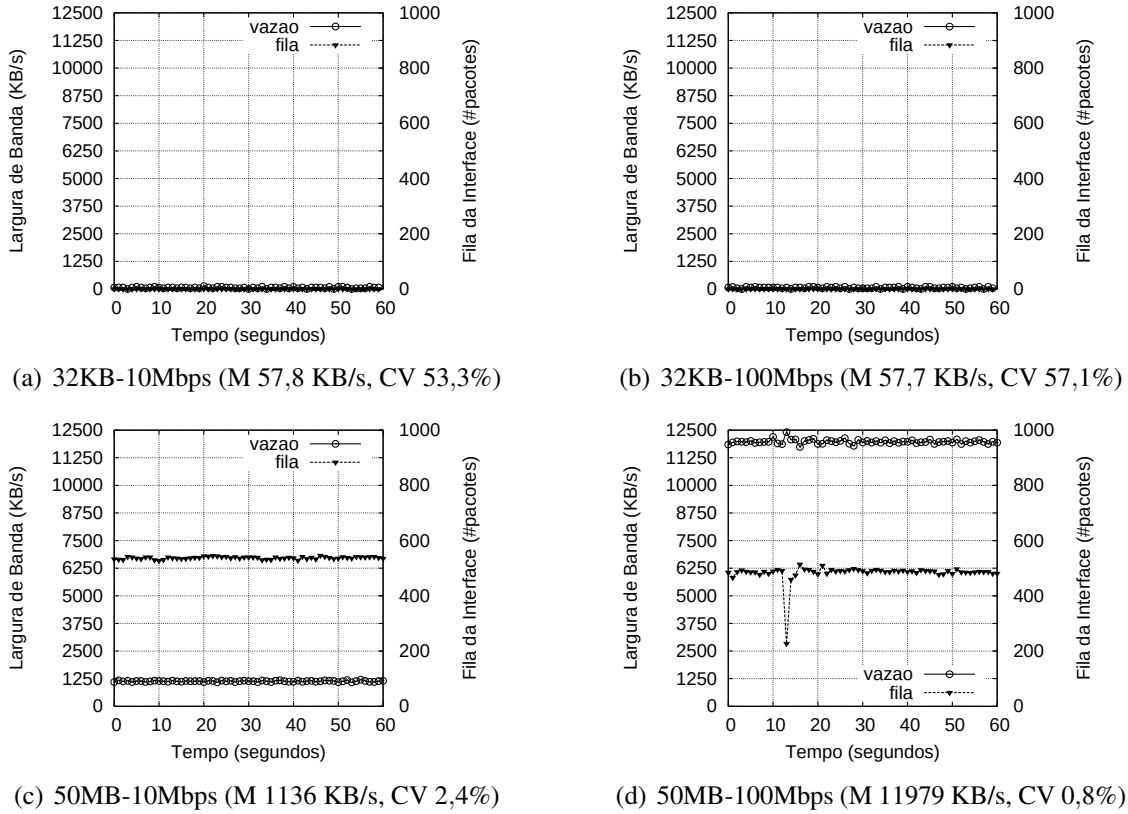
3.3. Diagnóstico do Provisionamento de Rede

Para validar o diagnóstico do provisionamento de rede, foi usada uma máquina virtual com capacidade de largura de banda limitada no MMV através do arquivo de configuração da MV. A banda alocada variou entre 10 Mbps e 100 Mbps, em intervalos de 10 Mbps. Definiram-se três cargas de trabalho, de acordo com o tamanho do arquivo requisitado ao servidor *web* (32 KB, 4 MB, 50 MB). Em todos os casos, havia cinco clientes simultâneos, com intervalo entre requisições entre 1 e 5 s, de acordo com uma distribuição uniforme.

Para representar um cenário que saturasse a rede, independente da largura de banda destinada à MV, os clientes requisitaram ao servidor arquivos de 50 MB. De modo complementar, o arquivo definido para subutilização também deveria manter este comportamento (de subutilização) para qualquer largura de banda provisionada, e assim utilizou-se arquivos de 32 KB. Por fim, em um cenário intermediário, foram usados arquivos de 4 MB, que deveriam saturar a rede quando a banda provisionada fosse pequena, mas não quando ela fosse aumentada além de certo ponto no intervalo entre 10 e 100 Mbps.

As requisições de 32 KB e 50 MB exibem comportamentos semelhantes, independente da largura de banda provisionada; assim, a Fig. 4 mostra os resultados somente para os extremos de cada cenário (10 Mbps e 100 Mbps). Os resultados estão de acordo com o esperado (superprovisionado para 32 KB, subprovisionado para 50 MB), conforme expresso pelas Eqs. (8) e (10): no primeiro caso (Figs. 4(a) e 4(b)) não há enfileiramento na interface de rede e o consumo de largura de banda não alcança o limite estabelecido, enquanto no segundo caso (Figs. 4(c) e 4(d)), verifica-se a ocorrência de fila e o consumo de banda mantém-se constante ($CV_{lb} < 5\%$) sobre a largura de banda provisionada.

Para o cenário com arquivos de 4 MB, foram obtidos três diagnósticos diferentes, dependendo da largura de banda alocada: subprovisionamento entre 10 e 50 Mbps, provisionamento adequado com 60 Mbps e superprovisionamento entre 70 e 100 Mbps. Os gráficos da Fig. 5 mostram o intervalo em que há a mudança no diagnóstico. Em termos da análise proposta na seção 2.2, verifica-se que, com largura de banda igual a



(a) 32KB-10Mbps (M 57,8 KB/s, CV 53,3%)

(b) 32KB-100Mbps (M 57,7 KB/s, CV 57,1%)

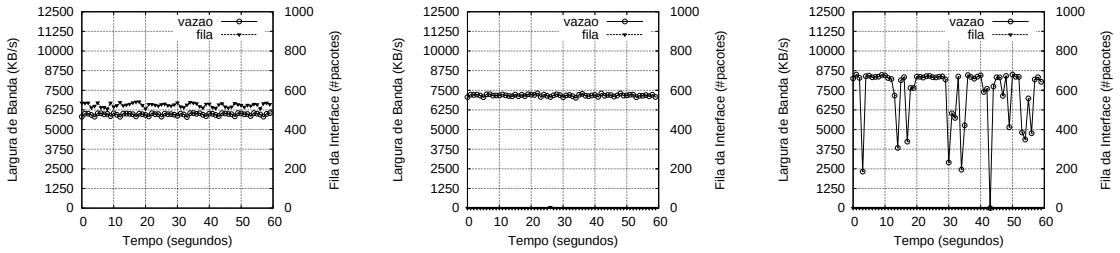
(c) 50MB-10Mbps (M 1136 KB/s, CV 2,4%)

(d) 50MB-100Mbps (M 11979 KB/s, CV 0,8%)

Figura 4. Consumo de largura de banda e enfileiramento nos ambientes saturado (50 MB) e não saturado (32 KB). M é a média do consumo de banda, e CV o coeficiente de variação. Para requisições a arquivos de 32 KB a banda provisionada nunca é alcançada, e para arquivos de 50 MB a banda provisionada é ocupada durante todo o intervalo de medição.

50 Mbps o ambiente está subprovisionado (Fig. 5(a)), ou seja, é necessário mais largura de banda, uma vez que o consumo de banda é constante ($CV_{lb} < 5\%$) e ocorre enfileiramento (Eq. (8)). Quando o limite estabelecido é de 60 Mbps (Fig. 5(b)), a Eq. (9) indica que o provisionamento realizado é adequado, pois o comportamento é constante e não há enfileiramento. A partir de 70 Mbps (Fig. 5(c)), a Eq. (10) indica que o recurso foi superprovisionado e a largura de banda pode ser reduzida, uma vez que o comportamento não é constante ($CV_{lb} > 5\%$). Portanto, pode-se concluir que o raciocínio subjacente às equações da seção 2.2 é consistente com os comportamentos verificados experimentalmente.

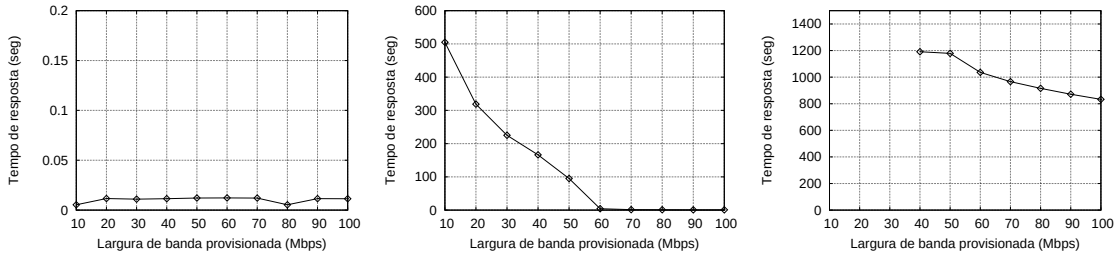
Para reforçar essa conclusão, observou-se o tempo de resposta da aplicação (no caso o *httperf*) em cada cenário (tamanho de arquivo e largura de banda alocada). A Fig. 6 apresenta a média dos tempos de resposta entre os cinco clientes. Para requisições de 32 KB (Fig. 6(a)), o tempo de resposta praticamente não é influenciado pela variação da largura de banda alocada, com ganhos na ordem de milissegundos. No cenário da Fig. 6(b), onde os arquivos requisitados são de 4 MB, verifica-se que o tempo de resposta cai rapidamente à medida em que a largura de banda disponível aumenta de 10 Mbps para 60 Mbps, e que os ganhos a partir de 60 Mbps são sensivelmente menores. Por fim, as requisições de 50 MB obtêm uma redução perceptível no tempo de resposta com o aumento da banda; neste gráfico, quando a largura de banda era inferior a 40 Mbps os



(a) 4MB-50Mbps (M 5958 KB/s, CV 1,3%) (b) 4MB-60Mbps (M 7173 KB/s, CV 1,9%) (c) 4MB-70Mbps (M 7276 KB/s, CV 26,5%)

Figura 5. Representação de cenário com busca por largura de banda ideal, com requisições a arquivos de 4 MB. M é a média do consumo de banda, e CV o coeficiente de variação. O consumo é constante com banda de 50 Mbps e 60 Mbps, e inconstante com 70 Mbps. A partir de 60 Mbps, a fila na interface desaparece.

tempos de resposta foram maiores do que o intervalo de monitoração, e por isso os dados correspondentes não estão mostrados. Os resultados da Fig. 6 corroboram o diagnóstico do provisionamento de rede dado pelas Eqs.(8)–(10). Em particular, o gráfico da Fig. 6(b) demonstra que, dependendo do valor tarifado por Mbps, o ganho de desempenho pode não justificar um aumento de banda além de 60 Mbps.



(a) 32 KB (b) 4 MB (c) 50 MB

Figura 6. Média do tempo de resposta das requisições aos três tamanhos de arquivo (32 KB, 4 MB e 50 MB) de acordo com a largura de banda provisionada.

4. Trabalhos Relacionados

Os trabalhos relacionados discutidos nesta seção dividem-se em duas categorias: identificação da demanda de recursos de máquinas virtuais e provisionamento dinâmico de recursos em ambientes de nuvem.

Identificação da demanda. Métricas para identificar demandas de recursos de máquinas virtuais executando Linux são propostas em [van Riel 2006]. Para o processador é usada uma análise similar à apresentada na seção 2.1, envolvendo *%Idle* e *%Steal*; no entanto, não são definidos limiares para esses valores, apenas noções subjetivas como "*%Idle* alto" e "*%Steal* baixo". A identificação da demanda de memória se baseia em estatísticas (*refaults* e referências a páginas alocadas) que não estão disponíveis a aplicações de usuário nas versões atuais do Linux, e nem em outros SOs. O artigo menciona o uso de largura de banda/vazão como métrica para avaliar as demandas de disco e de rede, mas sem uma proposta concreta. Além disso, ele não apresenta resultados experimentais.

[Pras et al. 2009] propõe avaliar a variabilidade do consumo de largura de banda, associada à utilização média, para dimensionar enlaces de rede. A fila nas interfaces é usada como indicativo da variabilidade do consumo de banda. Em comparação com a proposta da seção 2.2, nota-se que tanto a variabilidade da largura de banda como a fila das interfaces são consideradas. No entanto, a medida da variabilidade é diferente, e em nosso trabalho o enfileiramento é usado para identificar o subprovisionamento da rede.

Alguns trabalhos [Du et al. 2011, Nikolaev e Back 2011] propõem o uso de contadores de desempenho de *hardware* – registradores que contabilizam a ocorrência de eventos de baixo nível, como *cache hits/misses* e instruções executadas – para a monitoração de ambientes virtuais. Embora a monitoração em nível de *hardware* geralmente tenha um *overhead* menor do que em nível de sistema operacional (como a proposta neste artigo), contadores de desempenho são mais apropriados para *profiling* de aplicações individuais, enquanto o nosso foco está na análise de MVs como um todo. Ademais, as métricas usadas em nosso trabalho são bem mais portáveis entre monitores de máquinas virtuais e arquiteturas de *hardware* do que os contadores de desempenho: mesmo existindo no processador, estes nem sempre estão acessíveis às máquinas virtuais, além de serem específicos de cada processador (o conjunto de contadores disponíveis varia mesmo entre diferentes processadores da mesma arquitetura, como Intel x86).

Provisionamento dinâmico de recursos. Uma ferramenta para realizar provisionamento de recursos orientado ao cumprimento de SLA é apresentada em [Buyya et al. 2011]. As requisições de recursos do usuário são alocadas sob a avaliação do cumprimento de SLA e métricas de qualidade de serviço (QoS). Assume-se que o cliente conhece as necessidades de suas aplicações, e que a ferramenta conhece as aplicações em execução. As principais diferenças em relação à proposta deste artigo são que, no nosso caso, as métricas monitoradas não estão associadas a uma aplicação específica, e o cliente não precisa conhecer seus SLOs *a priori*, pois o provisionamento é diagnosticado em tempo de execução.

O uso de técnicas de aprendizado de máquina para provisionamento de recursos em nuvens é proposto por [Rao et al. 2011, Kundu et al. 2012, Vasić et al. 2012]. Esses trabalhos associam a utilização de recursos com medidas de desempenho das aplicações (como tempo de resposta) e usam classificadores estatísticos para determinar a classe à qual pertence uma aplicação e, consequentemente, os recursos que devem ser alocados para sua execução. Além do foco em aplicações e não em máquinas virtuais, os resultados demonstram que essa abordagem funciona para cargas de trabalho estáticas, mas tem problemas com cargas variáveis e/ou desconhecidas.

Em [Heo et al. 2009] é proposta uma ferramenta para gerenciar a alocação de processador e memória a MVs sobre a plataforma Xen, possibilitando o *overbooking*. A utilização de ambos recursos é observada periodicamente; a partir dessas observações e de um valor objetivo, calcula-se a quantidade de recursos necessária. No entanto, como é considerada apenas a média de utilização e não a sua variação, o tempo de resposta da aplicação pode ser degradado. Além disso, em processador é necessário observar o tempo de resposta da aplicação, o que difere da proposta deste artigo.

[Sudevalayam e Kulkarni 2011] usam monitoração para avaliar o impacto no consumo de processador quando duas máquinas virtuais que se comunicam entre si são alocadas no mesmo *hardware*, e propõe um modelo analítico para estimar a utilização de

CPU quando as MVs são alocadas na mesma máquina física ou em máquinas separadas. A análise se restringe à plataforma Xen, e o único recurso considerado é o processador.

Um algoritmo para provisionar máquinas virtuais levando em conta a correlação na demanda por recursos entre as MVs é proposto em [Halder et al. 2012]. MVs correlacionadas, que são aquelas com picos de demanda nos mesmos intervalos de tempo, são alocadas em máquinas físicas distintas, enquanto MVs não correlacionadas podem ser alocadas na mesma máquina. Além de considerar apenas processador, o algoritmo pressupõe cargas bem conhecidas, que se repetem sazonalmente, uma premissa que não é necessária em nossa abordagem.

5. Conclusão

Para que clientes de provedores IaaS obtenham um bom custo-benefício de suas soluções em nuvem, é necessário equilibrar desempenho e custo, e para isso é fundamental conhecer suas necessidades de recursos computacionais. Este trabalho apresenta uma importante contribuição nesse sentido, fornecendo a clientes IaaS um diagnóstico sobre o provisionamento de processador e rede de suas máquinas virtuais. A abordagem proposta se baseia na análise de métricas disponíveis nas máquinas virtuais para determinar se esses recursos estão adequados, sub- ou superprovisionados diante da demanda existente. A avaliação de processador leva em conta a média e a variação da utilização do recurso, enquanto a avaliação de rede se baseia na variação do consumo de largura de banda e no enfileiramento observado nas interfaces virtuais. Resultados experimentais com Linux e Xen demonstram que a abordagem proposta é capaz de diagnosticar corretamente o provisionamento em diferentes cenários de carga.

O diagnóstico fornecido permite que um cliente IaaS adeque sua alocação de recursos e/ou seus acordos e objetivos de nível de serviço (SLAs/SLOs) estabelecidos com o provedor. Uma extensão natural deste trabalho seria um mecanismo automatizado de gerência da elasticidade de recursos na nuvem que incorpore esse diagnóstico em suas decisões. Outra perspectiva futura é investigar como oferecer um diagnóstico para o provedor IaaS sobre o provisionamento dos recursos aos seus diferentes clientes e MVs; para isso, será necessário definir métricas adequadas e que sejam visíveis ao provedor, estabelecendo a relação entre essas métricas, o provisionamento dos recursos e o desempenho resultante.

Referências

Apparao, P., Makineni, S., e Newell, D. (2006). Characterization of network processing overheads in Xen. In *Proc. VTDC'06*, p. 2.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., e Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

Beloglazov, A. e Buyya, R. (2010). Energy efficient resource management in virtualized cloud data centers. In *Proc. IEEE/ACM CCGRID'10*, pp. 826–831.

Butler, B. (2012). Gartner's IaaS magic quadrant: A who's who of cloud market. *Network World*. `http://bit.ly/UlMy9s`. Acessado em 29/nov/2012.

Buyya, R., Garg, S., e Calheiros, R. (2011). SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *Proc. Intl. Conf. on Cloud and Service Computing (CSC)*, pp. 1–10.

Du, J., Sehrawat, N., e Zwaenepoel, W. (2011). Performance profiling of virtual machines. *SIGPLAN Not.*, 46(7):3–14.

Halder, K., Bellur, U., e Kulkarni, P. (2012). Risk aware provisioning and resource aggregation based consolidation of virtual machines. In *Proc. IEEE CLOUD'12*, pp. 598–605.

Heo, J. Zhu, X. Padala, P. e Wang, Z. (2009). Memory overbooking and dynamic control of Xen virtual machines in consolidated environments. In *Proc. IFIP/IEEE IM'09*, pp. 630–637.

Hoch, D. (2010). *Linux System and Performance Monitoring*. In *LinuxCon 2010*, `http://ufsdump.org/papers/linuxcon2010-linux-monitoring.pdf`.

Jacobson, V. (1988). Congestion avoidance and control. In *Proc. SIGCOMM'88*, pp. 314–329.

Kundu, S., Rangaswami, R., Gulati, A., Zhao, M., e Dutta, K. (2012). Modeling virtualized applications using machine learning techniques. *SIGPLAN Not.*, 47(7):3–14.

Nikolaev, R. e Back, G. (2011). Perfctr-Xen: a framework for performance counter virtualization. In *Proc. ACM VEE'11*, pp. 15–26.

Padala, P., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A., e Salem, K. (2007). Adaptive control of virtualized resources in utility computing environments. In *Proc. ACM EuroSys'07*, pp. 289–302.

Pras, A., Nieuwenhuis, L., van de Meent, R., e Mandjes, M. (2009). Dimensioning network links: a new look at equivalent bandwidth. *IEEE Network*, 23(2):5–10.

Rao, J., Bu, X., Xu, C.-Z., e Wang, K. (2011). A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In *Proc. IEEE MASCOTS'11*, pp. 45–54.

Sauvé, J., Marques, F., Moura, A., Sampaio, M., Jornada, J., e Radziuk, E. (2005). SLA design from a business perspective. *Ambient Networks*, LNCS 3775, pp. 72–83.

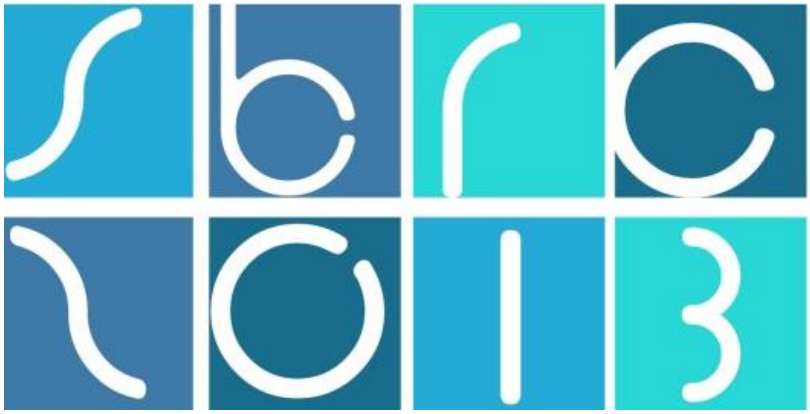
Sudevalayam, S. e Kulkarni, P. (2011). Affinity-aware modeling of CPU usage for provisioning virtualized applications. In *Proc. IEEE CLOUD'11*, pp. 139–146.

Suleiman, B., Sakr, S., Jeffery, R., e Liu, A. (2012). On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. *Journal of Internet Services and Applications*, 3:173–193.

van Riel, R. (2006). Measuring resource demand in Linux. In *Proc. Ottawa Linux Symposium*.

Vasić, N., Novaković, D., Miučin, S., Kostić, D., e Bianchini, R. (2012). Dejavu: accelerating resource allocation in virtualized environments. *SIGARCH Comp. Arch. News*, 40(1):423–436.

Wood, T., Cherkasova, L., Ozonat, K., e Shenoy, P. (2008). Profiling and modeling resource usage of virtualized applications. In *Proc. ACM/IFIP/USENIX Middleware'08*, pp. 366–387.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 15

Computação Ciente de Contexto

Um Middleware para Provisionamento de Contextos para Redes Veiculares*

Fabrício A. Silva[1,2], Thais Regina M. B. Silva[2], Linnyer B. Ruiz[3] e Antonio A. F. Loureiro[1]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)
[2]Campus Florestal – Universidade Federal de Viçosa (UFV)
[3]Departamento de Informática - Universidade Estadual de Maringá (UEM)
{fabricio.asilva,loureiro, thaisrb}@dcc.ufmg.br, linnyer@gmail.com

Resumo. *As aplicações para redes veiculares são consideradas cientes de contexto pois precisam de informações de interesse sobre os veículos, motoristas, passageiros e o ambiente para operarem satisfatoriamente e assim beneficiarem seus usuários, seja indicando a melhor rota ou alertando sobre acidentes e congestionamentos. Dessa forma, essas aplicações precisam de um provedor de contextos responsável por coletar, analisar e disponibilizar esse tipo de dado para as mesmas. Além disso, as aplicações para redes veiculares apresentam uma demanda por contextos lógicos (isto é, contextos que não são coletados por sensores físicos) e são consideradas coletivas, uma vez que seus recursos podem ser compartilhados simultaneamente por dois ou mais usuários. Este trabalho apresenta um middleware, chamado ConProVa, composto por módulos capazes de lidar com diversas questões relativas ao provisionamento de contextos, incluindo a inferência de contextos lógicos. O ConProVa também possui funcionalidades para detectar e tratar situações de conflitos de interesse, comuns em aplicações cientes de contexto coletivas. Uma instância do middleware foi avaliada para uma aplicação de detecção de congestionamento causado por acidentes e os resultados mostraram que foi possível aumentar a satisfação dos usuários, reduzindo o tempo total de trajeto e a emissão de gás carbônico.*

Abstract. *Vehicular ad-hoc network applications are context-aware since they need environmental and local contexts to operate properly and help their users on many aspects, like indicating the best route to take or warning about accidents and traffic congestion situations. Therefore these applications need a context provider system responsible for collecting, analyzing, reasoning and making the contexts available to them. In addition to this, vehicular applications require logical contexts (i.e., contexts that are not collected directly from physical sensors) and are collective because their resources are shared among many users. In this work we propose a smart context provisioning middleware, called ConProVA, that, in addition to provide contexts to the applications, it is able to infer logical contexts and deal with conflicts of interest. An instance of ConProVA was evaluated through simulation considering an accident detection and avoidance application. The results showed that it was possible to improve the users' satisfaction by reducing total travel time and carbon dioxide emission when accidents occur.*

*Este trabalho conta com o apoio financeiro do CNPq por meio do INCT NAMITEC, processo 573.738/2008-4

1. Introdução

O estudo das redes veiculares está se tornando mais comum a cada dia. Esse tipo de rede tem como objetivo permitir a comunicação sem fio entre veículos automotores e também entre veículos e estações fixas infra-estruturadas situadas nas margens das ruas e estradas [Campista et al. 2009]. Muitas aplicações podem ser desenvolvidas para as redes veiculares e assim trazer benefícios para os usuários, como por exemplo identificação e controle de congestionamento, assistência no controle de vagas de estacionamento, auxílio e alertas de riscos de acidentes, disponibilização de conteúdo interativo aos passageiros, dentre várias outras. A utilidade desse tipo de aplicação para a sociedade com o objetivo de se ter uma melhoria da qualidade e da segurança no trânsito está fazendo com que vários grupos de pesquisadores se dediquem a essa área.

As aplicações para redes veiculares são consideradas cientes de contexto. Para funcionarem corretamente e de acordo com as necessidades, as aplicações desenvolvidas para esse tipo de rede requerem dados de interesse chamados contextos dos veículos, de seus usuários e do ambiente. Exemplos de tais contextos incluem a velocidade, posição, direção, nível de combustível e aceleração do veículo, o nível de sonolência e as intenções dos motoristas, e o clima (chuvoso ou ensolarado, por exemplo) no ambiente em que o veículo se encontra. Sem essas e outras informações contextuais, seria mais difícil tomar decisões acertadas.

Outra característica que pode ser observada nas aplicações de redes veiculares é a sua demanda por contextos lógicos. Contextos lógicos são informações contextuais que não podem ser coletadas diretamente por sensores físicos, e portanto devem ser inferidas com o uso de alguma técnica computacional [Ye et al. 2012]. Como exemplos de contextos lógicos no escopo das redes veiculares, pode-se citar a atividade desempenhada pelos usuários, as intenções do motorista, a ocorrência de um acidente e a identificação de uma vaga de estacionamento livre. Com o uso desse tipo de contexto em conjunto com outras informações contextuais, as decisões tomadas tendem a ser mais precisas.

Além de serem cientes de contexto, as aplicações para redes veiculares também podem ser classificadas como coletivas. Esse tipo de rede é composta por vários veículos automotores disputando recursos comuns como largura de banda, espaço em ruas, conteúdos online, dentre outros. Com isso, conflitos de interesse podem ocorrer quando mais de um veículo está interessado no mesmo recurso que é restrito e não pode ser alocado a todos os interessados. Por exemplo, quando dois ou mais veículos possuem interesse em uma vaga de estacionamento em determinada região e obviamente uma única vaga não pode ser alocada a mais de um veículo ao mesmo tempo, é importante decidir de maneira justa qual veículo será atendido.

O objetivo desse trabalho é definir e avaliar um *middleware* inteligente, chamado ConProVA (do inglês **Con**text **Pro**visioning for **V**ehicular **A**pplications), para o provisionamento de contextos para aplicações de redes veiculares. Por serem cientes de contexto, essas aplicações precisam de uma funcionalidade para prover os contextos a elas. Esse provedor deve ser inteligente no sentido de ser capaz de inferir contextos lógicos e resolver conflitos de interesse quando necessário, já que essas são duas demandas das redes veiculares, como descrito anteriormente. O ConProVA foi projetado para atender a essas demandas, sendo também capaz de divulgar, por meio do paradigma de comunicação *Publish/Subscribe*, eventos ocorridos de acordo com as solicitações dos interessados. Além

disso, o *middleware* é flexível para ser adaptada a diferentes aplicações e demandas e se preocupa com o volume de dados transmitidos para não sobrecarregar a rede.

Um *middleware* pode ser considerado uma boa solução para provisionamento de contexto, pois permitirá que os desenvolvedores das aplicações foquem somente na aplicação sem se preocuparem com a coleta, processamento e armazenamento dos elementos de contexto. Além disso, um *middleware* irá permitir a comunicação entre tecnologias heterogêneas de diferentes fabricantes, já que atua em uma camada entre a aplicação e a plataforma de hardware, permitindo uma maior integração dos veículos e consequentemente uma melhor qualidade dos serviços.

Este artigo é composto por cinco seções e está organizado como descrito abaixo. A seção 2 apresenta os principais trabalhos relacionados encontrados na literatura. A seção 3 descreve os requisitos, arquitetura e detalhes do *middleware* proposto. A seção 4 contém todos os detalhes da avaliação realizada para verificar a viabilidade da proposta e os resultados obtidos. Finalmente, a seção 5 apresenta os comentários finais e alguns potenciais trabalhos futuros.

2. Trabalhos Relacionados

Muitos pesquisadores já propuseram *middlewares* para redes veiculares. No entanto, conforme descrito abaixo, as propostas não contemplam aspectos importantes que são cobertos pelo ConProVA.

[Nour et al. 2011] propuseram um *middleware* responsável por coletar, armazenar e disponibilizar contextos para as aplicações em redes veiculares. Esse *middleware* é organizado em camadas com responsabilidades específicas como coletar, filtrar, analisar e armazenar contextos. Além da organização em camadas, não foram apresentados detalhes e nenhuma avaliação foi realizada.

O sistema CarTel, proposto por [Bychkovsky et al. 2006], é composto por um computador embutido com sensores responsáveis pela coleta de dados dos carros como posição, disponibilidade de conexão WiFi, desempenho do motor, dentre outros. Os dados coletados são enviados a um servidor por meio de conexões sem fio intermitentes. Além da proposta, não foram apresentados mais detalhes e apenas uma versão simplificada da solução foi avaliada em uma demonstração.

[Riva 2006] propôs Contory, um *middleware* para prover contextos para *smartphones*. Apesar de não ser específico para redes veiculares, essa proposta apresenta características relacionadas ao trabalho que valem a pena descrevê-la. No Contory, são considerados três opções para coleta de dados: internos (locais em relação ao dispositivo), externos (coletados de um servidor) e distribuídos (coletados de elementos de rede conectados via Bluetooth ou WiFi). Além disso, foi proposta uma definição de campos que representam um contexto, como tipo, valor, data/hora da coleta, tempo de vida, origem e um campo opcional. Também foi proposta uma linguagem de consulta similar ao SQL que permite buscar e filtrar contextos de acordo com critérios escolhidos pelo interessado.

[Leontiadis et al. 2009] argumentam em seu trabalho que o paradigma de comunicação *Publish/Subscribe* (Pub/Sub) [Eugster et al. 2003] é o mais adequado para redes veiculares uma vez que cada veículo está interessado em eventos específicos e podem se inscrever para receber notificações sobre esses eventos. Além disso, dada a ca-

racterística assíncrona desse paradigma, um veículo pode se inscrever para algum evento mesmo se o fornecedor do mesmo não estiver conectado simultaneamente, o que tende a ser comum nas redes veiculares já que a topologia é dinâmica devido a mobilidade dos veículos. Dadas essas justificativas, foi proposto um *middleware* que adota o paradigma Pub/Sub e que considera a localização e o horário das publicações e notificações. Foram propostas as primitivas de comunicação sobre como se inscrever e como notificar a ocorrência de eventos. Além disso, foi proposta uma solução flexível para o casamento entre uma notificação e uma inscrição utilizando árvore de mapeamento. Porém, esse *middleware* assume que todos os contextos estão disponíveis para uso e possui foco somente na parte de comunicação.

Além das propostas de *middlewares* descritas, é importante apresentar trabalhos relacionados à inferência de contextos lógicos e resolução de conflitos, que são dois tópicos importantes para a definição do ConProVA. [Ye et al. 2012] fizeram um levantamento detalhado de várias técnicas e aplicações para inferência de contextos lógicos. Dentre essas técnicas está a *Naive Bayes*, que é adotada neste trabalho. Porém, nenhuma das aplicações de inferência descritas pode ser considerada flexível para diferentes demandas e aplicações, como é proposto no ConProVA. Em relação à identificação e resolução de conflitos, o trabalho desenvolvido por [Silva et al. 2010] propõe uma arquitetura que adota vários algoritmos em conjunto para identificar e resolver conflitos de acordo com a disponibilidade de recursos em aplicações ubíquas. Além dessa proposta para aplicações ubíquas, não foram encontrados outros trabalhos com o objetivo de resolver conflitos em redes veiculares.

Todas as propostas descritas possuem suas contribuições para o estado da arte da área. Porém, elas não tratam questões importantes para as redes veiculares. Nenhuma solução trata da demanda das aplicações por contextos lógicos. Além disso, não é considerada a natureza coletiva das redes veiculares, em que conflitos de interesse podem ocorrer pela demanda por recursos restritos como vagas de estacionamento e largura de banda. Por último, também não é tratado o problema da grande quantidade de informação redundante que pode ser gerada em casos de tráfego intenso. Todos esses problemas são considerados e tratados pelo ConProVA, como será descrito na próxima seção.

3. ConProVA

Esta seção descreve os detalhes do *middleware* proposto, chamado ConProVA (**Con**text **Pro**visioning for **V**ehicular **A**pplications). Dadas as características das redes veiculares, os principais requisitos do ConProVA são modularidade, inferência de contextos lógicos, tratamento de conflitos de interesse e compartilhamento de informação entre veículos. Além de atender a esses requisitos, o *middleware* proposto é flexível para ser adaptado de acordo com diferentes aplicações e demandas.

3.1. Arquitetura do ConProVA

ConProVA foi organizado em componentes com características e interfaces específicas, como mostra a figura 1. É assumida uma arquitetura de comunicação cliente/servidor com múltiplos servidores em que os clientes são os veículos e os servidores são as estações fixas infraestruturadas localizadas nas ruas e estradas. Essas estações possuem maior capacidade computacional, são estáticas e conectadas entre si. Os veículos são carros ou qualquer outro automóvel com capacidade de comunicação sem fio, um sistema de

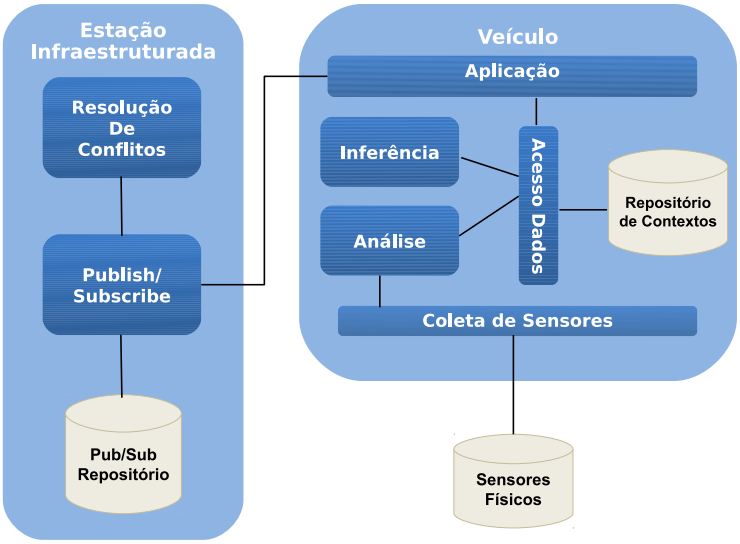


Figura 1. Arquitetura de componentes do ConProVA

posicionamento global e uma unidade de processamento. Os veículos são capazes de trocar mensagens por meio de comunicação sem fio com as estações fixas e entre si.

A vantagem em se utilizar uma arquitetura de comunicação cliente/servidor da maneira proposta é o fácil compartilhamento de informações entre os veículos pelos servidores com o mínimo de troca de mensagens, economizando assim largura de banda, que é um recurso escasso nas redes veiculares. De outra forma, seria preciso uma grande quantidade de mensagens roteadas entre múltiplos veículos. Por outro lado, essa arquitetura requer uma infraestrutura física que tem um custo de implantação. No entanto, considerando as grandes e médias cidades já cobertas pela rede celular, o custo dessa infraestrutura poderá ser reduzido mesmo considerando que as redes veiculares utilizam tecnologia de comunicação diferente.

Os componentes que fazem parte do ConProVA são:

Coleta de Sensores: responsável por implementar as interfaces entre o *middleware* e os dispositivos de sensores físicos como GPS, sensor de velocidade, uma rede de sensores sem fio, dentre outros;

Análise: responsável por processar os dados coletados pelos sensores para organizar os contextos ou agregar os dados relacionados. Pode utilizar os contextos armazenados no repositório de contextos para agregá-los com os novos dados e assim manter um histórico de eventos;

Acesso aos Dados: responsável por controlar o acesso ao repositório de contextos. Também é responsável por representar os dados coletados em forma de contextos de acordo com a especificação definida por [Riva 2006] e adotada pelo ConProVA por se tratar de uma representação completa e validada, composta por itens que descrevem informações espaciais, temporais e ambientais dos contextos;

Inferência: responsável por inferir contextos lógicos com base em dados contextuais existentes. A técnica selecionada para ser adotada por esse módulo é *Naive Bayes*, que é

baseada na regra de Bayes:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \tag{1}$$

onde X é a evidência e H é a hipótese. No escopo do ConProVA, a evidência é a informação contextual conhecida e a hipótese é o contexto lógico a ser inferido. Em outras palavras, é calculada a probabilidade de um contexto lógico ser válido dado o conhecimento de um conjunto de dados contextuais obtido por outros meios.

A técnica de *Naive Bayes* foi escolhida por ser um modelo de classificação simples e por assumir a independência condicional entre as evidências. Dessa forma, o cálculo é reduzido a:

$$P(H|X) = P(H) \times \prod_{1 \leq i \leq n} P(x_i|H) \tag{2}$$

onde x_i é o valor de um dos atributos da evidência (isto é, dados contextuais conhecidos no ConProVA) e n é a quantidade de evidências existentes. Com isso, é necessário saber somente a probabilidade das evidências quando é conhecida a hipótese ($P(x_i|H)$). Apesar de sua simplicidade, *Naive Bayes* tem alcançado bons resultados quando comparada a outras técnicas de classificação em aprendizado de máquinas, de acordo com [Han and Kamber 2006].

Para cada contexto lógico a ser inferido, é necessário conhecer os valores probabilísticos dos atributos da evidência usados pelo *Naive Bayes*. Além disso, esses valores diferem de contexto para contexto e também de aplicação para aplicação. Por exemplo, a probabilidade de ocorrência de acidentes depende das características específicas de cada cidade. Para se ter uma solução flexível do módulo de inferência, um arquivo de configuração em formato XML foi definido para conter as informações de entrada do *Naive Bayes* para cada contexto lógico a ser inferido. Este arquivo deve ser editado e configurado de acordo com as necessidades específicas de contextos lógicos pelo projetista da aplicação.

Resolução de Conflito: responsável por identificar e resolver conflitos de interesse quando necessário. Vários recursos disponíveis para redes veiculares são restritos em quantidade. Com isso, quando mais de um veículo tem interesse em um mesmo recurso que não possa ser alocado a todos os interessados, um conflito ocorre e deve ser solucionado. O módulo de resolução de conflitos é responsável por tratar esse tipo de problema.

Similarmente à inferência de contextos lógicos, as características dos conflitos variam de acordo com as aplicações. Portanto, esse módulo também deve ser flexível o suficiente para ser adotado por diferentes aplicações com o mínimo de esforço possível. Para atender a essa demanda, o ConProVA utiliza uma função de utilidade configurável. Essa função é definida como:

$$U = \sum_{1 \leq i \leq n} c_i \times w_i \tag{3}$$

onde $c_i \in C$, $w_i \in W$, C é o vetor de contextos usados para o cálculo da utilidade e W é o vetor de pesos para cada contexto em C, e $|W| = |C| = n$, que é a quantidade de contextos utilizados para o cálculo da utilidade. Os vetores C e W são definidos pelo projetista de acordo com as características das aplicações e dos conflitos em um arquivo de configuração em formato XML.

O valor da utilidade é calculada para todos os veículos em conflito e indica a importância dada a cada um deles para o recurso disputado. Com as utilidades dos veículos calculadas, o algoritmo soluciona o conflito selecionando os M veículos interessados com maior utilidade, onde M é a quantidade disponível do recurso que causou o conflito.

***Publish/Subscribe*:** responsável pelo compartilhamento de informações entre os veículos. Um dos objetivos do *middleware* ConProVA é disponibilizar contextos e eventos para outros veículos, contribuindo assim para as tomadas de decisões. Para isso, o paradigma de comunicação *Publish/Subscribe* baseado em conteúdo foi utilizado por se tratar de uma solução interessante para redes veiculares. Algumas das suas principais características, como representação das mensagens, função de casamento entre eventos e interesses e a disponibilização dos eventos aos interessados, são baseadas no trabalho desenvolvido por [Leontiadis et al. 2009], que já definiu e avaliou um esquema de comunicação Pub/Sub para redes veiculares. É assumido que todos os veículos possuem um sistema de navegação com GPS e mapa detalhado da cidade em que se encontram. Além disso, as estações fixas possuem o mapa detalhado e são capazes de calcular rotas (incluindo alternativas) para atender aos veículos. Abaixo é descrito como cada funcionalidade desse módulo foi definida e a figura 2 ilustra um exemplo simplificado do funcionamento desse módulo.

Subscrição: quando um veículo tem interesse em alguma informação, ele cria uma mensagem de subscrição e a envia para as estações fixas da sua vizinhança naquele momento. Essa mensagem contém campos que permitem a identificação da subscrição, os eventos de interesse, a localização do veículo, seu destino final e a sua identificação, dentre outros. Para manter a subscrição sempre atualizada, os veículos periodicamente enviam seus interesses às estações próximas. O intervalo entre um envio e outro é dinâmico e inversamente proporcional à velocidade do veículo, uma vez que quanto mais rápido o veículo estiver, mais rápido seu estado e localização irão mudar. Por outro lado, um veículo mais lento não terá alterações significativas no seu estado, incluindo localização, em um curto período e portanto o intervalo pode ser maior. Essa dinamicidade no intervalo de reenvio das subscrições reduz a quantidade de mensagens redundantes enviadas, principalmente em situações de tráfego intenso, diminuindo assim o uso de largura de banda. A periodicidade do envio é dada pela fórmula $intervalo = R/v$ onde v é a velocidade do veículo e R é o raio de cobertura das estações fixas. Caso $intervalo < min$ ou $intervalo > max$, o valor do intervalo é atribuído ao limite inferior (min) ou superior (max), respectivamente. Os valores de R, min e max são definidos especificamente para cada aplicação.

Recepção de Subscrição: quando uma estação fixa recebe uma mensagem de subscrição de um veículo, primeiramente ela verifica no seu repositório de Pub/Sub se a mensagem já foi recebida anteriormente e pode ser descartada. Em caso negativo, é verificado também o prazo de expiração da mensagem para que sejam descartadas mensagens já expiradas. Caso a mensagem não seja descartada por nenhum dos dois motivos descritos, ela é armazenada no repositório e compartilhada com as outras estações fixas, e será utilizada posteriormente quando algum evento ocorrer.

Publicação: quando algum evento ocorre, o veículo prepara uma mensagem de publicação contendo seu identificador, o contexto que originou o evento, a sua localização, o tempo de expiração do evento, e envia essa mensagem às estações de sua vizinhança. Quando o evento não é mais válido, o veículo então envia uma mensagem cancelando a

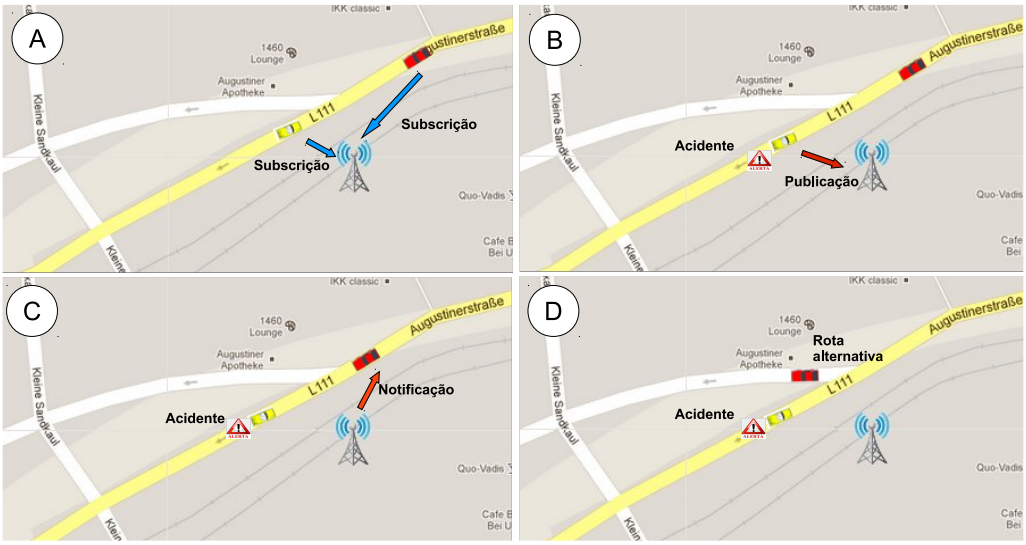


Figura 2. (A) os veículos enviam mensagem de subscrição à estação fixa para receberem notificação sobre o trajeto. (B) Em seguida, um acidente ocorre e é inferido pelo veículo amarelo, que envia uma publicação à estação. (C) A estação verifica quais veículos possuem interesse nessa publicação (no caso, o veículo vermelho), e então envia uma notificação ao mesmo. (D) Finalmente, ao receber a notificação, o veículo vermelho segue por uma rota alternativa.

publicação enviada anteriormente.

Recepção de Publicação: quando uma estação fixa recebe uma mensagem de publicação, as subscrições existentes no repositório são comparadas para verificar se há casamento de alguma subscrição com a publicação recebida. Se o evento representado pela publicação recebida possui um número restrito de recursos (por exemplo, o evento pode ser a disponibilidade de uma vaga de estacionamento ou a indicação de uma rota alternativa com restrição de capacidade) e o número de subscrições interessadas no evento é maior que o número de recursos disponíveis, então o módulo de resolução de conflito é invocado. Após ser executado, o módulo de resolução de conflitos irá retornar uma lista de subscrições que deverão ser atendidas. Para finalizar, a estação envia uma mensagem de notificação a todos os veículos atendidos contendo os detalhes do evento.

Recepção de Notificação: quando um veículo recebe uma notificação, ele verifica se o seu interesse no evento é ainda válido ou não. Se for, ele responde à estação notificadora confirmando a sua aceitação da notificação.

4. Avaliação

O objetivo da avaliação realizada é mostrar como a adoção do ConProVA pode aumentar a satisfação dos usuários das redes veiculares. Para isso, uma aplicação de inferência de acidentes e indicação de congestionamentos causados pelos mesmos foi implementada em um ambiente de simulação. É importante informar que não é objetivo deste trabalho propor uma solução para esse problema específico, mas sim mostrar que o ConProVA pode ser utilizado para resolvê-lo. No entanto, para o cenário avaliado neste trabalho, é possível demonstrar que o *middleware* proposto pode ser configurado de acordo e obter bons resultados.

ConProVA foi simulado por meio da ferramenta de simulação de rede OMNET++ [Varga 2001] e o gerador de tráfego SUMO [Behrisch et al. 2011]. As simulações foram realizadas considerando um modelo de tráfego real da cidade de Colônia, na Alemanha [Uppoor and Fiore 2011]. Esse modelo de tráfego foi escolhido por ser validado e estar disponível publicamente. Como as pesquisas relacionadas a redes veiculares são incipientes, existem poucos modelos disponíveis, e muitos deles não são validados.

4.1. Definições do ConProVA

A seguir é descrito como cada um dos principais módulos do ConProVA foram configurados nos cenários de simulação.

Aplicação: Uma aplicação de identificação e indicação de congestionamento causado por acidentes foi implementada. Os veículos que adotam ConProVA são capazes de inferir a ocorrência de acidentes, o que é considerado um contexto lógico já que não existe sensor capaz de medir essa informação, e publicar esse evento para que outros veículos evitem o local do acidente, seguindo rotas alternativas até o destino final. Foi considerada a existência de veículos comuns e outros de emergência, como ambulâncias e viaturas.

Módulo de Inferência: Este módulo é responsável por inferir a ocorrência de acidentes. Para isso, o arquivo de configuração do módulo foi criado contendo as probabilidades de entrada para o modelo de *Naive Bayes*. Esses valores probabilísticos de entrada foram calculados com base em informações da prefeitura da cidade de Colônia [Hall 2012]. Segundo essas informações, durante o ano de 2010 ocorreram 43345 acidentes nessa cidade. Além disso, de acordo com o mapa da cidade utilizado nas simulações, Colônia possui 823 semáforos e 69205 vias (ruas, avenidas, dentre outros). Com base nesses números, nos detalhes da cidade e informações intuitivas, como a maior probabilidade de ocorrência de acidentes em dias de semana e em horários de maior movimento, foi possível estimar as probabilidades utilizadas como entrada para o *Naive Bayes*.

O modelo utilizado neste trabalho considera que um veículo pode estar se movendo em baixa velocidade principalmente por duas razões: ocorrência de acidente ou situação de semáforo. Então, dada a evidência da velocidade baixa do veículo, é calculada a probabilidade de o mesmo estar em situação de acidente ou de semáforo. A maior probabilidade é a mais provável e então considerada válida como contexto lógico. Em outras palavras, na fórmula 2, $H = \{A, S\}$, onde A e S significam *acidente* e *semáforo* respectivamente, e $X = VB_t$ onde VB_t significa *Velocidade Baixa em um intervalo de tempo t*. Ou seja, a ocorrência de acidente ou situação de semáforo são as hipóteses (contextos lógicos a serem inferidos) e o período de tempo em velocidade baixa é a evidência (dados contextuais conhecidos).

Para se ter uma maior precisão da inferência, a variável de evidência VB_t é composta por três atributos, VB_{t_1}, VB_{t_2} e VB_{t_3}, que definem a quantidade de tempo em que o veículo está se movendo em baixa velocidade e $t_1 < t_2 < t_3$. Com isso, é possível ter mais informações sobre a situação do veículo e assim aumentar a chance de uma estimativa correta. Nesse caso, a evidência VB_{t_i} com $i \in \{1, 2, 3\}$ será válida se o veículo estiver em velocidade baixa por um tempo t em que $t_{i-1} < t < t_i$ ($t_0 = 0$ por definição). Um veículo é considerado em velocidade baixa se está 10% abaixo da velocidade média da via em que se encontra.

Substituindo as hipóteses e evidências na fórmula 2, temos então:

$$P(H = h|VB_t) = P(H = h) \times \prod_{1 \leq i \leq 3} P(VB_{t_i}|h) \quad (4)$$

onde $h \in \{A, S\}$ é a hipótese, que pode ser *acidente* ou *semáforo*. O valor máximo entre $P(H = A|VB_t)$ e $P(H = S|VB_t)$ é considerado o mais provável e é inferido como contexto lógico.

Módulo de Resolução de Conflitos: Como descrito anteriormente, a aplicação implementada identifica alguma situação de congestionamento causado por acidentes e publica esse evento para que outros veículos evitem o local do acidente seguindo rotas alternativas. No entanto, algumas vias das rotas alternativas podem ter uma capacidade máxima e, caso muitos veículos sigam por essa via, a mesma pode se sobrecarregar, causando congestionamento. Nesse caso, o módulo de resolução de conflito deve decidir quais veículos deverão seguir quais rotas, de forma a não sobrecarregá-las.

Para ajudar aos veículos de emergência (ambulâncias e viaturas policiais) alcançarem seus destinos o quanto antes, a informação de prioridade é utilizada como contexto na função de utilidade do módulo de resolução de conflitos. Além disso, a distância entre os veículos e as rotas alternativas também é usada. Nesse caso, veículos com maior prioridade e mais próximos das melhores rotas serão alocados para elas, e a quantidade de veículos alocada será tal que não sobrecarregará a rota, evitando que as rotas alternativas se congestionem também.

Então, na função de utilidade da fórmula 3, temos que $C = \{prioridade, distancia\}$, $W = \{w_p, w_d\}$ e o cálculo é dado por:

$$U = prioridade \times w_p + distancia \times w_d \quad (5)$$

onde w_p é o peso dado ao valor da prioridade e w_d é o peso dado à distância do veículo para a rota em questão. Nos cenários simulados, $w_p >> w_d$ para dar preferência aos veículos de emergência, que serão atendidos a não ser quando estiverem muito distantes das melhores rotas.

Módulo *Publish/Subscribe*: Todos os veículos periodicamente enviam uma mensagem de subscrição para receberem notificações de eventos relacionados à situação do tráfego na sua rota. O intervalo entre o envio das subscrições é dinâmico, calculado de acordo com a definição na seção 3, e varia entre $min = 10s$ e $max = 50s$, tendo o raio de cobertura das estações infraestruturadas atribuído para $R = 400m$.

As estações infraestruturadas armazenam as subscrições recebidas em seus respectivos repositórios. Quando um evento (isto é, um acidente) é inferido pelo módulo de inferência, o veículo responsável envia uma publicação descrevendo o evento às estações. Quando uma publicação é recebida por uma estação infraestruturada, ela busca em seu repositório por subscrições que tenham interesse nesse evento. Em caso de encontrar, os veículos que enviaram tais subscrições são notificados por meio de mensagens de notificação pela estação.

Antes de enviar a notificação, a estação invoca o módulo de resolução de conflitos que seleciona os veículos de forma a satisfazer a função de utilidade e não sobrecarregar as rotas alternativas. Então, além de informar aos veículos sobre o evento, a estação também indica qual rota alternativa os mesmos devem seguir.

Tabela 1. Configuração da Simulação

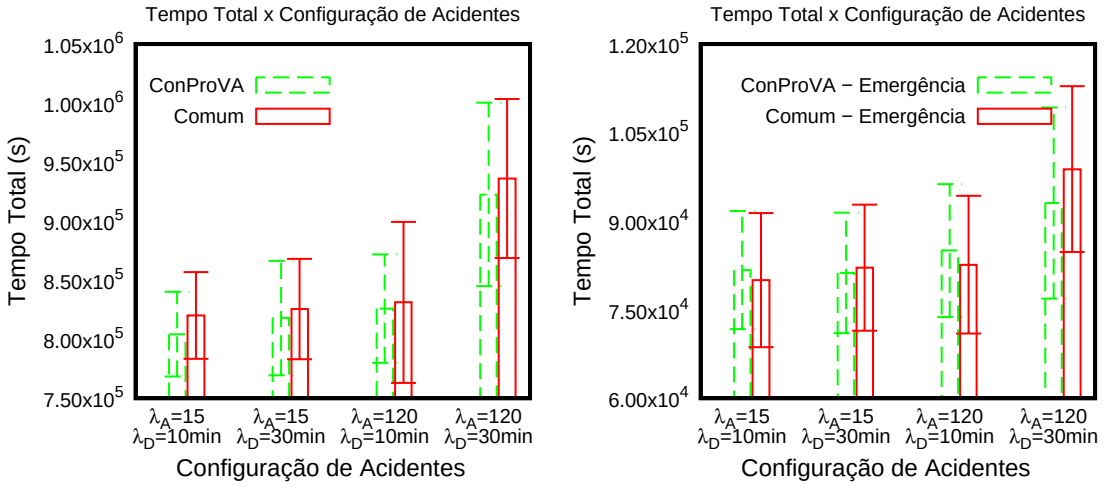
Parâmetro	Valor
Percentual de veículos de emergência	10%
Quantidade de acidentes	*Poisson* com $\lambda = \lambda_A \in \{15, 120\}$
Duração dos acidentes	*Poisson* com $\lambda = \lambda_D \in \{10min, 30min\}$
Localização das estações fixas	Em grade, cobrindo toda a região
Protocolos de comunicação	IEEE 1609.4/802.11p [Eckhoff and Sommer 2012]
Cenários	**ConProVA**: versão completa do ConProVA **Comum**: Nenhuma das funcionalidades do ConProVA

4.2. Configurações da Simulação

Para avaliar a adoção do *middleware* proposto, foram realizadas simulações com o modelo real de tráfego da cidade de Colônia, na Alemanha [Uppoor and Fiore 2011]. Mais de 2000 veículos partem de suas posições iniciais entre as 6:00 e 6:30 da manhã. A simulação termina quando todos os veículos chegam ao seu destino final, o que representa aproximadamente 1h30min, dependendo da quantidade e da gravidade dos acidentes ocorridos. A quantidade e a gravidade (medida em tempo de duração do acidente, ou em outras palavras, o tempo gasto para que o acidente seja removido e as vias envolvidas sejam liberadas) dos acidentes segue uma distribuição de *Poisson* com o parâmetro λ selecionado para cobrir situações variadas, com muitos ou poucos acidentes, e com a duração dos mesmos variando para simular diferentes gravidades. O valor de λ para a quantidade de acidentes foi escolhido com base em estatísticas da prefeitura da cidade de Colônia. A tabela 1 apresenta as configurações utilizadas.

4.3. Resultados

Cada cenário foi simulado 33 vezes e os gráficos apresentam as médias e os respectivos desvios padrão.



(a) Resultados de todos os veículos (b) Resultados para veículos de emergência

Figura 3. Tempo total de trajeto. λ_A = parâmetro da distribuição de Poisson para o número de acidentes e λ_D = parâmetro da distribuição de Poisson para a duração dos acidentes

Tempo total de trajeto: O tempo total de trajeto mede quanto tempo todos os veículos

precisam para alcançar os seus respectivos destinos finais. Em outras palavras, é a soma do tempo gasto por todos os veículos até que eles alcancem o seu destino final. Como o modelo de mobilidade utilizado é de um dia real, os mesmos veículos possuem a mesma origem e destino em todas as simulações. A figura 3(a) mostra que a adoção do ConProVA ajudou a reduzir o tempo médio total em todos os cenários. Isso ocorre pois, devido à inferência e notificação de acidentes, alguns veículos trocaram seu trajeto para uma rota alternativa, evitando assim os congestionamentos causados pelos acidentes.

Considerando somente os resultados dos veículos de emergência apresentados na figura 3(b), ConProVA contribuiu para a redução do tempo total quando a duração dos acidentes é maior (λ_D = 30min). Quando os acidentes são menos severos e o congestionamento causado por eles é rapidamente solucionado, às vezes é mais adequado simplesmente esperar na rota congestionada ao invés de escolher uma rota alternativa que pode ser mais demorada em situações normais de tráfego.

Emissão de CO_2 e distância total: A emissão de CO_2 mede a quantidade de gás carbônico originado dos veículos que foi depositada no ar durante o trajeto dos mesmos. Essa informação é relevante nos dias atuais devido à grande preocupação com a qualidade do ar e a sustentabilidade. Para se ter uma ideia, na cidade de Belo Horizonte-MG, no ano de 2010 foram emitidos 3,75 milhões de toneladas de CO_2 no ar, sendo que 71% desse montante é proveniente de veículos[1]. Esse número é 18% maior se comparado com o ano de 2007. Portanto, é de interesse da sociedade que se tenha soluções para diminuir a emissão de gases poluentes no ar.

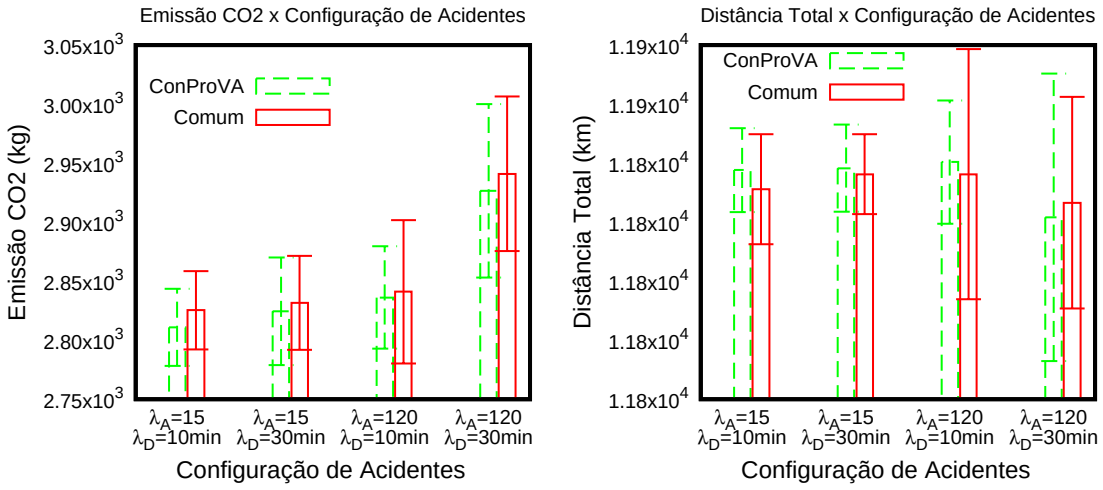
Nesse trabalho, quantidade de CO_2 emitida é calculada com base em vários fatores como a velocidade e aceleração dos veículos, de acordo com a fórmula definida por [Cappiello et al. 2002]. Como apresentado na figura 4(a), os cenários em que os veículos adotam o ConProVA emitiram menos CO_2 do que os cenários comuns. Isso ocorre pois, em geral, os veículos que adotam o ConProVA gastam menos tempo no tráfego por escolherem rotas alternativas quando acidentes ocorrem e são inferidos. Essa diminuição da emissão de CO_2 ocorre mesmo com os veículos tendo que percorrer uma distância maior, como mostra a figura 4(b), devido às rotas alternativas serem, em geral, mais longas que as rotas tradicionais.

Sobrecarga de comunicação: Para avaliar o impacto da adoção do ConProVA em relação à largura de banda necessária, o número de mensagens trocadas foi contabilizado e é apresentado na tabela 2. Como pode ser observado, quanto maior a probabilidade de acidentes (λ_A = 120), maior o número de mensagens trocadas pois são enviadas mais mensagens de subscrição e publicação. Porém, a taxa de mensagens trocadas por unidade de tempo (aproximadamente 1 mensagem a cada 23 segundos) não é significativa para sobrecarregar a rede, sendo que os benefícios vistos anteriormente superam essa troca extra de mensagens.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou ConProVA, um *middleware* para provisionamento de contextos para redes veiculares. ConProVA foi definido considerando as características específicas das redes veiculares e a sua arquitetura é flexível para ser facilmente adotado por diferen-

[1]fonte: 2º Inventário de Emissão de Gases de Efeito Estufa, realizado em Belo Horizonte em Dezembro de 2012



(a) Emissão total de CO_2 (b) Distância total percorrida

Figura 4. Emissão de CO_2 e distância percorrida. λ_A = parâmetro da distribuição de Poisson para o número de acidentes e λ_D = parâmetro da distribuição de Poisson para a duração dos acidentes

Tabela 2. Número médio de mensagens/segundo enviadas por veículo

λ_D	λ_A	**Subscrição Msg/Segundo/veículo**	**Publicação Msg/Segundo/veículo**
10min	15	0,043± 0,012	0,0005±0,003
10min	120	0,043±0,013	0,0008±0,003
30min	15	0,042±0,013	0,0006±0,004
30min	120	0,044±0,015	0,0014±0,005

tes aplicações. Além de possibilitar o provisionamento de contextos físicos e de implementar o paradigma de comunicação *Publish/Subscribe* para publicação e notificação de eventos, o *middleware* proposto possui duas outras funcionalidades importantes em redes veiculares: inferência de contextos lógicos e resolução de conflitos de interesse. Essas duas funcionalidades são os principais diferenciais do ConProVA em relação a soluções encontradas na literatura.

Simulações feitas utilizando um cenário real de tráfego mostraram que a adoção do ConProVA possibilitou o aumento da satisfação dos usuários ao diminuir o tempo que os veículos gastam para chegar ao destino final e reduzir a emissão de gás carbônico quando acidentes ocorrem e consequentemente, causam congestionamento.

Como trabalhos futuros, pode-se listar o refinamento dos módulos de inferência e resolução de conflitos, a proposta de técnicas para medir a qualidade dos contextos (QoC) providos e a adoção do ConProVA junto a outras aplicações.

Referências

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). Sumo - simulation of urban mobility: An overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain.

Bychkovsky, V., Chen, K., Goraczko, M., Hu, H., Hull, B., Miu, A., Shih, E., Zhang, Y., Balakrishnan, H., and Madden, S. (2006). The cartel mobile sensor computing

system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 383–384, New York, NY, USA. ACM.

Campista, M. E. M., Moraes, I. M., Rubinstein, M. G., and Duarte, O. C. M. B. (2009). Redes Veiculares: Princípios, Aplicações e Desafios. In *Minicursos do XXVII Simpósio Brasileiro de Redes de Computadores.*, chapter 5, pages 199–254. Recife.

Cappiello, A., Chabini, I., Nam, E., Lue, A., and Abou Zeid, M. (2002). A statistical model of vehicle emissions and fuel consumption. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 801 – 809.

Eckhoff, D. and Sommer, C. (2012). A Multi-Channel IEEE 1609.4 and 802.11p EDCA Model for the Veins Framework. In *5th ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2012)*. ACM.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131.

Hall, C. C. (2012). Cologne city hall. Available in http://www.stadt-koeln.de/en/.

Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Leontiadis, I., Costa, P., and Mascolo, C. (2009). A hybrid approach for content-based publish/subscribe in vehicular networks. *Pervasive and Mobile Computing*, 5(6):697 – 713. PerCom 2009.

Nour, S., Negru, R., Xhafa, F., Pop, F., Dobre, C., and Cristea, V. (2011). Middleware for data sensing and processing in vanets. In *Emerging Intelligent Data and Web Technologies (EIDWT), 2011 International Conference on*, pages 42 –48.

Riva, O. (2006). Contory: A middleware for the provisioning of context information on smart phones. In van Steen, M. and Henning, M., editors, *Middleware 2006*, volume 4290 of *Lecture Notes in Computer Science*, pages 219–239. Springer Berlin / Heidelberg.

Silva, T. R. M. B., Ruiz, L. B., and Loureiro, A. A. F. (2010). Uma arquitetura para resolução de conflitos coletivos em sistemas ubíquos e cientes de contexto. In *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 119–132.

Uppoor, S. and Fiore, M. (2011). Large-scale urban vehicular mobility for networking research. *2011 IEEE Vehicular Networking Conference (VNC)*, pages 62–69.

Varga, A. (2001). The OMNET++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference*, pages 319–324, Prague, Czech Republic. SCS – European Publishing House.

Ye, J., Dobson, S., and McKeever, S. (2012). Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing*, 8(1):36–66.

Uma Interface de Prototipagem para Aplicações Pervasivas

David Barreto[1], **Matheus Erthal**[1], **Douglas Mareli**[1], **Orlando Loques**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
CEP: 24210 – 240 – Niterói – RJ – Brasil

{dbarreto, merthal, dmareli, loques}@ic.uff.br

Resumo. *Este artigo descreve a Interface de Prototipagem e Gerenciamento de Aplicações Pervasivas (IPGAP), que tem como objetivo prover uma plataforma de suporte à construção, teste e execução de aplicações para ambientes inteligentes (smart ambients). Para prover essa funcionalidade, a ferramenta proposta facilita a simulação de sensores e atuadores bem como meios para visualizar a interação com componentes reais presentes no ambiente. Assim, o desenvolvedor poderá construir suas aplicações sem a necessidade de se ter a infraestrutura completa de um ambiente inteligente.*

Abstract. *This article describes the Pervasive Applications Prototyping and Management Interface (IPGAP) that aims to provide a platform to support construction, test and execution of applications for smart ambients. In order to provide these features capabilities, our tool helps to perform simulation of sensors and actuators as well as means to visualize the interaction of real components which are inside the ambient. This way the developer will be able to construct applications without having a complete smart ambient infrastructure.*

1. Introdução

Desde as propostas de Mark Weiser na década de 1990 [Weiser 1991], os pesquisadores da área de computação ubíqua/pervasiva vêm propondo mudanças na interação homem-máquina, visando tornar o uso de dispositivos cada vez mais transparente no ambiente. Isso possibilita ao usuário, manter o foco na tarefa a ser realizada e não na ferramenta para realizá-la. A partir dessas ideias surgiu o conceito de ambientes inteligentes [Augusto and McCullagh 2007], onde sensores e atuadores interconectados em rede são capazes de fornecer informações relevantes sobre o ambiente para aplicações e usuários, bem como, efetivamente, agir neste ambiente e alterar seu estado.

O mercado de aplicações para plataformas móveis vem caminhando na direção acima descrita, com milhões de aplicações desenvolvidas e distribuídas para os usuários nos últimos anos, provendo serviços que cada vez mais estão se inserindo em seu cotidiano. Esse sucesso se deve aos crescentes avanços nas tecnologias de comunicação e, sobretudo, ao surgimento de sistemas operacionais mais adequados para os dispositivos móveis, como Google Android, Apple iOS e Microsoft Windows Phone. Incluem-se neste número diversas aplicações interessantes como, por exemplo, um aplicativo para identificação de estresse no usuário através da captação de sua voz pelo microfone do aparelho [Lu et al. 2012], e um aplicativo que adquire a frequência cardíaca através do LED da câmera de um *smartphone* [Gregoski et al. 2012]. Entretanto, de modo geral essas aplicações ainda são auto-contidas, ou seja, não compartilham as informações geradas, nem expõem seus serviços no ambiente a fim de cooperar com outros aplicativos

e provisionar serviços diferenciados para o usuário. O grande desafio da computação ubíqua/pervasiva é justamente utilizar essas aplicações integradas a um ambiente inteligente, fornecendo seus serviços e informações a outras entidades.

Em comparação ao avanço no desenvolvimento de aplicações para o mercado de dispositivos móveis, as aplicações ubíquas ainda são escassas no mercado. Podemos citar como causas desse efeito a dificuldade em integrar os dispositivos que compõem uma aplicação ubíqua, a falta de ferramentas adequadas para a criação e integração dessas aplicações, e a dificuldade em depurá-las [Weis et al. 2007]. Além disso, um ambiente de testes contendo todos os dispositivos e a infraestrutura necessária para realizá-los pode ser inviável financeiramente, ao passo que um ambiente construído em pequena escala pode não ser suficiente para testar os diversos cenários possíveis em um ambiente inteligente.

Para resolver esses problemas, é proposta uma ferramenta de suporte à construção de protótipos de aplicações pervasivas chamada IPGAP (Interface de Prototipagem e Gerenciamento de Aplicações Pervasivas), que fornece ao desenvolvedor um ambiente de testes para suas aplicações de maneira rápida e com baixo custo. Além disso, a IPGAP oferece um conjunto de serviços básicos para gerenciamento dos recursos do ambiente (como descoberta e registro), APIs para invocação remota de operações e comunicação por eventos, e um suporte para interpretação de contexto, contendo uma GUI (*Graphic User Interface)* para composição de regras. Assim, o desenvolvedor poderá utilizar a infraestrutura provida para criar uma aplicação pervasiva mais facilmente.

Consideremos uma aplicação em que uma mídia (e.g. vídeo, música) é retransmitida automaticamente para o aparelho mais próximo ao usuário, assim que ele se ausenta do local onde está sendo originalmente reproduzida (Seção 4.1). O desenvolvedor dessa aplicação, além de contar com uma API que possibilita reunir as informações de sensores e permitiar a atuação nos recursos do ambiente, poderá testar sua aplicação na interface de prototipagem de forma rápida e barata, sem a necessidade de montar uma infraestrutura completa para um ambiente inteligente.

O restante deste artigo está organizado como a seguir. Na Seção 2, apresentaremos uma visão geral dos principais conceitos utilizados como base para o desenvolvimento da IPGAP. Na Seção 3, veremos mais detalhes sobre o funcionamento da ferramenta, seus conceitos, características e exemplos de utilização. Mostraremos na Seção 4 uma avaliação da IPGAP através de aplicações desenvolvidas. Os trabalhos relacionados serão apresentados e discutidos na Seção 5, e as conclusões e trabalhos futuros, na Seção 6.

2. Visão Geral da Infraestrutura da IPGAP

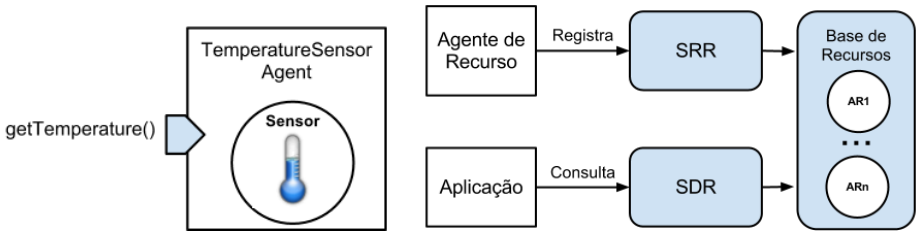
A IPGAP é pautada em conceitos desenvolvidos em nosso grupo de pesquisa, que foram postos em prática através do projeto **SmartAndroid**[1], desenvolvido para comprovação de conceitos propostos em três dissertações de Mestrado [Barreto 2012, Erthal 2012, Mareli 2012]. O projeto SmartAndroid contempla o desenvolvimento das APIs e todo o *framework* proposto, além da interface de prototipagem, utilizando a plataforma Android. Veremos a seguir uma breve descrição dos conceitos que servem de base para as implementações realizadas.

[1]Para mais informações visite www.tempo.uff.br/smartandroid

2.1. Agentes de Recurso

Sensores, atuadores, dispositivos e eletrodomésticos inteligentes, além de módulos de *software* que forneçam algum serviço para o ambiente, são definidos como *recursos*. Estes são encapsulados em Agentes de Recursos (AR), que podem ser compreendidos como elementos que expõem informações dos recursos juntamente com sua interface, de forma que outras entidades possam acessá-las de maneira uniforme [Sztajnberg et al. 2009].

Os ARs escondem detalhes de baixo nível do recurso encapsulado, diminuindo significativamente a complexidade de integração de um recurso no sistema. Por exemplo, os detalhes da coleta de dados de um sensor de temperatura seriam conhecidos apenas pelo seu AR, que se encarrega de fornecer uma interface simples para que os outros componentes do sistema tenham acesso as informações desse sensor. Na Figura 1(a) temos um AR que encapsula um sensor de temperatura, expondo o método `getTemperature()`.



(a) Um Agente de Recurso para um sensor de Temperatura

(b) Serviço Básicos do *framework*

Figura 1. Componentes do Sistema

2.2. Serviços Básicos

É necessário que os recursos do ambiente sejam descobertos para que as aplicações possam utilizá-los. Por esse motivo existe o *Serviço de Descoberta de Recursos* (SDR), que permite que estes sejam localizados por meio de consultas a uma base, populada com as referências dos ARs pelo *Serviço de Registro de Recursos* (SRR) (Figura 1(b)) [Mareli 2012].

A descoberta pode ser realizada através de vários tipos de consulta, que retornam como resultado referências para os ARs que satisfazem os critérios da busca. Algumas consultas envolvem o tipo dos recursos, que são caracterizados através da definição de uma ontologia mínima [Bezerra 2011]. Na Figura 2 vemos a assinatura dos principais métodos de busca do SDR.

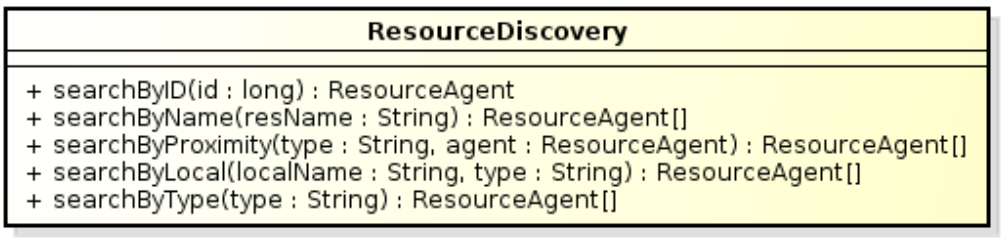


Figura 2. Métodos de Busca do SDR

Como exemplo de uso do mecanismo de descoberta, apresentamos uma aplicação de monitoramento de pacientes [Carvalho et al. 2010]. Essa aplicação utiliza sensores para inferir o estado de saúde do paciente, através da coleta contínua de dados do ambiente (e.g. temperatura, umidade), dados fisiológicos (e.g. pressão arterial, frequência cardíaca), bem como um plano de cuidados – um conjunto de prescrições feitas por um profissional de saúde, contendo os medicamentos que o paciente deve tomar, medições fisiológicas, exercícios físicos, e outras recomendações com seus respectivos horários.

Para aumentar a adesão do paciente ao tratamento, é salutar que este seja alertado no momento em que deve realizar uma tarefa, através de um dispositivo como TV, celular, *tablet*, etc. Porém, nem sempre um dispositivo está à vista do paciente. Por exemplo, enviar a mensagem *"Está na hora de tomar o remédio"* para o celular do paciente, pode não ser eficaz se o paciente não tiver o hábito de estar próximo ao celular. Por outro lado, enviar a mesma mensagem para todos os dispositivos de visualização da residência, apesar de à primeira vista ser mais eficaz, pode ser um estorvo para o paciente e os outros moradores da casa, além de ser uma abordagem muito intrusiva podendo causar constrangimentos e exposição desnecessária do paciente. Utilizando-se o SDR, a aplicação poderia fazer a consulta: *"o dispositivo de visualização mais próximo do paciente"* para obter uma referência para este recurso, e assim, enviar a mensagem de alerta. Dessa forma a aplicação seria menos intrusiva e reduziria a possibilidade do paciente não perceber o alerta. Um trecho de código com exemplos de uso do SDR poderá ser encontrado na Seção 4.1.

2.3. Contexto

Aplicações sensíveis ao contexto são integradas com o mundo físico, e respondem a estímulos do ambiente obtidos através de sensores. Essa é uma caracterização fundamental para sistemas ubíquos/pervasivos. Assim, podemos dizer que essas aplicações são interessadas nas informações dos recursos que sejam relevantes para o sistema, ou seja, no contexto dos recursos e do ambiente [Abowd et al. 1999]. Em uma aplicação feita para um ambiente inteligente, por exemplo, o contexto poderia ser: se uma lâmpada está acesa, o canal em que se encontra uma TV, a temperatura de um ar-condicionado, entre outros.

As aplicações construídas através do *framework* proposto, podem utilizar regras envolvendo o contexto dos recursos, conhecidas como *regras de contexto*. Essas regras são compostas por condições obtidas de informações de contexto dos ARs envolvidos e de um temporizador. Um *interpretador de contexto* avalia constantemente as condições da regra. Se verificado que a condição é verdadeira, este notifica os ARs interessados, ou seja, aqueles que vão de fato efetuar uma ação no ambiente (Figura 3). Essa notificação é realizada através do padrão *publish-subscribe*, largamente utilizado em sistemas distribuídos.

As regras de contexto são criadas na IPGAP através de uma GUI, que permite a seleção das condições da regra intuitivamente. Ao acionar-se o ícone do dispositivo desejado são exibidas suas informações de contexto, que podem então ser selecionadas e incluídas na regra. Uma regra pode envolver o contexto de diversos dispositivos, combinados por conectivos lógicos. A partir da descrição da regra, as referências dos ARs envolvidos são obtidas automaticamente, e a regra é então executada. Assim, podem ser criadas regras simples, como desligar aparelhos que não estão sendo utilizados por

um certo tempo, ou complexas, como acionar uma ambulância caso um morador tenha sua situação de saúde identificada como crítica, através da monitoração de seus dados fisiológicos [Copetti et al. 2012]. Um estudo mais profundo sobre interpretação de contexto pode ser encontrado em [Erthal 2012].

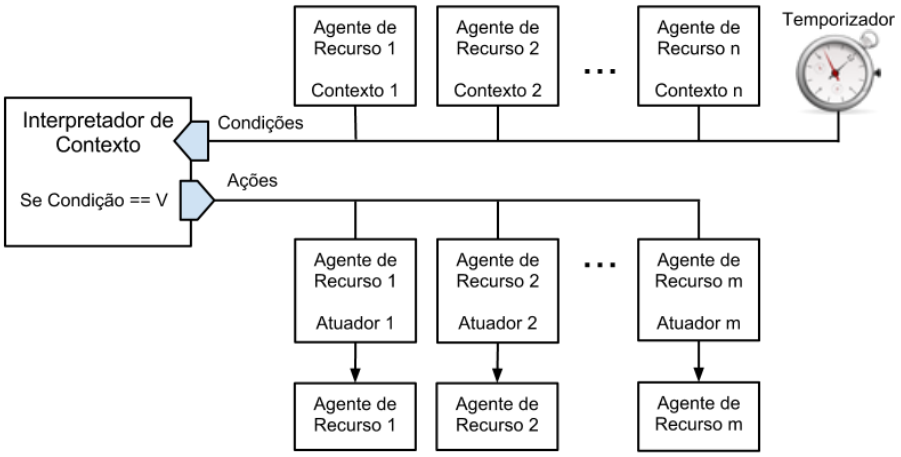


Figura 3. Interpretação de uma Regra de Contexto

3. A Interface de Prototipagem de Aplicações Pervasivas

A IPGAP foi desenvolvida para fornecer a seus usuários um mundo em que os dispositivos interajam entre si, e são facilmente acessíveis utilizando a API provida no *framework*. Um desenvolvedor de aplicações para ambientes ubíquos/pervasivos pode testar suas aplicações no ambiente da IPGAP, de modo a avaliá-las e, consequentemente, aperfeiçoá-las. Isso seria um desafio se fosse preciso desenvolver toda infraestrutura a partir do zero.

Através de aparelhos de fácil aquisição, que podem simular vários dispositivos de um ambiente inteligente, pode-se criar diversos cenários a baixo custo. Por exemplo, nossa prova de conceito foi implementada sobre o sistema operacional Android, sendo utilizado um *tablet* para visualização da ferramenta de prototipagem (Figura 4). Os simuladores dos dispositivos (ver Seção 3.4.1) foram instalados em celulares de baixo custo conectados por uma rede sem fio.

A utilização desses aparelhos diminui grande parte do custo de desenvolvimento (financeiro e temporal) do projeto, pois não exige a aquisição de dispositivos reais, como sensores e atuadores, o que é útil em fases iniciais. A ferramenta também dá suporte à integração de dispositivos reais no ambiente juntamente com os dispositivos simulados, criando assim um ambiente híbrido. Para as aplicações este fato é totalmente transparente, pois permite que os dispositivos reais e simulados sejam facilmente intercambiáveis. É importante salientar que os serviços básicos do sistema – como o SDR, SRR e os interpretadores de contexto vistos na Seção 2 – podem ser executados em máquinas mais robustas, assegurando, assim, requisitos de qualidade de serviço e tolerância a falhas.

3.1. A IPGAP para o Desenvolvedor

O desenvolvedor tem a capacidade de, através da IPGAP, visualizar o ambiente alvo na tela de um computador ou *tablet*, em um formato similar ao de uma planta baixa

Figura 4. IPGAP: Visão do mapa do ambiente

(Seção 3.3). O ambiente pode então ser populado na tela com os recursos necessários à aplicação que está sendo desenvolvida. Por exemplo, para que uma aplicação de controle de iluminação seja testada, é necessário que se tenha lâmpadas disponíveis, além de sensores que detectem a presença de pessoas nos cômodos e avatares para representar essas pessoas. O desenvolvedor pode adicionar todos esses recursos no mapa da casa e executar sua aplicação. Para testá-la, uma das alternativas seria utilizar a IPGAP para movimentar o avatar no mapa, e observar os sensores detectando a presença ou ausência do indivíduo, bem como as lâmpadas se acendendo e apagando.

3.2. A IPGAP para o Usuário Final

A IPGAP possui a característica de ser um *software* intuitivo, por conta de sua GUI. Isso faz com que seja uma ferramenta muito conveniente para o usuário final (entenda-se "usuário final" como um utilizador do ambiente inteligente, potencialmente leigo na área da computação.), pois se torna um poderoso controle remoto de todos os dispositivos de sua residência. Assim, o usuário final pode, via rede sem fio, manipular os dispositivos e criar regras de contexto que atuarão em seu ambiente. Dessa forma, pode-se configurar todo o ambiente de acordo com as preferências do usuário, através de um dispositivo móvel como um *smartphone* ou *tablet*.

A situação ideal é que este usuário possa controlar toda sua casa remotamente, de qualquer parte do mundo através da internet. É claro que isso envolve uma série de questões sobre segurança de rede, níveis de permissões, entre outras. Estes aspectos são abordados com detalhes em [Mareli 2012].

3.3. Representação do ambiente

Anteriormente, mencionamos que os recursos do ambiente poderiam ser visualizados através de representações gráficas na tela de um dispositivo (como um *tablet*), para que o desenvolvedor ou usuário final possa manipulá-los. Essas representações são posicionadas em um mapa esquemático do ambiente inteligente. Cada cômodo no mapa tem

sua área pré-definida pelo usuário em um editor de mapas[2], que também é responsável por definir a aparência da casa representada. Na Figura 4 vemos o mapa do ambiente, populado com alguns recursos rodando em um *tablet*

Os recursos são exibidos no mapa pela IPGAP através de ícones, de acordo com o desejo do usuário, e suas posições são aproximadas em relação a posição real do recurso em metros. Dessa forma, o mapa do ambiente se transforma em uma espécie de "área de trabalho", semelhante à de sistemas operacionais como Windows, Linux e Android, contendo ícones que podem ser acionados para chamar outros processos.

Salientamos que os recursos representados no mapa não são meramente imagens, sendo na verdade um espelho do ambiente inteligente. Os recursos estão registrados no sistema, e podem ser descobertos e utilizados por aplicações.

3.4. Aplicativos

Nossa ferramenta utiliza o conceito de aplicativos, atualmente atribuído ao universo dos *smartphones* e *tablets*, embora exista na grande maioria dos Sistemas Operacionais de propósito geral há bastante tempo (com uma semântica ligeiramente diferente). No contexto das aplicações móveis, existem lojas virtuais (como Google Play e Apple Store) que disponibilizam uma base com milhares de aplicativos que podem ser baixados pelos usuários. Dessa forma, pode-se facilmente customizar o ambiente com a instalação de novos *softwares*.

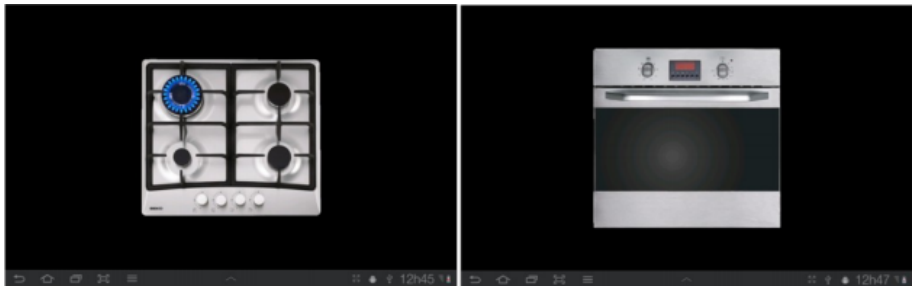


Figura 5. Aplicativo do fogão em um *tablet*. Visão das bocas e do forno

No contexto da IPGAP, um aplicativo possui uma GUI que representa a interface de um determinado recurso, além de poder invocar operações do AR do mesmo (potencialmente em outro *host*), o que nos permite manipular e visualizar seu estado interno de forma intuitiva. Essa funcionalidade se dá através de primitivas de comunicação síncrona, semelhante às utilizadas no RPC (*Remote Procedure Call*) e RMI (*Remote Method Invocation*). Assim como nas abordagens do RPC e RMI, existe um *stub* – um componente que possui a mesma interface do recurso-alvo, porém sua implementação contém chamadas remotas para este recurso, agindo como um *proxy*.

Imagine um aplicativo capaz de exibir a interface de um fogão (Figura 5), onde o usuário pode através dela, acender uma boca ou o forno do aparelho. Ao acionar-se alguma funcionalidade do aplicativo do fogão, essa operação deve invocar a sua correspondente no AR do fogão real (que pode ser embutido no *hardware* do fogão real pelo seu fabricante. Ver Seção 3.5), conforme mostramos no esquema da Figura 6. Note que

[2]Foi o utilizado o *software* **Tiled** (www.mapeditor.org) para criação do mapa do ambiente de testes.

para isso, o aplicativo do fogão deve possuir um objeto *stub* que encaminha a chamada para o fogão real. Um aplicativo pode ser acionado através da IPGAP por meio de sua representação no mapa, acionando seu ícone (ver Seção 3.3), passando a exibir a interface do dispositivo que representa, como na Figura 5.

O caminho inverso também deverá ocorrer, ou seja, o que acontecer no fogão real também deve ser refletido no aplicativo do fogão. Isso é fruto de uma comunicação assíncrona estabelecida entre o fogão real e o aplicativo, provida por uma implementação do padrão *publish-subscribe*. Assim, dizemos que o aplicativo em questão é **interessado** nas mudanças que ocorrerem no fogão real, ou seja, sempre que o fogão real alterar seu estado, o aplicativo será notificado através de um evento

3.4.1. Simuladores

É comum que o desenvolvedor necessite de diferentes entidades presentes em um ambiente inteligente para testar suas aplicações. Isso pode se tornar inviável conforme o número de entidades cresce, pois envolve os custos com os equipamentos e principalmente com a integração destes. Por esse motivo a IPGAP inclui um conjunto de aplicações (que pode ser constantemente ampliado) que simulam os principais dispositivos presentes em um ambiente inteligente, como lâmpada, fogão, TV, ar-condicionado assim como sensores de localização, temperatura, umidade, entre outros. Dessa forma, o desenvolvedor pode criar suas aplicações utilizando esses componentes – chamados de simuladores – como se fossem os equipamentos reais.

Figura 6. Aplicativo do fogão atuando no fogão real

Ressaltamos que os simuladores expõem seus serviços e o seu contexto no ambiente através de ARs, o que torna uma possível troca de um simulador por dispositivo real transparente para o desenvolvedor. Os simuladores fornecidos com a IPGAP são também considerados aplicativos. Um aplicativo simulador deve conter um código para gerar valores (aleatoriamente ou segundo diretivas de simulação) além de conter o AR do referido recurso simulado. Já um aplicativo comum, contém um *stub* que encaminha os parâmetros passados nas operações chamadas para o AR do recurso alvo, através de invocações remotas.

Um desenvolvedor pode, seguindo um estilo bem definido de programação, criar seus próprios aplicativos. Um exemplo desse caso seria um centro de pesquisas que está desenvolvendo um novo tipo de sensor. O primeiro passo seria utilizar um simulador para

este sensor e testá-lo no ambiente da IPGAP juntamente com os demais recursos. Dessa forma pode-se testar o sensor antes mesmo de se ter um protótipo físico completo.

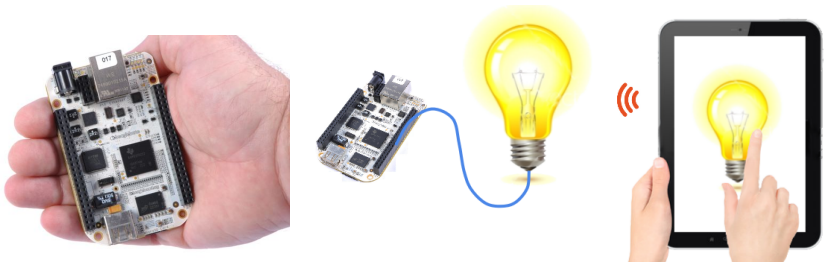
3.4.2. Instalação dos Aplicativos

Como citado anteriormente, nossa ferramenta permite a instalação de aplicativos no ambiente de prototipagem, de maneira similar ao que ocorre nos sistemas operacionais para plataformas móveis. Dessa forma o desenvolvedor tem a possibilidade de estender esse ambiente segundo as suas necessidades. Idealmente, os usuários da IPGAP podem baixar os aplicativos e/ou simuladores em uma loja aos moldes da Google Play e Apple Store, onde pode-se fazer buscas por aplicativos, visualizar sua descrição, fazer o download e assim, instalá-los na IPGAP. Esse processo se dá de forma automatizada, o que torna essa opção útil para o usuário final, podendo este, acrescentar os aplicativos que quiser na interface de visualização do ambiente.

3.5. Utilização de Recursos Externos

Recursos reais como eletrodomésticos e sensores podem fazer parte do sistema, enriquecendo assim os testes realizados na aplicação a ser desenvolvida, além de possibilitar o uso da IPGAP pelos usuários finais para controlar dispositivos de ambientes inteligentes. Para isso os dispositivos devem possuir um chip embutido em seu próprio *hardware*, que implemente as APIs padronizadas do *framework*.

Uma alternativa seria desenvolver um componente (*wrapper*) que implemente ambas as APIs (a API de nossa proposta e a API nativa do recurso desejado). Quando esse componente recebe uma chamada através de nossa API, ele encaminha a chamada para o recurso-alvo, utilizando a API de seu fabricante. Ou seja, o componente faz a "tradução" de uma chamada do nosso sistema para uma chamada nativa da API do aparelho em questão. Essa técnica é mais viável nas etapas de desenvolvimento, pois não depende de que se tenha um recurso que utiliza o padrão do sistema nativamente.



(a) Beaglebone (b) Esquema com Beaglebone e Lâmpada

Figura 7. Experimentos com Beaglebone

Nosso grupo está realizando testes com o Beaglebone [BeagleBoard 2012] com o objetivo de construir protótipos de diversos dispositivos. O Beaglebone (Figura 7(a)) é uma placa de desenvolvimento de tamanho reduzido e baixo custo, que contém pinos para entrada e saída e um processador ARM AM335x, que suporta a execução de sistemas como Ubuntu e Android. Através desse equipamento podemos, por exemplo, conectar uma lâmpada comum e controlá-la através de um relé ligado aos pinos de I/O do Beaglebone. A Figura 7(b) ilustra essa possibilidade.

4. Aplicações

Um conjunto de aplicações pervasivas foi construído utilizando-se nossa API e a IPGAP como plataforma de testes. Através do desenvolvimento dessas aplicações foi possível avaliar a viabilidade da proposta, assim como a flexibilidade e usabilidade da mesma. Foram implementadas aplicações para controle dos dispositivos da casa (como um controle remoto), controle de iluminação da residência e jogos *multiplayer* onde usuários do ambiente podem descobrir jogadores e jogar em conjunto, entre outras aplicações.

Uma das aplicações implementadas, nomeada de **MediaFollowMe**, proporciona ao usuário final a facilidade de repassar áudio/vídeo/imagens (ou qualquer outro tipo de mídia) de um aparelho para outro automaticamente, levando em conta informações como proximidade do dispositivo em relação ao usuário, como se a mídia o seguisse. Exemplificaremos esta aplicação a seguir.

4.1. MediaFollowMe

O usuário final está em casa assistindo a um filme em seu *Blu-ray* na TV da sala, quando em um ponto crítico da trama precisa se deslocar rapidamente para outro cômodo. Ele inicia através de seu *smartphone* o aplicativo **MediaFollowMe** e escolhe dentre uma lista de dispositivos presentes no ambiente (exibida na tela do aparelho) o *Blu-ray* da sala como fonte emissora da mídia.

A partir daí, a aplicação não necessita mais da interação direta do usuário, atuando autonomamente no ambiente. As referências dos ARs que representam o *Blu-ray* da sala e o próprio usuário são obtidas através do SDR, para saber de onde obter a mídia e para onde deve transmiti-la (conforme a posição do usuário). Além disso, a aplicação deve possuir as referências dos ARs dos sensores de presença da casa, a fim de receber um evento (ou seja, ser notificada) quando o usuário entrar ou sair de um cômodo. Note que pessoas também são representadas no sistema através de ARs. O AR que representa uma pessoa, na verdade, agrega diversos outros ARs, como os de sensores corporais, emissores de RF-ID, entre outros dispositivos.

Listagem 1. Obtendo as referências e registrando interesse nos eventos

```
1 ResourceAgent[] pSensors = discovery.searchByType("PresenceSensor");
2
3 for (sensor : pSensors) {
4   sensor.registerStakeholder("IN", this);
5   sensor.registerStakeholder("OUT", this);
6 }
```

Após obter todas as referências que precisa, a aplicação MediaFollowMe se registra como interessada em receber eventos dos ARs dos sensores de presença do ambiente. Na Listagem 1 vemos um trecho de código exemplificando esse processo. Na linha 1 é feita uma busca por todos os sensores de presença do ambiente através do SDR. Para cada sensor obtido, o MediaFollowMe se registra como interessado em eventos de entrada (linha 4) e saída (linha 5) de usuários nos cômodos onde se encontram os sensores.

No momento em que o usuário deixar a sala, o sensor de presença correspondente detectará que alguém saiu, e notificará através de um evento todos os interessados nessa informação, incluindo o MediaFollowMe. Os eventos devem informar quem é o recurso

que o originou (neste caso, o sensor de presença da sala), qual o contexto alterado (entrada ou saída do cômodo) e o qual o valor alterado (qual usuário se locomoveu). Assim, a aplicação receberá a referência do AR do usuário que saiu do cômodo em questão, e pausará a mídia proveniente do *Blu-ray* da sala, devido à ausência do indivíduo.

Dessa forma, quando o usuário adentra a cozinha, o sensor de presença deste cômodo detecta a sua chegada e notifica à aplicação via eventos. A aplicação busca o dispositivo mais próximo do usuário e descobre uma TV na cozinha. Agora, o MediaFollowMe pode realizar uma chamada remota para o *Blu-ray* (fonte dos dados) solicitando que ele mude o receptor de vídeo da TV da sala para a TV da cozinha, e retoma a reprodução do filme (que estava em pausa). Isso é feito automaticamente e sem fio, resultando na transferência da mídia para esta TV.

Na Listagem 2 apresentamos o tratamento dos eventos recebidos pela aplicação. O método de *callback* **notificationHandler** é definido na API do *framework* e deve ser implementado pelos objetos que desejam receber notificações de eventos. Este método passa como parâmetros o AR que gerou o evento, o contexto que foi alterado e o valor que foi alterado. No MediaFollowMe esses parâmetros são o AR do sensor de presença que gerou o evento, se o evento é de entrada (IN) ou saída (OUT), e qual usuário entrou ou saiu do cômodo. A classe **Person** (linha 4) é uma subclasse de **ResourceAgent**, portanto pode ser passada como parâmetro na consulta de proximidade (linha 12), que retorna uma lista de recursos ordenados por proximidade em relação ao recurso passado (neste caso o usuário). Na linha 6 a aplicação testa se o usuário que entrou/saiu do cômodo em questão é o usuário requerido. Por fim, na linha 13 é feita uma chamada para o AR do *Blu-ray* solicitando que o *stream* da mídia seja repassado para o dispositivo de visualização mais próximo do usuário, e na linha 14, é realizada uma chamada remota ao *Blu-ray* para reproduzir a mídia.

Listagem 2. Tratamento dos eventos e consulta por dispositivo mais próximo

```
1  @Override
2  public void notificationHandler(ResourceAgent res, String context,
       Object obj) {
3    ...
4    Person p = (Person) obj;
5
6    if (p.getName().equals(USER_NAME) {
7      if (context.equals("OUT") {
8
9        bluray.pause();
10     } else if(context.equals("IN") {
11
12       ResourceAgent[] views = discovery.searchByProximity("View", p);
13       bluray.streamTo(views[0]);
14       bluray.play();
15     }
16   }
17 }
```

Foi possível observar a aplicação em funcionamento e o comportamento dos recursos envolvidos através da IPGAP. Ao movimentar-se o avatar do usuário pelos cômodos da casa, pôde-se visualizar a mídia sendo exibida no dispositivo de visualização mais

próximo a ele, conforme esperado.

5. Trabalhos Relacionados

Podemos encontrar na literatura outras propostas de ferramentas para prototipagem de aplicações pervasivas (veja [Tang et al. 2010] para uma *survey*). Em [Armac and Retkowitz 2007, Van Nguyen et al. 2009] encontramos ferramentas de teste e a avaliação de serviços para ambientes inteligentes. Entretanto, não incluem uma abordagem de descoberta de dispositivos recém adicionados no ambiente. A IPGAP abrange essa questão através do SDR e SRR (Seção 2).

Já em [Zhang et al. 2010] e [Fu et al. 2011] encontramos simuladores para ambientes inteligentes baseados em OSGi (*Open Service Gateway Initiative*), oferecendo a possibilidade de adição e remoção de dispositivos sem alterar o código do sistema, através da modificação de arquivos de configuração. Porém, estes trabalhos não focam no perfil do usuário final (Seção 3.2), pois não proveem uma interface clara para que o usuário configure seus dispositivos.

Em [Bruneau and Consel 2012] é apresentado um simulador para aplicações pervasivas, parametrizado por diretivas escritas em uma linguagem de configuração própria, porém não oferece suporte à configuração dinâmica das entidades do sistema. Além disso o trabalho não deixa claro a possibilidade de transparência entre dispositivos reais e simulados, recursos providos pela IPGAP.

Outra vertente de trabalhos sobre simulação de sistemas pervasivos pode ser encontrada nos trabalhos [Barton and Vijayaraghavan 2002, Nishikawa et al. 2006]. Esses projetos possuem uma visualização 3D do ambiente em que o usuário tem a possibilidade de controlar um avatar em uma visão de 3ª pessoa, como nos jogos de computador. Assim como a IPGAP, estes projetos permitem que se configure dispositivos no sistema e se interaja com eles. Porém, as ferramentas propostas nestes trabalhos focam em testes de interação dos dispositivos e a simulação do ambiente físico, não se preocupando com sensibilidade ao contexto, ou seja, não consideram um ambiente adaptativo onde o contexto pode causar a alteração do estado dos recursos.

6. Conclusão e Trabalhos Futuros

Neste artigo foi apresentada a IPGAP – Interface de Prototipagem e Gerenciamento de Aplicações Pervasivas – que visa auxiliar o desenvolvedor a construir e testar aplicações para ambientes inteligentes. A IPGAP permite que a criação de protótipos funcionais seja realizada em menos tempo e à baixo custo, através do uso de um misto de simuladores e dispositivos reais, e uma API para gerenciamento do ambiente. As aplicações desenvolvidas podem ser facilmente instaladas no ambiente inteligente, permitindo que sejam realizados testes com uma infinidade de cenários. Além disso, aplicativos para gerenciamento dos recursos do ambiente (como fogão e TV) podem ser instalados na IPGAP, permitindo que seja estendida de acordo com as necessidades do desenvolvedor, sem necessidade de alteração no código.

Entre as principais contribuições deste trabalho está a utilização de uma abordagem onde os recursos são descobertos no ambiente de forma autônoma, permitindo assim a inclusão de novos recursos sem que para isso tenha-se que recompilar o código

da IPGAP ou mesmo interromper seu funcionamento. Além disso, um diferencial desta proposta é o foco também no usuário final, que pode através da IPGAP controlar o ambiente inteligente em que se encontra. As próximas etapas do projeto incluem o tratamento de informações de contexto mais complexas e a implementação de uma visualização 3D do ambiente inteligente na interface, além da utilização de uma abordagem baseada em contratos [Carvalho et al. 2011], para adaptação dinâmica da aplicação às necessidades do usuário final, possibilitando também a criação de subsistemas.

Referências

Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In Gellersen, H.-W., editor, *Handheld and Ubiquitous Computing*, volume 1707 of *Lecture Notes in Computer Science*, pages 304–307. Springer Berlin Heidelberg.

Armac, I. and Retkowitz, D. (2007). Simulation of Smart Environments. In *IEEE International Conference on Pervasive Services*, pages 257–266. IEEE.

Augusto, J. and McCullagh, P. (2007). Ambient intelligence: Concepts and applications. *Computer Science and Information Systems/ComSIS*, 4(1):1–26.

Barreto, D. (2012). *Uma Interface de Prototipagem e Gerenciamento para Aplicações Pervasivas*. Dissertação de mestrado em andamento, Instituto de Computação – Universidade Federal Fluminense.

Barton, J. J. and Vijayaraghavan, V. (2002). Ubiwise, a ubiquitous wireless infrastructure simulation environment. Technical Report HPL-2002-303, Hewlett-Packard Laboratories, Palo Alto.

BeagleBoard (2012). Open hardware physical computing on arm and linux. http://beagleboard.org/.

Bezerra, L. N. (2011). *Uso de ontologia em serviço de contexto e descoberta de recursos para autoadaptação de sistemas*. Dissertação de mestrado, Universidade do Estado do Rio de Janeiro.

Bruneau, J. and Consel, C. (2012). Diasim: a simulator for pervasive computing applications. *Software: Practice and Experience*.

Carvalho, S. T., Erthal, M., Mareli, D., Sztajnberg, A., Copetti, A., and Loques, O. (2010). Monitoramento Remoto de Pacientes em Ambiente Domiciliar. In *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC 2010*, pages 1005–1012, Gramado, RS, Brasil.

Carvalho, S. T., Murta, L., and Loques, O. (2011). A contract-based approach for managing dynamic variability in software product line architectures. Technical report, Laboratorio Tempo – Universidade Federal Fluminense.

Copetti, A., Leite, J., Loques, O., and Neves, M. (2012). A decision-making mechanism for context inference in pervasive healthcare environments. *Decision Support Systems*.

Erthal, M. (2012). *Interpretação de contexto em ambientes ubíquos inteligentes*. Dissertação de mestrado em andamento, Instituto de Computação – Universidade Federal Fluminense.

Fu, Q., Li, P., Chen, C., Qi, L., Lu, Y., and Yu, C. (2011). A configurable context-aware simulator for smart home systems. In *2011 6th International Conference on Pervasive Computing and Applications*, pages 39–44. IEEE.

Gregoski, M. J., Mueller, M., Vertegel, A., Shaporev, A., Jackson, B. B., Frenzel, R. M., Sprehn, S. M., and Treiber, F. A. (2012). Development and validation of a smartphone heart rate acquisition application for health promotion and wellness telehealth applications. *Int. J. Telemedicine Appl.*, 2012:1:1–1:1.

Lu, H., Frauendorfer, D., Rabbi, M., Mast, M. S., Chittaranjan, G. T., Campbell, A. T., Gatica-Perez, D., and Choudhury, T. (2012). Stresssense: detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 351–360. ACM.

Mareli, D. (2012). *Um framework de desenvolvimento de aplicações ubíquas em ambientes inteligentes*. Dissertação de mestrado em andamento, Instituto de Computação – Universidade Federal Fluminense.

Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., and Ito, M. (2006). UbiREAL: Realistic Smartspace Simulator for Systematic Testing. In Dourish, P. and Friday, A., editors, *UbiComp 2006: Ubiquitous Computing*, volume 4206 of *Lecture Notes in Computer Science*, pages 459–476. Springer Berlin / Heidelberg.

Sztajnberg, A., Rodrigues, A. L. B., Bezerra, L. N., Loques, O. G., Copetti, A., and Carvalho, S. T. (2009). Applying context-aware techniques to design remote assisted living applications. *International Journal of Functional Informatics and Personalised Medicine*, 2(4):358.

Tang, L., Yu, Z., Zhou, X., Wang, H., and Becker, C. (2010). Supporting rapid design and evaluation of pervasive applications: challenges and solutions. *Personal and Ubiquitous Computing*, 15(3):253–269.

Van Nguyen, T., Kim, J. G., and Choi, D. (2009). ISS: The Interactive Smart home Simulator. In *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, volume 03, pages 1828–1833.

Weis, T., Knoll, M., Ulbrich, A., Muhl, G., and Brandle, A. (2007). Rapid prototyping for pervasive applications. *Pervasive Computing, IEEE*, 6(2):76 –84.

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, 3:94–104.

Zhang, L., Suo, Y., Chen, Y., and Shi, Y. (2010). SHSim: An OSGI-based smart home simulator. In *2010 3rd IEEE International Conference on Ubi-Media Computing*, pages 87–90. IEEE.

Um *Framework* de Desenvolvimento de Aplicações Ubíquas em Ambientes Inteligentes

Douglas Mareli[1], Matheus Erthal[1], David Barreto[1], Orlando Loques[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

{dmareli,merthal,dbarreto,loques}@ic.uff.br

Resumo. *Com os recentes avanços da computação móvel e nas tecnologias de comunicação sem fio, percebe-se o surgimento de um cenário favorável à construção de aplicações ubíquas. Este trabalho propõe um novo framework para a construção de tais aplicações, provendo um ferramental conceitual e de implementação. São propostas abstrações que possibilitam aos desenvolvedores lidar com os recursos distribuídos no ambiente de maneira simples e homogênea, e interpretar as informações de contexto. A fim de se demonstrar a viabilidade da proposta, os conceitos foram implementados em uma plataforma chamada SmartAndroid. Sobre esta plataforma foi implementada uma interface de prototipagem de aplicações ubíquas onde configurações de ambientes podem ser testadas antes da aquisição de todos os dispositivos; e uma interface de composição de regras de contexto, onde usuários finais podem definir suas preferências no ambiente.*

Abstract. *Due to recent advances in mobile computing and wireless communication technologies, we can see the emergence of a favorable scenario for building ubiquitous applications. This work proposes a new framework that aims at building those applications, providing a set of concepts and implementation tools. We propose abstractions that allow developers to handle the resources spread in the environment in a simple and homogeneous way, and to interpret context information. In order to demonstrate the feasibility of the proposal, the concepts were implemented in a platform named SmartAndroid. A ubiquitous applications prototyping interface was implemented over this platform to allow testing of different environment configurations before purchasing all devices; and also a context rules composition interface, where end users can define their preferences in the environment.*

1. Introdução

A Computação Ubíqua, como proposta por Weiser na década de 1990 [Weiser 1991], descrevia uma mudança no paradigma de interação entre o usuário e os sistemas computacionais. Weiser previu o surgimento do que chamou de "computação calma", onde a interação entre os usuários e os computadores ocorre de forma natural, sem ações explícitas. Uma aplicação ubíqua identifica as necessidades de seus usuários coletando, por meio de sensores, as informações do seu contexto de execução, e as atende provendo serviços, por meio de atuadores, os quais incluem diversos tipos de interfaces.

A construção e a manipulação de aplicações ubíquas representam grandes desafios para desenvolvedores, especialmente em termos do conhecimento técnico exigido e

da disponibilidade de dispositivos reais durante o desenvolvimento da aplicação. Alguns desses desafios podem ser assim destacados: (i) há dificuldades em se estabelecer um protocolo comum de comunicação entre os componentes do sistema distribuído, por conta da *heterogeneidade dos dispositivos envolvidos*; (ii) a interatividade das aplicações ubíquas é dificultada dependendo da quantidade e da *variedade de informações de contexto e serviços* disponíveis no ambiente; (iii) o desenvolvimento e o teste de aplicações exigem uma alta *disponibilidade de recursos*, como por exemplo, sensores (e.g., presença, iluminação, temperatura), atuadores (e.g., chaves, alarmes, *smart-tvs*), incluindo novos dispositivos embarcados, ou ainda de espaços físicos, tais como uma casa para aplicações do tipo *smart home*.

Muitos trabalhos têm por objetivo definir *frameworks* voltados à construção e ao gerenciamento de aplicações ubíquas [Helal et al. 2005, Cardoso 2006, Ranganathan et al. 2005]. Em [Augusto and McCullagh 2007] são apontados desafios na aquisição de conhecimentos do ambiente. Em [Helal et al. 2005] é proposto um *middleware* entre a camada física (compreendida pelos sensores e atuadores) e a camada de aplicação (onde se encontram o ambiente de desenvolvimento e as aplicações ubíquas). Em [Cardoso 2006] são propostos serviços para gerenciar componentes representativos do ambiente no nível de *middleware*. Este artigo aborda o conceito de Ambientes Inteligentes (AmbI), onde uma variedade de dispositivos está disponível, como por exemplo, em casas inteligentes (ou *smart homes*), com televisores, termômetros, *smartphones*, e outros, os quais podem ser descobertos e configurados de acordo com suas especificidades. O trabalho de [Ranganathan et al. 2005], em especial, se preocupa em organizar estes componentes de forma a facilitar suas manipulações; a estruturação proposta permite ampliar o escopo de operações de suporte de um sistema ubíquo. Estes trabalhos, no entanto, não têm como foco a integração das questões acima identificadas: *heterogeneidade dos dispositivos*, *variedade de informações de contexto e serviços*, e *disponibilidade de recursos*.

Este artigo apresenta a proposta de um novo *framework* para o desenvolvimento de aplicações ubíquas em AmbI. O objetivo é fornecer suporte à programação, teste e execução de aplicações, permitindo lidar de forma consistente com sistemas de grande complexidade. Este *framework* destaca-se por tratar dos desafios já identificados na Computação Ubíqua [de Araujo 2003]. A *heterogeneidade de dispositivos* é tratada através da definição de um Modelo de Componentes Distribuídos, no qual o componente básico tem uma estrutura uniforme definida como um Agente de Recurso (AR), proposto inicialmente em [Cardoso 2006] como uma entidade de coleta de informações de contexto. Neste trabalho, o AR, além de manter informações de contexto, age como um componente que encapsula o código do dispositivo a ele associado, incluindo os aspectos de interação com os componentes da aplicação. Para a questão da *variedade de informações de contexto e serviços*, é proposto um Modelo de Contexto que define o armazenamento e a distribuição de tais informações, e provê o suporte para a criação de regras de contexto. Finalmente, em relação à questão sobre a *disponibilidade de recursos*, o *framework* inclui uma aplicação de Interface de Prototipagem e Gerenciamento de Aplicações Pervasivas (IPGAP) [Ferreira et al. 2013], voltada à visualização e ao teste de aplicações ubíquas, mesclando componentes reais e virtuais.

Os conceitos do *framework* foram concretizados sobre uma plataforma denomi-

nada *SmartAndroid*[1], desenvolvida no contexto do projeto. A implementação deste projeto tem viabilizado avaliações no sentido de provar conceitualmente que o *framework* facilita o processo de construção de aplicações ubíquas. Uma das avaliações ocorreu durante o processo de transformação de uma aplicação com funcionamento estritamente local em uma aplicação ubíqua. Em outra avaliação, uma aplicação foi construída para verificar a viabilidade de implantação do Modelo de Contexto, por meio da exploração dos mecanismos de comunicação utilizados no Modelo de Componentes Distribuídos.

O artigo está assim organizado: a Seção 2 apresenta os conceitos básicos que orientam o desenvolvimento deste trabalho; a Seção 3 apresenta a arquitetura geral do *framework* incluindo o Modelo de Componentes Distribuídos e o Modelo de Contexto; na Seção 4 é apresentada a IPGAP; a Seção 5 apresenta uma prova de conceito demonstrando a viabilidade da construção de aplicações ubíquas utilizando o *framework* e uma aplicação que explora as principais características do AmbI; a Seção 6 apresenta uma comparação com trabalhos relacionados; e a Seção 7 apresenta as conclusões e trabalhos futuros.

2. Conceitos Básicos

Os *frameworks* para aplicações ubíquas utilizam em geral conceitos de Inteligência Ambiental [Augusto and McCullagh 2007], de Computação Sensível ao Contexto [Dey et al. 2001] e de Prototipagem de Aplicações Ubíquas [Weis et al. 2007]. A Inteligência Ambiental define o AmbI, que é um espaço onde estas aplicações funcionam. O comportamento de sistemas ubíquos é definido a partir de técnicas aplicadas na Computação Sensível ao Contexto. A prototipagem, por sua vez, é utilizada para manipular e testar o funcionamento do conjunto de aplicações no ambiente.

Os sistemas com enfoque na Computação Ubíqua aplicada no AmbI são baseados geralmente em uma arquitetura em camadas. Na camada inferior encontra-se o espaço físico com seus ocupantes, e em uma camada acima estão os sensores coletando informações de contexto e os atuadores provendo serviços para atender às necessidades destes ocupantes. Uma camada intermediária (*middleware*) é definida entre as tomadas de decisão e as interações com o ambiente, incluindo também conceitos e mecanismos para a construção de software dessa classe de sistemas. As decisões podem ser tomadas por um ocupante ou através de mecanismos de inteligência artificial. Há propostas na literatura que caracterizam este tipo de ambiente, dentre eles o termo "*smart home*" (casa inteligente) se mostra como um dos mais conhecidos [Helal et al. 2005, Augusto and McCullagh 2007, Ranganathan et al. 2005].

O *framework* proposto neste artigo envolve, além de um *middleware* com suporte a serviços, os conceitos de Computação Sensível ao Contexto e de comunicação e interação, apresentados nas próximas subseções.

2.1. Computação Sensível ao Contexto

O contexto exerce um papel de fundamental importância na Computação Ubíqua. Sintetizando propostas anteriores, Dey e Abowd [Dey et al. 2001] definiram que contexto é qualquer informação relevante usada para caracterizar a situação de entidades, especificamente: pessoas, lugares e coisas. No AmbI, "Lugares" são os cômodos, os andares de

[1] www.tempo.uff.br/smartandroid

uma edificação, ou espaços em geral que possibilitam a localização de outras entidades; "Pessoas" são indivíduos que povoam o ambiente e interagem com o mesmo; e "Coisas" são representações virtuais de objetos físicos ou componentes de software.

A sensibilidade ao contexto está em se determinar o que o usuário está tentando realizar a partir da aquisição de contexto. Por exemplo, se uma pessoa sai de casa e deixa a torneira aberta, provavelmente a tenha esquecido nesse estado. Neste caso, um sistema sensível ao contexto faz o que qualquer pessoa faria se detectasse esta situação (i.e., fecharia as torneiras). A aquisição do contexto de forma automatizada contribui para a construção de aplicações para AmbI que, de outra maneira, seriam inviáveis, por exigir a entrada de dados ou comandos diretamente por parte dos usuários. O *framework* proposto neste artigo oferece um suporte para a declaração e consulta destas informações (ver a Seção 3.1.1), além da definição de Interpretadores de Contextos (ver a Seção 3.2.2), que facilitam a construção de regras de contexto para atuar no AmbI.

2.2. Comunicação e Interação

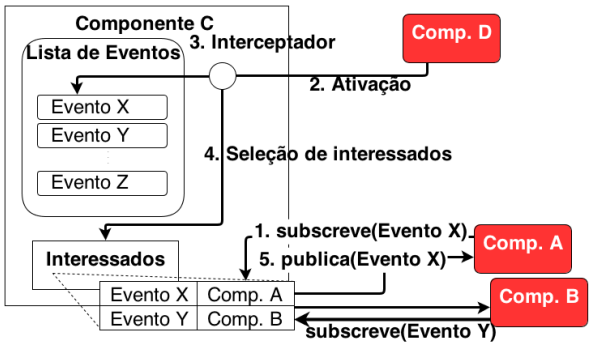


Figura 1. Esquema de *publish-subscribe*

De modo a atender requisitos típicos de ambientes distribuídos, foram incluídos dois mecanismos no *framework* proposto: a invocação remota de procedimentos (*Remote Procedure Call* – RPC) e a comunicação por eventos (ou interesses) seguindo o paradigma *publish-subscribe* [Eugster et al. 2003]. O mecanismo de RPC é utilizado para estabelecer a comunicação síncrona direta entre os componentes do AmbI. Uma entidade invocadora utiliza um *proxy* da entidade invocada, nele estando contida a sua interface de chamadas de procedimentos públicos. A chamada e o retorno do procedimento são enviados através de mensagens serializadas (no *SmartAndroid* através da notação JSON). Este mecanismo pode ser usado para enviar comandos aos dispositivos, por exemplo, fechar uma torneira, ou alterar remotamente o *setup* de um ar-condicionado.

O paradigma *publish-subscribe* é utilizado no *framework* proposto para a aquisição do status de componentes do ambiente. A Figura 1 ilustra sua inserção no esquema de componentes. Inicialmente, os Componentes A e B se subscrevem aos Eventos X e Y, respectivamente, do Componente C (Passo 1) e esta subscrição é armazenada no campo de Interessados. Posteriormente, o Componente D envia um comando que ativa o Evento X (Passo 2). Em seguida, ocorre a interceptação (Passo 3) e o Componente A, interessado neste evento (Passo 4), é notificado (Passo 5).

Como suporte, a comunicação no *framework* necessita de uma infraestrutura básica de rede. Por conveniência, foi adotado Wi-Fi, que garante o mínimo de segurança

contra acesso de agentes externos à rede do ambiente e agrega benefícios de mecanismos de criptografia, como a WPA.

3. Descrição do *Framework*

O *framework* provê facilidades e padrões de programação tipicamente requeridos em aplicações sensíveis ao contexto focadas em AmbI. Dentre as facilidades está a capacidade de abstrair detalhes das partes da aplicação que envolve a comunicação, a aquisição de contexto e a localização de recursos. Como abstração aos recursos são utilizados componentes chamados Agentes de Recursos (AR). O AR é a unidade básica de modularização do *framework* , sendo utilizado na modelagem, implementação e gerenciamento das aplicações. Por exemplo: sensores de temperatura da casa, de vazamento de gás na cozinha, um medidor de pressão arterial, um atuador para fechar as janelas, etc.

Para consolidar a proposta do *framework*, seus conceitos e mecanismos foram implementados numa plataforma que denominamos *SmartAndroid*. Essa opção se beneficia da acessibilidade da tecnologia *Android* [Saha 2008] facilitando a integração de dispositivos disponíveis em seu ecossistema e o desenvolvimento de sistemas embarcados. Isto permitiu uma construção rápida da plataforma facilitando a experimentação de aplicações ubíquas sofisticadas. Deve ser ressaltado que o *framework* usa conceitos de implementação imediata em outras plataformas distribuídas atuais. Adicionalmente, a técnica de *Wrappers* pode ser usada para integrar componentes de outras tecnologias em aplicações *SmartAndroid*. A arquitetura do *framework* é definida a partir de um Modelo de Componentes Distribuídos (Seção 3.1) e de um Modelo de Contexto (Seção 3.2).

3.1. Modelo de Componentes Distribuídos

Como visto anteriormente na Seção 2, sistemas para AmbI são geralmente estruturados nas camadas física, de *middleware* e de aplicação (Figura 2). O Modelo de Componentes Distribuídos segue esta estrutura para mapear as aplicações ubíquas no AmbI. Neste Modelo são definidos: a estrutura do AR e o Suporte ao Gerenciamento de Agentes de Recursos (SGAR).

O conceito de AR foi definido para padronizar os componentes de interação do ambiente, e dessa forma resolver a questão da *heterogeneidade dos dispositivos*. A sua estrutura geral, como ilustrada na Figura 2 (a), é composta por nome único (“tvFamília”), hierarquia de tipos (“\Visual\TV”), localização corrente (“Sala”), classe (classe TV) com variáveis e métodos, e uma lista de interessados em informações de contexto. A hierarquia de tipos, que é composta por uma sequência de classes, foi inspirada na ontologia descrita em [Ranganathan et al. 2005] para classificar entidades (como os ARs). Esta hierarquia foi organizada para possibilitar a consulta e instanciação de ARs a partir de propriedades associadas a uma classe específica. Na Seção 3.1.1 é apresentado o SGAR que se beneficia desta estrutura do AR.

A lista de interessados de um AR é uma estrutura de suporte ao mecanismo de comunicação por eventos apresentado na Seção 2.2. O Modelo de Contexto (ver Seção 3.2) se beneficia deste mecanismo, dado que as mudanças no contexto de uma instância de AR são publicadas como eventos para outras instâncias.

Na Figura 2 (b) são apresentados componentes típicos de uma *smart home*. Na camada de recursos estão componentes físicos ou conceituais. Os recursos conceituais

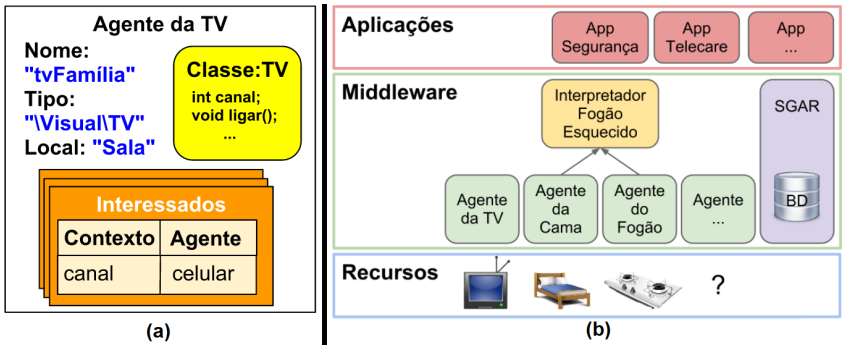


Figura 2. Arquitetura do *Framework*: (a) Estrutura do AR; (b) Camadas

Tabela 1. Operações do SGAR

Serviço	Operação	Argumentos	Descrição
SRR	*register*	Dados do AR	Insere dados do AR no Repositório
SRR	*unregister*	Dados do AR	Remove AR do Repositório
SDR	*search*	Consulta	Busca ARs por tipo, local e/ou nome
SLR	*getPlaces*	Nenhum	Retorna a lista de espaços do Mapa
SLR	*searchByProximity*	Posição	Retorna ARs mais próximos
AR	*subscribe*	AR e Evento	Adiciona interesse do AR no Evento
AR	*publish*	Evento e Valor	Notifica ARs interessados no Evento

são aqueles simulados através de outro tipo de dispositivos (por conveniência usamos *smartphones* e *tablets*) ou criados através da IPGAP (ver Seção 4). A figura apresenta como exemplos de recursos um televisor, uma cama e um fogão. No *middleware* são apresentados os ARs representativos destes recursos, o SGAR e uma entidade de interpretação de contexto, a qual é definida no nível da aplicação, mas com o suporte da camada intermediária. O interpretador descrito na figura verifica se o ocupante da casa esqueceu o fogão ligado e, para isso, avalia se ele está deitado na cama e se o fogão está ligado durante determinado tempo; uma vez passado o tempo limite, ações são disparadas para acordá-lo, ou para desligar o fogão, atendendo às preferências do usuário. Os ARs referentes à cama e ao fogão coletam o status destes recursos físicos e os divulgam para ARs interessados. Isto permite que estas informações sejam utilizadas pelos componentes da camada da aplicação. Ainda no *middleware*, o SGAR (ver Seção 3.1.1) é definido como suporte para aplicações criarem, consultarem e instanciarem ARs.

3.1.1. Suporte ao Gerenciamento de Recursos

O SGAR é o responsável por gerenciar o conjunto de ARs no AmbI. O controle de registro e descoberta de ARs segue a ideia geral sobre serviços que manipulam recursos apresentada em [Cardoso 2006]. Neste trabalho, o SGAR é composto por três componentes, ou serviços básicos: o Serviço de Registro de Recurso (SRR), o Serviço de Descoberta de Recursos (SDR) e o Serviço de Localização de Recursos (SLR). A Tabela 1 destaca as principais operações de cada serviço de suporte. O SRR é utilizado para registrar (*register*) ou remover (*unregister*) ARs no ambiente, o SDR localiza recursos para permitir o acesso

remoto (*search*) e o SLR os localiza através de posições físicas e indica os recursos mais próximos (*searchByProximity*). Estas operações manipulam dados representativos dos respectivos ARs no Repositório de Recursos. As operações "*subscribe*" e "*publish*" são comuns a todos os ARs e funcionam com base no mecanismo de comunicação por eventos descrito na Seção 2.2. Os componentes (A, B, C, D) na Figura 1 correspondem aos ARs e os Eventos correspondem à suas respectivas mudanças de estados.

O Repositório de Recursos, como apresentado na Figura 3, possui o Diretório de Recursos e o Mapa, o qual contém representado o conjunto de espaços físicos do AmbI. O Diretório contém os nomes das instâncias de AR existentes no ambiente, armazenados conforme seus respectivos tipos. No caso da figura há o tipo TV com a "tvFamília" e o tipo *tablet* com o "iPad" e o "Galaxy". Os dados consistem do nome, tipo, localização e a referência de acesso. A referência de acesso permite a interação de instâncias de ARs através do RPC (ver Seção 2.2). Cada espaço do Mapa possui uma área e um nome. No caso de uma *smart home* os cômodos da casa representam estes espaços. Cada entrada do Mapa referencia o conjunto de ARs contidos no respectivo espaço. Os ARs representando dispositivos com mobilidade têm suas referências atualizadas dinamicamente.

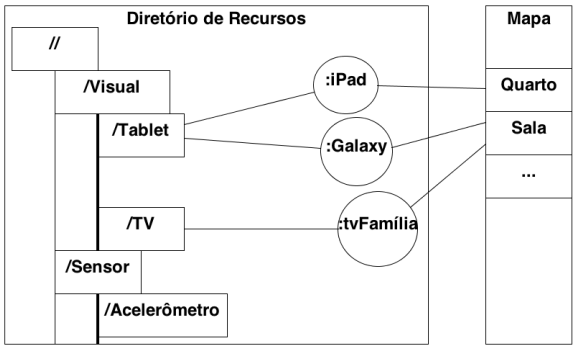


Figura 3. Repositório de Recursos

3.2. Modelo de Contexto

O Modelo de Contexto foi elaborado com o objetivo de tratar a questão da *variedade de informações de contexto e serviços* presentes no AmbI. Neste modelo, o *framework* utiliza uma abordagem flexível, onde o mecanismo de comunicação por eventos (visto na Seção 2.2) possibilita que ARs divulguem mudanças de contexto. A informação pode estar em qualquer lugar do AmbI, contudo, é provido um nível de abstração tal que a subscrição ou consulta a qualquer informação de contexto é feita de maneira padronizada. O processo de aquisição do contexto envolve a descoberta do AR de interesse através dos serviços da SGAR, com a posterior subscrição do mesmo. A flexibilidade da abordagem adotada possibilita a utilização de uma unidade centralizadora para gerenciar as informações de contexto, caso seja conveniente, segundo uma arquitetura de memória compartilhada do tipo *blackboard* [Winograd 2001].

3.2.1. Variáveis de Contexto e Operações

As informações de contexto respectivas de cada AR são expostas através de Variáveis de Contexto (VC). Por exemplo, um agente para uma *smart-tv* pode prover VCs para: a

programação que está sendo exibida, a programação agendada, se a própria televisão está ligada, se está gravando alguma programação, e outras. Em outros termos, tudo o que diz respeito ao estado da televisão e que pode efetivamente ser coletado.

Uma Operação (OP) tem o papel de expor uma funcionalidade (ou serviço) do AR, possibilitando às aplicações interagirem ativamente no ambiente. Por exemplo, uma televisão integrada ao sistema pode oferecer OPs para desligá-la, mudar de canal, gravar alguma programação, mostrar uma mensagem na tela, perguntar algo ao usuário, gerar um alerta, pausar a programação, etc.

Tanto VCs como OPs definem interfaces, ou portas do AR. Diferentes aplicações, instaladas no mesmo ambiente podem usar estas interfaces para interagir com o próprio ambiente. A Figura 4 representa a subscrição às VCs "Em uso" do AR da cama e "Ligado" do AR do fogão. Na mesma figura pode-se observar a atuação no ambiente ao se utilizar as OPs "Mostrar mensagem" da televisão e "Disparar" do despertador. O Interpretador e o Atuador também representados na figura serão descritos na Seção 3.2.2.

As questões de segurança, inerentes ao problema, são resolvidas com duas técnicas: a própria segurança da rede e com a criação de domínios dentro de um AmbI. A segurança da rede (criptografia de pacotes em redes LAN, como WPA2) evita que aplicações estrangeiras possam acessar os recursos de um ambiente. Domínios podem ser usados para isolar aplicações específicas dentro do AmbI, como, por exemplo, uma aplicação de monitoramento de pacientes [Carvalho et al. 2010].

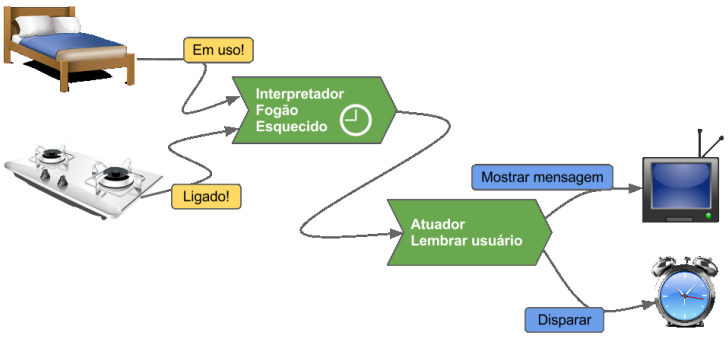


Figura 4. Interpretador de Regra

3.2.2. Interpretação de Contexto

A interpretação de contexto tem a função de agregar informações de contexto provenientes de diferentes fontes, considerando também a passagem de tempo, e avaliá-las segundo alguma lógica específica. O suporte à interpretação de contexto pelo *framework* possibilita uma separação de interesses, onde os desenvolvedores abstraem a implementação de regras de contexto e focam na lógica da aplicação.

Em nossa proposta, como opção básica, a interpretação do contexto é desempenhada por entidades chamadas Interpretadores de Contexto (IC), que avaliam essas informações e notificam agentes atuadores interessados (ver outras opções na Seção 5.2). Atuadores são quaisquer ARs que se subscrevem em ICs para desempenhar ações; sejam estas ações no nível de software (e.g., guardar no histórico, enviar para um servidor

remoto) ou no nível do ambiente, ao se chamar OPs de outros ARs (e.g., mostrar uma mensagem na televisão, disparar o despertador, mudar a temperatura do ar-condicionado).

Considere a seguinte regra simples: se uma pessoa (que mora sozinha) ligou o fogão, se deitou na cama, e passaram-se 15 minutos, então dispare o alarme do despertador e mostre na televisão a mensagem "O fogão foi esquecido ligado!". Este exemplo poderia ser implementando no sistema como representado na Figura 4. O IC recebe notificações da cama e do fogão com valores atualizados das VCs "ligado" e "em uso", respectivamente, e resolve internamente a temporização monitorada destas VCs. Uma vez que o IC tenha avaliado a regra como verdadeira durante o tempo de 15 minutos (previamente declarado), ele notifica o atuador "Lembrar usuário". O atuador, por sua vez, invoca as OPs "Mostrar mensagem" da TV e "Disparar" do despertador.

O IC pode ser encapsulado em um AR, assim estendendo suas funcionalidades. A arquitetura do IC é composta de um módulo que recebe as notificações e atualiza os valores em *cache*; uma estrutura de dados em árvore que não só armazena as referências para as VCs e os valores atualizados, mas também a lógica da regra; um módulo que avalia a árvore a cada atualização de valores; e um módulo que gerencia os temporizadores, controlado pelo módulo de avaliação.

A interpretação de contexto visa não só a construção de regras de contexto por parte das aplicações, mas também a definição das preferências dos usuários finais no sistema. Uma GUI está sendo desenvolvida para possibilitar aos usuários sem experiência técnica criarem, editarem, desabilitarem etc., regras para o seu dia-a-dia. A GUI permitirá que, com poucos toques, um usuário possa selecionar ARs em um mapa da casa, escolher as VCs, comparar com valores ou outras VCs (operadores de comparação: $=, \neq, <, >, \leq$ e $\geq$), montar a expressão lógica (operadores lógicos: "*E*", "*OU*", "*NÃO*"), e definir um conjunto de ações a serem desempenhadas no sistema. A definição do conjunto de ações pode ser feita tanto ao se escolher ARs atuadores para entrar em ação (e.g., Atuador Lembrar Usuário), como selecionando ARs no mapa e suas OPs em seguida (e.g., ar-condicionado - mudar temperatura - 20º). Através do mecanismo de subscrição os ICs notificarão os ARs atuadores, que executarão tarefas no AmbI através de chamadas RPC.

4. Prototipagem de Aplicações Pervasivas

A prototipagem de aplicações pervasivas é fundamental para a depuração e teste em um AmbI. A questão de *disponibilidade de recursos* é resolvida através da IPGAP [Ferreira et al. 2013], um aparato ferramental que proporciona ao desenvolvedor de aplicações ter acesso a estes benefícios. Além disso, a IPGAP disponibiliza uma Interface Gráfica de Usuário (*Graphic User Interface* – GUI) que proporciona ao usuário final controlar remotamente recursos do ambiente.

Na Figura 5 é ilustrado um *tablet* executando o aplicativo da IPGAP e *smartphones* emulando recursos como TV, lâmpada e termômetro. A IPGAP faz o mapeamento do ambiente e permite visualizar e instanciar cada um destes elementos e representá-los na sua GUI. Outros recursos, além dos representados pelos *smartphones*, podem ser simulados virtualmente por componentes de software. Assim, o desenvolvedor de aplicações não fica limitado pela disponibilidade de recursos físicos podendo simular os elementos indisponíveis.

O *framework* proposto permite criar aplicações de forma liberal, nas dimensões e

Figura 5. Interface de Prototipagem

quantidades desejadas para um ambiente real. Através da IPGAP, as restrições e viabilidade do funcionamento das aplicações desenvolvidas podem ser verificadas em ambiente próximo ao real. Isto permite avaliar o desempenho das implementações antes da sua implantação efetiva.

5. Avaliação

A avaliação do *framework* foi realizada em duas fases. Primeiro foi avaliado o fator de transparência de comunicação na construção de aplicações ubíquas. Esta avaliação ocorre através do processo de transformação de uma aplicação estritamente local *Android* em uma aplicação ubíqua através do *SmartAndroid*. Depois é avaliada a transparência no desenvolvimento de aplicações sensíveis ao contexto e manipulação de informações promovida pelo Modelo de Contexto.

5.1. Prova de Conceito do *Framework*

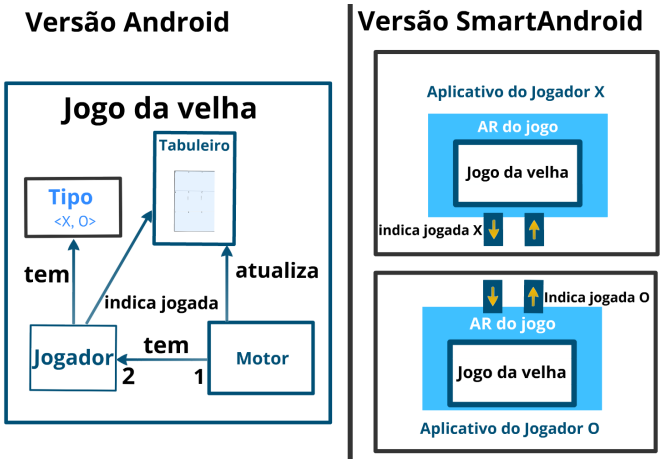


Figura 6. Comparação entre os modelos das aplicações *Android* e *SmartAndroid*

Para a primeira avaliação utilizou-se a aplicação do jogo da velha como base. Em sua versão *Android* (Figura 6), os jogadores interagem através de uma mesma tela de dispositivo (*tablet* ou *smartphone*). Na versão desenvolvida com o *SmartAndroid* os jogadores interagem sobre um mesmo tabuleiro em dispositivos diferentes. A Figura 6 mostra detalhes relevantes do modelo de aplicação da versão *Android* e apresenta modificações

sobre esta estrutura na versão *SmartAndroid*. Na versão original destacam-se os seguintes componentes: o Motor do Jogo (Motor), a estrutura do Tabuleiro e a identidade dos Jogadores que pode ser do tipo "X" ou "O" (Tipo <X,O>). O Motor atualiza o estado do Tabuleiro que é mostrado na GUI da aplicação. O Tipo do Jogador é verificado para indicar qual símbolo da jogada é marcado no Tabuleiro ("X" ou "O").

Na versão *SmartAndroid* as principais estruturas do jogo da velha são encapsuladas no AR do jogo. O AR do Jogo é responsável por interceptar as indicações de jogadas dos respectivos jogadores representados no dispositivo ("X" ou "O") e encaminhar para o outro jogador. O outro jogador recebe estas jogadas através de seu AR que as encaminha ao Motor de jogo que efetiva a atualização do tabuleiro com a jogada do adversário. A etapa de inicialização da nova aplicação foi desenvolvida a partir das operações definidas na Tabela 1. O Código 1 apresenta a descoberta de ARs de outros jogadores na Linha 1, o registro do AR do jogador local na Linha 2, e a subscrição de interesses em jogadas entre ARs de jogo do AmbI nas Linhas 5 e 6. Após isto, nenhuma alteração no nível de comunicação ou contexto torna-se necessária. Logo, podemos comprovar que o uso de AR tornou transparente a programação da aplicação ubíqua.

Código 1. Inicialização do jogo da velha no *SmartAndroid*

```
1 List ARs = SDR.search(TYPE, ARJogo) //Consulta por tipo
2 SRR.register(arJogo) //Registro do Agente do Jogo
3 if (ARs.size>0): //Caso haja outros jogadores...
4     for each iAR in ARs: //Subscrição entre ARs
5         iAR.subscribe("indica jogada", arJogo)
6         arJogo.subscribe("indica jogada", iAR)
```

5.2. Aplicação de Controle de Iluminação Residencial

Dentre os protótipos construídos como prova de conceito, foi criada a Aplicação de Controle de Iluminação Residencial (*Smart Light Controller* – SmartLiC) com o intuito de promover economia no consumo de energia elétrica reduzindo gastos com iluminação. Através de sensores de presença, identifica-se a presença/ausência de pessoas no cômodo, conforme a pessoa sai de um cômodo e passam-se T_{max} unidades de tempo sem que ela volte, então a luz daquele cômodo é desligada.

A Figura 4 representa um caso simples de utilização de interpretadores, onde este contém uma regra apenas e as atuações ocorrem em separado, desempenhadas pelos agentes atuadores. A fim de se prover maior flexibilidade para os desenvolvedores, o *framework* possibilita também que a interpretação de contexto seja desenvolvida no nível da aplicação, contudo, aproveitando o mecanismo de comunicação por eventos e compilações de regras. Note que esta abordagem possibilita a utilização de um motor de regras (e.g., Jess, Drools), se for conveniente.

O Código 2 ilustra uma possível implementação da interpretação de contexto pela aplicação SmartLiC. Na etapa de inicialização, a aplicação utiliza o SLR para localizar os cômodos (Linha 2), e obter a lâmpada e o sensor de presença ("lampAR" e "presAR") de cada um, utilizando a busca do SDR (Linhas 4 e 5). Para evitar a repetição do código da regra, suas referências e a respectiva ação a ser desempenhada (desligar a lâmpada referenciada – "lampAR.turnOff") são adicionadas na lista de configurações ("configList") (Linha 6). A regra é criada em "rule" como:

"*lâmpada ligada E o cômodo desocupado POR T_{max} unidades de tempo*", e desempenha uma relação lógica entre as diferentes VCs e a temporização da regra (Linha 7). Em seguida, subscreve-se às VCs dos ARs de interesse, começando o processamento da regra (Linha 8).

Na etapa de avaliação, também descrita no Código 2, é definido um bloco de comando para o tratamento das notificações recebidas dos ARs de interesse (lâmpadas e sensores de presença subscritos). Uma vez recebido um evento com a atualização de uma VC de um dos ARs ("ar_ref") (Linha 11), é selecionada a configuração de "configList" respectiva (Linha 12). Segue-se a avaliação da regra ("*evaluate*"), utilizando a configuração obtida, e considerando os temporizadores (Linha 13). A etapa de avaliação é bloqueante no caso da existência de temporizadores associados, e a invocação das ações (OPs) ocorre na mesma *thread* que a avaliação. A temporização ocorre enquanto a expressão da regra é válida, se eventos chegarem antes do fim da temporização e isto fizer com que a expressão da regra seja invalidada, então o temporizador é finalizado e as ações não são disparadas.

Código 2. Interpretação de Contexto no SmartLiC

```
1  //Inicialização
2  placeList = SLR.getPlaces() //Obtém cômodos
3  for each place from placeList: //Seleciona um cômodo da lista
4      lampAR = SDR.search(Lamp, place) //Obtém lâmpada do cômodo
5      presAR = SDR.search(Presence, place) //Obtém sensor de presença do cômodo
6      configList.add(lampAR, presAR, lampAR.turnOff) //Guarda configuração
7  rule = Lamp.on ∧ ¬(Presence.occupied) ∧ T > T_max //Função de regra
8  subscribeTo(configList) //Subscreve às lâmpadas e sensores de presença
9
10 //Avaliação da regra de contexto
11 on event received (ar_ref, vc, value): //Quando chega uma notificação
12     config = configList(ar_ref, vc) //Seleciona a configuração do AR
13     if (evaluate(rule, config)) then: //Avalia regra com configuração
14         invoke(config.getActions()) //Invoca ações
```

O SmartLiC é um projeto simples, mas que possibilita a avaliação de boa parte do ferramental proposto pelo *framework* e implementado no *SmartAndroid*. O foco tomado na implementação foi economia de energia, porém, outras funcionalidades podem ser agregadas ao serviço. Por exemplo, uma aplicação de segurança do tipo "engana ladrão", onde lâmpadas da casa são acendidas e apagadas seguindo uma ordem pré-estabelecida ou informações de um histórico, objetivando simular a presença de pessoas na casa quando os proprietários estão ausentes.

6. Trabalhos Relacionados

Em [Helal et al. 2005] é proposta uma arquitetura de camadas semelhante ao já apresentado neste trabalho. Além destas, há a camada de conhecimento que tem função similar ao SGAR, e a camada de contexto que tem função similar ao Modelo de Contexto, mas atua em uma granularidade maior. Nosso *framework* descreve mecanismos para construção e instalação de novas aplicações, algo que em [Helal et al. 2005] não fica evidente sobre como pode ser feito. Um conjunto de operações de alto nível para ambientes inteligentes (espaços ativos) é proposto em [Ranganathan et al. 2005]. As operações básicas são semelhantes às funções de um sistema operacional, só que ao invés de manipular recursos,

entidades do ambiente são manipuladas. Em nosso *framework* há as funções apresentadas na Tabela 1. Outras funções como parar, iniciar, suspender, reiniciar são definidas por cada AR e aplicação ubíqua do sistema, permitindo uma distribuição do controle de serviços do ambiente.

Boa parte dos *frameworks* propostos evita levantar pontos sobre a sobrecarga de comunicação ocorrida em sistemas ubíquos. O artigo [Villanueva et al. 2009] é uma proposta com abordagem distribuída para a parte de suporte (SGAR), onde cada componente possui o suporte replicado e que utiliza comunicação *multicast* para realizar os serviços. O custo de espaço desta abordagem é muito alto devido à ocorrência de replicação de dados sobre o ambiente, e acaba ocasionando uma sobrecarga no consumo de energia devido ao excesso de comunicação. O JaCa-Android [Santi et al. 2011] utiliza o esquema de captura de mudança de estado em recursos através de escuta de eventos, semelhante à comunicação por eventos apresentada na Seção 2.2, mas limitado a componentes em um mesmo dispositivo. Além de não ser voltada a sistemas distribuídos, esta proposta tem sua abordagem limitada pela a plataforma *Android* por utilizar mecanismos específicos da tecnologia para a captura de eventos.

O *framework* busca como diferencial que novas aplicações ubíquas possam ser concebidas de forma dinâmica e adaptativa. A proposta disponibiliza ao desenvolvedor de aplicações um ferramental para montar AmbIs personalizados. O desenvolvedor adquire como benefício à separação de interesses, que reduz a carga de codificação de requisitos essenciais de um sistema (e.g. segurança, manutenção), e possui ainda um suporte para construção de regras sobre o contexto do ambiente, permitindo prover serviços em alto nível aos usuários do sistema inteligente.

7. Conclusão e Trabalhos Futuros

Neste artigo, foi apresentado o *Framework* de Desenvolvimento de Aplicações Ubíquas em Ambientes Inteligentes, com abstrações que visam facilitar o desenvolvimento e a implantação de aplicações que interajam de forma ubíqua com o ambiente e seus ocupantes. Através do *framework*, o desenvolvedor cria serviços que atendem a requisitos preestabelecidos de acordo com as características de um AmbI. Após a concepção de serviços para o AmbI, a implantação ocorre a partir da instalação do conjunto de aplicações sobre uma infraestrutura de rede Wi-Fi segura. Os serviços, depois de implantados, podem ser manipulados pelos ocupantes através da IPGAP e das interfaces de operações de cada aplicação instalada. Na avaliação, que se deu por meio das implementações das aplicações do jogo da velha com múltiplos participantes e do SmartLiC, foi constatada a qualidade do *framework* em lidar com problemas de um AmbI.

A proposta possui potencial para ampliar suas funcionalidades e atender a outros desafios importantes na área de Computação Ubíqua e Computação Sensível ao Contexto. A segurança atual está limitada às configurações da rede Wi-Fi, futuramente pretende-se agregar ao suporte restrições de acesso a determinados ocupantes e entre diferentes domínios de aplicações. A economia de energia em dispositivos pode ser aprimorada reduzindo a sobrecarga de comunicação em quantidade e qualidade. Na parte de Computação Sensível ao Contexto, pretende-se identificar as preferências do usuário de forma menos intrusiva através de técnicas de mineração de dados aliadas à aprendizagem de máquina. Para permitir um controle de serviços maior no nível do usuário, está

sendo definida uma interface amigável para a definição de regras de contexto. Como resultado desses incrementos, aplicações críticas como um sistema de assistência domiciliar a saúde [Carvalho et al. 2010] serão beneficiadas.

Referências

Augusto, J. and McCullagh, P. (2007). Ambient intelligence: Concepts and applications. *Computer Science and Information Systems/ComSIS*, 4(1):1–26.

Cardoso, L. (2006). Integração de serviços de monitoração e descoberta de recursos a um suporte para arquiteturas adaptáveis de software. Dissertação de mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, Rio de Janeiro.

Carvalho, S., Erthal, M., Mareli, D., Sztajnberg, A., Copetti, A., and Loques, O. (2010). Monitoramento remoto de pacientes em ambiente domiciliar. *XXVIII SBRC-Salao de Ferramentas, Gramado, RS, Brasil*.

de Araujo, R. (2003). Computação ubíqua: Princípios, tecnologias e desafios. In *XXI Simpósio Brasileiro de Redes de Computadores*, volume 8, pages 11–13.

Dey, A., Abowd, G., and Salber, D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, 16(2):97–166.

Eugster, P., Felber, P., Guerraoui, R., and Kermarrec, A. (2003). The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131.

Ferreira, D. B., Erthal, M., Mareli, D., and Loques, O. (2013). Uma interface de prototipagem para aplicações pervasivas. In *SBRC 2013 ()*.

Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., and Jansen, E. (2005). The gator tech smart house: A programmable pervasive space. *Computer*, 38(3):50–60.

Ranganathan, A., Chetan, S., Al-Muhtadi, J., Campbell, R., and Mickunas, M. (2005). Olympus: A high-level programming model for pervasive computing environments. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 7–16. IEEE.

Saha, A. (2008). A Developer's First Look At Android. *Linux For You*, (January):48–50.

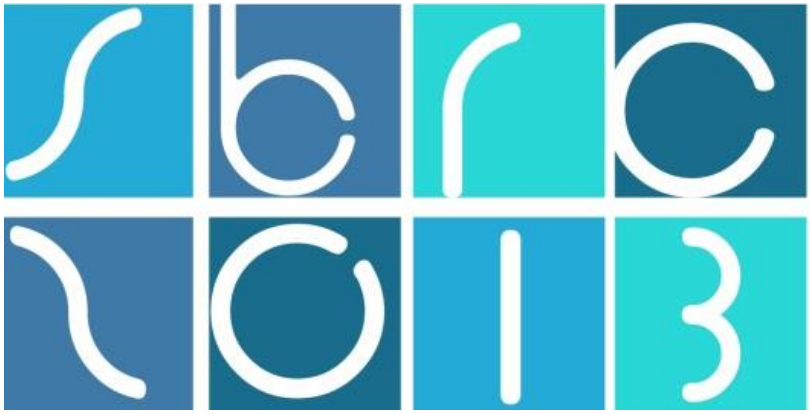
Santi, A., Guidi, M., and Ricci, A. (2011). Jaca-android: an agent-based platform for building smart mobile applications. *Languages, Methodologies, and Development Tools for Multi-Agent Systems*, pages 95–114.

Villanueva, F., Villa, D., Santofimia, M., Moya, F., and Lopez, J. (2009). A framework for advanced home service design and management. *Consumer Electronics, IEEE Transactions on*, 55(3):1246–1253.

Weis, T., Knoll, M., Ulbrich, A., Muhl, G., and Brandle, A. (2007). Rapid prototyping for pervasive applications. *Pervasive Computing, IEEE*, 6(2):76–84.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.

Winograd, T. (2001). Architectures for context. *Human-Computer Interaction*, 16(2):401–419.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 16

Rádios Cognitivos e Redes Tolerantes a Atrasos

Uma Solução para Gerenciamento de Dispositivos de Rádio Cognitivo Baseada na MIB IEEE 802.22

Lucas Bondan[1], Maicon Kist[1], Rafael Kunst[1], Cristiano B. Both[1], Juergen Rochol[1], Lisandro Z. Granville[1]

[1]Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brasil

{lbondan, maicon.kist, rkunst, cbboth, juergen, granville}@inf.ufrgs.br

Abstract. *Cognitive Radio technology allows wireless devices to transmit information while the channels are not in use at time. However, it is necessary a system that helps the dynamic configuration process of the cognitive devices for proper operation of cognitive radio networks. In this paper, a management system for cognitive radio devices is proposed based on the IEEE 802.22 MIB. The system's main objective is to manage and to monitor the spectrum sensing process. The results obtained in simulations show that the system enables to maximize the throughput of the devices by setting the sensing window according to the confidence on the state of a channel, i.e., free or busy.*

Resumo. *A tecnologia de Rádio Cognitivo permite que dispositivos sem fio transmitam informações enquanto os canais não estiverem em uso em um determinado instante de tempo. Entretanto, para o correto funcionamento das redes de rádios cognitivos, é necessário um sistema que auxilie no processo de configuração dinâmica dos dispositivos cognitivos. Neste trabalho é proposto um sistema de gerenciamento especializado para dispositivos de rádio cognitivo baseado na MIB IEEE 802.22. O sistema tem como objetivo principal o gerenciamento e a monitoração do processo de sensoriamento espectral. Os resultados, obtidos através de simulações, mostram que o sistema possibilita maximizar a vazão na transmissão de informações dos dispositivos, ajustando a janela de sensoriamento de acordo com os níveis de confiança para percepção de que um determinado canal está ocupado.*

1. Introdução

Técnicas de acesso dinâmico ao espectro de frequências (*Dynamic Spectrum Access* - DSA) estão sendo utilizadas para melhorar a eficiência de modernos dispositivos de redes sem fio [Zhao e Sadler 2007]. Os dispositivos que implementam essas técnicas, possuem a capacidade de acessar dinamicamente uma determinada frequência do espectro que não está em uso em um determinado instante de tempo. Atualmente, a principal tecnologia que provê a implementação de técnicas de acesso dinâmico ao espectro é chamada de Rádio Cognitivo [Wang *et al.* 2011]. Essa tecnologia permite que os dispositivos ajustem o seu funcionamento para maximizar suas taxas de transmissão e minimizar a interferência em relação a outros dispositivos sem fio [Khalid e Anpalagan 2010]. Entretanto, para o correto funcionamento das redes de rádios cognitivos é necessário um sistema de gerenciamento que auxilie no processo de configuração dinâmica dos dispositivos cognitivos.

As características dinâmicas dos dispositivos de rádio cognitivo exigem a configuração de vários parâmetros que devem ser gerenciados e monitorados. Neste sentido,

o primeiro grupo de trabalho IEEE a padronizar a utilização da tecnologia de rádio cognitivo foi o 802.22, através da definição de uma *Management Information Base* (MIB) para gerenciar e monitorar dispositivos de uma rede de rádios cognitivos [IEEE 802.22 2011]. Um exemplo da necessidade de um sistema de gerenciamento em redes de rádios cognitivos está no processo conhecido como sensoriamento espectral. Neste processo, o dispositivo cognitivo deve analisar, durante um intervalo de tempo, se um determinado canal está em uso ou não [Yucek e Arslan 2009]. A confiabilidade do resultado dessa análise está diretamente relacionada à duração do sensoriamento, ou seja, quanto mais tempo de sensoriamento, maior o nível de confiança. Considerando apenas esses dois parâmetros de configuração, tempo e confiabilidade do sensoriamento, já percebe-se a necessidade de um sistema especializado de gerenciamento para a configuração do dispositivo e posterior monitoramento da suas funcionalidades, a fim de garantir o correto funcionamento dos dispositivos.

A comunidade científica em gerenciamento de redes apresenta poucos trabalhos sobre redes de rádios cognitivos. Neste artigo, as poucas propostas encontradas na literatura são classificadas de acordo com dois escopos: Global e Local. No primeiro escopo, encontram-se os trabalhos publicados por Wang *et al.* [Wang *et al.* 2008] e Stavroulaki *et al.* [Stavroulaki *et al.* 2012], nos quais o enfoque é no gerenciamento global de redes de rádios cognitivos. No segundo escopo, os trabalhos publicados por Wang *et al.* [Wang *et al.* 2010] e por Potier e Quian [Potier e Qian 2011] investigam como devem ser gerenciadas as informações e os parâmetros em um dispositivo de rádio cognitivo em específico. Entretanto, não se tem conhecimento de um sistema de gerenciamento especializado para a configuração e monitoração de dispositivos de rádio cognitivo, nem mesmo como esse sistema deve ser projetado, considerando a dinamicidade do sensoriamento espectral.

A principal contribuição deste artigo é propor um sistema de gerenciamento para dispositivos de rádio cognitivo considerando um escopo Local. O sistema especializado baseia-se nas informações definidas na MIB IEEE 802.22, sendo o objetivo principal deste artigo o gerenciamento e a monitoração do processo de sensoriamento espectral. Para atender à dinamicidade dos parâmetros de configuração de dispositivos de rádio cognitivo, o sistema foi modelado considerando um gerenciamento automático. Desta forma, o sistema proposto possibilita a configuração automática dos dispositivos baseada em regras para adequação de parâmetros de configuração. Os resultados obtidos através de simulações mostram que o sistema possibilita maximizar a vazão na transmissão dos dispositivos, ajustando a janela de sensoriamento e garantindo a confiabilidade na detecção de canal ocupado.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta a fundamentação teórica sobre redes de rádios cognitivos, bem como o estado da arte sobre as pesquisas em gerenciamento de redes de rádios cognitivos. Na Seção 3, é descrita a arquitetura da solução proposta e o protótipo desenvolvido como prova de conceito. Os detalhes sobre o modelo do sistema e os resultados obtidos são apresentados na Seção 4. Por fim, a Seção 5 conclui o presente trabalho e apresenta perspectivas para trabalhos futuros.

2. Fundamentação Teórica

Nesta seção são abordados os elementos que definem os conceitos sobre dispositivos de rádios cognitivos e seu gerenciamento. Na Subseção 2.1, são descritos os principais mecanismos presentes em rádios cognitivos, que tornam possível a comunicação entre estes

dispositivos. Na Subseção 2.2, são abordados os trabalhos relacionados ao gerenciamento de redes compostas por dispositivos de rádio cognitivo.

2.1. Gerência em Rádios Cognitivos

O rádio cognitivo pode ser definido como um dispositivo capaz de mudar seus parâmetros dinamicamente, através da análise do ambiente em que se encontra, sem interferir na operação dos demais dispositivos. As características fundamentais do rádio cognitivo são a capacidade cognitiva e a reconfigurabilidade [Akyildiz *et al.* 2008]. Através da capacidade cognitiva, o dispositivo pode selecionar o melhor canal disponível para transmissão em um determinado instante de tempo. Além disso, a reconfigurabilidade permite que o rádio cognitivo ajuste os parâmetros mais adequados para a utilização do espectro selecionado.

Com a mudança dinâmica dos parâmetros nas redes de rádios cognitivos, os dispositivos dotados da capacidade cognitiva devem ser capazes de modificar suas configurações para se adaptar às mudanças no ambiente em que se encontram, por exemplo ajustando a frequência em que estão trabalhando. Isso gera a necessidade de gerenciar essas configurações, a fim de manter a rede de comunicação operando de maneira satisfatória, sem interferir em transmissões de outros dispositivos que eventualmente utilizem o mesmo canal [Coutinho *et al.* 2012]. Dentre os trabalhos para gerenciamento das informações, encontra-se a MIB IEEE 802.22 [IEEE 802.22 2011]. A organização das informações no formato de uma MIB facilita o acesso e manipulação dessas informações, possibilitando o desenvolvimento de sistemas de gerenciamento para redes de rádios cognitivos. A MIB IEEE 802.22 é organizada em sete grupos, separados de acordo com a função exercida por cada elemento que compõe a rede: estação radio-base (*Base Station* - BS) ou dispositivo nas dependências do usuário (*Customer Premisse Equipment* - CPE). Esses grupos são descritos a seguir:

- *wranDevMib*: apresenta objetos contendo informações comuns a todos os dispositivos da rede;
- *wranIfBsMib*: apresenta objetos referentes exclusivamente à operação da BS;
- *wranIfBsSfMgmt*: apresenta objetos contendo informações sobre o fluxo de serviços de configuração, instanciação e gerência da BS;
- *wranIfCpeMib*: apresenta objetos referentes à operação móvel ou fixa dos CPEs;
- *wranIfSmMib*: apresenta objetos relacionados ao gerenciamento da utilização do espectro de frequências;
- *wranIfSsaMib*: apresenta objetos relacionados ao processo de sensoriamento espectral;
- *wranIfDatabaseServiceMib*: apresenta objetos relacionados ao serviço de banco de dados com informações gerais da rede.

Alguns dos objetos presentes na MIB IEEE 802.22 fornecem informações relevantes para maximizar o desempenho da rede. Dentre esses objetos, pode-se citar o grupo que provê informações sobre o sensoriamento do espectro (*wranIfSsaMib*), bem como objetos que provêm informações sobre a vazão obtida na transmissão de dados pelos dispositivos (*wranCpeTxThroughput*, por exemplo). A Figura 1 ilustra a estrutura dos sete grupos da MIB IEEE 802.22, apresentando alguns dos objetos que foram analisados no desenvolvimento deste artigo.

Dentro desses sete grupos, pode-se destacar alguns objetos importantes para o desenvolvimento deste artigo, visto que impactam diretamente na configuração e na aplicação do sensoriamento espectral. Por exemplo, os que se referem a configuração da janela

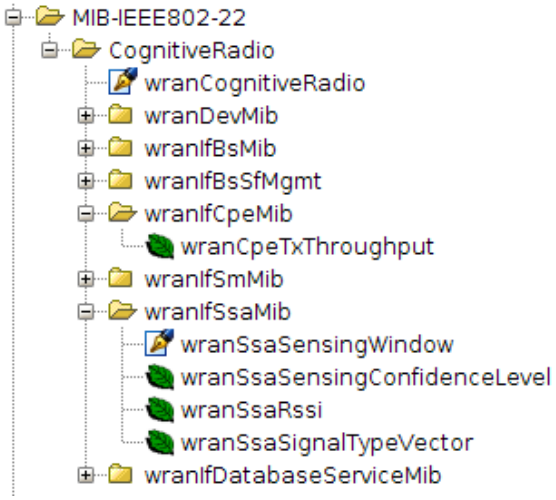


Figura 1. MIB IEEE 802.22

de sensoriamento (*wranSsaSensingWindow*). Esta janela determina por quanto tempo o dispositivo de rádio cognitivo irá analisar o canal, a fim de descobrir se o canal está ocupado. A MIB apresenta também o objeto que provê o resultado da vazão obtida por cada dispositivo na transmissão de dados (*wranCpeTxTrougput*). O nível de confiança na detecção de canais ocupados também está presente (*wranSsaSensingConfidendeLevel*), assim como o objeto que define a intensidade do sinal reconhecido no sensoriamento (*wranSsaRssi*) e o vetor que identifica o tipo de sinal recebido (*wranSsaSignalTypeVector*), como por exemplo *Orthogonal Frequency Division Multiplexing* (OFDM) ou microfone sem fio.

Além da MIB IEEE 802.22, a literatura sobre gerência de redes apresenta propostas para o gerenciamento de redes de rádios cognitivos. A Subseção 2.2 apresenta os principais trabalhos sobre esse tópico de pesquisa na literatura.

2.2. Trabalhos Relacionados

Neste artigo, o gerenciamento de dispositivos de rádio cognitivo foi classificado em dois escopos. O primeiro, chamado escopo Global, tem como objetivo gerenciar as informações pertinentes às redes de rádios cognitivos. Essas informações precisam ser compartilhadas por todas as redes, por exemplo, as frequências em uso nas redes adjacentes e geolocalização dos dispositivos. O segundo escopo é definido como Local, sendo responsável por gerenciar informações presentes em cada dispositivo que compõe a rede. Este escopo é composto por informações armazenadas localmente em cada dispositivo, como os objetos presentes na MIB IEEE 802.22.

A literatura referente ao escopo Local, foco de investigação deste artigo, apresenta poucos trabalhos publicados. Um desses trabalhos é o artigo de Potier e Quian [Potier e Qian 2011], que apresenta um protocolo para gerência de dispositivos de rádio cognitivo, definindo a atuação do rádio cognitivo nas diferentes camadas de protocolos. O protocolo proposto pelos autores é elaborado de forma que as decisões tomadas não afetem a operação dos demais dispositivos que compõem a rede de rádios cognitivos. Entretanto, a solução não apresenta análises sobre a correlação entre os parâmetros de configuração, como por exemplo o impacto da alteração do tempo de sensoriamento no desempenho da rede.

Outro trabalho sobre gerenciamento local foi publicado por Wang *et al.* [Wang *et al.* 2010], que propõe uma arquitetura para manipulação local das informações dos dispositivos. Essa arquitetura possibilita que o dispositivo de rádio cognitivo configure automaticamente seu acesso à rede. Baseado na arquitetura proposta, os autores elaboraram um sistema multiprocessado em um único chip (*Multiprocessor System-on-Chip* - MPSoC). Embora a arquitetura proposta no trabalho considere que os dispositivos necessitam de gerenciamento, o trabalho não indica como esse gerenciamento deve ser realizado.

Baseado na literatura sobre gerenciamento de rádio cognitivo, pode-se perceber a necessidade de um sistema de gerência Local, que considere a dinamicidade do sensoriamento espectral. Dessa forma, este artigo propõe um sistema de gerenciamento capaz de ajustar automaticamente os parâmetros de configuração necessários para maximizar a vazão na transmissão de dados do dispositivo de rádio cognitivo, enquanto o nível de confiabilidade é mantido. Os detalhes do sistema proposto são descritos na Seção 3.

3. Sistema de Gerenciamento para Dispositivo de Rádio Cognitivo

Nesta seção estão descritos os detalhes do sistema de gerenciamento proposto para dispositivos de rádio cognitivo. Inicialmente, na Subseção 3.1 é apresentada e discutida a arquitetura do sistema. Em seguida, na Subseção 3.2, são descritos detalhes relativos ao desenvolvimento do protótipo utilizado para comprovar o funcionamento da solução proposta.

3.1. Arquitetura

O sistema de gerenciamento proposto neste artigo foi projetado dentro do escopo Local de redes de rádio cognitivos. Uma rede de rádios cognitivos é composta por dois tipos de dispositivos: BS e CPE. Cada CPE possui um conjunto de informações gerenciáveis, isso é, informações que representam a configuração ativa no dispositivo, como por exemplo o tamanho da janela de sensoriamento. Além disso, o dispositivo armazena as informações obtidas após o processo de sensoriamento, como a taxa de acerto na identificação dos canais livres.

A BS caracteriza-se por concentrar todas as informações da rede em um sistema de comunicação sem fio. Essa característica torna a BS o melhor elemento para implantação do sistema de gerenciamento. Dessa forma, o sistema de gerenciamento proposto opera na BS. Assim, o sistema é capaz de gerenciar todos os dispositivos que fazem parte da rede de rádios cognitivos, bem como processar as informações obtidas e realizar os ajustes de configuração necessários.

O sistema de gerenciamento proposto baseia-se nas configurações estabelecidas no início da operação de gerência, realizadas pelo administrador do sistema. Posteriormente, todo o processo de gerência das operações é realizado de maneira automática. A operação automática do sistema de gerenciamento é proposta devido a grande variabilidade dos parâmetros de configuração que os dispositivos cognitivos possuem. Apesar de não necessitar da intervenção do administrador da rede durante sua execução, o sistema de gerenciamento pode ter suas operações paralisadas a qualquer momento, permitindo que o administrador configure os parâmetros do sistema de acordo com a necessidade. A Figura 2 ilustra em detalhes a arquitetura do sistema de gerenciamento proposto.

A operação do sistema de gerenciamento é formada por quatro módulos: (i) Interface, (ii) Regras, (iii) Coleta e (iv) Configuração. O sistema inicia sua operação pelo

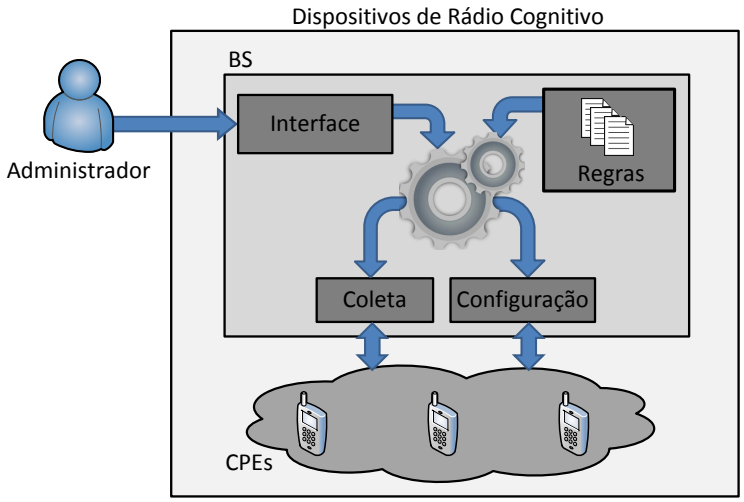


Figura 2. Arquitetura do sistema de gerência proposto

módulo de Interface. Nesse módulo, o administrador configura os valores desejados para os fatores utilizados nas Regras de configuração do rádio cognitivo. Essas regras utilizam limiares para ajustar os valores dos parâmetros de configuração dos dispositivos, podendo ser descritas pelo próprio administrador da rede. Os módulos de Coleta e Configuração têm como objetivo obter informações e aplicar ajustes na configuração dos dispositivos de rádio cognitivo. Baseado na arquitetura proposta, foi desenvolvido um protótipo do sistema de gerenciamento. Os detalhes dos protótipo são descritos na subseção a seguir.

3.2. Protótipo

Para comprovar o funcionamento do sistema de gerenciamento proposto, foi desenvolvido um protótipo em linguagem de programação C baseado na arquitetura descrita na Subseção 3.1. As informações são obtidas por um gerente através de um agente para dispositivos de rádio cognitivo, desenvolvido através do *Simple Network Management Protocol* (SNMP) [IETF 1988]. Este protocolo provê um meio de comunicação simples e eficiente entre os dispositivos gerenciados e o sistema de gerenciamento. As informações no agente são organizadas em uma MIB, que possui objetos baseados na MIB IEEE 802.22, de acordo com a representação da Figura 1. As informações do processo de sensoriamento espectral são obtidas através da simulação de um detector de energia no *software* Matlab.

O sistema de gerenciamento precisa ser configurado apenas durante sua inicialização, onde são informados os valores de ajuste dos parâmetros de configuração. Esses valores são utilizados pelas regras de configuração, respeitando os limiares descritos nestas regras. Para validação do sistema de gerenciamento, o protótipo desenvolvido apresenta atualmente duas regras de configuração, voltadas para o sensoriamento espectral: (i) baseada nos limiares da vazão do dispositivo e (ii) baseada nos limiares do nível de confiança para identificação de ocupação dos canais de rádio frequência. Entretanto, novas regras de configuração podem ser adicionadas facilmente. Após a coleta das informações, o sistema de gerenciamento aciona o módulo das regras de configuração. Esse módulo calcula qual a melhor configuração para os parâmetros do agente de acordo com os limiares previamente definidos.

Os limiares estabelecidos consideram os valores máximos e mínimos para os parâ-

metros analisados. Por exemplo, de acordo com a norma IEEE 802.22, o nível mínimo de confiança no sinal recebido para identificação de canal ocupado é de 67% [IEEE 802.22 2011]. Assim, existe uma relação direta entre a vazão obtida na transmissão e o nível de confiança para a percepção se determinado canal está livre ou não. Esse nível de confiança está relacionado com o tamanho da janela de sensoriamento. Por exemplo, ao diminuir a janela de sensoriamento para maximizar a vazão, deve-se considerar que essa diminuição afetará a confiança do resultado do sensoriamento espectral.

A partir das especificações do protótipo, na próxima seção são descritos os detalhes sobre a modelagem do ambiente de simulação para o sistema de gerenciamento. Além disso, são apresentados e discutidos os resultados obtidos através das simulações realizadas dentro do ambiente definido.

4. Avaliação e Resultados

Esta seção apresenta os detalhes sobre as simulações realizadas para validar o funcionamento do sistema. Além disso, os resultados obtidos são apresentados e discutidos. A Subseção 4.1 apresenta os detalhes da modelagem do sistema de gerenciamento para rádio cognitivo. Na Subseção 4.2 são discutidos os resultados obtidos pela simulação, através da análise de desempenho do sistema.

4.1. Modelagem do Sistema de Gerenciamento

As simulações realizadas neste trabalho têm como objetivo principal avaliar o funcionamento do sistema de gerenciamento proposto, utilizando duas regras de configuração. A primeira regra visa maximizar a vazão, ajustando a janela de sensoriamento em relação ao nível de confiança para detectar canais ocupados. Por outro lado, a segunda regra objetiva maximizar o nível de confiança do sensoriamento espectral. Sendo assim, nesta seção, é apresentada a modelagem do sistema de gerenciamento utilizada para a avaliação de desempenho.

Inicialmente, é necessário modelar um sinal de transmissão sem fio para ser utilizado na simulação como prova de conceito. Dessa forma, é importante ressaltar que qualquer tipo de sinal pode ser utilizado no sistema proposto. A Equação 1 apresenta o sinal modelado $S(t)$, de acordo com a definição de Liang *et al.* [Liang *et al.* 2008].

$$S(t) = \cos\left(2\pi \int_{0}^{t} [f_c + f_\Delta s(n)]dn\right) \tag{1}$$

onde, t é um instante de tempo, f_c é a frequência portadora, f_Δ é o desvio de frequência e $s(n)$ é o sinal de origem. A detecção de $S(t)$ pelo receptor é realizada de acordo com uma taxa de amostragem N, definida pela a Equação 2.

$$N = \tau.2.f_{max} \tag{2}$$

onde, τ representa a duração do sensoriamento espectral, em milissegundos, e f_{max} indica a frequência máxima do sinal amostrado. Considerando a amostragem realizada, a capacidade de transmissão de um determinado canal $C(t)$, em Mbit/s, pode ser obtida através do teorema de Shannon:

$$C(t) = B.log_2\left(1 + \frac{S(t)}{R(t)}\right) \tag{3}$$

onde B representa a largura de banda do canal e $\frac{S(t)}{R(t)}$ a relação entre o nível do sinal e do ruído, medida em um instante de tempo t, sendo que $R(t) \neq 0$. Essa relação é obtida com base no indicador de potência do sinal recebido (*Received Signal Strength Indication* - RSSI). Entretanto, para a obtenção da taxa efetiva de transmissão da rede de rádios cognitivos, é necessário considerar a duração da janela sensoriamento espectral (τ), para identificar se um determinado canal está livre ou ocupado. Durante o tempo de sensoriamento, não é possível realizar transmissões, portanto a duração de uma transmissão de dados (T_D), em milissegundos, pode ser definida como:

$$T_D = Q_D - \tau \tag{4}$$

onde Q_D é a duração de um quadro, em milissegundos e $Q_D > \tau$. Dessa forma, a capacidade de transmissão de dados em cada quadro (C_Q) pode ser obtida através da Equação 5.

$$C_Q = C(t).T_D \tag{5}$$

Assim, a capacidade total de transmissão (C'), em Mbit/s, descontando-se a janela de sensoriamento, é definida na Equação 6. Através desta equação, o sistema de gerenciamento proposto neste trabalho analisa a capacidade de transmissão do dispositivo em relação ao tempo da janela de sensoriamento.

$$C' = C_Q.\frac{10^3}{Q_D}, \quad (Q_D > 0) \tag{6}$$

Após a definição da capacidade total de transmissão, é necessário relacioná-la com a confiança na detecção do sinal. Essa confiança é baseada no nível de similaridade da energia do sinal antes da transmissão e após a amostragem realizada pelo dispositivo receptor do sinal. O nível de energia médio do sinal amostrado é calculado através da Equação 7.

$$M(y) = \frac{1}{N}\sum_{n=1}^{N}|y(n)|^2 \tag{7}$$

onde, $y(n) = s(n) + u(n)$ é o somatório do sinal transmitido ($s(n)$) com os ruídos e interferências que o compõe ($u(n)$). Como o ruído analisado no sinal recebido é do tipo *Additive White Gaussian Noise* (AWGN), a relação não resultará em um valor de sinal recebido muito acima do transmitido. Dessa forma, considerando-se que $M(y)'$ é a energia medida antes da transmissão e $M(y)''$ é a energia medida após, tem-se o nível de confiança (ς), dado pela Equação 8.

$$\varsigma = 1 - \left|1 - \frac{M(y)''}{M(y)'}\right|, \quad (M(y)' > 0) \tag{8}$$

No sistema de gerenciamento proposto neste artigo, τ pode ser alterado automaticamente. O valor desta alteração automática é chamado de ajuste da janela de sensoriamento e o intervalo entre cada ajuste é definido como período de *polling*. A relação entre a vazão e o nível de confiança do sensoriamento pode ser obtida pela normalização dos valores obtidos através das regras de configuração. Esses valores são representados pela tupla $v = [C', \varsigma]$, onde $v \in V$ e V é um conjunto de tuplas. A primeira regra prioriza maximizar a vazão, onde o parâmetro secundário é o nível de confiança. Dessa forma, a tupla resultante da regra da vazão (RC') é definida por:

$$RC' = V_k \rightarrow \forall i \neq k, V_{C',k} > V_{C',i} \quad \wedge \quad V_{\varsigma,k} > \varsigma_{\min} \tag{9}$$

onde, $\varsigma_{\min}$ é o menor nível de confiança tolerado pelo sistema de gerenciamento. Por outro lado, a segunda regra prioriza obter o maior nível de confiança, deixando a vazão como parâmetro secundário. Relacionando-se os valores obtidos em RC', pode-se obter uma taxa relativa à regra da vazão ($R'C'$):

$$R'C' = \frac{RC'_{C'}}{RC'_{\varsigma}} \tag{10}$$

De forma análoga à regra de priorização da vazão, pode-se obter a tupla ótima para a regra de priorização da confiança ($R\varsigma$). Tem-se que $R\varsigma$ é representada pela Equação 11:

$$R\varsigma = V_k \rightarrow \forall i \neq k, V_{\varsigma,k} > V_{\varsigma,i} \tag{11}$$

Baseando-se novamente na tupla v, pode-se calcular a taxa relativa à regra do nível de confiança ($R'\varsigma$). Essa taxa é definida pela Equação 12:

$$R'\varsigma = \frac{R\varsigma_{\varsigma}}{R\varsigma_{C'}} \tag{12}$$

Finalmente, pode-se encontrar a melhor relação entre $R'C'$ e $R'\varsigma$. Essa relação é obtida pelo valor mínimo da diferença entre o resultado das duas regras de configuração, como pode ser observado na Equação 13. Dessa forma, a menor discrepância entre os valores da tupla v representa a melhor configuração possível entre a vazão e o nível de confiança do sensoriamento.

$$R = \min\{R'C', R'\varsigma\} \tag{13}$$

A modelagem do sistema descrita nesta subseção é validada em um ambiente de simulação. Esse ambiente e a avaliação de desempenho do sistema de gerenciamento são apresentados e discutidos na próxima subseção.

4.2. Avaliação de Desempenho

A avaliação de desempenho do sistema de gerenciamento proposto foi realizada em um ambiente de simulação. A Tabela 1 apresenta os principais parâmetros do sistema, bem como os limiares utilizados nas regras de configuração. Os parâmetros e limiares utilizados neste trabalho são baseados na norma IEEE 802.22, que define o padrão para redes sem fio regionais (*Wireless Regional Networks* - WRAN) [IEEE 802.22 2011]. Estes parâ-

metros e limiares também caracterizam as suposições iniciais para a operação do sistema de gerenciamento. A janela de sensoriamento foi configurada entre 50μs e 5ms. O valor mínimo do nível de confiança para a detecção de um canal ocupado foi de 67%, considerando um ruído AWGN de 20dB. A duração do quadro foi definida em 10^{-3}s, o ajuste da janela de sensoriamento variando de 10^{-6} até 10^{-5}s e o período de *polling* foi ajustado entre 10^{-1}s e 1s. Além disso, os experimentos foram repetidos até alcançar um intervalo de confiança de 95%.

Parâmetro	**Valor**
Janela de sensoriamento	$[5.10^{-5}$:$5.10^{-3}]$s
Nível de confiança mínimo	67%
Ruído	AWGN
Relação Sinal/Ruído	20dB
Duração do Quadro	10^{-3}s
Ajuste da janela de sensoriamento	$[10^{-6}$:$10^{-5}]$s
Período de *Polling*	$[10^{-1}$:1]s
Intervalo de confiança	95%

Tabela 1. Limiares considerados para simulação

O sistema de gerenciamento foi analisado considerando quatro abordagens: (i) o comportamento da tupla v em função do tamanho da janela de sensoriamento, (ii) regras de configuração em relação ao ajuste da janela de sensoriamento, (iii) sobrecarga do tráfego SNMP para o ajuste da janela de sensoriamento e (iv) o tempo de convergência entre as regras de configuração, considerando o período de *polling*.

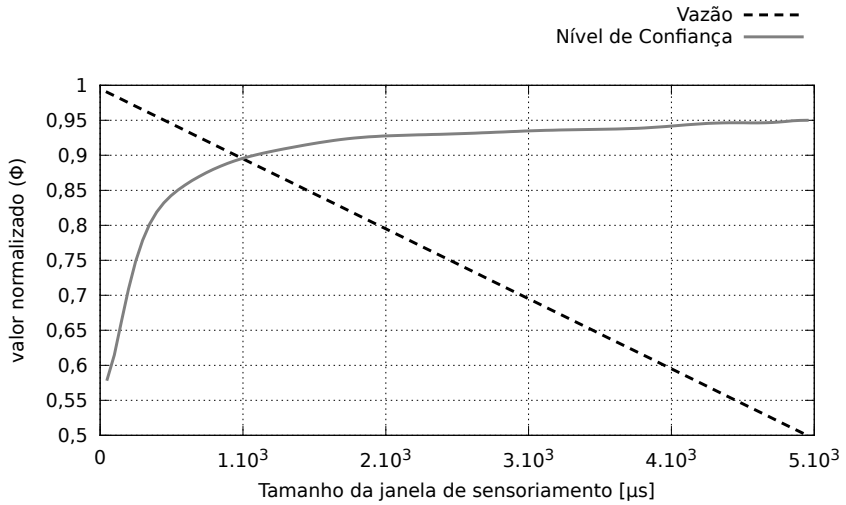


Figura 3. Relação da vazão e do nível de confiança

A primeira análise refere-se ao comportamento da tupla v em função do tamanho da janela de sensoriamento e da relação normalizada entre o percentual da vazão e do nível de confiança. Esta relação normalizada é definida como Φ. Na Figura 3, observa-se a tendência dos valores normalizados da vazão, onde a medida em que o tamanho da janela de sensoriamento aumenta, a vazão máxima obtida é reduzida linearmente e os valores normalizados do nível de confiança apresentam um crescimento logarítmico. De acordo com as tendências do comportamento da tupla v, pode-se encontrar o ponto de

inflexão entre as curvas de vazão e nível de confiança. Esse ponto indica o valor da janela de sensoriamento onde a vazão e o nível de confiança possuem a menor variação entre si. O sistema de gerenciamento proposto para o sensoriamento espectral objetiva configurar automaticamente a janela de sensoriamento, segundo as duas regras de configuração.

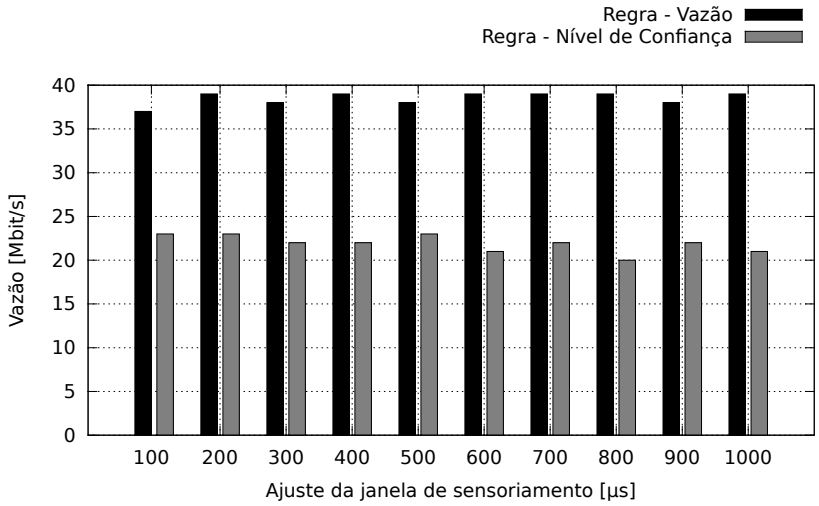


Figura 4. Vazão obtida para regras de confiança

A segunda avaliação de desempenho analisa as regras de configuração em relação ao ajuste da janela de sensoriamento. Essa analise é realizada em duas etapas, primeiramente considerando a vazão e posteriormente o nível de confiança. Na Figura 4 pode-se observar a vazão em detrimento ao ajuste da janela de sensoriamento. Nessa análise, pode-se confirmar que a utilização da regra da vazão aumenta a taxa de transmissão de dados em relação a regra do nível de confiança. No cenário analisado, esse aumento é em média 16 Mbit/s. Outra possível análise está relacionada ao ajuste da janela de sensoriamento. Diferentemente do esperado, esse ajuste não afeta significativamente a vazão do dispositivo gerenciado. Esse comportamento ocorre devido ao tamanho da janela de sensoriamento final ser semelhante para os diversos ajustes. A diferença está na quantidade de mensagens SNMP trocadas para configurar o tamanho a janela de sensoriamento.

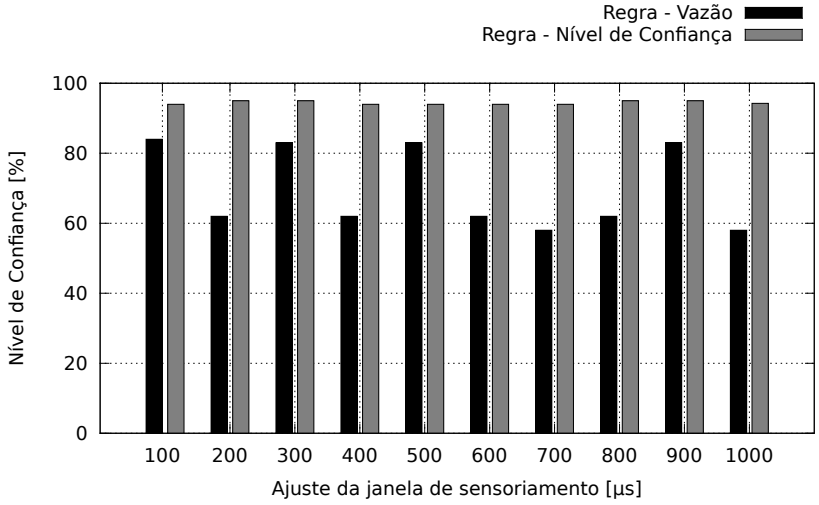


Figura 5. Nível de confiança obtido para regras de confiança

Na Figura 5, pode-se observar o nível de confiança em relação ao ajuste da janela de sensoriamento para as regras de configuração. Percebe-se que o percentual de confiança é de aproximadamente de 95%, quando utiliza-se a regra do nível de confiança. Além disso, ao utilizar essa regra, o percentual de confiança é similar para todos os ajustes da janela de sensoriamento, ou seja, existe uma convergência em torno de uma confiabilidade máxima. Por outro lado, pode-se observar que existe uma variabilidade no percentual do nível de confiança para a regra da vazão. Considerando-se essa regra percebe-se uma variação de aproximadamente 23% no nível de confiança, entre os diversos ajustes da janela de sensoriamento. Essa variabilidade é decorrente do crescimento logarítmico do nível de confiança, como apresentado na Figura 3, onde o nível de confiança para uma janela de sensoriamento inferior a $1.10^3\mu$s possui um crescimento acentuado. Isto é, para valores de janela de sensoriamento próximos de $1.10^3\mu$s, a variação do nível de confiança é maior.

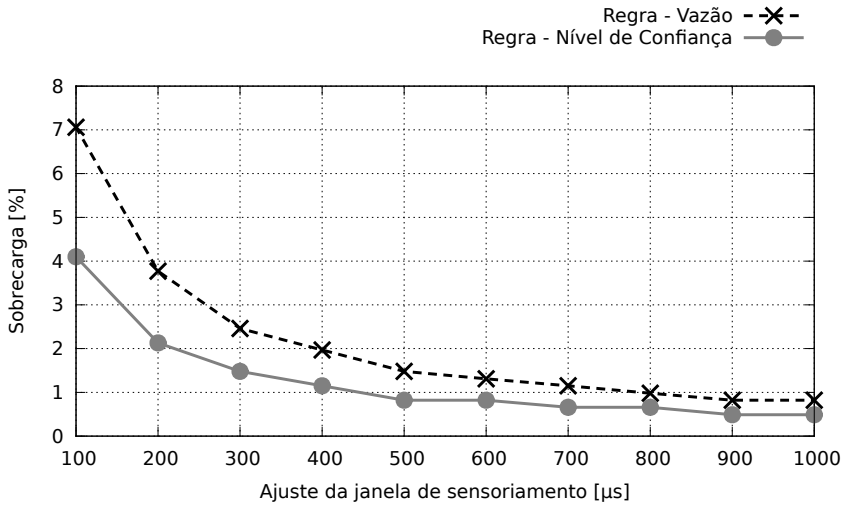


Figura 6. Sobrecarga do tráfego SNMP

A terceira análise deste artigo investiga a sobrecarga do tráfego SNMP gerada pelo sistema de gerenciamento e do dispositivo de rádio cognitivo. O cálculo da sobrecarga baseia-se na quantidade de pacotes SNMP em relação a capacidade máxima no canal de transmissão ($C(t)$), em Mbit/s, definido pela Equação 3. A Figura 6 apresenta o percentual da sobrecarga em função do ajuste da janela de sensoriamento para as regras de configuração. A investigação confirma que a sobrecarga de tráfego SNMP decresce à medida em que o valor do ajuste da janela de sensoriamento aumenta. Esta redução na sobrecarga ocorre pois, quanto maior o valor do ajuste na janela de sensoriamento, menos tráfego SNMP é necessário para alcançar o tamanho final da janela de sensoriamento. Outra análise que pode-se realizar está relacionada as diferenças de sobrecarga em relação as regras de configuração. A sobrecarga para a regra do nível de confiança apresenta-se menor do que a sobrecarga para a regra da vazão. Esse comportamento é justificado devido ao nível de confiança apresentar um crescimento logarítmico, como apresentado na Figura 3. Dessa forma, é necessário um menor ajuste total da janela de sensoriamento para alcançar o tamanho da janela de sensoriamento desejada.

A quarta e última análise realizada neste trabalho tem como objetivo encontrar a relação ideal entre a máxima vazão e o melhor nível de confiança. Nesta análise considera-se o período de *polling*, isto é o intervalo entre cada ajuste da janela de sensoriamento. Essa relação foi definida segundo a formulação descrita na Subseção 4.1,

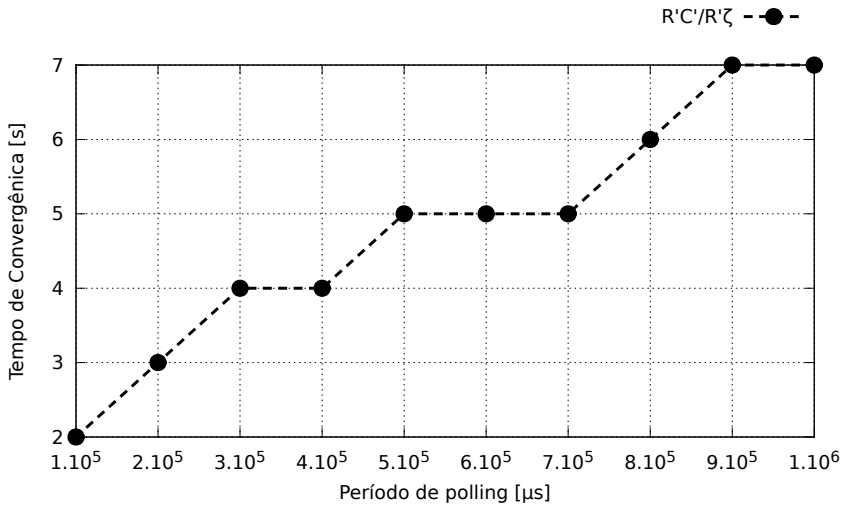


Figura 7. Tempo de convergência em relação as tuplas $R'C'$ e R'_ς

onde matematicamente pode-se chegar ao ajuste ideal para a janela de sensoriamento.

Considerando a relação entre as tuplas RC' e R_ς, o ajuste da janela de sensoriamento calculado para tupla RC' é de 900μs, já para a tupla R_ς o ajuste é de 500μs. A Figura 7 apresenta os tempos de convergência considerando o período de *polling* para a melhor relação entre as tuplas RC' e R_ς. Além disso, pode-se observar o aumento no tempo de convergência quando o período de *polling* também aumenta. Este comportamento ocorre devido ao sistema de gerenciamento esperar o período de *polling* para realizar um novo ajuste na janela de sensoriamento.

5. Conclusões e Trabalhos Futuros

A proposta deste artigo apresenta um sistema de gerenciamento para radio cognitivo capaz de atuar com a dinamicidade das configurações de dispositivos cognitivos. Para tando, foram estudados os objetos da MIB IEEE 802.22 referentes ao processo de sensoriamento espectral. O sistema proposto mostrou-se capaz de melhorar o desempenho da transmissão de informações, ajustando os valores de configurações dos dispositivos de acordo com regras de configuração. Para prova de conceito, foi elaborado um protótipo, o qual ajusta o tamanho da janela de sensoriamento utilizando regras para maximização da vazão ou do nível de confiança. Acredita-se que o protótipo é capaz de operar de maneira satisfatória também em ambientes reais, uma vez que os resultados obtidos por simulação refletem em parte, as condições obtidas em campo.

Futuramente, pretende-se estender o estudo dos objetos que compõe a MIB IEEE 802.22, analisando como os outros parâmetros de configuração podem ser utilizados para melhorar o desempenho da operação da rede. Com isso, será possível desenvolver novas regras de configuração, baseando-se em um conjunto maior de parâmetros. Além disso, pretende-se aprimorar o sistema proposto para que o mesmo possa atuar no escopo Global de redes de rádios cognitivos. Para tanto, o sistema deverá considerar como as decisões tomadas dentro de uma rede afetam a operação de redes adjacentes. Pode-se ainda aumentar o nível de abstração na descrição das regras de configuração, através da descrição destas regras na forma de políticas.

Agradecimentos

Os autores gostariam de agradecer a Digitel S.A., Fundação de Apoio a Pesquisa do Estado do Rio Grande do Sul (FAPERGS), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Coordenação de Aperfeiçoamento de Nível Superior (CAPES) pelo apoio financeiro.

Referências

Akyildiz, I., Lee, W.-Y., Vuran, M., e Mohanty, S. (2008). A survey on spectrum management in cognitive radio networks. *IEEE Communications Magazine*, páginas 40–48.

Coutinho, P. S., da Silva, M. W. R., e de Rezende, J. F. (2012). Mecanismo de Handoff de Espectro para Rádios Cognitivos. *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 44–56.

IEEE 802.22 (2011). IEEE Standard for Information Technology - Telecommunications and information exchange between systems Wireless Regional Area Networks (WRAN) - Specific requirements Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Policies and Procedures for Operation in the TV Bands. *IEEE Std 802.22*, páginas 1–680.

IETF (1988). RFC 1067 - Simple Network Management Protocol. *Disponível em: http://www.ietf.org/rfc/rfc1067.txt*. Acesso em dezembro de 2012.

Khalid, L. e Anpalagan, A. (2010). Emerging cognitive radio technology: Principles, challenges and opportunities. *Computers & Electrical Engineering*, páginas 358–366.

Liang, Y.-C., Zeng, Y., Peh, E., e Hoang, A. T. (2008). Sensing-Throughput Tradeoff for Cognitive Radio Networks. *IEEE Transactions on Wireless Communications*, páginas 1326–1337.

Potier, P. e Qian, L. (2011). Network management of cognitive radio ad hoc networks. *Proceedings of the 4th International Conference on Cognitive Radio and Advanced Spectrum Management*, páginas 1–5.

Stavroulaki, V., Bantouna, A., Kritikou, Y., Tsagkaris, K., Demestichas, P., Blasco, P., Bader, F., Dohler, M., Denkovski, D., Atanasovski, V., Gavrilovska, L., e Moessner, K. (2012). Knowledge Management Toolbox: Machine Learning for Cognitive Radio Networks. *IEEE Vehicular Technology Magazine*, páginas 91–99.

Wang, C.-X., Chen, H.-H., Hong, X., e Guizani, M. (2008). Cognitive radio network management. *IEEE Vehicular Technology Magazine*, páginas 28–35.

Wang, J., Ghosh, M., e Challapali, K. (2011). Emerging cognitive radio applications: A survey. *IEEE Communications Magazine*, páginas 74–81.

Wang, S., Xie, L., Liu, H., Zhang, B., e Zhao, H. (2010). Acra: An autonomic and expandable architecture for cognitive radio nodes. *International Conference on Wireless Communications and Signal Processing (WCSP)*, páginas 1–5.

Yucek, T. e Arslan, H. (2009). A survey of spectrum sensing algorithms for cognitive radio applications. *IEEE Communications Surveys Tutorials*, páginas 116–130.

Zhao, Q. e Sadler, B. (2007). A Survey of Dynamic Spectrum Access. *IEEE Signal Processing Magazine*, páginas 79–89.

Redes de Rádios Cognitivos Utilizando Sequências de Saltos de Canais Baseadas em Papéis*

Raphael Melo Guedes[1], Marcel William Rocha da Silva[2], Pedro Smith Coutinho[1], José Ferreira de Rezende[1]

[1]COPPE – Universidade Federal do Rio de Janeiro (UFRJ)

[2]DTL - IM – Universidade Federal Rural do Rio de Janeiro (UFRRJ)

{raphael,marcel,coutinho,rezende}@land.ufrj.br

Abstract. *In a multi-channel scenario, where nodes hop periodically among channels without coordination, the rendezvous is the first step in establishing a communication. Several studies that deal with this problem do not deal with broadcast transmissions, which are very useful in sending control messages. An intuitive solution consists of all nodes converge to a unique and synchronized sequence. However, this unique sequence is inefficient for sending data messages in unicast, since it requires negotiation packet-by-packet for selecting the transmission channel of the data message thus the capacity of multiple channels would be used. In this work we propose a role-based policy that defines the on-demand use of different sequences, alleviating the media access problem and enabling the occurrence of parallel transmissions on different channels. This proposal has been implemented in the ns-2 in a cognitive radios mesh network scenario, where the presence of primary users favors the use of channel hopping.*

Resumo. *Em um cenário multi-canal, onde os nós saltam periodicamente entre canais sem coordenação, o encontro entre os nós (rendezvous) é a primeira etapa no estabelecimento de uma comunicação. A maioria dos trabalhos que lidam com este problema não tratam de transmissões em broadcast, as quais são bastante úteis no envio de mensagens de controle. Uma solução intuitiva consiste em todos os nós convergirem para uma sequência única e sincronizada. No entanto, esta sequência única é ineficiente para o envio de mensagens de dados em unicast, pois requer a negociação pacote-a-pacote para a escolha do canal de transmissão da mensagem de dados a fim de que a capacidade dos múltiplos canais seja utilizada. Assim, neste trabalho propomos uma política de papéis que define o uso de diferentes sequências sob demanda, aliviando o problema de acesso ao meio e possibilitando a ocorrência de transmissões em paralelo em diferentes canais. Esta proposta foi implementada no ns-2 em um cenário de redes em malha de rádios cognitivos, onde a presença de usuários primários favorece ainda mais o uso do mecanismo de saltos de canais.*

1. Introdução

Devido à capacidade de acesso dinâmico ao espectro (*Dynamic Spectrum Access* - DSA) dos rádios cognitivos, eles representam uma solução promissora ao problema de escassez

*Este trabalho recebeu recursos do CNPq, CAPES, FAPERJ, FINEP e RNP.

e subutilização do espectro. Esses rádios, ou usuários secundários (USs), possuem permissão para acessar faixas do espectro de frequências licenciadas (*i.e.*, canais) durante os períodos em que os usuários licenciados, ou usuários primários (UPs), daquela faixa não a utilizam [Akyildiz et al. 2006]. Portanto, os USs devem suspender sua operação e migrar para outro canal disponível sempre que um UP estiver presente. Neste cenário multicanal, o encontro entre pares de nós num determinado canal (*rendezvous*) é o primeiro passo para o início de uma comunicação [Theis et al. 2011].

O *rendezvous* pode ser demorado quando não se possui nenhuma informação prévia de quais ou qual canal está livre para o uso. Assim, muitos trabalhos em DSA recorrem ao uso de um canal de controle exclusivo para troca de mensagens de coordenação entre os USs cuja finalidade é facilitar a ocorrência de *rendezvous*. Porém, devido a disponibilidade dinâmica dos canais, qualquer canal selecionado como canal de controle pode estar sendo compartilhado com usuários UPs, podendo permanecer inacessível durante longos períodos. Ou seja, um único canal de controle não é o mais aconselhável em redes DSA. Logo, diversas propostas adotam a ideia de um canal de controle que usa saltos de canais (*channel hopping*), na intenção de tornar-se imune à seleção de um único canal e garantir um encontro mais rápido entre os nós, visto que estes saltam por um conjunto de sequências de canais (*rendezvous channels*) aumentando as chances de um encontro.

Diversos algoritmos foram propostos na literatura para a geração de sequências aleatórias de canais que permitem garantir um tempo máximo para o encontro (*Maximum-Time-To-Rendezvous* - MTTR) entre dois nós. Esses algoritmos são comumente referenciados como algoritmos de *rendezvous* (RV). Formalmente, a sequência de saltos de canais de cada nó é estabelecida na forma (*slot*, canal) e, quando ocorre um casamento entre *slot* e canal de ambos os nós, eles podem então estabelecer uma comunicação e trocar informações, sejam dados ou controle.

Porém, um problema encontrado em grande parte dos algoritmos de RV é o fato deles não lidarem diretamente com transmissões *broadcast*. Assim, mensagens de controle que devem ser enviadas por um nó a todos os seu vizinhos, tais como mensagens de *hello*, RTS/CTS, informações de estado de enlace, entre outras, exigem a ocorrência de múltiplos RVs nó-a-nó. Estes múltiplos RVs *unicast* atrasam a entrega da mensagem a todos os nós e podem prejudicar o estabelecimento de alguma comunicação futura.

Para este problema, uma solução intuitiva seria todos os nós adotarem uma sequência única como canal de controle. Neste caso, uma transmissão *broadcast* ocorreria de forma natural. No entanto, ao se adotar esta solução, os nós passariam a negociar, neste canal de controle, o canal (ou a sequência de canais) a ser utilizado(a) nas futuras transmissões *unicast*. No caso do nó ser um elemento intermediário que encaminha pacotes para diferentes próximos saltos, essa negociação teria que ser feita praticamente pacote-a-pacote, elevando o número de requisições de acesso ao canal de controle e provocando um problema conhecido como gargalo do canal de controle [So and Vaidya 2004, Mo et al. 2005, Luo et al. 2006].

A solução proposta neste trabalho utiliza a ideia dos nós assumirem diferentes papéis no decorrer do tempo, os quais são associados ao uso de diferentes sequências de saltos de canais. Conforme o estado do nó e a necessidade de uso dos canais para transmissão, ele chaveia entre diferentes sequências de saltos de canais. Assim, quando o nó

deseja enviar uma mensagem de controle em *broadcast*, ele chaveia de forma assíncrona para uma sequência de saltos computada localmente e que permite a ele atingir o maior número de vizinhos, evitando os canais com maior probabilidade de uso por UPs. Quando o nó estiver em repouso, ele permanece na sequência definida pelo algoritmo de RV utilizado. Finalmente, para o envio de mensagens em *unicast*, o nó transmissor deve chavear para a sequência de repouso do nó receptor.

O problema da transmissão em *broadcast* pode ser resolvido conforme proposto em [Guedes et al. 2012], onde seleciona-se de maneira gulosa os canais utilizados de forma a relacionar o conhecimento das sequências dos nós vizinhos e a estimativa de uso do canal pelos nós UPs. Adotando esta prática, de mudança de sequências conforme a necessidade, diminui-se a competição pelo acesso ao meio além de favorecer possíveis transmissões em paralelo, inclusive entre nós vizinhos. Além disso, não gera a necessidade de negociação prévia para uma transmissão *unicast*, pois o nó TX deve simplesmente chavear para a sequência de repouso do nó RX.

Neste artigo, a solução proposta e a solução intuitiva foram implementadas no simulador *ns-2*, onde foi necessário incluir: um modelo de interferência físico baseado na SINR (*Signal-to-Interference-plus-Noise Ratio*) [Brar et al. 2006], um novo agente gerador de tráfego em *broadcast*, a possibilidade dos nós saltarem de canal segundo diferentes sequências de saltos especificadas, os algoritmos de *rendezvous* para a geração dessas sequências, o algoritmo proposto em [Guedes et al. 2012] para a geração da sequência de *broadcast*, e a coleta das métricas de desempenho para a comparação das duas soluções.

A Seção 2 descreve os principais trabalhos que tratam do problema de *rendezvous*. A Seção 3 apresenta o modelo do sistema sob qual reside a proposta deste artigo. A Seção 4 descreve o mecanismo proposto neste trabalho e a Seção 5 descreve o ambiente de simulação usado na avaliação de desempenho da solução proposta e de uma solução intuitiva para o problema tratado. Em seguida, a Seção 6 traz os resultados das simulações realizadas, assim como uma discussão a respeito. E, por fim, a Seção 7 apresenta as conclusões deste trabalho.

2. Trabalhos Relacionados

Diversos trabalhos em rádios cognitivos (RCs) adotam a técnica de *channel hopping* precisamente para evitar os canais ocupados pelos UPs, permitindo o encontro desses RCs em outros canais. Além disso, essa abordagem permite a comunicação simultânea de múltiplos RCs na mesma área de interferência, através da atribuição de diferentes sequências de saltos de canais para cada nó. Vários trabalhos na literatura tratam o problema de *rendezvous*, cuja diferença entre eles está na utilização de diferentes estratégias na construção de suas sequências de salto de canais [Silvius et al. 2008, Bahl et al. 2004, DaSilva and Guerreiro 2008, Bian et al. 2011, Cormio and Chowdhury 2010, Theis et al. 2011, Lin et al. 2011, Zhang et al. 2011, Guedes et al. 2012]. Esses trabalhos são avaliados e comparados de acordo com as métricas de tempo máximo necessário para que dois nós se encontrem (MTTR - *Maximum-Time-To-Rendezvous*) e a distribuição dos encontros nos diferentes canais da sequência de saltos. Quanto menor é o MTTR e quanto mais distribuídos são os encontros nos diferentes canais, melhor é o algoritmo.

A maior parte dos algoritmos de *rendezvous* (RV) existentes assume um sistema

de divisão do tempo em *slots*, em que a cada *slot* os RCs saltam entre os canais na tentativa de encontrar seus potenciais vizinhos. Em um esquema onde cada RC salta entre canais aleatoriamente [Silvius et al. 2008], quando dois RCs saltam para o mesmo canal no mesmo instante, um encontro (RV) ocorre, caso o canal esteja livre da presença de algum UP. Caso contrário, os nós devem continuar "saltando" até que ocorra o RV.

Algoritmos de RV podem ser classificados como síncronos ou assíncronos, dependendo da simultaneidade no início das suas sequências de saltos. Em sistemas síncronos, uma coordenação inicial entre os usuários é necessária antes do *rendezvous* para estabelecer o sincronismo entre os *clocks* dos RCs [DaSilva and Guerreiro 2008, Bian et al. 2011, Bahl et al. 2004, Guedes et al. 2012]. Desta forma, as trocas de canal e os *slots* ficam consequentemente alinhados no tempo.

Um exemplo de algoritmo de RV é o *modular clock* (*mclock*) [Theis et al. 2011]. Este algoritmo usa aritmética modular e números primos para garantir um MTTR que seja satisfatório. O algoritmo começa pela seleção de um canal de forma aleatória, a partir da lista de canais disponíveis. Após cada intervalo de tempo (*slot*), o CR salta $r_i \ mod \ p_i$ canais percorrendo esta lista, onde p_i é o menor número primo maior que o número de canais. Após $2 \times p_i$ *slots*, um novo valor de r_i é escolhido aleatoriamente dentro do intervalo $[0, p_i)$ [Guedes et al. 2012].

Em [Bian et al. 2011], é apresentado um esquema baseado em sistema de *quoruns* para construir a sequência de salto de canais. O objetivo é aumentar o número de sobreposições entre as múltiplas sequências. Um *quorum* é definido como um elemento de um sistema S que satisfaz a propriedade de interseção: $p \cap q \neq \emptyset$, $\forall p, q \in S$ [Lo 2011]. Por exemplo, a partir de $Z = \{0, 1, 2\}$ pode-se formar o conjunto de *quoruns* $S = \{\{0, 1\}, \{0, 2\}, \{1, 2\}\}$ e cada elemento de S dá origem a uma sequência, conforme a Figura 1. Assim temos u, v e w que são sequências que apresentam sobreposição de canais entre elas e, cada sequência pode ser atribuída a um nó como uma sequência de salto de canais. Na figura, os campos em branco representam o caso em que o canal será atribuído aleatoriamente.

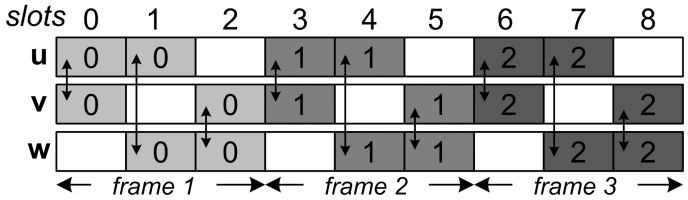


Figura 1. Exemplo de sequências a partir do conjunto de *quoruns*.

Outro algoritmo de RV é o ETCH (*Efficient Channel Hopping for Communication Rendezvous*)[Zhang et al. 2011]. Este algoritmo consiste de três partes: escalonamento, atribuição de canais e execução da sequência. Para conseguir uma melhor média de TTR, o ETCH constrói um conjunto de $2N$ sequências, contendo $2N - 1$ *slots*, onde N é o número de canais, de forma que cada sequência apresente sobreposição de canais com as demais em diferentes *slots*. Para isso, as sequências são construídas em dois passos, no primeiro passo, ele gera $2N - 1$ escalonamentos e, no segundo passo, é feita a atribuição de canais. Ao fim do processo, todos os nós dispõem de um conjunto

de sequências iguais. Na execução, primeiramente, o nó seleciona aleatoriamente uma sequência que irá seguir, depois de passar por todos os *slots* desta sequência, ele realiza uma seleção aleatória entre as sequências restantes, e passa a utilizar esta sequência recém escolhida [Zhang et al. 2011].

Um problema, mencionado em [Theis et al. 2011], é que grande parte dos algoritmos de RV não aborda a questão de transmissões em *broadcast*, ou melhor, o caso do *rendezvous* entre múltiplos usuários. Uma solução pode ser encontrada em [Lin et al. 2011], onde inicialmente cada nó possui uma sequência de canais, gerada com parâmetros próprios do algoritmo. A cada encontro dois-a-dois entre os nós, eles comparam seus parâmetros e um dos nós passa a utilizar a sequência do outro. Para esta comparação, a solução define um conjunto de critérios de desempate dois-a-dois entre os parâmetros. Desta forma, ao fim de alguns encontros, os nós convergem para uma sequência única, e assim, podem enviar mensagens em *broadcast* usando essa sequência. Uma dificuldade é que o tempo de convergência pode ser grande, uma vez que a ordem em que os encontros ocorrem influencia no processo de desempate. Além disso, o uso de uma única sequência cria o problema de múltiplos acessos ao meio compartilhado.

Algoritmo 1: Seleção de sequência de *Broadcast*.

input: $RADIOS$, $nó - broadcaster$, $SEQSIZE$, $CANAIS$, $SEQ(canal, slot)$, $P_{idle}(nó, canal)$
init $sequence = \{\ \}$
init $reachedNodes = \emptyset$
for *(nó in* $RADIOS$*)* **do**
  init $P_{idle}(nó)acumulada = 0.0$
while $(|sequence| < SEQSIZE)$ **do**
  selecione canal C onde $nó - broadcaster$ alcança o maior número esperado de vizinhos $\notin reachedNodes$
  for *(nó in vizinhos alcançados)* **do**
    $P_{idle}(nó)acumulada = P_{idle}(nó)acumulada + P_{idle}(nó, C)$
    if $(P_{idle}(nó)acumulada >= 1.0)$ **then**
      adicione $nó$ a $reachedNodes$
  acrescente C a $sequence$
  if $(reachedNodes == RADIOS)$ **then**
    $reachedNodes = \emptyset$
retorna $sequence$

Para evitar o problema trazido pelo uso da sequência única, em [Guedes et al. 2012], propomos uma solução que consiste em definir uma sequência especial, específica para cada nó, para a transmissão de pacotes em *broadcast*. Esta sequência pode ser utilizada em conjunto com qualquer algoritmo de RV e tem como objetivo permitir o encontro com o maior número de vizinhos em poucos *slots*. A proposta considera ainda estimativas de uso dos canais pelos UPs para evitar canais pouco disponíveis. O Algoritmo 1 proposto para a construção da sequência de *broadcast* utiliza como entradas as sequências dos vizinhos, que podem ser recriadas depois da troca de parâmetros do algoritmo de RV em eventuais encontros, e também a estimativa

de utilização dos canais. Esta estimativa da atividade de cada canal, mais precisamente a probabilidade de um canal estar livre da atividade de usuários primários, é indicada como P_{idle} no algoritmo. No trabalho, comprovou-se através de simulações a eficiência do uso dessa sequência, num cenário de único salto, em relação ao uso de múltiplos *rendezvous* dois-a-dois.

Neste artigo, utilizamos a ideia do nó dispor de mais de uma sequência e, a partir de um esquema de papéis, decidir qual sequência utilizar. Assim, o modelo adotado neste trabalho assume uma topologia de múltiplos saltos onde os nós enviam pacotes de dados em *unicast* e de controle em *broadcast*. Maiores detalhes sobre o modelo e a proposta do uso de papéis serão apresentados respectivamente nas duas próximas seções.

3. Modelo do Sistema

No modelo de rede de rádios cognitivos adotado neste trabalho, consideramos um cenário com N_s usuários secundários (rádios cognitivos) e N_p usuários primários (rádios licenciados), ambos utilizando uma faixa do espectro licenciado composta por C canais. Os usuários secundários acessam os C canais utilizando técnicas de salto de canais, onde o tempo é dividido em *slots* de duração τ. Durante cada *slot*, um único canal c_i ($i = 1, 2, ..., N$) é utilizado pelo usuário secundário, o qual mantém sua única interface de rádio sintonizada neste canal. A duração τ de um *slot* é dada pelo tempo necessário para a transmissão de um pacote de dados seguido pela mensagem de confirmação (*Ack*). As mensagens de dados enviadas pelos usuários secundários possuem tamanho fixo.

Neste modelo, assumimos que os UPs apenas mudam de estado nas transições entre os *slots* e não trocam de canal durante sua operação. Assim, modelamos o padrão de atividade de cada canal $c_i \in C$ de acordo com um modelo de transição de dois estados, como apresentado na Figura 2. Os estados `ON` e `OFF` representam respectivamente os períodos de tempo em que o canal está ocupado (UPs em atividade) ou livre (UPs ociosos). Este modelo `ON-OFF` é amplamente adotado na literatura e é uma boa aproximação para muitos cenários e tecnologias legadas [Wellens et al. 2009].

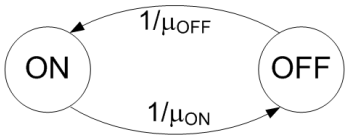


Figura 2. Modelo de atividade nos canais.

Os canais utilizados em cada *slot* de tempo pelo usuário secundário i são representados pela sequência de saltos de canal $S_i = s_i(1), s_i(2), ..., s_i(M)$, onde M é o tamanho da sequência. Assumimos que os USs estão sincronizados, ou seja, as trocas de canal são simultâneas e as sequências de saltos estão alinhadas. Quando o canal c_k usado na posição t da sequência de saltos de dois USs i e j é o mesmo ($s_i(t) = s_j(t) = c_k$), a comunicação entre estes USs se torna possível desde que não haja nenhum primário ativo naquele mesmo canal. De maneira genérica, n USs podem estar sintonizados no canal c_k no *slot* t ($s_1(t) = s_2(t) = ... = s_n(t) = c_k$).

A fim de evitar colisões de pacotes nos *slots*, assumimos que os USs utilizam um mecanismo de controle de acesso ao meio do tipo CSMA/CA (*Carrier Sense Multiple*

Access with Collision Avoidance). Assim, no início de cada *slot*, os USs que desejam transmitir devem escutar o meio livre por um intervalo de tempo escolhido aleatoriamente. Caso o meio permaneça livre durante todo o intervalo, a transmissão é realizada. Caso outra transmissão seja detectada durante este intervalo, o US adia a transmissão do pacote para o próximo *slot*, onde o transmissor novamente disputará o acesso ao meio.

No caso de transmissões de pacotes *unicast*, o US transmissor aguarda por um reconhecimento. Caso o reconhecimento não chegue, o US disputará novamente o meio nos próximos *slots* para realizar novas tentativas de retransmissão do pacote, até que um certo número de tentativas seja atingido. Neste caso, o pacote será descartado. Nenhum controle de erro é usado na transmissão de pacotes em *broadcast*. Em resumo, utilizando este modelo CSMA/CA em *slots*, os USs disputam o acesso aos C canais para a transmissão de pacotes em *unicast* e *broadcast*. Para a transmissão destes dois tipos de mensagem, os USs comutam entre papéis, de acordo com o funcionamento descrito na próxima seção.

4. Proposta

O uso do mecanismo de saltos de canais pelos USs consiste em todos os nós saltarem entre canais, evitando aqueles canais ocupados pelos UPs, seguindo uma sequência de saltos definida por um algoritmo de RV. Toda vez que ocorre um encontro entre dois nós, mensagens de controle ou de dados podem ser trocadas entre eles. Como explicado anteriormente, a maioria dos algoritmos de RV propostos na literatura não tratam do problema da transmissão simultânea para múltiplos usuários, denominada neste trabalho de transmissão em *broadcast*. Assim, quando um nó deseja enviar uma mensagem a todos os seus vizinhos, esta transmissão é feita nó-a-nó, a partir de sucessivos *rendezvous*. Este procedimento leva a grandes atrasos na disseminação da informação, prejudicando o funcionamento da rede.

Como mencionado na Seção 2, um solução para esse problema é apresentada em [Lin et al. 2011]. Nessa proposta, os nós convergem para uma sequência única, facilitando assim o envio de mensagens em *broadcast*. Essa solução, doravante denominada de solução intuitiva, tem como vantagem permitir o envio imediato da mensagem a todos os nós da vizinhança, aumentando a taxa de entrega dessas mensagens. No entanto, o uso dessa sequência única se aplicada a todas as transmissões *unicast* da rede provoca um problema de contenção no acesso ao meio em todos os canais da sequência, o que limita a capacidade da rede. Uma forma de se evitar esse problema seria os nós negociarem, usando o canal de controle fornecido pela sequência única, o canal ou sequência de canais a ser utilizado(a) para as transmissões *unicast* entre dois nós. No entanto, conforme a carga de mensagens *unicast*, a quantidade de negociações no canal de controle pode ser grande, gerando o problema conhecido como gargalo do canal de controle. No caso de um nó com função de roteador, negociações podem ser necessárias pacote-a-pacote.

A proposta neste artigo consiste na definição de uma política de papéis/estado, no qual cada nó mapeia o uso de uma sequência diferente em decorrência do seu objetivo. Desta forma, definimos três possíveis estados para o nó: repouso ou modo de recepção (RX), TX *unicast* e TX *broadcast*. De acordo com o estado do nó, ele decide por uma sequência mais apropriada que evite ou cause menos competição no meio. Assim, o nó utiliza a sequência denominada sequência de repouso quando não possui nada a enviar, ou seja, em modo de repouso. A sequência de repouso é aquela gerada pelo algoritmo de

RV utilizado. Quando o nó deseja enviar uma mensagem de *broadcast*, ele utiliza a sua sequência de *broadcast*, definida conforme [Guedes et al. 2012], e quando deseja enviar uma mensagem *unicast* o nó transmissor chaveia para a sequência de repouso do nó de destino.

A intenção desta estratégia baseada em papéis é aliviar o acesso ao meio e possibilitar transmissões em paralelo, através da atribuição de sequências distintas. Pois diferentemente do caso de uma única sequência, cada nó utilizará uma sequência de *broadcast* própria originada a partir de sua vizinhança. Outra vantagem é de não criar a necessidade de uma negociação prévia pacote-a-pacote para uma transmissão *unicast*, pois na necessidade de envio de uma mensagem o nó transmissor chaveia para a sequência de repouso do nó receptor desejado, sequência esta esperada e válida para o modo de recepção. Esta abordagem reduz a necessidade de troca de informações, mensagens de controle, entre os nós para um agendamento de alguma transmissão futura.

Nas duas próximas seções, apresentaremos o ambiente de simulação utilizado e os resultados numéricos de comparação do desempenho da nossa proposta e da solução intuitiva, respectivamente.

5. Ambiente de Simulação

O simulador utilizado para a avaliação e comparação das soluções foi o *ns-2* [NS-2], sendo necessário, para isto, implementar um conjunto de extensões a esse simulador. Primeiramente, um modelo de interferência físico baseado na SINR (*Signal-to-Interference-plus-Noise Ratio*) [Brar et al. 2006] foi implementado para ser usado com a camada MAC 802.11 padrão do simulador. Este modelo permite computar a interferência acumulada sofrida por uma transmissão, com a qual é possível calcular a SINR de um determinado quadro e, por conseguinte, a probabilidade dele ser recebido com sucesso.

Em seguida, a camada MAC 802.11 foi modificada para permitir que os nós trocassem de canal a cada *slot*, segundo diferentes sequências de saltos, e se comportassem de acordo com o modelo CSMA/CA em *slots*, como detalhado na Seção 3. Para a geração das sequências de saltos, diversos algoritmos de RV (*rendezvous*) foram implementados (*quorum*, ETCH, *mclock*, etc.), incluindo o algoritmo proposto em [Guedes et al. 2012] para a geração da sequência de *broadcast*. Implementou-se também um novo agente gerador de tráfego em *broadcast*, que gera pacotes periódicos com intervalos de tempo $\mu \pm jitter$. Os pacotes em *broadcast* gerados são recebidos pelo próprio agente, o qual coleta e armazena estatísticas por vizinho a respeito dos pacotes recebidos.

O cenário utilizado nos experimentos foi de redes em malha sem fio de múltiplos saltos. Para a geração das topologias foi utilizado um modelo de posicionamento que garante a conectividade entre os nós, evitando que o grau máximo entre todos os nós ultrapasse um determinado valor. Assim, seguindo esse modelo, um nó ao ser disposto aleatoriamente em uma determinada área, somente será mantido se estiver dentro da área de cobertura de pelo menos um outro nó e obedecer o grau máximo de vizinhança.

Os usuários primários presentes nos cenários simulados têm a atividade regida por um modelo `ON-OFF`, assim como descrito na Seção 3. O tempo médio de permanência nos estados `ON` e `OFF` eram dados por variáveis aleatórias exponenciais com médias μ_{ON} e μ_{OFF}, respectivamente. Os UPs simulados possuíam um raio de influência, denominado

R_I. Os USs dentro do raio de influência de um UP ficam impossibilitados de transmitir e receber pacotes naquele canal utilizado pelo UP enquanto o mesmo encontra-se no estado `ON`. Vale destacar que neste modelo, a probabilidade do canal estar livre para um US sobre a influência de um UP é dada por $p_{Idle} = {\mu_{OFF}}/{(\mu_{ON}+\mu_{OFF})}$.

Todos os nós da rede em malha sem fio geram tráfego *broadcast* de acordo com uma determinada carga. Vale ressaltar que esse tráfego é de apenas um salto, ou seja, ele é gerado a partir de um nó para todos os seus vizinhos, não sendo reencaminhado por esses nós. Da mesma forma, o tráfego *unicast* é gerado por todos os nós, tendo como destino um dos vizinhos de cada nó, o qual é escolhido aleatoriamente. Esse tráfego é do tipo CBR cuja carga pode ser controlada pela variação do intervalo entre a geração dos pacotes. Como todo o tráfego gerado é de apenas um único salto, os resultados não são influenciados pelo protocolo de roteamento utilizado, mas apenas pela carga imposta à rede, a qual é variada durante a avaliação de desempenho para ambos os tráfegos.

6. Resultados de Simulação

Em todas as simulações realizadas, 25 nós são dispostos em uma área quadrada de 1000 metros de lado, seguindo o modelo de posicionamento descrito na seção anterior. O grau máximo utilizado para a geração das topologias foi igual a 6. Ainda, um UP para cada canal disponível é disposto aleatoriamente dentro dessa área, o qual tem uma raio de influência R_I igual a 250 metros. O modelo `ON-OFF` de atividade dos UPs é parametrizado pela fixação do parâmetro μ_{OFF} em 20 *slots* e a posterior escolha aleatória da probabilidade de cada primário estar ocioso (p_{Idle}) dentro do intervalo $(0, 1)$. Com estes valores é possível calcular o parâmetro μ_{ON} de cada UP através da fórmula apresentada na seção anterior.

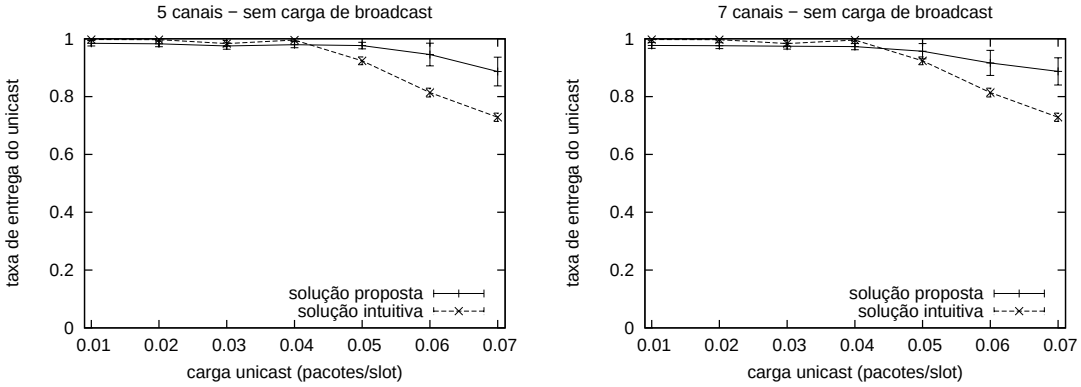
Cada experimento é executado por 50000 *slots* de tempo e as métricas avaliadas foram: a taxa de entrega do tráfego *unicast*, e a taxa de entrega e o atraso, em número de *slots*, do tráfego *broadcast*. Essas métricas de desempenho são mostradas em função da carga *unicast* gerada na rede e do número de canais. Cada valor mostrado nos gráficos equivale a média de 30 rodadas e as barras de erro correspondem ao intervalo de confiança calculado com um nível de confiança de 95%.

As sequências de saltos de canais utilizadas na solução proposta foram atribuídas a partir do algoritmo de RV *mclock* [Theis et al. 2011]. Obtivemos resultados também para o algoritmo ETCH que apresentou melhor taxa de entrega em *broadcast* que o *mclock*, porém a taxa de entrega *unicast* empatou com a solução intuitiva.

A carga de tráfego *unicast* é mostrada na abscissa de todos os gráficos em termos do número de pacotes por *slot*, o qual é variado entre 0.01 e 0.07. A carga do tráfego *broadcast*, μ, assume três valores distintos: 0 (sem carga), 500 (alta carga) e 1000 (média carga). Esses valores correspondem ao intervalo entre geração de pacotes *broadcast*, em número de *slots*. O valor do $jitter$ somado ao intervalo μ é sorteado no intervalo $[-0.05\mu, +0.05\mu]$.

Os gráficos da Figura 3 mostram a taxa de entrega do tráfego *unicast* para sequências de 5 e 7 canais quando nenhum tráfego *broadcast* é gerado. Os resultados demonstram que a solução proposta apresenta um desempenho superior com o aumento da carga, pois consegue aliviar o compartilhamento no meio pelo fato de cada nó utilizar diferentes sequências de canais a cada instante. Essas sequências correspondem às

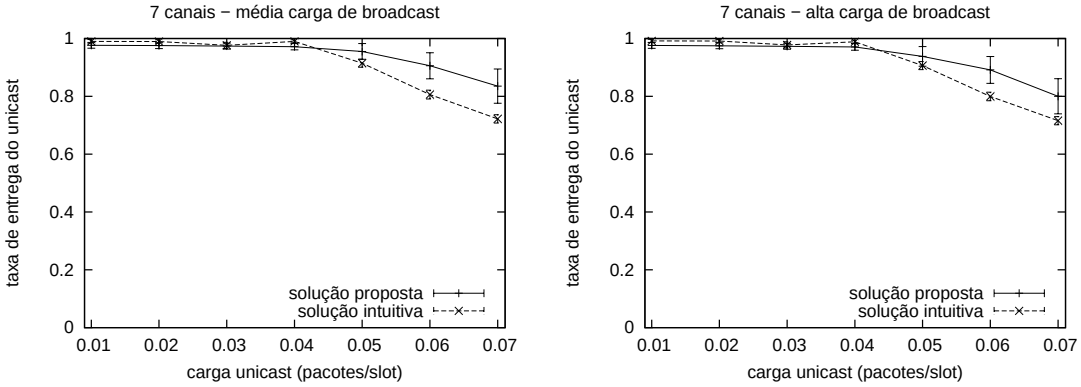
sequências de repouso de seus destinatários. Esse ganho ocorre mesmo quando simultaneamente o receptor chaveia de sequência por ser também um transmissor. O aumento do número de canais de 5 (Fig. 3(a)) para 7 (Fig. 3(b)) tem pouca influência nesse resultado.



(a) 5 canais (b) 7 canais

Figura 3. Taxa de Entrega do Tráfego *Unicast* Sem Carga de *Broadcast*

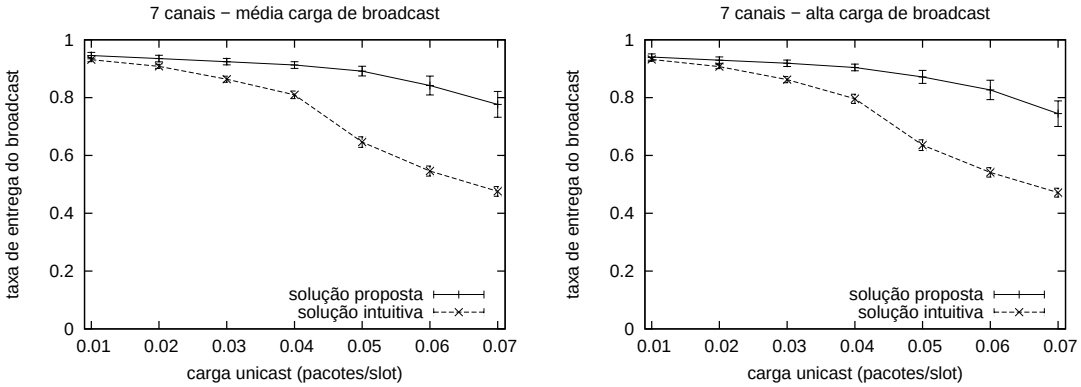
Nos próximos dois conjuntos de resultados (Figuras 4 e 5), a influência do tráfego *broadcast*, em conjunto com o tráfego *unicast*, é analisada. Com relação ao tráfego *unicast* (Figura 4), podemos notar que o acréscimo do tráfego *broadcast* tem um impacto reduzido na taxa de entrega do primeiro. No entanto, podemos ver que a solução proposta é a mais afetada. Isso se deve ao fato de uma única mensagem *broadcast* ter de ser retransmitida em todos os canais da sequência de *broadcast*, o que não acontece no caso da solução intuitiva. Esta característica também afeta a recepção de pacotes *unicast*, já que vizinhos de um US tem maior dificuldade de encontrá-lo em sua sequência de repouso quando ele está utilizando a sequência de *broadcast*. No entanto, de maneira geral a solução proposta continua sendo superior à solução intuitiva nessa métrica pelo fato de diminuir o compartilhamento do meio nos diversos canais, aumentando a capacidade da rede em escoar o tráfego gerado.



(a) 7 canais - média carga (b) 7 canais - alta carga

Figura 4. Taxa de Entrega do Tráfego *Unicast* com Carga de *Broadcast*

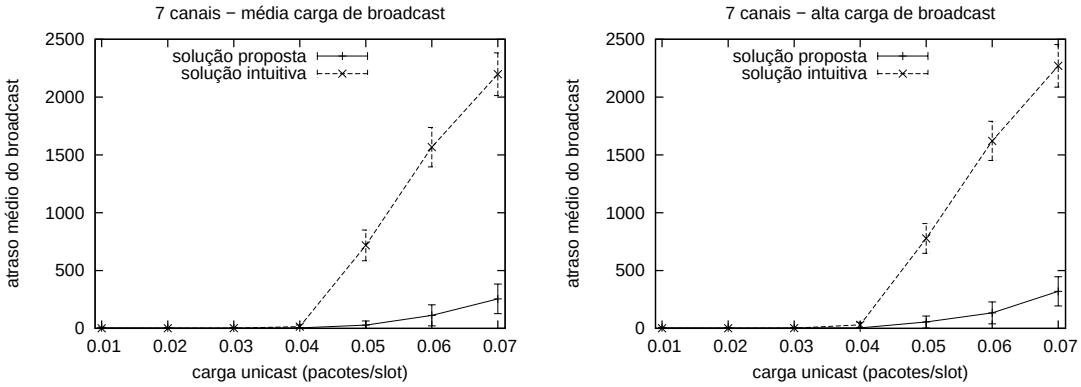
Pelos gráficos das Figuras 6(a) e 6(b), podemos verificar que apesar da solução intuitiva permitir que todos os seus vizinhos sejam alcançados por uma única transmissão,



(a) 7 canais - média carga (b) 7 canais - alta carga

Figura 5. Taxa de Entrega do Tráfego *Broadcast*

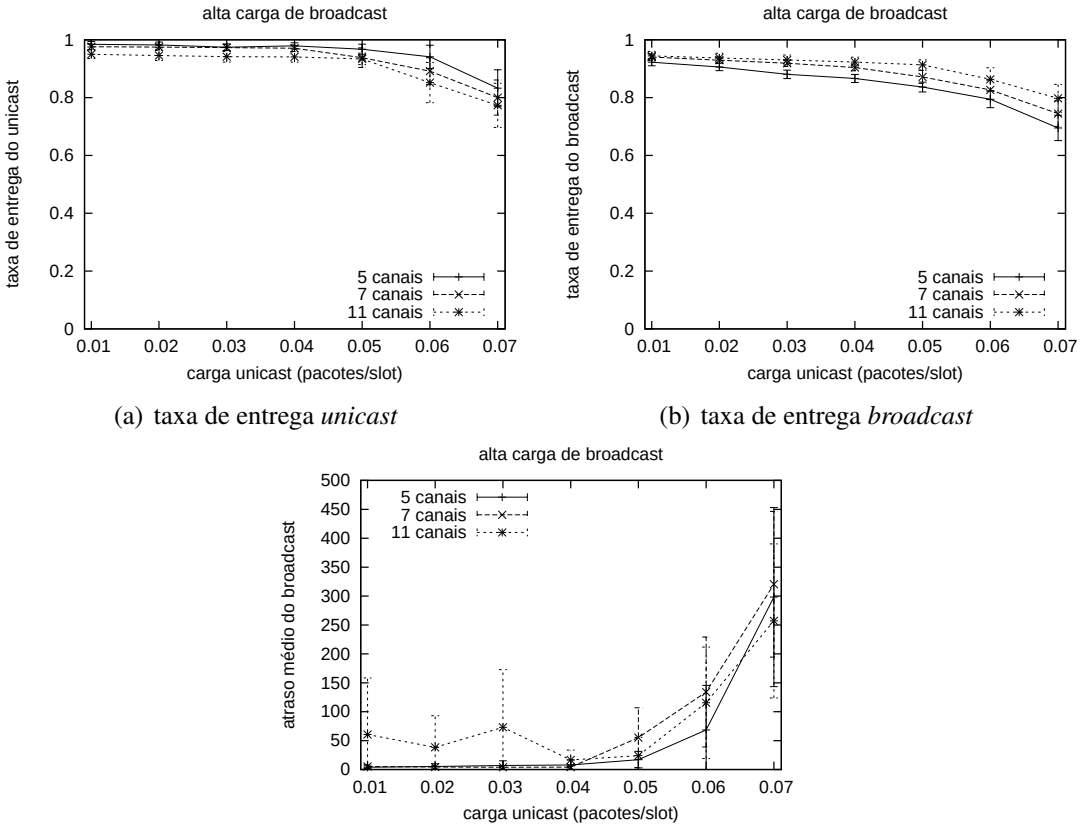
ela apresenta um pior desempenho em termos do atraso médio das mensagens *broadcast*. Quando a carga *unicast* ultrapassa um determinado valor, em torno de 0.04 pacotes/*slot* por nó, o atraso médio dessas mensagens cresce de forma acentuada, assim como a sua taxa de entrega, como evidenciado pelas Figuras 5(a) e 5(b). Portanto, a solução proposta favorece o tráfego *broadcast* em situações de alta carga na rede, aumentando a taxa de entrega e diminuindo o atraso médio na entrega das mensagens.



(a) 7 canais - média carga (b) 7 canais - alta carga

Figura 6. Atraso Médio do Tráfego *Broadcast*

Os gráficos da Figura 7 mostram as três métricas de desempenho para a solução proposta em função da carga *unicast* com um número crescente de canais. Desses gráficos podemos notar que quanto maior o número de canais, maior é a taxa de entrega do tráfego *broadcast*. No entanto, esse aumento no número de canais afeta negativamente a taxa de entrega do tráfego *unicast*, ou seja, a capacidade disponível na rede. A explicação desse fenômeno é a mesma dos resultados das Figuras 4 e 5. Com o aumento do número de canais, maior será o tamanho da sequência de *broadcast*, e mais tempo o US permanecerá fora da sequência de repouso e terá dificuldade em receber pacotes *unicast*. Desta forma, existe um compromisso com relação ao número de canais a serem utilizados nas sequências de saltos da solução proposta.



(a) taxa de entrega *unicast*

(b) taxa de entrega *broadcast*

(c) atraso médio *broadcast*

Figura 7. Influência do Número de Canais

7. Conclusão

O rádio cognitivo é uma tecnologia que permite um tipo de rede definido como DSA. Devido as suas regras de funcionamento, uma abordagem favorável é o uso de sequência de saltos de canais para acesso ao meio. Porém isso dificulta o encontro entre os nós, assim, diversos algoritmos de *rendezvous* foram criados. No entanto nestes algoritmos, as transmissões ou trocas de mensagens para múltiplos destinatários não é bem explorada. Um exemplo disso, são as transmissões de pacotes em *broadcast*, que acabam sendo inadequadas ou ineficientes em face aos procedimentos de saltos de canais.

Deste modo, propomos uma forma de acesso ao meio que se vale de algum algoritmo de *rendezvous* e da criação de uma sequência de *broadcast*. Nesta proposta, a partir de um critério de tipo de mensagem e/ou destinatário cria-se uma abordagem por papéis que define diferentes sequências de saltos de canais a fim de facilitar o compartilhamento do meio. Esta política de acesso foi comparada ao caso mais intuitivo de uso de apenas um canal ou uma mesma sequência de canais, que a princípio é uma boa opção, porém não atende bem a situações onde o acesso ao meio é muito exigido.

A partir dos resultados pode-se constatar que a política de saltos de canais empregada demostrou uma melhora na taxa de entrega de mensagens, seja de *broadcast* quanto de *unicast*, ao mesmo tempo em que gerou uma redução no atraso médio de recepção das mensagens para múltiplos usuários.

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 Int. J. of Comp. and Telecom. Networking*, 50:2127–2159.

Bahl, P., Chandra, R., and Dunagan, J. (2004). SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proceedings of ACM MobiCom*, pages 216–230, New York, NY, USA. ACM.

Bian, K., Park, J.-M., and Chen, R. (2011). Control channel establishment in cognitive radio networks using channel hopping. *IEEE Journal on Selected Areas in Communications*, 29(4):689–703.

Brar, G., Blough, D. M., and Santi, P. (2006). Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks. In *Proceedings of ACM MobiCom*, pages 2–13, New York, NY, USA. ACM.

Cormio, C. and Chowdhury, K. R. (2010). Common control channel design for cognitive radio wireless ad hoc networks using adaptive frequency hopping. *Ad Hoc Networks*, 8(4):430–438.

DaSilva, L. A. and Guerreiro, I. (2008). Sequence-based rendezvous for dynamic spectrum access. In *Proceedings of IEEE DySPAN*, pages 1 –7.

Guedes, R. M., da Silva, M. W. R., Coutinho, P. S., and de Rezende, J. F. (2012). Agnostic broadcast rendezvous for cognitive radio networks using channel hopping. In *Proceedings of IEEE LCN*, pages 647–654.

Lin, Z., Liu, H., Chu, X., and Leung, Y.-W. (2011). Jump-stay based channel-hopping algorithm with guaranteed rendezvous for cognitive radio networks. In *Proceedings of IEEE INFOCOM*, pages 2444–2452.

Lo, B. F. (2011). A survey of common control channel design in cognitive radio networks. *Elsevier Physical Communication*, 4:26–39.

Luo, T., Motani, M., and Srinivasan, V. (2006). Cam-mac: A cooperative asynchronous multi-channel mac protocol for ad hoc networks. In *Proceedings of IEEE BROADNETS*, pages 1 –10.

Mo, J., So, H.-S. W., and Walrand, J. (2005). Comparison of multi-mhannel mac protocols. In *Proceedings of ACM MSWiM*, pages 209–218, New York, NY, USA. ACM.

NS-2. The network simulator - ns-2. `http://www.isi.edu/nsnam/ns/` - último acesso em 07/12/2012.

Silvius, M. D., Ge, F., Young, A., MacKenzie, A. B., and Bostian, C. W. (2008). Smart radio: Spectrum access for first responders. In *Proceedings of SPIE Defense and Security Conference*.

So, J. and Vaidya, N. H. (2004). Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *Proceedings of ACM MobiHoc*, pages 222–233, New York, NY, USA. ACM.

Theis, N. C., Thomas, R. W., and DaSilva, L. A. (2011). Rendezvous for cognitive radios. *IEEE Transactions on Mobile Computing*, 10(2):216–227.

Wellens, M., Riihijärvi, J., and Mähönen, P. (2009). Empirical time and frequency domain models of spectrum use. *Physical Communication*, 2(1-2):10–32.

Zhang, Y., Li, Q., Yu, G., and Wang, B. (2011). ETCH: Efficient channel hopping for communication rendezvous in dynamic spectrum access networks. In *Proceedings of IEEE INFOCOM*, pages 2471 –2479.

Uma avaliação do uso de mecanismos de custódia compartilhada em redes tolerantes a atrasos e desconexões

Ely S. Miranda [1,2], Juliano F. Naves[2,3], Igor M. Moraes[3], Pedro B. Velloso[3]

[1]Instituto Federal do Piauí

[2]Instituto Federal de Rondônia

[3]Instituto de Computação-PGC
Universidade Federal Fluminense

ely.miranda@ifpi.edu.br,{jfischer,igor,velloso}@ic.uff.br

Resumo. *Este artigo avalia o uso da transferência de custódia entre nós para aumentar o desempenho em redes tolerantes a atrasos e desconexões. Para isso, dois mecanismos são considerados. O primeiro, chamado de LJC (*Limited Joint Custody*), propõe o uso da custódia compartilhada e implementa um esquema de replicação controlada para limitar o número de custódias por agregado na rede. O segundo, chamado de FCF (*Forward Custody First*), é uma política de encaminhamento que prioriza os agregados sob custódia. Através de simulações, o desempenho destes mecanismos é avaliado para diferentes protocolos de roteamento em dois cenários distintos baseados em registros reais de mobilidade. Os principais resultados mostram que ambos os mecanismos aumentam a taxa de entrega e também reduzem o atraso nos dois cenários analisados com menor sobrecarga. A combinação dos mecanismos FCF-LJC quando comparada a mecanismos que não usam custódia proporciona um aumento de até 94% na taxa de entrega e reduz em até 23% e 73% o atraso de entrega e sobrecarga de controle, respectivamente.*

Abstract. *This paper evaluates the use of custody transfer between nodes to improve performance in delay tolerant networks. Two mechanisms are considered in the analysis. The first one, named LJC (Limited Joint Custody), proposes the use of joint custody and implements a controlled replication scheme to limit the number of custodians per bundle in the network. The second one, named FCF (Forward Custody First), is a forwarding policy that prioritizes bundles in custody. Through simulations, the performance of these mechanisms is evaluated for different routing protocols in two different scenarios based on real traces. Main results show that both mechanisms increase the delivery rate and also reduce the delay in both scenarios with lower overhead. The combination of FCF-LJC mechanisms, compared to implementations with no custody, provides up to 94% higher delivery rate and decreases by 23% and 73% the delivery delay and control overhead, respectively.*

1. Introdução

As redes tolerantes a atrasos e desconexões (*Delay/Disruption Tolerant Networks* - DTNs) vêm recebendo cada vez mais atenção da comunidade acadêmica [Khabbaz et al. 2012, Oliveira et al. 2007], especialmente com o aumento do interesse por redes veiculares, redes oportunistas e redes de sensores acústicas subaquáticas.

Essas redes são caracterizadas por operarem com conectividade intermitente, o que pode causar longos atrasos e baixas taxas de entrega de mensagens [Fall 2003]. Como resultado, a tradicional pilha de protocolos TCP/IP é pouco eficiente neste cenário, pois nela assume-se que há sempre um caminho fim-a-fim entre a origem e o destino. Assim, o princípio básico das redes DTN é adotar o paradigma armazena-carrega-e-encaminha (*store-carry-and-forward*) para lidar com o problema da conectividade intermitente. Nesse paradigma, os nós encapsulam as mensagens em agregados e são dotados de *buffers*. Assim, os nós mantêm esses agregados em seu *buffer* durante períodos sem conectividade e tentam encaminhá-los quando um contato é estabelecido com um nó vizinho de acordo com uma métrica definida pelo protocolo de roteamento. Além disso, encaminhar um agregado significa enviar uma cópia desse agregado e não descartá-lo no nó encaminhador. Neste contexto, um agregado é removido do *buffer* quando o seu tempo de vida expira, ele chega a seu destino ou quando é descartado por políticas de descarte disparadas quando o *buffer* atinge certo nível de ocupação [Naves et al. 2012].

Muitos mecanismos de roteamento e encaminhamento para redes DTN foram propostos recentemente [Khabbaz et al. 2012, Fall e Farrell 2008]. Eles abordam diversas questões como, por exemplo, políticas de encaminhamento e descarte, conhecimento prévio de trajetórias de nós e previsão de contatos. Um mecanismo simples, porém pouco explorado, é a transferência de custódia [Fall et al. 2003]. Basicamente, esse mecanismo consiste em atribuir a custódia de um agregado a um nó específico. Consequentemente, esse agregado não pode ser descartado prematuramente por políticas de descarte. Trabalhos anteriores tentam aumentar a confiabilidade da rede usando mecanismos de custódia exclusiva. No entanto, poucos esforços foram dedicados a pesquisas usando mecanismos de custódia compartilhada e a políticas de encaminhamento baseadas em custódia para aumentar a probabilidade de entrega e reduzir o atraso na entrega dos agregados. Assim, em um primeiro esforço, Miranda *et al.* propõem os mecanismos LJC (*Limited Joint Custody*) e FCF (*Forward Custody First*) [Miranda et al. 2012]. O LJC é um mecanismo de custódia compartilhada que limita o número de custódias por agregado na rede. O FCF, por outro lado, é uma política de encaminhamento que prioriza o encaminhamento de agregados sob custódia. No entanto, os autores apresentam uma análise preliminar avaliando apenas os mecanismos propostos em um protocolo epidêmico. Portanto, o principal objetivo deste artigo é avaliar, através de simulações, o comportamento e o impacto do uso dos mecanismos LJC e FCF em protocolos mais eficientes propostos na literatura. Para a avaliação, foram escolhidos um protocolo de replicação probabilística, representado pelo PRoPHET [Lindgren et al. 2003], e outro de replicação controlada, representado pelo Spray and Wait [Spyropoulos et al. 2005]. A análise considera o desempenho dos mecanismos em relação à taxa de entrega, ao atraso médio de entrega e à sobrecarga de mensagens, usando dois registros reais de mobilidade. Além disso, avaliou-se o impacto dos mecanismos para diferentes volumes de tráfego. O objetivo é verificar o comportamento do uso de mais de um nó custódio por agregado em situações nas quais o transbordamento de *buffer* é mais frequente. Os principais resultados mostram que os mecanismos LJC e FCF combinados obtêm taxas de entrega superiores em até 94% quando comparados a implementações que não usam custódia. Além disso, reduzem o atraso de entrega em até 23% devido à priorização no encaminhamento de agregados sob custódia. Os mecanismos avaliados também apresentam uma sobrecarga de controle menor, chegado a reduzir em até 73% o número de cópias transmitidas para cada agregado

entregue. Por fim, o resultado mais interessante mostra que a combinação FCF-LJC sobre o protocolo PRoPHET, consegue a melhor eficiência em todas as métricas analisadas, tornando-se a melhor configuração possível dentre as avaliadas.

O restante desse trabalho está organizado como a seguir. A Seção 2 apresenta os trabalhos relacionados à esta pesquisa. A Seção 3 detalha o mecanismo de custódia compartilhada e a política de encaminhamento baseada em custódias. Na Seção 4, são descritos os cenários e o ambiente de simulação nos quais os mecanismos foram avaliados. A Seção 5 apresenta e discute os resultados obtidos nas simulações. Por fim, a Seção 6 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Vários trabalhos abordam os problemas de roteamento e encaminhamento em redes DTNs e têm como objetivo aumentar a taxa de entrega de mensagens [Cao e Sun 2012, Spyropoulos et al. 2010]. No entanto, apenas alguns pesquisadores exploram o uso da transferência de custódia para atingir tal objetivo. Inicialmente, os mecanismos de transferência de custódia foram introduzidos para aumentar a confiabilidade em DTNs. Fall *et al.* propõem um mecanismo chamado de custódia exclusiva, no qual definem que originalmente apenas um nó, chamado de nó custódio, é responsável por armazenar um agregado [Fall et al. 2003]. O foco deste artigo, no entanto, é o ganho de desempenho que pode ser obtido com a transferência de custódia. Chuah *et al.*, por exemplo estudam um mecanismo de transferência de custódia para aumentar a taxa de entrega de protocolos de roteamento em DTNs [Chuah et al. 2006]. Mais especificamente, eles concluem que o uso da transferência de custódia pode aumentar significativamente a taxa de entrega em redes esparsamente conectadas com taxas de sobrecarga aceitáveis. Contudo, esse não trabalho aborda a transferência de custódia compartilhada. Essa ideia é mencionada originalmente por Fall *et al.*, que sugerem o uso de mais de um nó custódio por agregado [Fall et al. 2003]. No entanto, eles não propõem qualquer mecanismo específico e não avaliam o impacto do uso de custódia compartilhada no desempenho da rede. Este artigo analisa um mecanismo de custódia compartilhada que usa um esquema de replicação controlada e avalia seu desempenho em relação a outras implementações. Portanto, esses trabalhos são complementares a este artigo.

Em [Spyropoulos et al. 2010] os autores definem uma taxonomia para protocolos de roteamento em DTNs. Em uma das categorias propostas, chamada de replicação com cópia limitada(*copy-limited replication*), um número de tíquetes de encaminhamento é associado a cada cópia de um agregado. Esses tíquetes definem o número de cópias extras dos agregados que os nós podem replicar e encaminhar. Wang *et al.* propõem um mecanismo de encaminhamento para redes DTN baseado no conceito de replicação controlada [Wang et al. 2012]. No entanto, nenhum desses trabalhos aplicam essa replicação controlada para agregados sob custódia. Este artigo avalia um mecanismo semelhante para controlar o número máximo de custódias por agregados ao invés do número de cópias por agregados. O objetivo é evitar congestionamentos prematuros devido à impossibilidade dos nós descartarem os agregados sob custódia.

Em virtude do paradigma armazena-carrega-e-encaminha inerente às redes DTN, políticas de encaminhamento podem ter um grande impacto sobre o desempenho geral da rede. Soares *et al.* apresentam diversas abordagens para o gerenciamento de

buffer e enfatizam a necessidade de políticas eficientes de encaminhamento e de descarte [Soares et al. 2010]. No entanto, todas essas políticas de encaminhamento não consideram agregados sob custódia. Este trabalho analisa o desempenho de um mecanismo de custódia compartilhada e de uma política que prioriza o encaminhamento de agregados sob custódia para aumentar a taxa de entrega e reduzir o atraso.

3. Política de encaminhamento baseada em custódia compartilhada

Esta seção apresenta um mecanismo de encaminhamento baseado em custódia compartilhada. Primeiramente, é descrito o esquema de custódia compartilhada, chamado *Limited Joint Custody* (LJC), que usa um mecanismo de replicação controlada para limitar o número de agregados sob custódia. Posteriormente, é descrita a *Forward Custody First* (FCF), que é uma política de encaminhamento que prioriza os agregados sob custódia.

3.1. O mecanismo *Limited Joint Custody* (LJC)

O conceito de custódia aplicado a redes DTN, proposto em [Fall et al. 2003], é uma abordagem que pode ser usada para aumentar o desempenho da rede. A ideia básica consiste em atribuir a custódia de um determinado agregado a um nó específico, denominado custódio. Consequentemente, esse nó deve manter no *buffer* todo agregado que está sob sua custódia até que ele alcance seu destino ou seu TTL expire. Assim, evita-se o descarte prematuro de agregados. Nessa abordagem, chamada custódia exclusiva, cada agregado é associado a um único nó. Além disso, quando um nó encaminha um agregado sob custódia, ele pode negociar a transferência da custódia para o próximo nó [Fall et al. 2003]. Neste trabalho, essa negociação é tratada como uma transferência simples em que, se há espaço no *buffer* do nó que receberá a cópia do agregado, ele aceita o pedido de custódia. Uma vez que a transferência é concluída, o agregado torna-se regular para o nó que realizou o encaminhamento.

Mecanismos de custódia compartilhada generalizam o conceito de custódia exclusiva, permitindo que os agregados tenham mais de um nó responsável por sua custódia. Isso significa que uma determinada quantidade de réplicas de um dado agregado pode estar sob custódia de diferentes nós. A ideia fundamental da custódia compartilhada é aumentar o número de réplicas sob custódia visando aumentar a probabilidade deste agregado ser entregue. O mecanismo *Limited Joint Custody* (LJC) usa a custódia compartilhada, porém limita o número de custódias por agregado. Assim, é possível controlar a replicação de um agregado para evitar congestionamentos. Esse controle é feito com um esquema baseado em tíquetes. Assim, no mecanismo LJC, quando um nó cria um agregado, ele recebe automaticamente a custódia deste agregado. Além disso, uma quantidade determinada de tíquetes está associada a este novo agregado sob custódia. O valor inicial do tíquete define o número máximo de custódias por agregado. Portanto, sempre que um nó encaminha um pacote sob custódia para um nó vizinho, ele deve compartilhar seus tíquetes com a nova réplica. No LJC emprega-se um mecanismo de compartilhamento de tíquetes análogo ao usado pelo protocolo de roteamento *Spray and Wait* [Spyropoulos et al. 2005] em sua variação binária. No mecanismo do LJC, o agregado original mantém $\lceil \frac{n}{2} \rceil$ tíquetes enquanto o agregado replicado recebe $\lfloor \frac{n}{2} \rfloor$ tíquetes. Quando a réplica de um agregado contém apenas um tíquete, o nó custódio deve transferir a custódia para o próximo contato, e, como consequência, o pacote no nó custódio anterior torna-se regular. A réplica recebida pelo novo custódio possui apenas um tíquete.

Assim, a partir do momento em que o número de tíquetes de um pacote sob custódia é igual a 1, o LJC passa a funcionar exatamente como o mecanismo de custódia exclusiva, em que a custódia é transferida a cada contato.

3.2. O mecanismo *Forward Custody First* (FCF)

A política de encaminhamento *Forward Custody First* (FCF) dá prioridade total aos agregados sob custódia. Assim, um nó encaminha primeiro os agregados que estão sob sua custódia. Agregados regulares, ou seja, agregados que não estão sob custódia, são encaminhados somente quando não há mais agregados sob custódia. A ideia chave da política FCF é baseada no fato de que agregados sob custódia não podem ser removidos do *buffer* a menos que cheguem ao seu destino ou que o TTL tenha expirado. Do contrário, agregados regulares podem ser descartados por políticas de descarte prematuramente. Assim, encaminhar primeiro um agregado sob custódia pode acelerar o tempo de entrega. Consequentemente, depois de entregue, nós podem remover esse agregado do *buffer*, disponibilizando espaço para novos agregados. Adicionalmente, pode-se aumentar a probabilidade de entrega.

Uma implementação básica dessa política consiste em duas filas isoladas. É necessário observar que em DTNs, quando um nó encaminha um agregado para um nó vizinho, o agregado não é removido do *buffer*, e consequentemente da fila de encaminhamento, a menos que o vizinho seja o destino final. A primeira fila mantém os agregados sob custódia (Fila de Custódias - FC) e a segunda mantém os agregados regulares (Fila de Regulares - FR). Ao estabelecer um contato com um nó vizinho, os agregados da FC são encaminhados primeiramente. Ainda na FC, os agregados com tíquetes de custódia com maior valor têm prioridade. Conforme explicado na seção anterior, o mecanismo LJC estabelece que ao encaminhar um agregado, seus tíquetes são divididos de forma binária. Com essa divisão e dando prioridade no encaminhamento aos agregados com mais tíquetes, a FC passa a ter um comportamento circular, pois um agregado recém-encaminhado tem seus tíquetes divididos e pode ir ao fim da fila. Com isso, obtém-se uma melhor distribuição dos agregados sob custódia na rede. Por fim, os agregados da FR são encaminhados usando o critério FIFO apenas quando a FC está vazia.

Neste artigo, os mecanismos LJC e FCF são utilizados de forma combinada visando analisar os ganhos de desempenho obtidos por cada um. Na Seção 5 os resultados da combinação desses dois mecanismos são apresentados.

4. Cenários de avaliação

Nas análises, considerou-se dois diferentes padrões de mobilidade extraídos de conjuntos de dados reais, denominados Rollernet [Tournoux et al. 2009] e Infocom06 [Hui e Lindgren 2008] . Em ambos os conjuntos de dados, que serão referenciados a partir daqui como cenários, os nós armazenam informações sobre todos os seus contatos. O cenário Rollernet foi coletado durante um circuito de patinação realizado em 2006 em Paris e o Infocom06 foi coletado na conferência IEEE Infocom em 2006. Esses dois cenários foram escolhidos com base nos seus peculiares padrões de mobilidade, os quais são indicados pela média de contatos por hora e o tempo de duração desses contatos, exibidos na Tabela 1. Assim, os mecanismos foram avaliados em um cenário onde os nós encontram-se com mais frequência, mas com pouco tempo de contato, Rollernet, e também em um cenário onde os nós possuem menos contatos, mas com tempos de contato

maiores possibilitando mais tempo para a troca de dados, Infocom06. O objetivo é investigar o desempenho dos mecanismos LJC e FCF em cenários com diferentes características de modo a identificar o impacto do número e tempo de contatos no comportamento de cada um deles.

Em [Miranda et al. 2012] as propostas foram analisadas utilizando-se apenas o protocolo *Epidemic*. Neste trabalho, verificou-se o desempenho dos mecanismos LJC e FCF com os protocolos PRoPHET [Lindgren et al. 2003] e *Spray and Wait* [Spyropoulos et al. 2005]. A escolha desses protocolos deu-se por serem bastante citados na literatura e por representarem respectivamente duas categorias distintas de disseminação de agregados em redes DTN: replicação probabilística e replicação controlada. Visando minimizar o efeito negativo do roteamento *Epidemic* com relação à sobrecarga inerente aos mecanismos de inundação, alternativas foram propostas baseadas em roteamento probabilístico, como é o caso do PRoPHET. Esse protocolo utiliza informações sobre históricos de encontros e, a cada contato, atualiza a probabilidade dos nós se encontrarem novamente. A medida que não há contatos entre dois nós, essa probabilidade é reduzida. Essas informações sobre a probabilidade de encontro são calculadas também de forma transitiva. Assim, a probabilidade de um agregado encontrar o nó destino de forma indireta também é mantida. O PRoPHET utiliza então essa probabilidade de encontro para encaminhar os agregados somente para nós com maior probabilidade de entrega. Apesar do PRoPHET minimizar os efeitos colaterais do roteamento epidêmico através do roteamento probabilístico, ainda assim não há um controle sobre a quantidade de replicações dos agregados. Os protocolos de replicação controlada, em especial o *Spray and Wait* (SnW), limitam a quantidade de cópias dos agregados na rede reduzindo bastante a sobrecarga enquanto mantém altas taxas de entrega e baixo atraso quando comparados aos de outras categorias de protocolos. O protocolo *Spray and Wait* trata-se uma melhoria do *Epidemic* cujo funcionamento consiste em duas etapas. Na primeira, denominada espalhamento, um nó replica L vezes o mesmo agregado. Nesse trabalho, essa distribuição é realizada de forma binária, ficando cada nó envolvido em um contato com metade dos tíquetes de replicação do um agregado de forma semelhante ao explicado na Seção 3.1. Na segunda, denominada espera, a disseminação dos agregados é interrompida. e os detentores das réplicas aguardam o encontro com o nó destino para entregá-las.

4.1. Ambiente de simulação

Os mecanismos LJC e FCF são avaliados através de simulações utilizando o simulador *The ONE (Opportunistic Network Environment)* [Keränen et al. 2009]. Nas simulações, as implementações nativas dos protocolos de roteamento PRoPHET e *Spray and Wait* são avaliadas nos cenários Rollernet e Infocom06. A transferência de custódia e o mecanismo de custódia compartilhada foram implementadas no ONE basicamente assumindo que os agregados sob custódia são descartados somente quando o TTL expira. Não são considerados o uso de *ACKs* enviados pelo nó destino como forma de remover os agregados dos *buffers*. Também foram introduzidos no simulador esquemas de prioridade em diferentes filas utilizadas pela política FCF conforme detalhado na Seção 3.2. No caso específico do protocolo PRoPHET, a política FCF é combinada com a política de encaminhamento GRTRMax, nativa do protocolo. Na política GRTRMax, o encaminhamento é feito apenas para nós que tenham maior probabilidade de contato com o nó destino dos agregados do que o nó que detém a mensagem. Alterou-se então seu com-

portamento para que, dentre os agregados mais prováveis de chegar ao destino a partir do contato atual, encaminhem-se primeiramente os sob custódia e, como critério de desempate, encaminhem-se os agregados com maior probabilidade de entrega.

Os parâmetros das simulações são definidos baseados em características específicas dos cenários. As diferentes durações dos experimentos apresentados por cada cenário implicam valores distintos de TTL, carga de tráfego e tamanhos de *buffer*, como indicado na Tabela 1. O TTL é definido baseado nos resultados apresentados por Naves *et al.* [Naves et al. 2012]. Com os valores de TTL considerados, aproximadamente 90% dos agregados chegam ao seu destino usando o protocolo *Epidemic* em ambos os cenários. Durante cada rodada de simulação, os pares fonte-destino de cada agregado são escolhidos aleatoriamente, com distribuição uniforme. O tamanho do *buffer* é proporcional à carga de tráfego. Especificamente para o protocolo *Spray and Wait* são definidas maiores cargas de tráfego visando colocar a rede sob situação de estresse e analisar principalmente o comportamento do uso de várias custódias em situações onde exista o transbordamento de *buffer*. Ainda para o protocolo *Spray and Wait*, o número máximo de replicações para cada agregado foi definido considerando-se a metade do número de nós.

Tabela 1. Parâmetros das simulações e dos cenários.

Parâmetros/Cenário	Rollernet	Infocom06
Duração ($\approx$)	3 horas	4 dias
Número de dispositivos	62	98
Número de contatos	15.803	74.224
Média de contatos por hora	5.704,96	796,21
Tempo de contato médio (s)	21,75	408
Tempo de contato máximo(s)	488	40.550
TTL (minutos)	60	2014
Número de agregados PRoPHET	500	5000
Número de agregados SnW	2000	20000
Tamanho dos agregados(MB)	1	1
Tamanho do Buffer(MB)	10-50	100-500

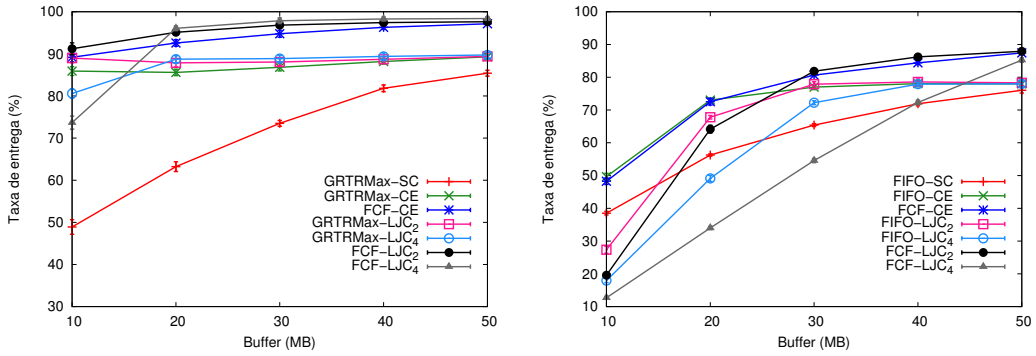
5. Resultados

Esta seção apresenta os resultados da avaliação de desempenho do mecanismo LJC e da política de encaminhamento FCF. O principal objetivo é analisar o impacto da custódia compartilhada e da política de encaminhamento no desempenho da rede em termos de taxa de entrega, sobrecarga e atraso de entrega. Primeiro, são analisados os mecanismos de custódia exclusiva e compartilhada. Assim, consideram-se os protocolos sem o uso de custódias (GRTRMax-SC e FIFO-SC), os protocolos com custódia exclusiva (GRTRMax-CE e FIFO-CE), e o mecanismo proposto LJC utilizando i agregados sob custódia (GRTRMax-LJC$_i$ e FIFO-LJC$_i$). Nesses casos, todos os mecanismos do protocolo PRoPHET usam GRTRMax e os do *Spray and Wait* usam FIFO como políticas de encaminhamento. Em seguida, é avaliado o impacto de priorizar os agregados sob custódia exclusiva usando a política de encaminhamento proposta FCF em lugar de GRTRMax e FIFO com ambos os protocolos (FCF-CE). Finalmente, as propostas LJC e FCF são incorporadas aos protocolos em questão (FCF-LJC$_i$) visando avaliar a custódia compartilhada e a priorização de agregados sob custódia operando simultaneamente. Em todas

as configurações mencionadas anteriormente, os agregados regulares são descartados dos *buffers* baseados em uma política FIFO. Usa-se um intervalo de confiança de 95% nos resultados e barras de erro são exibidas como linhas verticais em cada ponto dos gráficos.

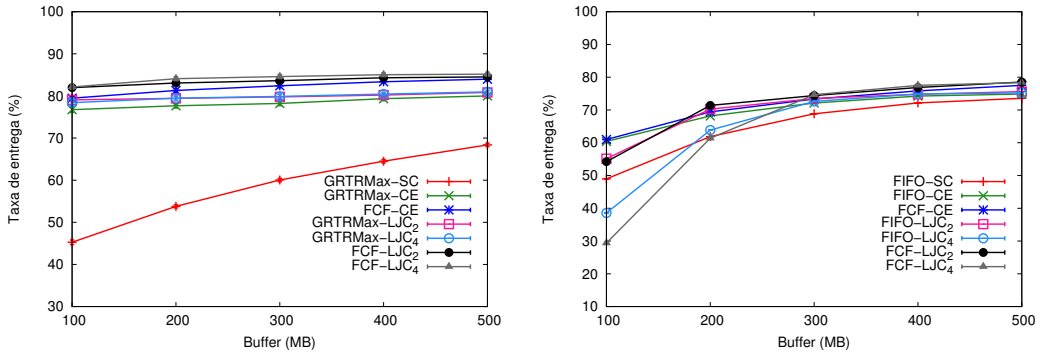
5.1. Taxa de Entrega

A taxa de entrega é a razão entre o número de agregados que atingem seu o nó de destino e o número de agregados enviados pelos nós de origem. Cópias não são computadas. As Figuras 1 e 2 mostram como o tamanho do buffer afeta a taxa de entrega para todos os mecanismos em ambos cenários. Dois aspectos podem ser claramente observados. Primeiro, quanto maior o tamanho do *buffer*, maior é a taxa de entrega para todos os mecanismos devido à redução do número de mensagens descartadas. Segundo, o uso dos mecanismos de custódia – exclusiva ou compartilhada – aumenta a taxa de entrega. Na verdade, os mecanismos de custódia garantem pelo menos uma cópia de cada agregado na rede e que essa cópia é descartada apenas se seu TTL expirar. Agregados regulares, ao contrário, podem ser descartados prematuramente por políticas de descarte. Assim, a disponibilidade de agregados sob custódia aumenta e, consequentemente, a probabilidade de um agregado chegar ao seu destino também aumenta.



(a) PRoPHET. (b) *Spray and Wait*.

Figura 1. Taxa de entrega no cenário Rollernet.



(a) PRoPHET. (b) *Spray and Wait*.

Figura 2. Taxa de entrega no cenário Infocom06.

As Figuras 1(a) e 2(a) mostram os resultados para os cenários Rollernet e Infocom para o protocolo PRoPHET. Os resultados no cenário Infocom06 com o PRoPHET são

bastante similares. O mecanismo proposto FCF-LJC$_i$ obtém a maior taxa de entrega do que os mecanismos de custódia exclusiva (FIFO-CE e FCF-CE) não importando o número de custódias por agregado.

LJC supera a custódia exclusiva porque incrementa o número de replicas dos agregados sob custódia pela rede e assim incrementa a probabilidade de entrega. Além do mais, com a política FCF, também se garante que os agregados sob custódia sejam encaminhados mais frequentemente que os agregados regulares devido à maior prioridade de encaminhamento. Portanto, com a combinação de LJC e FCF, temos mais agregados com prioridade na rede e assim, os mecanismos analisados atingem maiores taxas de entrega. O mecanismo FCF-LJC$_4$, com 4 custódias por agregado, apresenta a maior taxa de entrega para todos os tamanhos de buffer no protocolo PRoPHET e atinge nos cenários Rollernet e Infocom as taxas de entrega de aproximadamente 98% e 85% para os tamanhos de *buffer* de 50 MB e 500 MB.

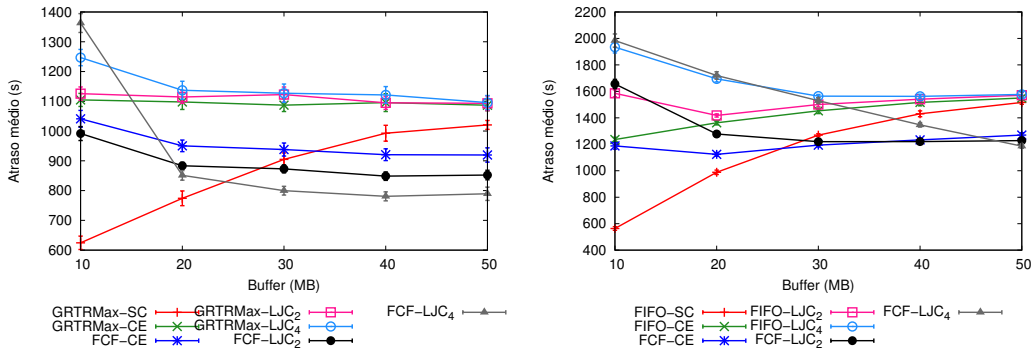
Os resultados para protocolo *Spray and Wait* são exibidos nas Figuras 1(b) e 2(b). As configurações FCF-CE e FCF-LJC$_2$ atingem taxas de entrega de aproximadamente 85% e 78% para os cenários Rollernet e Infocom06 respectivamente em seus maiores tamanhos de *buffer*. Porém, deve-se observar que o uso de custódia compartilhada, especialmente com 4 réplicas por agregado, é ineficiente para os tamanhos de *buffers* iniciais. Isso é justificado pela ação do mecanismo de replicação inerente ao protocolo ao limitar a propagação dos agregados. Com isso, ocorrem menos encaminhamentos, fazendo com que os tíquetes de custódia não sejam distribuídos. Consequentemente, os nós com *buffers* de tamanho reduzido ficam congestionados com agregados sob custódia, por não poderem realizar descartes. Esse congestionamento impede que novos agregados sejam recebidos e, consequentemente, a distribuição desses agregados na rede fica comprometida, tendo como principal consequência a redução da taxa de entrega. O mesmo não ocorre com a custódia exclusiva, pois a cada encaminhamento na fase de espalhamento, ocorre a transferência de custódia, deixando os agregados recém-encaminhados no nó de origem sujeitos ao descarte caso o *buffer* chegue ao seu limite.

É importante também evidenciar que a política de encaminhamento FCF tem o maior impacto na taxa de entrega que o mecanismo LJC para ambos os protocolos, como mostrado nas Figuras 1 e 2. Esse resultado deve-se à priorização dos agregados sob custódia. Essa priorização não somente aumenta a disponibilidade dos agregados, mas o mais importante, incrementa a probabilidade de entrega dos agregados, pois são mais prováveis de encontrar seu destino. Adicionalmente, FCF supera FIFO porque a política proposta não sofre do problema conhecido como cabeça de linha (*head-of-line problem*) [Ip et al. 2007]. Com FIFO, os agregados mais antigos no *buffer* são encaminhados primeiro durante um contato. Consequentemente, os agregados nas primeiras posições da fila são frequentemente encaminhados enquanto os que estão no fim da mesma raramente o são. Portanto, conclui-se que os mecanismos analisados aumentam a taxa de entrega para esses distintos cenários, em termos de tempo de contato e conectividade, para os protocolos PRoPHET e *Spray and Wait* quando comparados às configurações sem o uso de mecanismos de custódia.

5.2. Atraso de Entrega

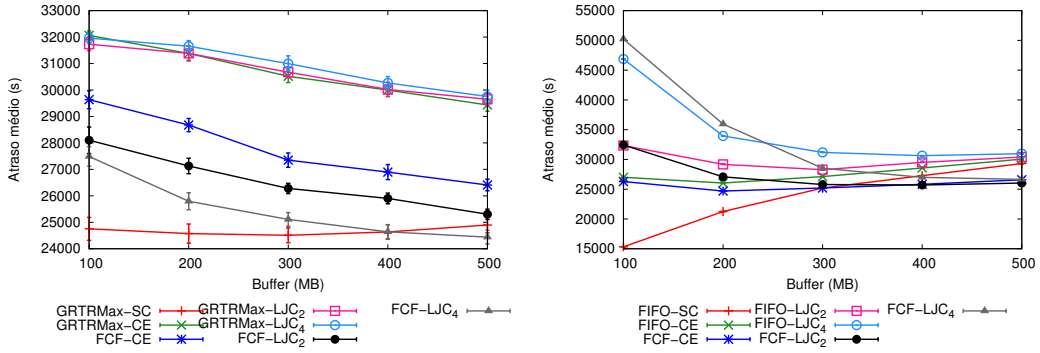
O atraso de entrega dos agregados é dado pelo intervalo entre o momento em que o agregado é enviado pelo nó de origem até o momento em que atinge seu destino.

As Figuras 3 e 4 exibem o atraso médio de todos os agregados entregues nos cenários Rollernet e Infocom06. A maioria dos resultados ratificam a ideia de que a política FCF reduz o atraso por priorizar os agregados sob custódia. No cenário Rollernet, Figura 3, para o tamanho de *buffer* igual a 50 MB, FCF-LJC$_4$ reduz 23% (Prophet) e 22% (SnW) os tempos de entrega quando comparado com todas às políticas baseadas em GRTRMax e FIFO. Para o tamanho de *buffer* igual a 500 MB no cenário Infocom06, Figura 4(b), a FCF-LJC$_2$ reduz em aproximadamente 11% o tempo de entrega dos agregados em relação às políticas baseadas em FIFO.



(a) PRoPHET. (b) *Spray and Wait*.

Figura 3. Atraso médio de entrega no cenário Rollernet.



(a) PRoPHET. (b) *Spray and Wait*.

Figura 4. Atraso médio de entrega no cenário Infocom06.

A política FCF garante que um agregado com tíquetes restantes será encaminhado pelo menos uma vez pelo seu nó receptor antes que seja movido para a fila de regulares. Assim, esse agregado pode ser encaminhado mais rapidamente, pois tem prioridade. Por outro lado, o problema denominado cabeça de linha experimentado pela política de encaminhamento FIFO aumenta o tempo de entrega dos agregados. Sem priorização, os agregados recebidos recentemente por um nó, devem esperar nos *buffers* até ficarem antigos suficiente para alcançar o início da fila. Uma vez no início da fila, o agregado é encaminhado. Esse atraso pelo qual os agregados passam depende tanto do tamanho do *buffer* quanto da dinâmica entre os contatos. Assim, quanto maior o tamanho do *buffer*, mais demorado é o tempo de espera por um encaminhamento. Os agregados também podem ser descartados por uma política de descarte durante esse período de espera, o que reduz a taxa de entrega.

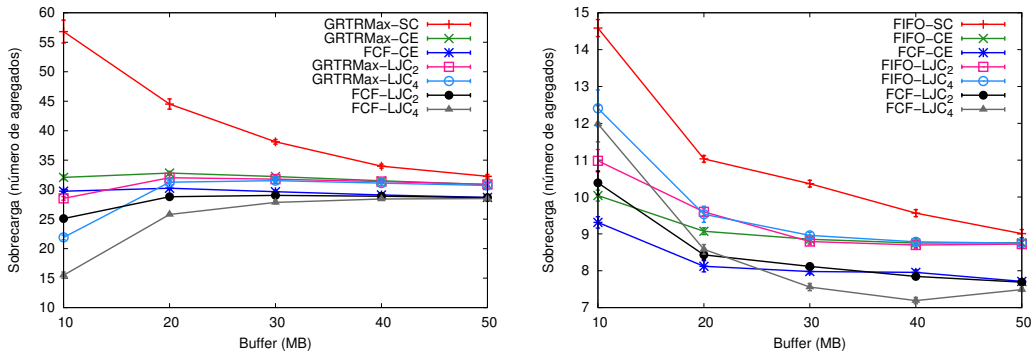
Particularmente, o problema cabeça de linha também explica o baixo atraso de entrega provido pelo mecanismo FIFO-SC quando comparada a todos os outros mecanismos baseados em FIFO nos dois cenários. De fato, FIFO-SC apenas entrega os agregados enviados durante a inicialização da rede, como observado por Lindgren e Phanse [Lindgren e Phanse 2006], o que é corroborado pelo baixo desempenho do mecanismo FIFO-CE em termos de taxa de entrega, como mostrado nas Figuras 1 e 2. Os primeiros agregados enviados pela rede experimentam baixo tempo de atraso porque os *buffers* estão quase vazios. Portanto, o atraso médio é baixo, uma vez que são levados em conta apenas os agregados entregues para calcular essa métrica.

Um ponto a ser observado é que no cenário Infocom06 com o protocolo PRoPHET (Figura 4(a)) o comportamento do atraso médio é distinto dos demais. O mecanismo sem custódia (GRTRMax-SC) possui o menor atraso. Isso se deve à atuação mais eficiente da política GRTRMax, pois nesse cenário há um número de contatos muito superior ao do Rollernet como mostra a Tabela 1. Assim, quanto mais contatos, mais efetivamente são mantidas as probabilidades de futuros encontros, ocasionando encaminhamentos para nós com maior probabilidade de entrega e com menor atraso. Já os mecanismos GRTRMax-LJC$_i$ possuem atraso maior devido ao mecanismo LJC propiciar maiores taxas de entrega (Figura 2(a)) em relação à GRTRMax-SC, o que resulta na entrega de agregados que inclusive possuem maior atraso. Por fim, os mecanismos FCF-LJC$_i$ ocupam posição intermediária devido à política FCF ser híbrida nesse protocolo, ou seja, primeiro os agregados sob custódia são encaminhados e, como critério de desempate, os com maior probabilidade de entrega. Ainda assim, percebe-se a tendência dos mecanismos que utilizam FCF superarem a implementação sem custódia, pois com valores de *buffers* a partir de 400 MB, o esquema FCF-LJC$_4$ passa a ter atraso inferior ao do mecanismo GRTRMax-SC.

5.3. Sobrecarga

A sobrecarga S é calculada de acordo com a seguinte equação: $S = \frac{a_{enc} - a_{ent}}{a_{ent}}$, onde a_{enc} é o número de agregados encaminhados e a_{ent} é o número de agregados entregues. Múltiplas cópias do mesmo agregado que tenham sido recebidos pelo nó destino não são computados. Portanto, a sobrecarga indica quantas ações de encaminhamento em média são necessárias para entregar um agregado.

A Figura 5 exibe a sobrecarga para o cenário Rollernet. Por limitação de espaço e similaridade entre os resultados nos dois cenários, os resultados para o cenário Infocom06 foram suprimidos. Claramente, o uso de custódia reduz a sobrecarga. Nesse caso, os agregados sob custódia não são descartados por políticas de descarte e assim os nós tendem a carregá-los por um período de tempo maior. Consequentemente, mais cópias desses agregados são encaminhadas e, como resultado, mais agregados atingem seus destinos, o que reduz a sobrecarga por agregado entregue. Os mecanismos que utilizam FCF, em geral, obtêm menor sobrecarga que os mecanismos que utilizam FIFO porque eles atingem maiores taxas de entrega. Todas as configurações FCF-LJC$_i$ superam as demais configurações em ambos os protocolos analisados. Isso é bem evidenciado para o tamanho de *buffer* igual a 10 MB, o mais restrito e onde ocorre o maior número de descartes. Para esse tamanho, FCF-CE obtém no *Spray and Wait* uma redução de aproximadamente 47% da sobrecarga em relação à implementação sem o uso de custódias. Para o protocolo PRoPHET essa melhoria é ainda mais evidente, chegando a FCF-LJC$_4$ a obter 73% de redução da sobrecarga quando comparado mecanismo FIFO-SC.



(a) PRoPHET. (b) *Spray and Wait*.

Figura 5. Sobrecarga no cenário Rollernet.

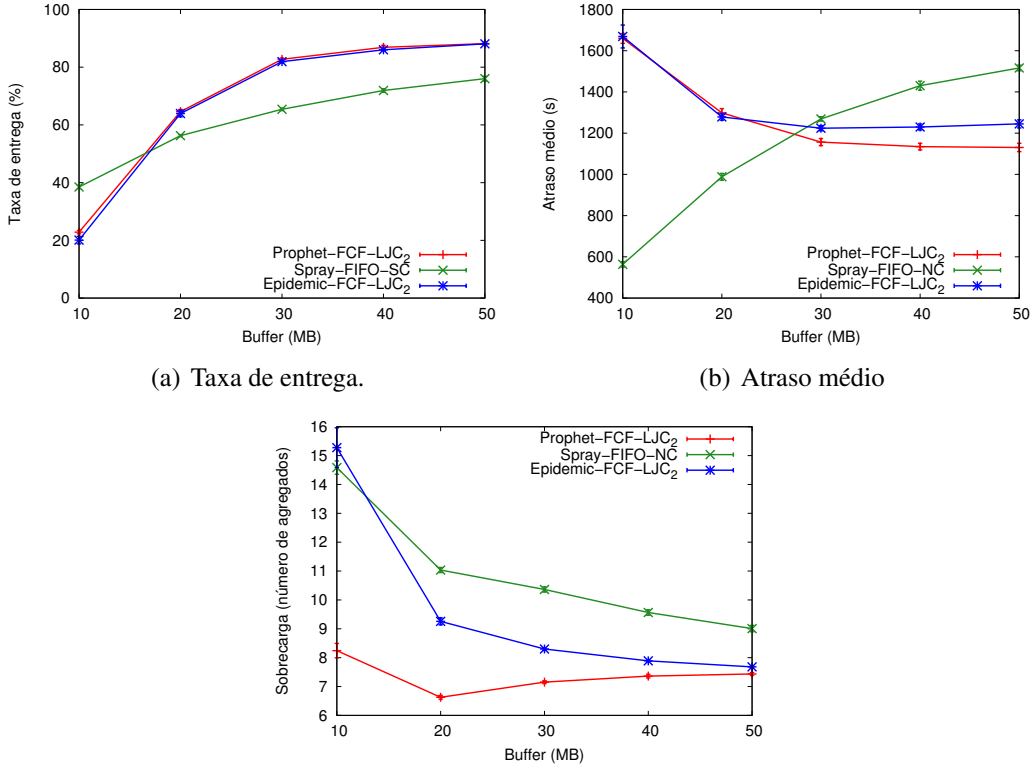
5.4. Uma alternativa ao *Spray and Wait*

Nesta seção é realizada uma comparação do mecanismo combinado FCF-LJC nos protocolos *Epidemic* e PRoPHET com a versão sem o uso de custódia do *Spray and Wait*. O intuito é mostrar que é possível oferecer uma taxa de entrega maior que o *Spray and Wait*, com uma sobrecarga menor e com atraso melhor para um maior volume de tráfego. A Figura 6 exibe essa comparação.

Dentre os mecanismos FCF-LJC$_i$, foi escolhido o FCF-LJC$_2$ por apresentar melhor desempenho geral com a configuração de 2000 agregados para o cenário Rollernet. As seguintes considerações são relacionadas ao tamanho de *buffer* de 50 MB. Em relação ao *Spray and Wait*, os protocolos PRoPHET e *Epidemic* apresentam uma taxa de entrega aproximadamente 12% maior. Verifica-se também que com o PRoPHET e *Epidemic* o atraso médio de entrega são aproximadamente 26% e 18% menores, respectivamente. Por fim, o mecanismo FCF-LJC$_2$ com os protocolos PRoPHET e *Epidemic* obtêm sobrecargas aproximadamente 18% e 15% menores que a apresentada pelo protocolo *Spray and Wait*. Assim, as propostas analisadas nesse trabalho apresentam relevantes melhorias de desempenho. Podemos concluir ainda que um protocolo híbrido baseado em replicação probabilística e com replicação controlada, como o mecanismo FCF-LJC$_2$ sobre o protocolo PRoPHET, se comporta como a melhor configuração possível dentre as analisadas e se destaca como uma alternativa ao protocolo de replicação controlada *Spray and Wait* em cenários com maior volume de tráfego.

6. Conclusões e Trabalhos Futuros

Nesse artigo, foram avaliados os mecanismos *Limited Joint Custody* (LJC) e *Forward Custody First* (FCF) para redes DTNs. O LJC emprega a custódia compartilhada e limita o número de custódias por agregado na rede. O FCF é uma política de encaminhamento que prioriza agregados sob custódia. Através de simulações, foi avaliado o desempenho de ambos os mecanismos com os protocolos PRoPHET e *Spray and Wait* em dois cenários reais de mobilidade. Os principais resultados mostram que a combinação de LJC e FCF proporciona um aumento significativo de desempenho em termos da taxa de entrega, atraso e sobrecarga. A explicação para tais ganhos é a seguinte. O LJC aumenta o número de réplicas de agregados sob custódia na rede. O FCF, por sua vez, dá total prioridade aos agregados sob custódia e, assim, encaminha esses agregados



(a) Taxa de entrega. (b) Atraso médio

(c) Sobrecarga

Figura 6. Comparação no cenário Rollernet entre os Protocolos *Epidemic* e PRoPHET com o uso de custódia e o *Spray and Wait* em sua versão padrão.

primeiro e mais frequentemente que os agregados regulares. Assim, a combinação FCF-LJC obtém mais agregados com prioridade na rede atingindo maiores taxas de entrega e menores atrasos. Outra conclusão importante é que, mesmo os protocolos de replicação controlada obtendo desempenho superior às demais categorias de protocolos, com FCF-LJC combinados ao protocolo de replicação probabilística PRoPHET, obtém-se uma alternativa mais eficiente do que o protocolo de replicação controlada *Spray and Wait* para cenários com maior número de agregados na rede. Também se verificou nesses cenários que não há melhoria no desempenho com o uso de mais de duas custódias por agregado. Os trabalhos futuros incluem incorporar e avaliar diferentes mecanismos concedendo ou revogando custódias de forma probabilística. Pretende-se também avaliar a negociação de transferência de custódia baseada em taxas de ocupação de *buffer* e o desempenho das propostas com protocolos baseados em métricas sociais.

7. Agradecimentos

Esse trabalho é apoiado pela CAPES, CNPq, FAPERJ, CTIC, Proppi/UFF e PROPI/IFPI. Os autores agradecem pelos dados obtidos do arquivo CRAWDAD do Dartmouth College.

Referências

Cao, Y. e Sun, Z. (2012). Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges. *IEEE Communications Surveys & Tutorials*, páginas 1–24.

Chuah, M.-C., Yang, P., Davison, B. D. e Cheng, L. (2006). Store-and-forward performance in a DTN. Em *IEEE VTC'06*.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. Em *ACM SIGCOMM'03*.

Fall, K. e Farrell, S. (2008). DTN: an architectural retrospective. *IEEE JSAC*, páginas 828–836.

Fall, K., Hong, W. e Madden, S. (2003). Custody transfer for reliable delivery in delay tolerant networks. Relatório técnico, Intel Research, Berkeley, California.

Hui, P. e Lindgren, A. (2008). Phase transition of opportunistic communications. Em *ACM Workshop on Challenged networks*.

Ip, Y.-K., Lau, W.-C. e Yue, O.-C. (2007). Forwarding and replication strategies for DTN with resource constraints. Em *IEEE VTC'07*.

Keränen, A., Ott, J. e Kärkkäinen, T. (2009). The ONE simulator for DTN protocol evaluation. Em *SIMUTools'09*.

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*, páginas 607–640.

Lindgren, A., Doria, A. e Schelén, O. (2003). Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, páginas 19–20.

Lindgren, A. e Phanse, K. S. (2006). Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. Em *COMSWARE'06*.

Miranda, E., Naves, J. F., Moraes, I. M. e Velloso, P. B. (2012). A joint custody-based forwarding policy for delay-tolerant networks. Em *IEEE GIIS'12*.

Naves, J. F., Moraes, I. M. e de Albuquerque, C. V. N. (2012). LPS and LRF: Efficient buffer management policies for delay and disruption tolerant networks. Em *IEEE LCN'12*.

Oliveira, C. T., Moreira, M. D. D., Rubinstein, M. G., Costa, L. H. M. K. e Duarte, O. C. M. B. (2007). Redes tolerantes a atrasos e desconexões. Em *Minicursos do SBRC*.

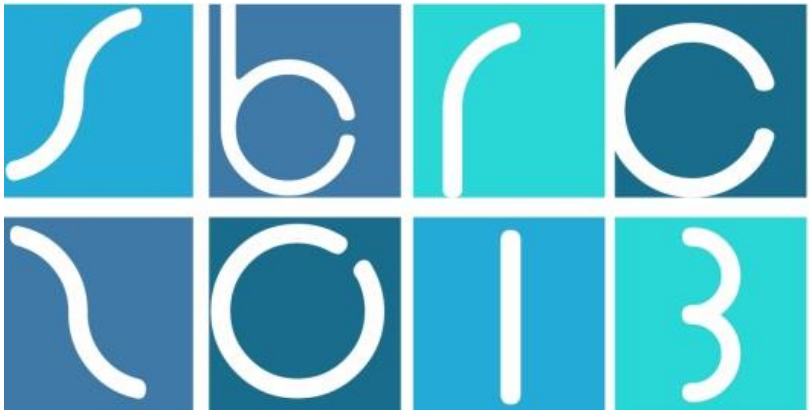
Soares, V. S., Farahmand, F. F. e Rodrigues, J. R. (2010). Performance analysis of scheduling and dropping policies in vehicular delay-tolerant networks. *International Journal on Advances in Internet Technology*, páginas 137–145.

Spyropoulos, T., Psounis, K. e Raghavendra, C. S. (2005). Spray and wait: an efficient routing scheme for intermittently connected mobile networks. Em *ACM SIGCOMM Workshop on Delay-tolerant networking*.

Spyropoulos, T., Rais, R. N. B., Turletti, T., Obraczka, K. e Vasilakos, A. (2010). Routing for disruption tolerant networks: taxonomy and design. *Wireless Networks*, páginas 2349–2370.

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'09*.

Wang, Y., Wu, J., Jiang, Z. e Li, F. (2012). A joint replication-migration-based routing in delay tolerant networks. Em *IEEE ICC'12*.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 17

Redes Definidas por Software 1

Uma arquitetura baseada em Redes Definidas por Software para isolamento de redes em datacenters virtualizados

Rogério V. Nunes[1], **Raphael L. Pontes**[1], **Dorgival Guedes**[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais – Belo Horizonte, MG – Brazil

{rogervn,raphaelluciano,dorgival}@dcc.ufmg.br

Abstract. *The increasing interest in Cloud Computing has brought new demands to providers of "Infrastructure-as-a-service" solutions. To host a large number of clients in the same datacenter, they require multi-tenant networks that can guarantee traffic isolation and scalability, with low costs. This paper describes a solution for this problem that uses Software Defined Networks (SDN). By using SDN, we can program the virtual switches at the physical servers to meet all those requirements, without demanding special hardware in the network. Experiments show good results with very little overhead, even preventing DoS attacks between tenants.*

Resumo. *O interesse crescente em Computação em Nuvem cria novas exigências para os provedores de soluções de "Infraestrutura-como-serviço". Para abrigar um grande número de clientes em um mesmo* datacenter*, eles requerem redes multi-cliente que possam garantir isolamento de tráfego e escalabilidade, preferencialmente a baixo custo. Este artigo apresenta uma solução para esse problema que usa Redes Definidas por Software (RDS). Com RDS, podemos programar os* switches *virtuais dos servidores físicos de uma forma elegante e flexível para atender a esses requisitos, sem exigir hardware especial na rede. Experimentos mostram bons resultados com muito pouco* overhead*, inclusive prevenindo ataques de negação de serviço entre usuários.*

1. Introdução

Os últimos anos têm presenciado a poplularização da computação em nuvem como um paradigma importante para a implementação de serviços na Internet, o qual exige recursos e soluções organizadas ao redor de sistemas distribuídos. Entre as diferentes formas de computação em nuvem, soluções de infraestrutura-como-serviço (IaaS) estão entre as mais bem sucedidas até o momento. Em IaaS, recursos computacionais são controlados por um modelo pague-pelo-que-usa, viabilizando a criação de soluções elásticas que se adaptam às demandas dos usuários de forma transparente.

Em um cenário como esse, grandes *datacenters* que usam técnicas de virtualização se mostraram a melhor forma de obter alta utilização de recursos mantendo um ambiente flexível. Nessas instalações, clientes do *datacenter* (em inglês denominados *tenants*, inquilinos) podem contratar máquinas virtuais (VMs) que são interconectadas para criar uma rede virtual, sobre a qual os serviços executam. Dois importantes requisitos para que esse tipo de solução funcione são o isolamento dos recursos dos inquilinos e a capacidade do sistema de escalar para acomodar um grande número deles. A primeira condição

é necessária para garantir que os serviços de cada inquilino não sejam afetados pelas ações dos demais (acidental ou maliciosamente). A segunda é essencial para tornar o modelo viável economicamente. Entretanto, apesar da virtualização de máquinas oferecer bom isolamento de CPU, memória e espaço de armazenamento, o acesso à rede é uma outra estória, especialmente quando escalabilidade e facilidade de gerência também são desejáveis [Greenberg et al. 2009].

Alguns novos protocolos e arquiteturas de redes vêm sendo propostos para endereçar esses problemas, mas eles exigem hardware novo para funcionar. Na maioria dos casos, atualizar todo o hardware de rede não é uma opção e soluções mais fáceis de gerenciar, escaláveis e eficientes, ainda não existem.

Neste trabalho propomos um novo sistema, *DCPortals*, que endereça aqueles fatores sem exigir novo hardware. Nossa solução é baseada na re-escrita de campos do cabeçalho do pacote, de forma a esconder a origem e o destino reais do hardware no núcleo da rede, ao mesmo tempo que também esconde o tráfego de cada rede virtual de quaisquer VMs que não pertençam ao mesmo inquilino.

Tendo isso em mente, o restante deste trabalho está organizado da seguinte forma: a seção 2 apresenta alguns conceitos básicos relacionados; a seção 3 descreve a arquitetura adotada e detalha a operação do sistema, cujo comportamento é avaliado na seção 4. Em seguida, a seção 5 coloca o *DCPortals* no contexto de outros trabalhos relacionados e a seção 6 apresenta nossas considerações finais, incluindo a discussão de trabalhos futuros.

2. Conceitos básicos

Alguns *datacenters* se valem do uso de VPNs sobre redes Ethernet para isolar o tráfego de cada inquilino. Apesar de ser uma solução relativamente simples, VPNs são limitadas pela definição do protocolo, que reserva apenas 12 bits para a identificação de diferentes redes virtuais. Outros usam soluções de roteamento da camada 3 para isolar tráfego, o que limita a capacidade do inquilino de configurar seu próprio esquema de endereçamento, exigindo configurações complexas à medida que o número de máquinas cresce. Em ambas as soluções, ainda há o problema de manipular os endereços de um grande número de VMs em uma única rede de *datacenter*. Em alguns casos, cada máquina física pode hospedar até centenas de VMs, cada uma com seu próprio endereço da camada 2 (MAC). Tabelas de encaminhamento na maioria dos *switches* Ethernet têm espaço limitado e o desempenho pode cair significativamente com a necessidade de mais *broadcasts*, já que os *switches* não conseguem apreender todos os endereços de todos os destinos possíveis. Soluções baseadas no empilhamento de protocolos (tunelamento) podem reduzir a necessidade dos *switches* da rede aprenderem todos os endereços, já que eles passam a precisar apenas de identificadores de túneis, mas têm limitações de desempenho e gerência. Uma boa revisão dessas técnicas é o trabalho de Cabuk *et al.* [Cabuk et al. 2007, Cabuk et al. 2008].

Para atingir o objetivo deste trabalho, utilizamos os *switches* virtuais presentes nos monitores de virtualização (hipervisores) presentes em cada máquina física. Esses *switches* podem ser facilmente atualizados para versões já disponíveis que implementam o modelo OpenFlow [McKeown et al. 2008]. Uma dessas versões é a Open vSwitch [Pfaff et al. 2009], que já se tornou padrão para hipervisores como o Xen e o KVM. Com OpenFlow, cada *switch* virtual exporta uma interface de programação para acesso às suas tabelas de encaminhamento. Usando essa interface, um controla-

dor pode informar ao *switch* como tratar pacotes que não casam com nenhum dos fluxos já identificados anteriormente. Para esses pacotes, ele pode determinar como aqueles pacotes (bem como os demais daquele fluxo) devem ser roteados, pode instruí-lo a re-escrever campos dos cabeçalhos baseado em parâmetros do fluxo, ou descartá-los. Enfim, é possível para o administrador do *datacenter* controlar como a rede encaminha cada fluxo de pacotes que a atravessa. Isso levou à criação do conceito de hipervisor de rede, um controlador (software) é capaz de isolar o tráfego dos inquilinos na rede assim como hipervisores de máquinas isolam VMs que compartilham CPU e memória. O sistema resultante é chamado de Rede Definida por Software (ou *Software Defined Network*, SDN) [Casado et al. 2010]. A abstração de hipervisores de redes definidas por software provê uma representação logicamente centralizada da rede, onde a configuração e o controle da rede podem ser realizados facilmente, enquanto mantém a escalabilidade da solução. Há benefícios importantes nesse enfoque:

- **Redução do uso da memória de *switches*:** ao re-escrever os cabeçalhos Ethernet podemos ocultar os endereços de camada 2 das VMs dos *switches* do núcleo da rede, reduzindo o número de entradas usadas em suas tabelas internas.
- **Criação de redes virtuais isoladas para cada inquilino:** ao controlar o encaminhamento de pacotes nos *switches* de borda podemos garantir que o tráfego de cada inquilino nunca alcançará VMs de outros inquilinos. Isso oferece tanto privacidade (VMs de outros inquilinos não podem acessar tráfego daquele inquilino) quanto proteção contra ataques (um inquilino malicioso não é capaz de direcionar seu tráfego para máquinas em outras redes virtuais).
- **Integração da configuração e gerência de VMs e da rede virtual:** *DCPortals* integra o hipervisor de rede com um controlador de virtualização bem conhecido, o OpenStack[1], de forma que a instanciação de máquinas virtuais possa ser integrada à configuração e operação da rede virtual. Se uma VM migra de um hospedeiro para outro, o hipervisor pode identificar isso automaticamente e reconfigurar os *switches* para tratar o tráfego daquele inquilino apropriadamente.
- **Implementação fácil das funcionalidades necessárias:** Por usar o paradigma SDN, podemos desenvolver algoritmos de controle que usam os recursos da rede da melhor forma. A re-escrita de pacotes se torna um procedimento simples e automatizado, em que *DCPortals* pode interagir com OpenStack para obter os parâmetros de configuração de que necessita. Isso simplifica as demandas sobre o hardware convencional no núcleo da rede, ao mesmo tempo que garante o isolamento de tráfego no nível das VMs. Além do mais, isso pode ser feito sem usar os rótulos de VLANs, que os libera para serem usados na implementação de um solução de roteamento multicaminhos baseado em VLANs, por exemplo, o que poderia ser facilmente integrado à nossa solução em *datacenters* que ofereçam redes Ethernet com canais redundantes [Mudigonda et al. 2010].

3. Detalhamento da solução

DCPortals foi implementado como um módulo construído sobre o hipervisor de SDN POX[2]. Ele consulta a base de dados do *OpenStack* diretamente para extrair a informação de que necessita sobre as máquinas virtuais e suas redes virtuais, tais como identificação

[1]http://www.openstack.org

[2]https://openflow.stanford.edu/display/ONL/POX+Wiki

do inquilino, outras VMs em uma dada rede virtual e, especialmente, a localização de cada VM. Com aquela informação, ele monta as mensagens OpenFlow para informar às Open vSwitches como tratar os fluxos de pacotes de/para uma dada VM. OpenStack controla o hipervisor Xen em cada hospedeiro, que configura sua Open vSwitch adequadamente. A figura 1 ilustra as relações entre os módulos.

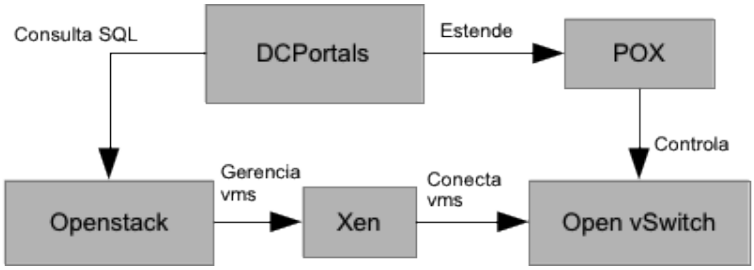


Figura 1. Arquitetura do sistema *DCPortals*.

O administrador do sistema usa a API OpenStack para disparar cada máquina virtual e indicar a rede virtual a que ela pertence. OpenStack seleciona o hospedeiro físico que executará a VM e envia a informação que ela precisa para ser disparada. O hipervisor Xen conecta a nova VM ao Open vSwitch da máquina física. Quando a nova VM envia seu primeiro pacote pela rede, o Open vSwitch identifica um novo fluxo e usa o protocolo OpenFlow para informar ao controlador POX, que extrai a informação sobre o fluxo a partir do pacote recebido. POX notifica o *DCPortals*, que inspeciona o pacote, consulta a base de dados OpenStack e envia outra mensagem OpenFlow de volta para o switch apropriado, dizendo como os demais pacotes daquele fluxo devem ser tratados. Como o fluxo é direcionado a uma VM em algum ponto da rede, *DCPortals* também já programa o Open vSwitch ao qual a VM de destino está conectado para também estar preparado para tratar os pacotes do fluxo. A seção a seguir fornece mais detalhes sobre esse processo e cada um dos aspectos principais do sistema.

3.1. A abstração de rede virtual

No *DCPortals*, a abstração de rede virtual oferece uma representação clara de como é definido o isolamento lógico entre as múltiplas redes de inquilinos que compartilham a mesma infraestrutura, independente da sua organização física. Cada inquilino vê suas máquinas como conectadas a um único *switch* virtual, separado dos demais, como ilustrado na figura 2. Máquinas conectadas àquela rede podem se comunicar livremente e todo seu tráfego é limitado às VMs conectadas ao *switch*, independente da estrutura que as interliga na topologia física. Essa abstração garante ao menos a segurança de uma rede local, sem forçar os inquilinos a se preocupar com a infraestrutura física, compartilhada.

A identificação das máquinas que forma uma dada rede é derivada da chave RSA de cada VM que é parte da configuração do OpenStack. Essa chave é usada para configurar o acesso do administrador a todas as VMs de um usuário por SSH. *DCPortals* assume que todas as VMs que compartilham uma mesma chave RSA pertencem a uma mesma rede (isolada). Tal premissa é razoável, considerando-se que todas as máquinas na rede local de um inquilino deveriam ser acessíveis para aquele inquilino. A chave RSA pode ser considerada um identificador opaco para o *switch* virtual daquela rede. Ao adotar essa

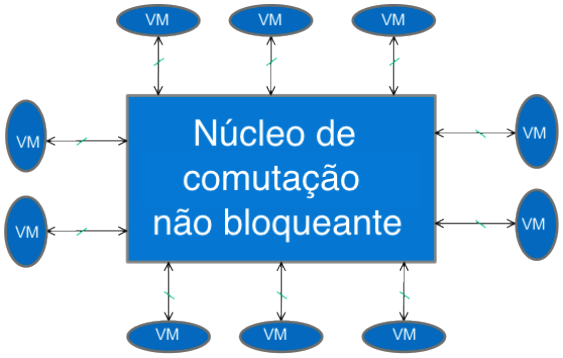


Figura 2. Abstração de rede virtual de cada inquilino.

solução, evitamos forçar cada inquilino a explicitamente fornecer uma descrição da rede, o que exigiria uma extensão do *esquema* da base de dados OpenStack[3]

Para verificar se os pacotes de uma VM A podem alcançar uma outra VM B, tudo que *DCPortals* tem que fazer é consultar a base OpenStack e verificar se as chaves RSA de A e B são iguais. Se esse for o caso, as máquinas estão na mesma rede virtual e tráfego de uma pode alcançar a outra, o que significa que o controlador de rede deve programar as Open vSwitches nas máquinas físicas que hospedam A e B de forma a garantir que pacotes possam fluir entre elas. Caso contrário, se A e B estão em redes diferentes, um pacote de A para B seria descartado pela primeira Open vSwitch no seu caminho.

3.2. Integração dos módulos

DCPortals usa a API do POX para manipular (receber, interpretar, construir e enviar) mensagens OpenFlow que controlam as Open vSwitches na rede, baseado nos fluxos identificados e na localização das VMs associadas. Ele basicamente reage a dois eventos definidos na interface POX, *ConnectionUp*, que é disparado quando um novo *switch* é conectado à rede do *datacenter*, e *PacketIn*, que é lançado quando um novo pacote chega ao controlador vindo de algum *switch*, indicando que um novo fluxo foi identificado e para o qual as regras de encaminhamento ainda não foram definidas. O primeiro evento é usado para construir as estruturas de dados necessárias para acompanhar o estado dos dispositivos, enquanto o segundo é essencial para fornecer as informações necessárias para se implementar o roteamento e isolamento de tráfego.

Como mencionado anteriormente, o sistema consulta a base de dados MySQL do OpenStack para recuperar informações sobre as VMs existentes cada vez que um novo fluxo é observado. Isso inclui, além da informação sobre a chave RSA da VM, a sua localização física e seus endereços IP e MAC. Além disso, *DCPortals* dispara consultas para obter informações sobre os hospedeiros físicos no sistema quando ele precisa identificar o destino de um novo fluxo.

Além da informação obtida do OpenStack, *DCPortals* também tem sua própria base de configuração. Ela é usada para armazenar a informação necessária para acessar a base MySQL, bem como os endereços MAC de todas as máquinas físicas que fazem parte da rede do *datacenter*. Isso é necessário para identificar a interface de controle de

[3]Desde que iniciamos este trabalho, o OpenStack já foi estendido com um módulo especial criado para descrever configurações de rede, o Quantum. Integrar o *DCPortals* a ele é um trabalho futuro previsto.

cada máquina, através da qual as mensagens OpenFlow são enviadas, separadamente da rede que contém o tráfego das VMs dos inquilinos.

3.3. Re-escrita de pacotes para isolamento das redes

Como discutido anteriormente, o uso de re-escrita de pacotes para implementar o isolamento de redes tem dois benefícios principais: garante que o tráfego de um inquilino fique fora do alcance dos demais e reduz a pressão sobre a memória dos dispositivos de rede, que não precisam lidar com os endereços MAC de todas as VMs no *datacenter*. Para conseguir isso, nós re-escrevemos os endereços de enlace (MAC) em todos os pacotes que atravessam uma Open vSwitch na borda da rede para remover a informação das VMs.

Para fazer isso, *DCPortals* considera que cada VM no *datacenter* é criada com um endereço IP único. Essa não é uma restrição séria para os inquilinos já que, por definição, todos os endereços IP válidos devem ser únicos. Se algum inquilino usa endereços IP restritos, é fácil assinalar para ele um segmento em uma faixa como a região 10/8. Nosso objetivo com isso é garantir que, dado um endereço IP na rede do *datacenter*, seja sempre possível identificar a VM exata associada a ele. Fazendo isso, deve ser sempre possível encontrar o endereço MAC correto (origem ou destino) para qualquer pacote, simplesmente verificando o endereço IP correspondente no pacote.

Quando uma máquina virtual VM_1, operando no hospedeiro físico $Host_1$, envia um pacote para outra máquina virtual VM_3 na mesma rede virtual, mas fisicamente localizada no hospedeiro físico $Host_2$, o pacote original enviado por VM_1 que alcança o *switch* virtual no $Host_1$ conterá os endereços MAC de VM_1 e VM_3, bem como os endereços IP de ambos. Para simplificar, consideremos que *DCPortals* já identificou o fluxo associado e programou os *switches* de borda apropriadamente. O Open vSwitch no $Host_1$, então, vai substituir os endereços MAC de VM_1 e VM_3 no cabeçalho Ethernet do pacote com os endereços MAC de $Host_1$ e $Host_2$, respectivamente. O pacote resultante é que atravessará o núcleo da rede, ainda carregando os endereços IP de VM_1 e VM_3. Entretanto, os endereços MAC que serão usados nas tabelas de aprendizado e encaminhamento dos *switches* no núcleo serão apenas aqueles dos hospedeiros físicos. Quando o pacote alcança $Host_2$, ele atravessará o *switch* virtual ali configurado, Nesse momento, ele verificará os endereços IP no pacote e re-escreverá os endereços MAC de VM_1 e VM_3 no cabeçalho. Esse é o pacote que será entregue a VM_3 nesse ponto. Note-se que que os endereços MAC das máquinas virtuais nunca cruzaram a rede física; mesmo assim, os pacotes só alcançaram destinos para os quais o sistema pode atestar sua conectividade segundo alguma rede virtual.

A figura 3 ilustra o processo para um pacote em um fluxo que já foi identificado e teve regras de re-escrita propagadas para os *switches* de borda apropriados, exatamente como discutido no parágrafo anterior. É importante lembrar que esse processo usa os endereços IP no pacote para garantir que os cabeçalhos originais possam ser recompostos. Isso significa que *DCPortals* só funciona com tráfego IP no momento. Não há, entretanto, aplicações relevantes em *datacenters* que não sejam baseadas em IP.

3.4. Tratamento de mensagens ARP e DHCP

A técnica de re-escrita de MACs descrita substitui endereços Ethernet (MAC) das máquinas virtuais em pacotes que já haviam sido construídos com aqueles endereços. Entretanto, para as VMs construírem aqueles pacotes pela primeira vez, elas devem aprender

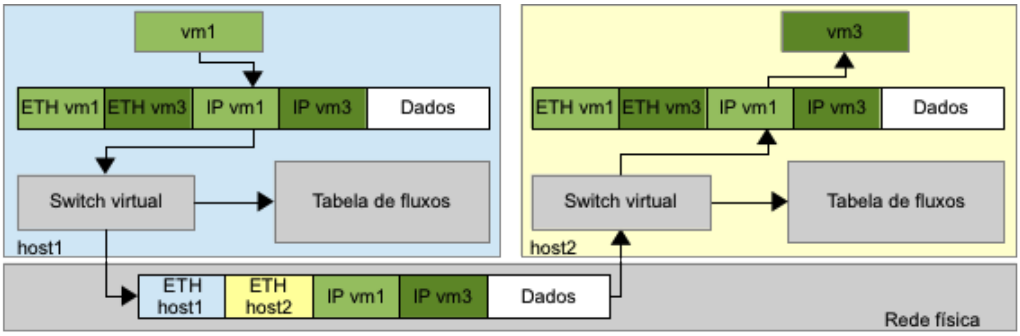


Figura 3. Re-escrita de cabeçalhos no *DCPortals*

o endereço MAC do destinatário. Em uma rede tradicional, isso seria conseguido através de uma mensagem de *broadcast* usando o protocolo ARP (*Address Resolution Protocol*). O protocolo se baseia em dois tipos de mensagens: *ARP request* e *ARP reply*. O primeiro é enviado para o endereço de *broadcast* da rede quando se precisa descobrir o endereço MAC associado a um certo endereço IP que se deseja contactar. O segundo é a resposta, enviada pela máquina alvo, informando seu endereço MAC.

Para o isolamento da rede virtual funcionar, não é aceitável que essas mensagens de *broadcast* viagem pela rede carregando os endereços de enlace das máquinas virtuais. *DCPortals* corrige esse problema interceptando toda comunicação ARP nos *switches* de borda e tratando-as diretamente. Já que o sistema tem acesso à base de dados do OpenStack, ele sabe como responder qualquer consulta ARP na rede. Tudo que ele precisa fazer é construir a mensagem de *ARP reply* apropriada e entregá-la diretamente à VM que iniciou uma consulta. Como as consultas são interceptadas no *switch* mais próximo do transmissor, as mensagens ARP nunca atravessam o núcleo da rede.

O protocolo DCHP, usado durante a configuração das máquinas virtuais na rede, também segue um padrão de funcionamento semelhante. Da mesma forma que no ARP, suas mensagens podem ser interceptadas e tratadas diretamente pelo *DCPortals* quando isso for interessante para o sistema.

3.5. Tratamento de outros *broadcasts*

Apesar da maioria das mensagens de *broadcast* em redes locais ser relacionada ao ARP ou ao DCHP, ainda devemos considerar como outras mensagens desse tipo serão tratadas (por exemplo, algum *broadcast* UDP gerado por uma aplicação). Quando um grupo de VMs é configurada em uma rede virtual, quaisquer pacotes desse tipo que ainda existam na rede devem ser entregues a todas as máquinas configuradas na mesma rede virtual — e apenas a elas. Entretanto, em um ambiente compartilhado complexo como uma rede de *datacenter*, esse não é o caso, já que tais pacotes seriam entregues a todas as máquinas conectadas à rede Ethernet física.

A figura 4 mostra a diferença entre o que acontece nesse caso em uma rede com e sem *DCPortals*. Na figura, máquinas virtuais de mesma cor representam máquinas em uma mesma rede virtual. VM_2 envia uma mensagem de *broadcast*. Idealmente, aquela mensagem deveria ser entregue apenas às outras máquinas na mesma rede virtual, VM_1, VM_4 e VM_8. Sem *DCPortals*, entretanto, todas as máquinas, independente de sua rede virtual, receberiam o pacote.

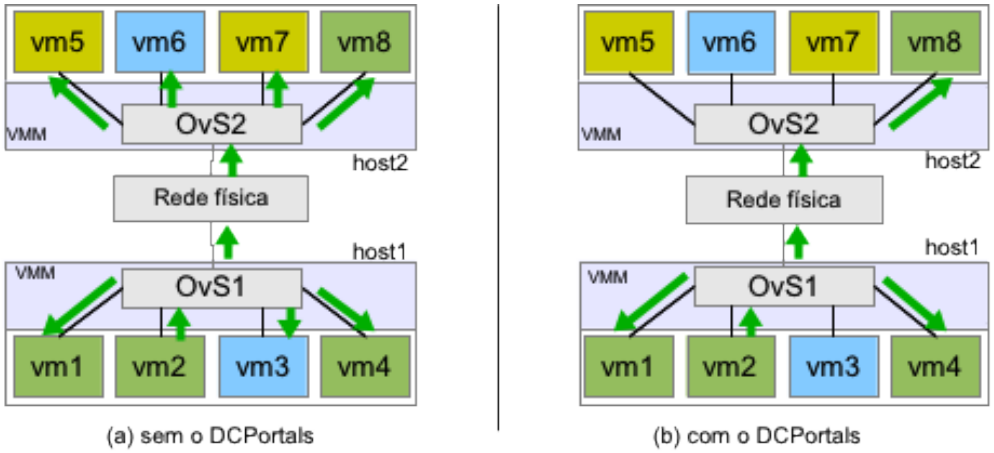


Figura 4. Diferença no tratamento de um *broadcast*.

Para conseguir o efeito desejado, primeiro o pacote de *broadcast* é inspecionado pelo *switch* virtual local ao hospedeiro que executa a VM transmissora. Ele é entregue, então, aos portos do *switch* que estejam conectados a outras VMs da mesma rede virtual e é enviado também pela interface externa real, se há outras VMs da mesma rede em outros hospedeiros físicos. Essa última mensagem alcançará a interface física de todos os hospedeiros da rede, como qualquer mensagem de *broadcast*. Dessa forma, ele atingirá todas as Open vSwitches na borda da rede. Cada uma delas verificará se existem outras VMs locais pertencentes à rede virtual de origem do pacote e entregarão o pacote apenas a tais VMs. No caso do exemplo da figura, o *switch* OvS_1 só entregará o pacote pelas portas conectadas às máquinas VM_1 VM_4, e à rede física.

Quando o pacote deixar a máquina hospedeira do transmissor, o processo de re-escrita de MAC age como anteriormente descrito, para remover o endereço MAC do transmissor, mas mantém o endereço de *broadcast* Ethernet como destino. Isso garantirá que todos os *switches* convencionais da rede física propagarão o pacote para toda a borda. Quando a mensagem de *broadcast* alcança um *switch* virtual da borda, o controlador recompõe o pacote original antes de entregá-lo às máquinas da mesma rede virtual. No nosso exemplo, o controlador comandaria o *switch* OvS_2 para entregar o pacote apenas pela porta conectada à VM_8.

Agindo assim, as mensagens continuam a ter o efeito de *broadcast*, mas serão entregues apenas às máquinas que realmente tenham sido configuradas como fazendo parte da rede virtual do transmissor, garantindo que nenhuma outra VM terá acesso a elas.

3.6. Ligação da rede virtual interna ao *datacenter* com a Internet

Na maioria dos casos, é necessário para cada máquina de uma aplicação em nuvem ter acesso ao mundo externo (à Internet), às vezes apenas para que o inquilino possa acessar e controlar suas máquinas, outras vezes para que todas possam oferecer uma interface para um serviço distribuído entre elas, como uma aplicação Web replicada, por exemplo. Os detalhes de como tais conexões são definidas dependem pesadamente da estrutura de cada *datacenter* e da sua política de serviço. Por exemplo, elas podem ser possíveis apenas

Máquina virtual	IP	Rede Virtual	Host físico
vm2	10.0.20.2	lan1	host1
vm3	10.0.20.3		
vm4	10.0.20.4		host2
vm5	10.0.20.5	lan2	
vm6	10.0.20.6		

Tabela 1. Distribuição de máquinas virtuais e redes entre as máquinas físicas usadas nos experimentos.

pela definição de uma segunda interface de rede em uma das máquinas virtuais, que seria a única máquina visível externamente e que passaria a agir como *gateway* entre as redes, responsável por rotear as mensagens entre a rede interna, virtual, e a rede externa.

Em sua implementação atual, *DCPortals* não requer que os inquilinos configurem uma máquina para lidar com o roteamento entre os dois domínios: ele seleciona uma máquina do *datacenter*, que pode ser a mesma responsável por executar o controlador, para rotear o tráfego entre a rede virtual e o restante da Internet. Essa máquina é capaz de "ver" todas as redes internas e tem regras específicas para controlar o tráfego que flui por ela (por exemplo, permitindo comunicação entre as redes de dois inquilinos de uma forma bem controlada). Dada a natureza flexível do paradigma SDN, esse *gateway* pode ser facilmente estendido para implementar regras de *firewall*, por exemplo.

4. Avaliação

Para realizar os experimentos de validação e confirmar a operação correta do sistema, utilizamos três máquinas, cada uma com duas interfaces de rede, conectadas a dois *switches* diferentes. Uma das redes resultantes foi usada para o tráfego de gerência (comandos OpenStack e OpenFlow) e a outra foi usada para a comunicação entre as VMs (rede operacional). Esse tipo de configuração é bastante usual em *datacenters* comerciais [Greenberg et al. 2009].

A rede de gerência usava endereços IP na faixa 10.0.254/24. As máquinas virtuais foram configuradas com duas redes virtuais isoladas, cada uma cobrindo duas máquinas físicas. Uma mesma faixa de endereços foi usada para todas as VMs, para acentuar a necessidade de isolamento de tráfego: sendo configuradas com endereços na mesma faixa, a não ser que algum mecanismo externo de isolamento esteja ativo, tráfego de cada VM poderia ser direcionado para qualquer outra máquina virtual. A distribuição de endereços entre as máquinas e de máquinas entre as redes é ilustrada na tabela 1.

Hospedeiros foram configurados com o sistema operacional Ubuntu versão 11.10 e os pacotes OpenStack, Xen e Open vSwitch obtidos dos repositórios oficiais. As máquinas virtuais foram configuradas com uma das imagens padrão do OpenStack, executando Ubuntu 10.10. Mais detalhes sobre a configuração desses sistemas está disponível em outro documento [Nunes 2012].

Usando esse ambiente nós executamos dois experimentos: o primeiro foi um teste de isolamento simples usando `ping` para o endereço de *broadcast* da rede e avaliação da interferência do sistema na latência de comunicação entre as VMs; o segundo testou o sistema sob um ataque de negação de serviço.

4.1. Verificação de isolamento e latência

O experimento consistiu em usar o comando `ping` para enviar uma mensagem *ICMP Request* para o endereço de *broadcast* da rede local a partir de uma das máquinas virtuais. O comportamento esperado para esse uso do `ping` é que cada máquina na mesma rede do transmissor deve responder com uma mensagem *ICMP Reply*, mesmo o transmissor[4]. Na saída gerada pelo `ping`, as múltiplas respostas para a mesma consulta devem aparecer com uma observação "DUP", já que no comportamento usual do programa elas são consideradas duplicatas (apesar das origens diversas). Com o *DCPortals*, mesmo com todas as máquinas configuradas na mesma faixa de endereços e compartilhando a mesma rede física, apenas VMs da mesma rede virtual do transmissor devem receber uma mensagem de consulta e, consequentemente, apenas elas devem responder. Nós iniciamos o comando `ping` de VM_2, na rede 1, e de VM_5, na rede 2. Sem *DCPortals*, ambas receberam respostas de todas as máquinas:

```
vm2:
    PING 10.0.20.255 (10.0.20.255) 56(84) bytes of data.
    64 bytes from 10.0.20.2: icmp_req=1 ttl=64 time=0.027 ms
    64 bytes from 10.0.20.4: icmp_req=1 ttl=64 time=2.60 ms (DUP!)
    64 bytes from 10.0.20.3: icmp_req=1 ttl=64 time=3.21 ms (DUP!)
    64 bytes from 10.0.20.6: icmp_req=1 ttl=64 time=3.22 ms (DUP!)
    64 bytes from 10.0.20.5: icmp_req=1 ttl=64 time=8.47 ms (DUP!)

vm5:
    PING 10.0.20.255 (10.0.20.255) 56(84) bytes of data.
    64 bytes from 10.0.20.5: icmp_req=1 ttl=64 time=0.027 ms
    64 bytes from 10.0.20.6: icmp_req=1 ttl=64 time=1.00 ms (DUP!)
    64 bytes from 10.0.20.3: icmp_req=1 ttl=64 time=1.86 ms (DUP!)
    64 bytes from 10.0.20.2: icmp_req=1 ttl=64 time=5.91 ms (DUP!)
    64 bytes from 10.0.20.4: icmp_req=1 ttl=64 time=10.5 ms (DUP!)
```

Isso confirma que realmente todas as máquinas são alcançáveis a partir das demais, apesar da configuração desejada em redes isoladas. Cada máquina recebeu uma cópia da mensagem de *broadcast* através da rede Ethernet compartilhada e respondeu diretamente ao transmissor.

Quando ativamos o *DCPortals* os resultados mudam, deixando claro que nem todas as máquinas foram alcançadas por cada mensagem de *broadcast*. A VM_2 recebeu apenas respostas das máquinas configuradas em sua própria rede virtual (inclusive de si própria), o mesmo acontecendo com VM_5. Isso confirma que, apesar de ocuparem a mesma faixa de endereços e compartilharem a mesma rede física, as máquinas em cada rede virtual foram isoladas corretamente.

```
vm2:
    PING 10.0.20.255 (10.0.20.255) 56(84) bytes of data.
    64 bytes from 10.0.20.2: icmp_req=1 ttl=64 time=0.026 ms
    64 bytes from 10.0.20.3: icmp_req=1 ttl=64 time=66.7 ms (DUP!)
    64 bytes from 10.0.20.4: icmp_req=1 ttl=64 time=88.5 ms (DUP!)

vm5:
    PING 10.0.20.255 (10.0.20.255) 56(84) bytes of data.
    64 bytes from 10.0.20.5: icmp_req=1 ttl=64 time=0.032 ms
    64 bytes from 10.0.20.6: icmp_req=1 ttl=64 time=103 ms (DUP!)
```

Traces de pacotes coletados em algumas das VMs (por consequência, dentro das suas redes virtuais) e na interface de rede das máquinas físicas (já no ponto em que pacotes

[4]Para esse experimento, o sistema operacional das VMs foi configurado para permitir mensagens ICMP para endereços de *broadcast*.

entram no núcleo da rede, após os *switches* virtuais da borda) confirmam o isolamento (detalhes omitidos por limitações de espaço).

O aumento dos tempos do ping com o *DCPortals* representam o custo de instalação do fluxo entre as duas VMs pelo sistema. Realizamos testes para avaliar o *overhead* da operação do controlador SDN, tanto durante o estabelecimento de um fluxo quando durante a transmissão de dados por um fluxo estabelecido. Houve, em média, um *overhead* de 7 ms em consultas ARP e 47 ms para o estabelecimento de um fluxo. Tais overheads estão limitados aos primeiros pacotes de um fluxo apenas. Uma vez estabelecida a entrada na tabela de encaminhamento, não houve diferenças significativas entre os dois sistemas.

4.2. Proteção contra ataques de negação de serviço

Uma motivação comum para a demanda de isolamento entre redes virtuais é a ameaça de que um inquilino possa iniciar um ataque de negação de serviço direcionado às máquinas de um outro inquilino. Em um ambiente de *datacenter* onde não há esse isolamento, um fluxo UDP criado de uma máquina atacante para uma rede alvo pode consumir banda da rede a ponto de fazer com que o serviço do inquilino atacado deixe de ser acessível. Um caso reportado semelhante, apesar da origem ter sido externa, ocorreu com o serviço BitBucket, enquanto ele usava a infraestrutura Amazon EC2 [BitBucket Attack 2012].

Tal problema não deveria ocorrer se as redes virtuais dos inquilinos fossem adequadamente isoladas, como é nosso objetivo com *DCPortals*. Para comprovar isso, neste experimento criamos um ataque UDP, direcionado a outra rede virtual. Para tornar o problema ainda mais grave, o fluxo UDP foi criado para o endereço de *broadcast* da rede destino, visando atingir todas as máquinas. A figura 5 mostra a configuração usada neste caso. Máquinas virtuais VM_2 e VM_3, localizadas no hospedeiro físico $host_1$ estabelecem uma transferência TCP entre eles. Ao mesmo tempo, VM_5, o atacante, inicia um fluxo UDP endereçado para o endereço de *broadcast* da rede. Usamos `iperf` para gerar os dois fluxos e medimos a vazão efetiva da conexão TCP entre VM_2 e VM_3. Limitamos a banda máxima de cada VM a 1 Gbps, um valor usual, equivalente à tecnologia da rede física mais comum atualmente.

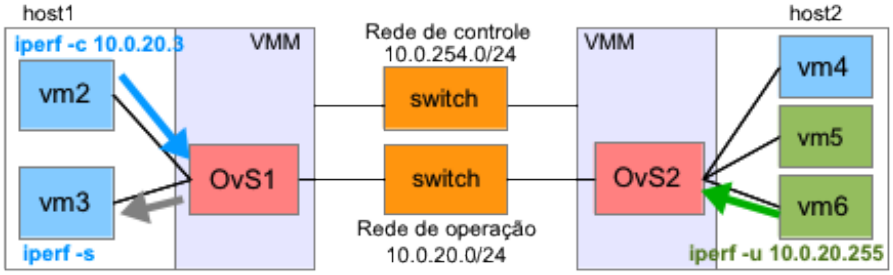


Figura 5. Experimento de negação de serviço. VM_6 inicia um alagamento UDP para o endereço de *broadcast* da rede, enquanto VM_2 e VM_3 (em outra rede virtual) estabelecem um fluxo TCP entre elas.

Cada teste durou 1.000 segundos e a vazão TCP foi medida a cada 3 segundos. Os primeiros 3 segundos foram descartados para eliminar ruídos devido a variações no disparo dos fluxos. Resultados são mostrados na tabela 2.

Cenário	Banda (Mbps)
Sem isolamento, sem ataque	928,29 ± 0,16
DCPortals, sem ataque	928,21 ± 0,22
Sem isolamento, sob ataque	41,21 ± 12,99
DCPortals, sob ataque	910,11 ± 0,28

Tabela 2. Vazão média do fluxo TCP observada em diferentes condições; erro considera intervalo de confiança de 99%.

Claramente, podemos ver a diferença entre os dois casos. O sistema sem isolamento sofre uma perda de aproximadamente 95% da vazão observada. Por outro lado, *DCPortals* sofre apenas uma perda de 5%. Com o isolamento, o fluxo UDP é bloqueado nos *switches* de borda, não sendo entregue às outras redes. A perda nesse caso é devida ao *overhead* no *switch* de borda para descartar os pacotes UDP que chegam. Considerando o sistema sem ataques, verificamos que não há diferença estatística entre os dois casos. Isso é o esperado, considerando-se a observação anterior sobre o baixo *overhead* durante a duração dos fluxos.

4.3. Escalabilidade

Durante os experimentos não tivemos recursos para realizar testes de escalabilidade extensos. Entretanto, acreditamos que a solução proposta tem boa escalabilidade. Em primeiro lugar, a operação do sistema e o encaminhamento dos fluxos já identificados reside nas Open vSwitches em cada máquina física que, por construção, estão entre os comutadores virtuais mais eficientes na atualidade [Pfaff et al. 2009]. Por outro lado, o controlador POX é capaz de processar mais de 30.000 fluxos por segundo, mais que suficiente para as demandas do *DCPortals* em um grande *datacenter*, com até dezenas de milhares de máquinas[5].

5. Trabalhos relacionados

Greenberg *et al.* [Greenberg et al. 2009] apresentam um interessante estudo sobre custos de um *datacenter* para computação em nuvem. Entre outras observações, eles identificam a rede como um dos principais desafios nesse contexto. Em particular, mencionam explicitamente a dependência as soluções atuais em relação a VLANs e os problemas associados a essa prática.

Uma das primeiras iniciativas visando uma técnica de isolamento de redes virtuais foi desenvolvida por um grupo do HP Labs [Cabuk et al. 2007], começando com a definição de domínios virtuais confiáveis (*Trusted Virtual Domains*, TVDs). Estes seriam seções de rede logicamente isoladas, de forma independente da topologia da infraestrutura. Para implementar esse isolamento, os autores criaram um módulo interno às máquinas virtuais que é responsável por todo o processamento relacionado ao isolamento de rede. Duas técnicas de isolamento, rótulos de VLANs e encapsulamento EtherIP, são comparadas. Aquele trabalho tem um objetivo semelhante ao *DCPortals*, mas as soluções consideradas têm limitações de escalabilidade e exigem alterações intrusivas no hipervisor (Xen). Um estudo comparativo mais longo do mesmo grupo menciona a técnica de re-escrita de endereços MAC no mesmo contexto das duas outras [Cabuk et al. 2008].

[5]http://www.noxrepo.org/pox/about-pox/

DCPortals usa o paradigma de redes definidas por software para resolver o problema de isolamento. Pettit *et al.* [Pettit et al. 2010] já discutiram a viabilidade desse tipo de enfoque para redes de *datacenters*, mas não apresentaram uma solução concreta. Duas aplicações do controlador SDN NOX, a *datacenters* foram apresentadas anteriormente, mas elas foram focadas na implementação de novas arquiteturas de rede [Tavakoli et al. 2009, Heller et al. 2010]. Diferente dessas soluções, *DCPortals* não exigem hardware com recursos OpenFlow no núcleo da rede e foca especificamente no isolamento de tráfego.

Um trabalho com motivações bastante similares ao aqui apresentado é certamente Netlord, desenvolvido por Mudigonda *et. al.* [Mudigonda et al. 2011]. Naquele trabalho, os autores usam uma solução baseada em tunelamento para obter isolamento sem exigir hardware especial na rede. Entretanto, a forma como aquela solução é implementada é bem diferente, ao usar uma extensão do hipervisor Xen especialmente desenvolvida para esse fim. Nós acreditamos que o uso de redes definidas por software é um enfoque mais elegante e flexível, sendo uma característica determinante do nosso trabalho. Isso simplifica a implementação e oferece um leque mais amplo de possibilidades de aplicação. *DCPortals*, por exemplo, funciona diretamente não apenas com Xen, mas com qualquer outro hipervisor que use a biblioteca libvirt e Open vSwitch, como é o caso do KVM.

6. Conclusão

Este trabalho descreve *DCPortals*, um sistema desenvolvido para fornecer isolamento de tráfego a redes virtuais em um ambiente de *datacenter* virtualizado. A arquitetura do sistema e os detalhes de implementação foram apresentados, bem como resultados que confirmam o isolamento oferecido, inclusive no contexto de um ataque de negação de serviço entre inquilinos. Avaliações também quantizaram o *overhead* durante o estabelecimento de fluxos, que são visíveis mas raros, e mostraram que durante operação normal os custos por fluxo são desprezíveis.

Como trabalhos futuros, continuamos melhorando o sistema; considerando sua integração com OpenStack, pretendemos estudar sua integração com Quantum, o componente de OpenStack recentemente anunciado para integração com controladores OpenFlow. Pretendemos também trabalhar na integração de *DCPortals*, que oferece isolamento de redes, com *Gatekeeper*, um sistema projetado para oferecer garantias de qualidade de serviço em uma rede de *datacenter* [Rodrigues et al. 2011].

Agradecimentos

Este trabalho foi parcialmente financiado pelo UOL (www.uol.com.br), através do programa UOL Bolsa Pesquisa, Fapemig, CNPq e Instituto Nacional de Ciência e Tecnologia da Web, InWeb (MCT/CNPq 573871/2008-6).

Referências

BitBucket Attack (2012). Bitbucket amazon ddos attack. `http://blog.bitbucket.org/2009/10/04/on-our-extended-downtime-amazon-and-whats-coming/`. Acessado em julho de 2012.

Cabuk, S., Dalton, C. I., Eduards, A. e Fischer, A. (2008). A comparative study on secure network virtualization. Technical Report HPL-2008-57, HP Laboratories.

Cabuk, S., Dalton, C. I., Ramasamy, H. e Schunter, M. (2007). Towards automated provisioning of secure virtualized networks. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, págs. 235–245, New York, NY, USA. ACM.

Casado, M., Koponen, T., Ramanathan, R. e Shenker, S. (2010). Virtualizing the network forwarding plane. In *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, PRESTO '10, págs. 8:1–8:6, New York, NY, USA. ACM.

Greenberg, A., Hamilton, J., Maltz, D. A. e Patel, P. (2009). The cost of a cloud: research problems in data center networks. *SIGCOMM Computer Communication Review*, 39(1):68–73.

Heller, B., Erickson, D., McKeown, N., Griffith, R., Ganichev, I., Whyte, S., Zarifis, K., Moon, D., Shenker, S. e Stuart, S. (2010). Ripcord: a modular platform for data center networking. *SIGCOMM Comput. Commun. Rev.*, 40:457–458.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. e Turner, J. (2008). Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38:69–74.

Mudigonda, J., Yalagandula, P., Al-Fares, M. e Mogul, J. C. (2010). Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, págs. 1–16, Berkeley, CA, USA. USENIX Association.

Mudigonda, J., Yalagandula, P., Mogul, J., Stiekes, B. e Pouffary, Y. (2011). Netlord: a scalable multi-tenant network architecture for virtualized datacenters. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, págs. 62–73, New York, NY, USA. ACM.

Nunes, R. V. (2012). Uma aplicação de redes definidas por software para a gerência de redes de datacenters virtualizados. Master's thesis, DCC/UFMG.

Pettit, J., Gross, J., Pfaff, B., Casado, M. e Crosby, S. (2010). Virtual switching in an era of advanced edges. In *Proceedings of the 2nd Workshop on Data Center - Converged and Virtual Ethernet Switching (DC CAVES)*, DC CAVES, págs. 1–7, Amsterdam, The Netherlands. ITC.

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M. e Shenker, S. (2009). Extending networking into the virtualization layer. In *8th ACM Workshop on Hot Topics in Networks (HotNets-VIII)*.

Rodrigues, H., Soares, P., Santos, J. R., Turner, Y. e Guedes, D. (2011). Isolamento de tráfego em ambientes virtualizados. In *Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas DIstribuídos (SBRC)*, págs. 1–14. SBC.

Tavakoli, A., Casado, M., Koponen, T. e Shenker, S. (2009). Applying NOX to the datacenter. In *Proceedings of workshop on Hot Topics in Networks (HotNets-VIII)*.

Redes Orientadas a Conteúdo Baseadas em Controladores Hierárquicos

João Vitor Torres, Lyno Henrique G. Ferraz, Otto Carlos M. B. Duarte

[1]Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ
Rio de Janeiro, Brazil
Email: {jvitor, lyno, otto}@gta.ufrj.br

Abstract. *Content Centric Network (CCN) routing schemes must learn routes to named data locations, so routers know where to send interest packets. Nevertheless, the huge amount of named data and non-aggregated prefixes challenge path evaluation, because routers proportionally store more routes and exchange more control messages. This article proposes a routing scheme based on control plane separation assuring router memory consumption proportional to traffic and reducing control message exchange for routing flatly named data. The routing scheme employs a hierarchy of multiple controllers which have two main functions: i) acquire topology and calculate routes, and ii) store named data locations. Distributed hash tables distribute the storage of named data locations efficiently. Furthermore, as the proposal runs on top of CCN, it preserves Content Centric Network features. The initial assessment points that the data location distribution and on demand route installation scheme produces scalable routing of named data.*

Resumo. *As Redes Orientadas a Conteúdo precisam aprender a localização dos conteúdos para definir rotas e encaminhar os pacotes de interesse. Contudo, a enorme quantidade de prefixos distintos de nome de conteúdo constitui um desafio para tarefa de localização, pois amplia proporcionalmente o número de rotas por roteador, e também de mensagens de controle trocadas pelos protocolos de roteamento. Este artigo propõe um esquema de roteamento para redes orientadas a conteúdo baseado em separação de planos que garante consumo de memória no roteador proporcional ao tráfego e número reduzido de mensagens de controle no roteamento de conteúdo com nomes planos. O esquema de roteamento proposto utiliza uma hierarquia de múltiplos elementos controladores com duas funções principais: i) monitorar a topologia e calcular as rotas, e ii) armazenar a localização dos conteúdos. Tabelas hash distribuídas armazenam as localizações dos conteúdos e todos os pacotes são orientados ao conteúdo, preservando as propriedades das Redes Orientadas a Conteúdo. A análise inicial indica que a proposta de separar e dividir a tarefa de localização de conteúdos aliada à instalação de rotas por demanda fornece roteamento de nomes planos com uso escalável e eficiente de recursos.*

1. Introdução

As Redes Orientadas a Conteúdo (*Content Centric Network* - CCN) [Jacobson et al. 2009b, Zhang et al. 2010] mudam drasticamente os princípios de

Este trabalho foi realizado com recursos da FINEP, FUNTTEL, CNPq, CAPES, FUJB, FAPERJ, CTIC e UOL.

roteamento, 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. Um dos principais desafios da rede orientada a conteúdo é a escalabilidade da localização e do roteamento 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. Esta estrutura permite a agregação dos nomes de conteúdo em seus prefixos comuns na direção do nível mais alto da hierarquia e a divulgação concisa de sumários de localização.

Para garantir esta agregação, a localização do conteúdo fica restrita a sua posição na hierarquia de nomes e não são divulgadas rotas para cópias fora do caminho até a fonte. Sob estas premissas, a proposta CCN usa esquemas de roteamento baseados em anúncios de prefixos de nomes tornando-os roteáveis. Estes esquemas, como por exemplo, *Open Shortest Path First for Named Data Network* (OSPFN) [Wang et al. 2012], herdam as características do IP vinculadas ao roteamento por disseminação e agregação de prefixos. Estas abordagens são sensíveis em relação à quantidade de prefixos não agregados de nomes de conteúdo, bem maior do que de prefixos IP, pois consomem recursos ao proativamente (periodicamente) difundir na rede a localização de todos os conteúdos. Além disso, a crescente mobilidade, a replicação e a hospedagem multidomicílio de conteúdos desagregam prefixos e aumentam ainda mais o consumo de recursos, pois exigem alto tráfego de controle e grandes tabelas de encaminhamento, o que as torna economicamente inviáveis atualmente [Perino e Varvello 2011].

Este artigo propõe um esquema de roteamento para Redes Orientadas a Conteúdo (CCN) baseado em controladores hierárquicos. A ideia básica é garantir a escalabilidade e uma melhor utilização de recursos através da separação e estruturação do plano de controle em uma hierarquia de elementos controladores para monitorar a topologia, calcular rotas entre nós, localizar e rotear conteúdos. A hierarquia divide cada nível em zonas permitindo sua programação conjunta a partir do nível superior. Em cada zona, inicialmente são instaladas rotas entre os nós roteadores e controladores. Em seguida os conteúdos são registrados e, finalmente as rotas para conteúdos podem ser instaladas por demanda dos consumidores de conteúdo.

O esquema de roteamento proposto não requer novas mensagens e, portanto, utiliza apenas os pacotes de interesse e os de dados próprios da rede CCN, preservando integralmente todas as propriedades CCN tais como: armazenamento local, controle de congestionamento, detecção de falha de rede e diversidade de caminhos. A semântica dos pacotes de interesse é estendida para reduzir a sobrecarga de controle ao embutir comandos de controle nos seus prefixos.

O armazenamento da localização de conteúdos é balanceado entre diversos elementos de tabelas *hash* distribuídas (*Distributed Hash Table* - DHT), sendo utilizada uma DHT em cada zona da hierarquia. No nível superior da hierarquia de controle, um processo distribuído garante a troca de informação entre controladores e convergência da rede. Isto permite a entrada e saída de conteúdos, roteadores, controladores e membros da DHT com redistribuição de carga.

O restante deste artigo está organizado da forma a seguir. Na Seção 2 os principais trabalhos relacionados são apresentados. A proposta é detalhada na Seção 3. Na Seção 4 discute-se o impacto das mensagens de controle, a mobilidade, a segurança, a falha e o particionamento da rede e a distribuição de carga. Por fim, na Seção 5 ressalta-se as principais conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Jacobson *et al.* propuseram o paradigma de paridade entre pacotes de interesse e de dados através do modelo de rede orientada a conteúdo (*Content Centric Networking* - CCN) [Jacobson et al. 2009b, Jacobson et al. 2009a]. A sua proposta resultou no projeto *Named Data Network* (NDN), cujo objetivo é construir uma nova arquitetura de rede [Zhang et al. 2010]. O modelo CCN permite, através de uma camada de estratégia, a rápida detecção de problemas na rede e uso de caminhos alternativos [Yia et al. 2012]. Contudo, 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].

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. Este esquema utiliza um sistema de DHT para prover este mapeamento [Baid et al. 2012]. D'Ambrosio *et al.* utiliza um esquema de múltiplas DHTs organizadas em hierarquia, na qual cada DHT mapeia a localização do conteúdo em identificadores topológicos de diferentes níveis, como por exemplo, máquina, rede e sistema autônomo [D'Ambrosio et al. 2011]. Este esquema distribui a tarefa de mapeamento em camadas e reduz o número de identificadores em cada nível.

Outras propostas também focam o conteúdo como o CCN, mas utilizam uma abordagem baseada numa arquitetura de subscrição com publicador e assinante [de Brito et al. 2012]. Nestas abordagens, ao contrário do CCN, o roteamento é iniciado do produtor para o consumidor de conteúdo e a tarefa de escalabilidade é entregar o conteúdo apenas para os respectivos assinantes. Carzaniga *et al.* compara a requisição de conteúdo por demanda e o balanço de fluxo do CCN com abordagens de subscrição sem este suporte, mas capazes de assinar canais de conteúdo com uma única requisição. Eles apresentam uma solução híbrida com uso reduzido e seletivo de estado nos roteadores quanto aos fluxos de pacotes [Carzaniga et al. 2011].

Propostas do tipo redes definidas por *software* (*Software Defined Networks* - SDN) empregam um controlador para, por demanda, instalar nos nós da rede regras de encaminhamento de pacotes por fluxo [Mattos et al. 2011, Fernandes et al. 2011, McKeown et al. 2008]. 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. As redes definidas por *software* são boas candidatas para redes da próxima geração [FITS 2012].

A proposta CCN garante o balanço de fluxo de pacotes através da paridade interesse e dado, o que é fundamental para prover adaptabilidade a falhas. Porém, os protoco-

los de roteamento tradicionais não são escaláveis para os requisitos de quantidade de prefixos distintos do CCN. O uso de DHTs nas propostas listadas acima fornece um serviço de resolução de nome, mas o encaminhamento dos pacotes é baseado nos identificadores topológicos de rede perdendo as propriedades CCN. As propostas SDN com separação de planos utilizam identificadores topológicos de rede para encaminhamento de pacotes e também perdem as propriedades CCN. Este artigo une a resolução de localização do conteúdo baseado em DHTs, a qual permite nomes planos de forma escalável, com a separação de planos e cálculo de rotas por controladores, os quais instalam por demanda nos comutadores regras de encaminhamento de pacotes baseadas diretamente nos nomes de conteúdo e mantém as propriedades CCN. A instalação de regras utiliza ainda pacotes de interesse com semântica estendida reduzindo a sobrecarga de comunicação ao embutir semântica adicional nos prefixos, prática introduzida na proposta Voice over CCN (VoCCN) [Jacobson et al. 2009a].

3. O Esquema de Roteamento Proposto

O esquema de roteamento proposto não requer nenhum tipo de pacote adicional e, portanto, é integralmente suportado apenas com os pacotes CCN de interesse e dados. O encaminhamento de pacotes em cada roteador segue o processamento padrão CCN: i) buscando e armazenamento cópias de conteúdo no repositório CS (*Content Store*), ii) utilizando as entradas na PIT (*Pending Interest Table*) para indicar o caminho reverso ao do interesse e, encaminhar para interface de saída na direção do consumidor do conteúdo o pacote de dados e, iii) encaminhar o pacote de interesse de acordo com a FIB (*Forwarding Information Base*) [Zhang et al. 2010]. O controle da rede se baseia na separação dos planos de dados e controle, usando, portanto, elementos específicos, chamados controladores, para instalação das rotas de encaminhamento no plano de dados, ou seja, as entradas na FIB dos roteadores. A interface de comunicação entre roteadores e controladores é baseada em pacotes de interesse nos quais se estendeu a semântica para obter tratamento específico nos nós da rede.

Existem três tipos de elementos de rede: roteador, controlador e nó de DHT. Os roteadores têm a função básica de encaminhamento de pacotes e, em especial, os roteadores diretamente conectados aos produtores de conteúdo são responsáveis pelo registro nos controladores da localização de conteúdos. Os controladores por sua vez, além de instalar regras na FIB dos roteadores, armazenam a localização dos conteúdos e calculam as entradas FIB de cada roteador até estes conteúdos. Este armazenamento é realizado utilizando um sistema de DHT.

Neste artigo, um nome ou prefixo de conteúdo se aplica a qualquer dado endereçável e alcançável, seja um arquivo, um serviço ou uma máquina na rede. Todos os roteadores e controladores possuem um ID, portanto são também endereçáveis na rede.

Os prefixos especiais utilizados no esquema são listados a seguir. O prefixo `"/HELLO"` busca o ID dos nós vizinhos imediatos; `"/router"` é seguido do ID do roteador e encaminha interesses para um roteador específico; `"/route"` é um comando de instalação de rota a ser processado pelos roteadores; `"/controller"` encaminha interesses para qualquer controlador e, se seguido do ID do controlador, aponta para um controlador específico; finalmente, o prefixo `"/register"` indica requisição de regis-

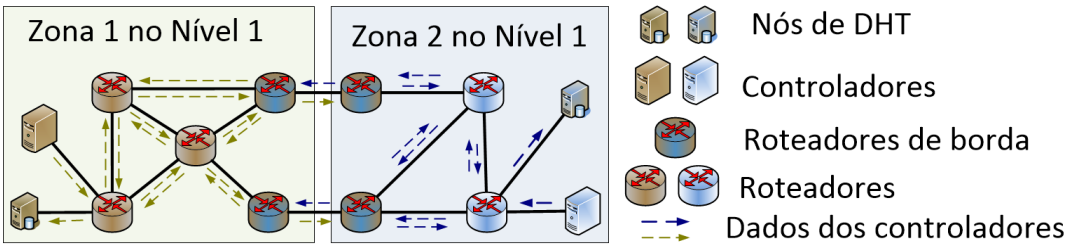


Figura 1. Exemplo de divisão de rotadores em zonas com um controlador, um nó DHT por zona e roteadores de borda. Os pacotes de interesse são omitidos para simplificação.

tro de novo nome de conteúdo.

Os roteadores inicialmente não tem nenhuma regra de encaminhamento na FIB, exceto as regras que endereçam a si próprios para processamento local, como `"/router/routerID"`, `"/route"`, `"/HELLO"` e `"/register"`.

A seguir, a estrutura de controladores hierárquicos e os procedimentos necessários para sua operação são apresentados. O uso de controladores hierárquicos tem três objetivos principais: i) reduzir o número de elementos de roteamento num único processo distribuído plano, ii) isolar as falhas as restringindo aos níveis correspondentes e iii) visar menor tempo de convergência.

3.1. Controladores Hierárquicos

Os roteadores da rede são divididos em zonas, sendo cada zona controlada por um controlador, ver Figura 1. Por sua vez, os controladores de zona são agrupados em zonas com nível hierárquico mais elevado, Zonas de nível 2, e um controlador gerencia um grupo de controladores de Nível 2 formando uma Zona de Nível 3. Este agrupamento hierárquico e sucessivo permite adicionar novos níveis conforme necessário, ver Figura 2. No nível mais baixo, cada controlador é responsável pelas rotas entre roteadores internos a sua zona. Para rotas externas a sua zona, o controlador consulta o seu controlador de nível hierárquico superior para obter o próximo salto. No nível de hierarquia mais alto, cada controlador obtém os seus controladores vizinhos de mesmo nível e troca diretamente com estes vizinhos informações sobre a topologia de zonas possibilitando montar rotas externas a sua zona sem auxílio de um controlador superior.

Cada zona hierárquica utiliza uma DHT para armazenar a localização dos conteúdos alcançáveis na hierarquia de nível inferior, onde o *hash* do prefixo do conteúdo é a chave da DHT e o ID do controlador responsável pelo prefixo é o valor da entrada correspondente da DHT. No nível mais baixo o valor desta entrada é o ID do roteador. A DHT é composta por elementos auxiliares proporcionando distribuição de carga e escalabilidade. A distribuição de chaves entre nós da DHT é ordenada de acordo com o seu procedimento particular e não impõe uma implementação específica.

3.1.1. Divisão e Topologia de Zonas

A divisão dos roteadores em zonas é definida pela ordem temporal de chegada das respostas dos controladores à inundação de pacotes de interesse iniciada para descoberta de controladores, procedimento denominado Descoberta de Controladores, de-

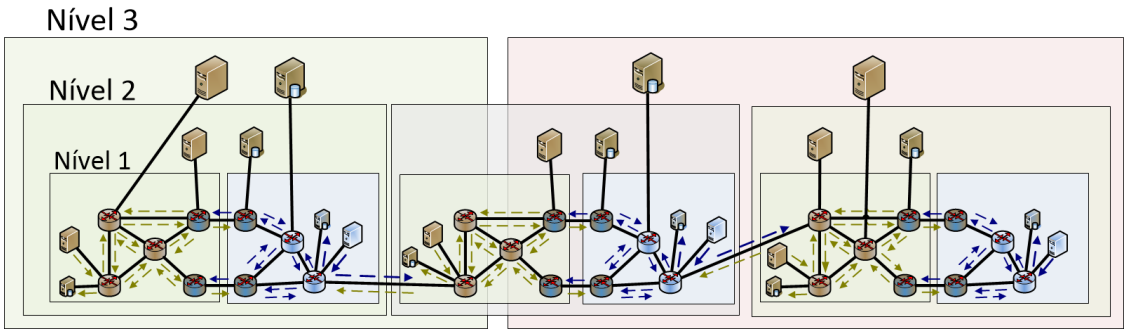


Figura 2. Exemplo em 3 níveis de divisão em controladores hierárquicos em zonas.

talhado a frente. Este procedimento atribui um controlador responsável por cada zona. Estes controladores, chamados de Controladores de Nível 1, replicam o procedimento de inundação na busca de seus controladores de nível superiores usando prefixos especiais do tipo `"/controller/levelX"`, onde `"X"` indica o nível superior. Assim, o procedimento de divisão em zonas se repete em cada nível. Este procedimento de inundação é repetido periodicamente e garante que os roteadores na borda entre zonas obtenham o controlador recebido pelo roteador vizinho. Os roteadores de borda informam seu controlador sobre os múltiplos controladores encontrados de cada nível definindo o último salto para a zona vizinha. Os controladores do nível `"X"` se registram nos controladores de nível `"X+1"` e, propagam a informação de vizinhança informada pelos roteadores.

No nível mais alto da hierarquia, para obter a topologia de controladores na rede, cada controlador envia para seus controladores vizinhos um pacote de interesse com o prefixo `"/controller/levelY/controllerID/topology"`. O controlador responde com a lista de controladores conhecidos e suas adjacências, permitindo o cálculo de rotas entre zonas. Além disso, os controladores também informam os controladores já conhecidos sobre novos controladores com um pacote de interesse para o prefixo `"/controller/controllerID/topologyUpdate"`. O controlador receptor confirma o recebimento e solicita a topologia atualizada.

3.1.2. Composição de Caminhos

Para comunicação entre zonas, cada controlador consulta seu controlador superior buscando a zona de próximo salto na direção do destino. No nível mais alto, todos controladores conhecem a topologia superior de zonas e calculam localmente a zona de próximo salto. Então, a zona de próximo salto é refinada a cada nível pelos controladores na hierarquia abaixo e, com a zona de próximo salto, o controlador de Nível 1 calcula e instala uma rota até o roteador de borda da sua zona na direção da zona de próximo salto. O roteador de borda da zona seguinte consulta seu próprio controlador para encaminhar adiante.

3.1.3. Operação da DHT

Os nós da DHT de cada zona se registram diretamente no respectivo controlador, o qual encaminha os pedidos e registros de conteúdo para o nó adequado. Os controladores

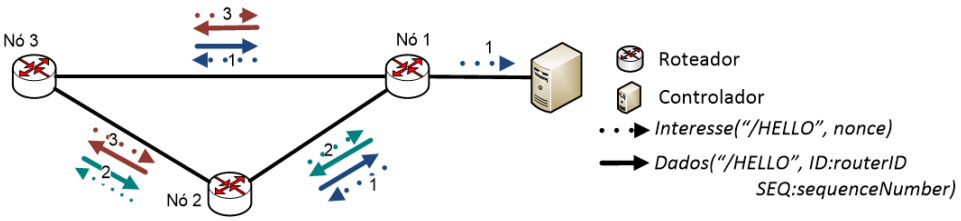


Figura 3. A Descoberta de Vizinhos através de difusão de pacotes de interesse `"/HELLO"`. Os números nas setas indicam o gerador do pacote de interesse.

do nível mais alto divulgam entre si os seus nós da DHT formando uma única DHT. Quando um nó entra ou sai da DHT, as chaves armazenadas por ele são redistribuídas. Para conteúdos acessíveis em zonas fora da hierarquia inferior, o controlador propaga o registro para o controlador e DHT superiores acrescentando seu ID ao registro. O nível de registro e busca de conteúdos é incluído no interesse para evitar a busca recursiva em todos os níveis. Os prefixos específicos para manipulação dos nós da DHT e seus registros não são ilustrados aqui por limitação de espaço.

3.2. Inicialização e Monitoração de Topologia

Nesta fase, os roteadores localizam o controlador e se registram. Com os registros, o controlador adquire a topologia da rede e calcula rotas até os roteadores dentro da sua zona. A fase possui três procedimentos: a Descoberta de Vizinhos com Hello, a Descoberta de Controlador e o Registro de Roteador. Estes procedimentos rodam periodicamente para manter uma visão atualizada da topologia. A seguir os procedimentos e algoritmos para descoberta de rotas são descritos.

3.2.1. O Procedimento de Descoberta de Vizinhos com Hello

Para descobrir seus vizinhos imediatos, os roteadores enviam periodicamente pacotes de interesse em todas as interfaces. Cada roteador vizinho responde com um pacote de dados contendo seu próprio ID e um número de sequência. A Figura 3 ilustra o fluxo de pacotes. O número de sequência fornece coerência de topologia, sendo atualizado sempre que um roteador detecta alteração de um vizinho ou interface. O roteador recebe o pacote de dados e instala uma entrada na FIB para o prefixo `"/router/routerID"` apontando para a interface de recebimento. O routerID, o sequenceNumber, a interface de conexão com o vizinho e uma métrica para a interface são armazenados em uma tabela de vizinhos. Um exemplo de métrica é o tempo de ida e volta até o vizinho, calculado a partir do intervalo entre envio do interesse e recebimento do pacote de dados. Através deste procedimento, todos os roteadores tem uma lista atualizada de vizinhos.

3.2.2. O Procedimento de Descoberta de Controlador

No procedimento de Descoberta de Controlador, os roteadores inundam a rede com pacotes de interesse por múltiplos caminhos até o controlador. O caminho de resposta mais rápida é escolhido por cada roteador. Estes caminhos permitem acessar o controlador e registrar os roteadores, conforme descrito na próxima seção.

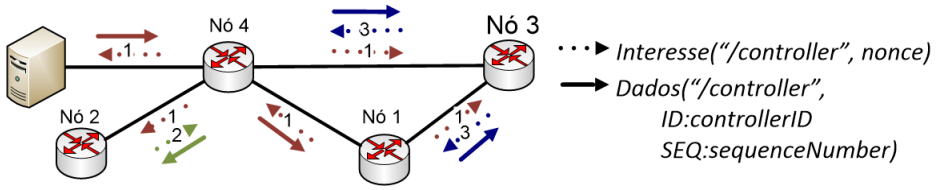


Figura 4. A Descoberta de Controlador através de inundação de pacotes de interesse para descoberta de caminhos. Inundação do nó 4 omitida.

Na inundação[1], os roteadores enviam pacotes de interesse de descoberta do controlador em todas as interfaces. Cada roteador, ao receber este pacote de interesse, adiciona uma entrada na sua *Pending Interest Table* (PIT) e responde a partir de sua CS ou encaminha o interesse para todas as demais interfaces. O controlador, ao receber o interesse, responde com um pacote de dados contendo seu ID e um número de sequência. O número de sequência é atualizado pelo controlador em intervalos ajustáveis de tempo. O roteador armazena o pacote de dados recebido na sua CS por um pequeno período e o encaminha de acordo com a PIT. A Figura 4 ilustra o fluxo de pacotes. O roteador instala ainda entradas na FIB para o prefixo `"/controller"` e `"/controller/controllerID"` através da interface de entrada do pacote de dados e armazena o ID e o número de sequência do controlador em uma tabela de controladores. Todos os roteadores enviam ou retransmitem o interesse, garantindo o recebimento do pacote e dados e a obtenção de rotas para o controlador. Pacotes de interesse atrasados são respondidos diretamente da CS reduzindo a sobrecarga do procedimento de inundação.

Periodicamente, cada roteador envia novos pacotes de interesse para descoberta e atualização de rotas para o controlador. Caso a resposta contenha um número de sequência igual ou inferior ao armazenado na sua tabela de controladores, o pacote de dados é assumido como antigo e o roteador não atualiza a FIB.

3.2.3. O Procedimento de Registro de Roteador

Assim que obtém o ID do controlador e um caminho para alcançá-lo, o roteador se registra no controlador enviando um pacote de interesse de semântica estendida com esta indicação. A Figura 5 ilustra o fluxo de pacotes, onde `"controllerID"` é o ID do controlador conhecido na tabela de controladores, `"registerRouter"` é uma palavra reservada que indica o pedido de registro para o roteador de ID `"routerID"`, e `"seq/sequenceNumber"` é o último número de sequência anunciado na Descoberta de Vizinhos. Cada roteador que encaminha o pacote de interesse em direção ao controlador, instala uma entrada na sua FIB para o prefixo `"/router/routerID"` através da interface de entrada do interesse. O controlador responde o interesse com um pacote de dados de confirmação e cria uma entrada na sua tabela de roteadores.

Após o registro, o controlador obtém os vizinhos deste roteador enviando um interesse específico para o roteador. Os roteadores no caminho já possuem entradas na FIB para o `"routerID"` e encaminham o interesse diretamente. O roteador `"routerID"` responde com um pacote de dados contendo o seu próprio ID, um número de sequência

[1]Não há sincronia entre roteadores para início simultâneo da inundação.

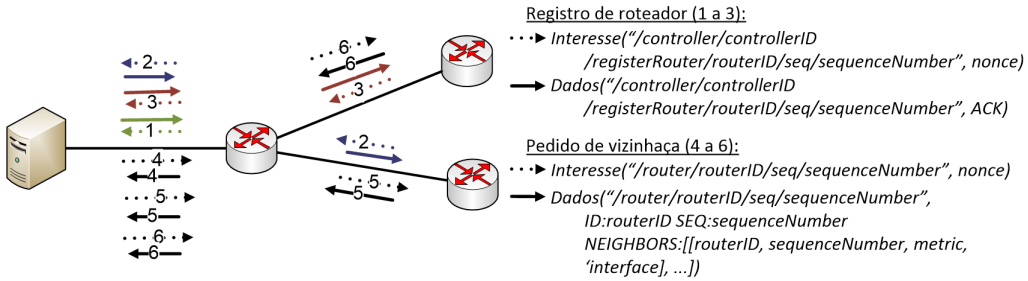


Figura 5. O controlador obtém a topologia da rede recebendo os registros dos roteadores e solicitando a lista de vizinhos de cada roteador.

e a sua lista de vizinhos, conforme Figura 5. Este pacote de dados atualiza o controlador com a lista de vizinhos do roteador. O número de sequência fornece ao controlador um parâmetro para comparar a coerência das informações enviadas por diferentes roteadores. Caso seja diferente, o controlador envia novo pacote de interesse para os roteadores que anunciaram o número de sequência antigo. Sempre que um roteador detecta uma mudança de topologia, o procedimento de registro é repetido, o número de sequência incrementado e o controlador obtém a topologia atualizada.

3.3. Roteamento para Obter Conteúdo

Os procedimentos da fase de Inicialização e Monitoração de Topologia permitem a todos os roteadores alcançar o controlador, atualizar a entrada e saída de nós e conexões e solicitar rotas. O controlador pode inferir a topologia da rede e, consequentemente, calcular as rotas entre dois pares quaisquer de roteadores. Contudo, na rede orientada a conteúdo o que interessa é a localização do conteúdo e o controlador não conhece a localização dos conteúdos. Para isto, os produtores precisam registrar os conteúdos no controlador e, assim o controlador pode instalar rotas para conteúdo nos roteadores.

3.3.1. Procedimento de Registro de Conteúdo

No registro de um novo conteúdo, o produtor envia um pacote de interesse com semântica estendida, Interesse 1 na Figura 6, onde `"register"` indica a intenção de registrar o nome `"myprefix"`. O roteador diretamente conectado recebe este pacote, adiciona uma entrada na PIT e também adiciona uma entrada na FIB com o nome `"myprefix"` apontando para a interface de entrada do interesse. Na sequência, ao invés de encaminhar o pacote de interesse para uma de suas interfaces, o roteador gera e envia um novo interesse indicando a localização do conteúdo na direção do controlador, Interesse 2 na Figura 6. Ao receber este interesse, o controlador armazena o `routerID` como a localização do conteúdo `"/myprefix"` na tabela de localização de conteúdos. O controlador responde para o roteador com um pacote de dados de confirmação e o roteador responde em confirmação para o produtor.

3.3.2. Procedimento de Instalação de Rotas

Para obter um conteúdo, o consumidor de conteúdo envia um pacote de interesse para a rede, Interesse 1 na Figura 7. O primeiro roteador recebe este pacote e adiciona

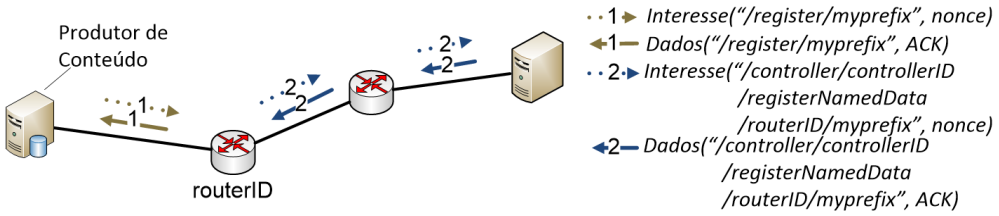


Figura 6. O Registro de Conteúdo realizado pelo roteador diretamente conectado ao Produtor de Conteúdo.

uma entrada na PIT. Caso não tenha entrada na FIB, o roteador solicita nova rota com um interesse de semântica estendida indicando este pedido, Interesse 2 na Figura 7. Neste pacote de interesse, `"routeFrom/sourceRouterID"` indica o roteador origem do pedido de rota. Visto que o prefixo `"wantedprefix"` já está registrado, o controlador sabe a rota para o destino. Então, o controlador calcula a melhor rota da origem até o destino e fornece a rota com um pacote de dados em resposta.

A resposta contém a informação a ser utilizada na instalação de rota, a qual abrange todo o caminho da origem ao destino. O controlador pode ainda incluir rotas alternativas no pacote de resposta ao pedido de rota, ver pacote de Dados 2 na Figura 7. Esta resposta inclui as palavras reservadas `"installRouteAndForwardInterest"` e `"installRoute"` para indicar o começo da rota, `"endroute"` para o fim da rota, `"routingPreference"` para a preferência de rota, e `"toprefix"` para indicar o prefixo desejado.

O roteador origem, ao receber o pacote de dados com a rota resposta, procura pelo seu ID e obtém o próximo salto. Então, o roteador adiciona uma entrada na sua FIB para o prefixo `"/wantedprefix"` através da interface de comunicação com o próximo salto.

Em seguida, o roteador origem usa a informação recebida do controlador para criar um pacote de interesse semanticamente estendido para instalar a rota nos roteadores do caminho. O próximo salto recebe este pacote de interesse e usa a informação contida no prefixo para criar uma entrada na FIB. Este procedimento se repete até o interesse atingir o roteador de destino, o qual já possui uma entrada na FIB para o `"/wantedprefix"`. Caso o prefixo especial use a palavra reservada `"installRouteAndForwardInterest"`, cada roteador no caminho adiciona uma entrada na PIT para o `"/wantedprefix"`, porém não adiciona entrada na PIT referente ao pacote de interesse de instalação de rota. A Figura 8 ilustra o fluxo de pacotes para a instalação de rota.

Caso o prefixo especial tenha a palavra reservada `"installRouteAndForwardInterest"`, o roteador destino também adiciona uma entrada para o `"/wantedprefix"` na sua PIT, cria e encaminha um pacote de interesse para o `"/wantedprefix"`, Item (b) na Figura 8 . Caso o prefixo especial contenha `"installRoute"`, o roteador destino apenas responde com um pacote de dados de confirmação, Item (a) na Figura 8.

4. Discussão

Esta seção discute algumas questões que afetam a operação do esquema.

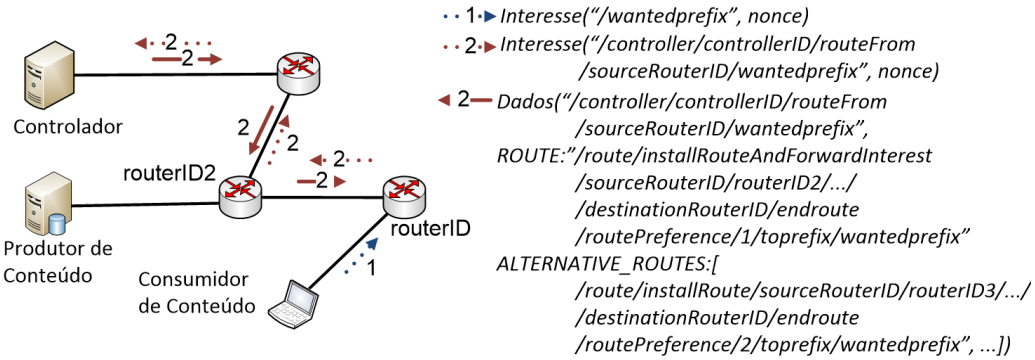


Figura 7. Solicitação de conteúdo do Consumidor (1) e Pedido de Rotas entre roteador e controlador (2).

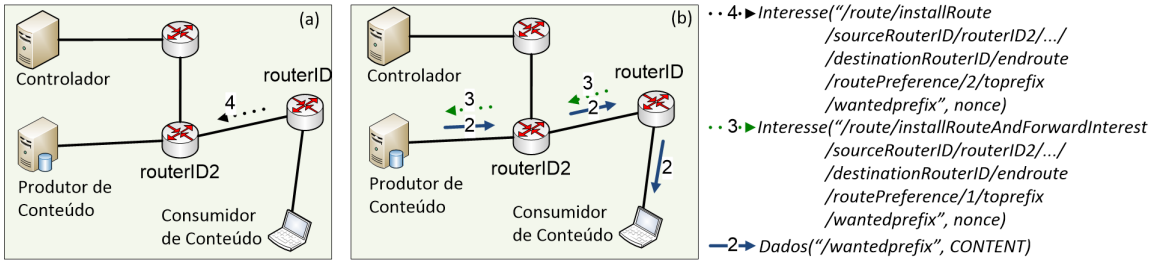


Figura 8. Instalação de Rotas. Item (a), instalação de rota sem busca de conteúdo. Item (b), instalação de rota e busca de conteúdo.

4.1. Número Estimado de Mensagens

O número de mensagens em cada procedimento é estimado a seguir. Na Descoberta de Vizinhos e na Descoberta de Controlador, o número é proporcional ao grau médio multiplicado pelo número de nós e se repete periodicamente. No Registro de Roteador, o número é proporcional ao número de nós multiplicado pela distância média entre roteador e controlador, ou diâmetro da rede. A taxa de alterações na topologia também afeta este último quantitativo.

No Registro de Conteúdo, o número de mensagens é proporcional à distância entre publicador e controlador e, também dependente do diâmetro da rede. A taxa de mobilidade de conteúdos também afeta este quantitativo. No Pedido de Conteúdo e Instalação de Rotas, o número é duplamente dependente do diâmetro da rede, primeiro na distância entre consumidor e controlador e segundo na distância entre consumidor e produtor. A correlação entre as solicitações de usuários afeta este quantitativo. É necessário ainda considerar as mensagens relativas à DHT.

Comparativamente, a divulgação de um prefixo não agregado via OSPF consome um número proporcional ao número de nós multiplicado pelo grau médio da rede. Outro ponto importante é a instalação de rotas com uma única consulta ao controlador que, comparada ao OpenFlow com consultas a cada salto, consome menos mensagens numa proporção relacionada ao diâmetro da rede.

4.2. Mobilidade

A mobilidade de consumidores e produtores afeta a entrega de conteúdos. No lado consumidor, basta o reenvio do interesse e novas rotas são instaladas. No lado produtor é mais complexo, pois é necessário informar a nova localização e apagar as rotas

já instaladas para a localização antiga. Para tanto, o produtor envia novo registro de seu conteúdo na nova localização e, o roteador da antiga localização responde aos interesses sem conteúdo com um pacote de dados `"NACK No Data"` que força a remoção das rotas até o consumidor.

Os controladores e a DHT atuam como um sistema de nome dinâmico fornecendo a localização. Isto permite o registro na DHT de cópias de conteúdo permitindo a busca de conteúdo fora do caminho até a fonte original. Para isto, DHT deve suportar múltiplas entradas de localização por conteúdo. O controlador ao receber múltiplas localizações tem opções para escolha da cópia mais adequada.

4.3. Segurança

Na rede CCN, todo produtor tem seu próprio par de chaves que pode ser usado para autenticar as mensagens. Porém, o controle da rede usando pacotes de interesse com prefixos semanticamente estendidos abre portas de ataque contra o funcionamento da rede. Estes pacotes alteram o comportamento dos elementos e normalmente não possuem informação assinada. Neste caso, os pacotes de interesse gerados devem carregar assinaturas conforme em VoCCN [Jacobson et al. 2009a]. Desta forma, os roteadores podem verificar que os pacotes de interesse foram assinados pelo controlador.

Na proposta CCN, ataques de negação de serviço (DOS) para conteúdo específico são naturalmente absorvidos com o controle de fluxo. Ataques distribuídos do mesmo tipo (DDOS) são absorvidos pela agregação de interesses e pelo controle de fluxo [Jacobson et al. 2009b]. Porém, ataques distribuídos para conteúdos distintos em um mesmo prefixo não são tratados originalmente. Nesta situação, a tabela PIT dos roteadores atendendo este prefixo cresce com o número de conteúdos solicitados, sendo que não há sequer verificação a priori da existência do conteúdo. Na nossa proposta, a existência do conteúdo é verificada no pedido de instalação de rotas, evitando na origem ataques usando nomes de conteúdo aleatórios.

4.4. Falhas e Partição da Rede

Falhas de circuitos ou nós podem ocasionar a partição da rede, isolando um ou mais controladores. A falha é detectada através da falta de resposta do controlador acima de um intervalo máximo de tempo. Caso ainda exista um controlador na partição, este pode ser encontrado através da Descoberta de Controlador. O controlador recebe os respectivos registros e obtêm a nova topologia. Os roteadores ligados ao produtor sempre possuem uma entrada FIB para conteúdos já registrados e renovam o registro no novo controlador. Caso a partição não possua controlador, os roteadores elegem um para assumir esta função. O processo de recuperação ocorre conforme a Inicialização e Monitoração de Topologia. No antigo controlador, os antigos registros de roteador e conteúdo expiram por tempo ou por sinalização de roteadores que detectem falha na busca de conteúdo não mais alcançável.

4.5. Distribuição de Carga

Em cenários de recuperação de falha ou inserção de novo controlador surge a necessidade de redistribuir os registros entre os controladores existentes e novos. Pela Descoberta de Controladores, os roteadores imediatamente conectados descobrem o novo

controlador. Esta descoberta é informada ao controlador existente no procedimento de Registro de Roteador.

Então o controlador existente pode solicitar diretamente ao novo controlador informações sobre capacidade total e livre de processamento, memória e disco, além do número de roteadores controlados e número de prefixos registrados. Baseado nestas informações, o controlador existente pode sinalizar a parte dos roteadores que alterne o registro para o novo controlador. Ao receber estes registros o novo controlador é informado sobre o controlador existente. A partir deste momento a comunicação entre os controladores é bidirecional estendendo o procedimento de distribuição de carga. Omitimos aqui o detalhamento de mensagens de sinalização deste procedimento.

Outro fator é a divisão dos registros de conteúdo entre os nós da DHT. A DHT é formada por elementos específicos e usa procedimentos próprios para distribuição de carga.

5. Conclusão

Este artigo propôs um esquema de roteamento para redes orientadas a conteúdo CCN superior a proposta OSPFN em termos de consumo de memória FIB e número de mensagens de controle. O esquema reusa a memória da tabela FIB dos roteadores instalando novas rotas em substituição das antigas a medida da necessidade. A substituição de rotas na FIB permite que a sua capacidade seja inferior ao exigido pelo OSPFN, o qual exige capacidade na FIB para o número total de conteúdos da rede com prefixos de nomes distintos. No esquema deste artigo, a demanda por capacidade de memória da FIB cresce de forma proporcional ao número de conteúdos com prefixos distintos em transito pelo roteador em simultâneo, isto é, de forma escalável com a demanda de tráfego.

O esquema proposto neste artigo separa as funções de roteamento em planos de controle e de dados. O plano de controle é composto por controladores estruturados em hierarquia. O total de mensagens da proposta é proporcional ao número de roteadores, níveis na hierarquia e prefixos distintos de conteúdo. Em redes com número de conteúdos de prefixos distintos muito maior do que de número de roteadores, o número de mensagens relativos ao registro e localização de conteúdo é dominante e proporcional ao diâmetro da rede. Este número é muito menor do que o exigido pelo OSPFN, proporcional ao número de roteadores da rede.

Como trabalhos futuros, pretende-se avaliar o desempenho da proposta através do simulador ndnSIM [Afanasyev et al. 2012], mensurando os efeitos de parâmetros como: tempo de armazenamento de rotas, padrões de mobilidade, número de nós por zona, tolerância a falhas, correlação de conteúdos solicitados, banda média por prefixo e os efeitos do número de níveis de controladores. Pretende-se também testar a proposta usando a distribuição CCNx [CCNx 2011] no Future Internet Testbed with Security (FITS) [FITS 2012].

Referências

Afanasyev, A., Moiseenko, I. e Zhang, L. (2012). ndnSIM: NDN simulator for NS-3. Technical report, University of California, Los Angeles.

Baid, A., Vu, T. e Raychaudhuri, D. (2012). Comparing Alternative Approaches for Networking of Named Objects in the Future Internet. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conferenceon*, pages 298 –303.

Carzaniga, A., Papalini, M. e Wolf, A. L. (2011). Content-based Publish/Subscribe Networking and Information-centric Networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ICN '11, pages 56–61. ACM.

CCNx (2011). CCNx Project. Disponível: http://www.ccnx.org/.

D'Ambrosio, M., Dannewitz, C., Karl, H. e Vercellone, V. (2011). MDHT: a Hierarchical Name Resolution Service for Information-centric Networks. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ICN '11, pages 7–12. ACM.

de Brito, G., Velloso, P. e Moraes, I. (2012). Redes Orientadas a Conteúdo: Um Novo Paradigma para a Internet. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, pages 211–264.

Fernandes, N., Moreira, M., Moraes, I., Ferraz, L., Couto, R., Carvalho, H., Campista, M., Costa, L. e Duarte, O. (2011). Virtual Networks: Isolation, Performance, and Trends. *Annals of Telecommunications*, 66:339–355.

FITS (2012). Future Internet Testbed with Security. Disponível: http://www.gta.ufrj.br/fits/.

Jacobson, V., Smetters, D. K., Briggs, N. H., Plass, M. F., Stewart, P., Thornton, J. D. e Braynard, R. L. (2009a). VoCCN: Voice-over Content-Centric Networks. ReArch '09, pages 1–6. ACM.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H. e Braynard, R. L. (2009b). Networking Named Content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12. ACM.

Mattos, D., Fernandes, N., da Costa, V., Cardoso, L., Campista, M., Costa, L. e Duarte, O. (2011). OMNI: OpenFlow MaNagement Infrastructure. In *Network of the Future (NOF), 2011 International Conference on the*, pages 52 –56.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., S. e Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Perino, D. e 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.

Wang, L., Hoque, A., Yi, C., Alyyan, A. e Zhang, B. (2012). OSPFN: An OSPF Based Routing Protocol for Named Data Networking. Technical report, University of Memphis and University of Arizona.

Yia, C., Afanasyevb, A., Moiseenkob, I., Wangc, L., Zhanga, B. e Zhangb, L. (2012). A Case for Stateful Forwarding Plane. Technical report, University of Arizona, University of California, Los Angeles and University of Memphis.

Zhang, L. et al. (2010). Named Data Networking (NDN) Project.

Redes Orientadas a Conteúdo: Abordagem no Nível de Enlace

Lisiane Maria Bannwart Ambiel[1], **Christian Esteve Rothenberg**[2], **Maurício Ferreira Magalhães**[1]

[1] Faculdade de Engenharia Elétrica e Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP)
Campinas, São Paulo, Brasil

[2]Fundação CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações
Campinas, São Paulo, Brasil

{lisiane,mauricio}@dca.fee.unicamp.br, esteve@cpqd.com.br

Abstract. *This paper proposes an architecture for content oriented networking at the link layer (Ethernet) without the use of network addressing schemes. Content Routers (CR) are the basis for this architecture and are in charge of packet caching and routing directly on content names. Different from IP environments where the destination address of the content source is known, the proposed link-level architecture requests content by controlled message flooding. Questions arise concerning the introduced overhead and the overall scalability. The paper proposes design options to contain the impacts of the content-oriented flooding approach and validates the prototype implementation in some scenarios compared to an IP approach. Results suggest that content-oriented IP-less architectures may be interesting for small networks such as as home networks that would largely benefit from a plug-and-play architecture that avoids network configuration and management as required with IP.*

Resumo. *O artigo propõe uma arquitetura de rede orientada a conteúdo no nível de enlace (Ethernet) sem uso de qualquer esquema de endereçamento. Os Content Routers (CR) são a base desta arquitetura, responsáveis pelo armazenamento de dados e roteamento de pacotes diretamente no conteúdo. Diferente do ambiente IP onde existe o conhecimento do endereço do provedor de conteúdo, a arquitetura proposta no nível de enlace requisita conteúdos através da inundação de mensagens de forma controlada. Esta condição traz consigo questões relacionadas à sobrecarga da rede e escalabilidade. O artigo propõe opções de projeto para minimizar o impacto da inundação de mensagens e valida a implementação do protótipo em alguns cenários comparando com uma abordagem IP. Os resultados sugerem que arquiteturas orientadas a conteúdo sem IP podem ser interessantes para redes de menor escala como home networks que se beneficiariam de uma arquitetura plug-and-play sem necessidade de configuração e gerenciamento como é o caso do IP.*

1. Introdução

As Redes Orientadas a Conteúdo (ROC) se apresentam como uma nova forma de pensar a Internet: mudam o paradigma de comunicação apresentando uma nova abordagem

com base no conteúdo independente de sua localização [de Brito et al. 2012]. A Internet foi projetada para interconectar computadores para aplicações como troca de correspondências e transferência de arquivos. Hoje é usada principalmente para disseminação e recuperação de conteúdos ou informação. Os usuários, em geral, não estão interessados em saber de onde vem a informação e somente precisam de uma garantia que a informação é autêntica. Esta abordagem torna a arquitetura da rede mais adequada para a distribuição de conteúdo e se utiliza de novos conceitos como conteúdo nomeado, roteamento baseado em nomes, segurança aplicada diretamente a conteúdos e armazenamento de dados nos nós intermediários da rede (*caching*).

Entre as arquiteturas propostas estão: DONA [Koponen et al. 2007], CCN [Jacobson et al. 2009], PSIRP [Tarkoma et al. 2009] e NetInf [Ahlgren et al. 2008]. Em comum estas arquiteturas utilizam nomes de conteúdo únicos e persistentes, modelo de segurança que permite verificação da integridade e origem do dado independente da fonte, simplicidade para suportar situações de mobilidade e múltiplos provedores (*multihoming*) e tolerância a falhas. Porém, ainda existem desafios para que estas redes sejam utilizadas em grande escala devido ao fato do número de conteúdos ser muito maior do que o número de endereços na rede atual, o que causa impacto no sistema de roteamento e resolução de nomes [Ahlgren et al. 2012]. Estudos indicam uma mudança no espaço de endereçamento de um bilhão de endereços IP para um trilhão de nomes de conteúdos [Perino and Varvello 2011]. Algumas questões são colocadas para pesquisa com relação à manutenção de estado por pacote, custos de processamento e de armazenamento no roteador, redução do tamanho de tabelas armazenadas de forma eficiente nos roteadores e como fazer buscas e operações em alta velocidade. Com relação às estratégias de encaminhamento é preciso investigar como descobrir um caminho para envio de uma requisição de um novo conteúdo e como escolher o melhor caminho quando se tem múltiplas opções [Yi et al. 2012].

Este trabalho propõe uma rede orientada a conteúdos no nível da camada de enlace (Ethernet) utilizando uma arquitetura de Content Routers (CRs) sem uso de endereçamento de rede com o objetivo de tornar o conteúdo mais importante. Desta forma não há necessidade de configuração de parâmetros de rede como no caso do IP. São utilizados mecanismos como: uso de identificadores planos para nomeação; armazenamento através de *caching* nos CRs; encaminhamento (*forwarding*) com base em tabela de roteamento ao invés de inundação (*flooding*) e entrega de conteúdo sem uso do endereço do cliente (*delivery*). A arquitetura dos CRs foi inicialmente definida por Wong et al. [Wong et al. 2011] como uma rede *overlay* sobre IP com suporte ao *caching* e utilização de nomes planos entre outras características. Diferente do ambiente IP onde existe o conhecimento do endereço do provedor, nesta proposta de arquitetura no nível de enlace (Ethernet) parte-se do princípio de que o conteúdo não é conhecido e a requisição de conteúdos é feita através de mecanismo de inundação de mensagens pelas interfaces nos CRs. Ao propor tal condição surgem questões relacionadas à quantidade de mensagens geradas na rede. Assim, para reduzir o número de mensagens, além do roteamento oportunístico baseado nas mensagens de RESPONSE, introduz-se: (a) primitivas para divulgação de conteúdos como forma de minimizar o impacto causado pelo *flooding*, (b) estrutura de dados compactadas (Bloom Filters) para manter a lista de requisições pendentes por interface; (c) conceito de Super CR como elemento especializado em *caching* para evitar inundação de mensagens, criando plano de controle e hierarquia.

Neste trabalho é estudado o comportamento de uma rede orientada a conteúdos através de experimentos com versões do protótipo para as alternativas IP e Ethernet. Entre outras dimensões de desempenho, são avaliadas a relação de *caching* disponível nos CRs sobre a quantidade de mensagens atendidas por um servidor de conteúdo e a distância percorrida em *hops* para atender a solicitação. Também é possível avaliar a relação do mecanismo de busca de conteúdo sobre a quantidade de mensagens geradas na rede. O tempo de transferência do conteúdo requisitado também pode ser observado nos diversos cenários porém deve ser considerado de forma relativa pois os CRs (protótipo) executam em ambiente virtual num mesmo servidor. A proposta no nível de enlace introduzida neste artigo apresenta-se como uma alternativa interessante para redes locais tais como *home networks* onde os custos relativos às dependências do IP (ex: gerência de rede, configuração de protocolos de roteamento e interfaces, segurança) seriam eliminados com uma orientação nativa da rede para um serviço de busca e entrega de conteúdo.

O artigo está organizado de acordo com a seguinte estrutura: a Seção 2 apresenta algumas definições relacionadas; na Seção 3 apresenta-se a arquitetura seguida da Seção 4 que descreve detalhes da implementação do protótipo; a Seção 5 apresenta a metodologia utilizada e o ambiente de teste; a Seção 6 apresenta uma discussão dos resultados obtidos. Ao final, na Seção 7 são apresentadas as conclusões e trabalhos futuros.

2. Background

Os seguintes conceitos chaves são propostos no contexto de Redes Orientadas a Conteúdo (ROC) [Ahlgren et al. 2012, de Brito et al. 2012]:

Objetos de dados nomeados: representam os conteúdos como, por exemplo, páginas WEB, documentos, filmes, fotos, músicas, enfim, todos os tipos de objetos que podem ser armazenados e acessados via computador. Um conteúdo mantém o seu nome independente da sua localização ou método de armazenamento.

Nomeação de conteúdos (*naming*): esquemas de nomeação que permitem identificar o conteúdo e requisitar sua distribuição à infraestrutura de rede. Esquemas básicos: nomeação plana, nomeação hierárquica e nomeação por atributo.

Roteamento baseado em nomes (*name-based routing*): a rede entrega os conteúdos requisitados por nome sem qualquer informação referente à localização, tanto de usuário quanto de armazenamento de conteúdos.

Armazenamento de conteúdos (*caching*): qualquer nó da rede pode atuar como um *cache* independente da aplicação ou protocolo. É aplicável a qualquer conteúdo, inclusive aos gerados pelo usuário cliente. Uma requisição por determinado conteúdo pode ser respondida por qualquer nó que tenha a cópia em seu *cache*.

Os mecanismos para nomeação podem ser classificados em três classes básicas: nomes planos, hierárquicos e baseados em atributos [Choi et al. 2011]. Cada uma das classes de nomes atende parcialmente os requisitos exigidos dos mecanismos de nomeação: persistência, escalabilidade e inteligibilidade ao usuário final. A nomeação hierárquica é legível e escalável, mas possui restrições ao uso persistente dos nomes. Esta restrição não existe no caso de nomes planos que apresentam vantagem com relação à segurança, mas seu uso depende de um processo de resolução entre um nome inteligível e o nome plano. O uso de atributos relacionados ao conteúdo pode facilitar a busca mas depende

de definições adequadas para evitar ambiguidades. Os mecanismos de roteamento de conteúdos podem ser classificados em dois grupos: roteamento não-estruturado, normalmente baseado em *flooding*, e roteamento hierárquico (ou estruturado) baseado em árvores hierárquicas ou tabelas hash distribuídas (*Distributed Hash Tables* - DHT).

Além destes, devem ser considerados como aspectos comuns aos projetos de ROC as primitivas e o modelo de segurança [Ghodsi et al. 2011]. As primitivas básicas utilizam o nome do conteúdo e o paradigma *publish/subscribe* onde o provedor do conteúdo não conhece a localização do solicitante e vice-versa. Da mesma forma provedor e solicitante de conteúdo não precisam estar *online* simultaneamente. No modelo de segurança o conteúdo é assegurado pelo provedor original de forma que os elementos da rede e os consumidores possam verificar a sua validade.

Fazendo uma avaliação sistemática dos componentes de hardware e software existentes nos roteadores para suporte às redes de conteúdo, Perino e Varvello [Perino and Varvello 2011] afirmam que o novo paradigma vai provocar mudanças nos roteadores: maior velocidade para suportar encaminhamento com base no nome do conteúdo e armazenamento de pacotes de dados; mudança no espaço de endereçamento (de um bilhão de endereços IP para um trilhão de nomes de conteúdos). Apesar do aumento da informação a ser armazenada, o *caching* local pode aliviar potencialmente a freqüência das operações de encaminhamento. Entre as soluções propostas, estratégias probabilísticas baseadas em estruturas como os Bloom Filters aparecem como abordagens promissoras. O trabalho [Perino and Varvello 2011] conclui que a tecnologia atual não está pronta para suportar as redes de conteúdo na escala da Internet, mas é viável no nível de uma rede de distribuição de conteúdos (CDN – *Content Delivery Network*) ou no escopo de um provedor de serviços (ISP – *Internet Service Provider*).

3. Arquitetura de CRs no Nível de Enlace

A arquitetura é composta por roteadores, os *Content Routers* (CR), que são os elementos de rede que realizam o encaminhamento de mensagens e o armazenamento (*caching*) oportunístico de conteúdos. A proposta original da arquitetura [Wong et al. 2011] é baseada numa rede *overlay* sobre IP com suporte ao *caching* e utilizando nomes planos entre outras características. Também são considerados elementos da rede: Cliente, que faz as requisições e Servidor, que tem o conteúdo original, faz o anúncio do mesmo e responde às requisições. A arquitetura se aplica aos ambientes IP e Ethernet com algumas diferenças que são explicitadas a seguir. Primeiramente, são introduzidas as seguintes definições:

***Data Chunk*:** Unidade básica de comunicação; parte de um conteúdo (ex. arquivo) que será identificado e requisitado. Cada *Data Chunk*, ou simplesmente *chunk*, é identificado pelo seu *chunkId*.

***Cryptographic Identificer* (CryptoId):** Identificação do conteúdo que resulta de um hash criptográfico aplicado a um bloco de dados (*Data Chunk*).

***Metadata*:** Estrutura de dados que agrega a lista de *chunkIds* relacionados a um conteúdo e outras informações adicionais como versão e validade. Informação gerada pelo provedor do conteúdo (Servidor) e que deve ser recuperado pelo Cliente para proceder às requisições dos *Data Chunks* por um mecanismo de resolução de nomes.

***Caching Threshold*:** Determina a probabilidade de se armazenar um conteúdo. Valor possível de ser configurado.

***Neighbor Zones* (NZones):** Total de CRs a visitar antes de encaminhar uma requisição para o servidor (ambiente IP) ou via *flooding* (ambiente Ethernet). Significa que a requisição será desviada na tentativa de buscar um conteúdo em um CR vizinho até o limite de NZones. Como esse número pode indicar uma área em volta do CR, usa-se o termo *zones*. Valor possível de ser configurado.

3.1. Características principais da rede

Na sequencia são apresentadas as principais características adotadas para a arquitetura.

Nomeação: utiliza identificador plano para nomear os conteúdos (chunkIds). Este tipo de identificação atende ao requisito de persistência pois está desacoplado da informação de localização e dá suporte ao modelo de segurança para rede de conteúdos.

Registro dos conteúdos: os servidores anunciam opcionalmente os conteúdos através da primitiva `ANNOUNCE_CONTENT` que contém o chunkId. Os roteadores armazenam a informação na tabela de rotas. Esta informação evita o *flooding* pois a requisição por um chunkId conhecido será encaminhada somente na direção do vizinho que repassou a mensagem de anúncio do conteúdo anteriormente.

Roteamento: somente as requisições são roteadas. O roteamento é feito exclusivamente com base nos *chunkId*. A tabela é preenchida com a informação anunciada pelos servidores ou de forma oportunística com a informação da mensagem `RESPONSE` que trafega na rede. Desta forma o CR cria o conhecimento das rotas para um conteúdo, mesmo que este não esteja disponível no cache local. A tabela mantém somente o melhor caminho para um conteúdo, ou seja, a menor distância em *hops*.

- No ambiente IP: se o conteúdo não é encontrado na tabela de rotas a requisição é encaminhada ao servidor (endereço de destino) com base no roteamento IP ativo na rede (ex: OSPF), ou seja, neste ambiente temos dois níveis de roteamento (ou roteamento híbrido): inicialmente no conteúdo e, caso este alcance o limite definido pela variável NZones, pelo roteamento IP na direção do servidor do conteúdo.
- No ambiente Ethernet: se o conteúdo não é encontrado na tabela de rotas é feita a inundação da requisição pelas interfaces disponíveis pois não existe o conhecimento de endereços como é o caso do ambiente IP. O roteador mantém informação sobre quais conteúdos (*chunkIds*) estão sendo aguardados por interface – equivalente a tabela PIT (Pending Interest Table) da arquitetura CCN [Jacobson et al. 2009] e não encaminha requisições por conteúdos que já estão na lista de pendências. Esta tabela é implementada com estruturas de Bloom filter [Tarkoma et al. 2012] para reduzir os requisitos de memória.

Mecanismo de entrega de dados:

- No ambiente IP o dado é roteado para o elemento que fez a requisição.
- No ambiente Ethernet o dado faz o caminho inverso ao da requisição. Ao receber uma mensagem `RESPONSE` o CR verifica se o chunkId é esperado e repassa a mensagem às interfaces que tem o chunkId na sua lista de pendências. Em seguida o chunkId é removido das listas.

***Caching*:** O objetivo desse armazenamento é que solicitações futuras pelo mesmo conteúdo sejam atendidas a partir dos CRs ao invés de percorrerem toda a rede até o servidor original, atendendo ao requisito de disponibilidade.

Segurança: Os chunkIds são nomes auto certificáveis. O sistema utiliza Merkle Tree e tem o objetivo de permitir a recuperação segura dos conteúdos de múltiplas fontes ou de fonte não conhecida. Cada conteúdo tem a informação de segurança associada permitindo que o receptor verifique a validade e a integridade do mesmo independente do local de recuperação. Atende ao requisito de autenticação [Wong et al. 2011].

Primitivas (API - *Application Programming Interfaces*): `REQUEST`, originada pelo Cliente que requisita um *chunkId*; `RESPONSE`, originada por um Servidor ou por um CR que tem o dado correspondente ao *chunkId* no seu cache em resposta a um `REQUEST`; `ANNOUNCE_CONTENT`, originada pelo Servidor para anunciar os *chunkIds* disponíveis, reconhecida pelos CRs; e `ANNOUNCE_SCR`, originada pelo CR que está configurado como Super CR, reconhecida pelos CRs.

3.2. Content Router (CR)

O CR faz o tratamento das mensagens que trafegam na rede conforme descrito a seguir.

As mensagens `REQUEST` são respondidas diretamente pelo CR quando o *chunkId* e o dado correspondente são encontrados na sua tabela de *caching* (*cache-hit*). Se o CR não tem o conteúdo (*cache-miss*), verifica a tabela de rotas e faz o roteamento da mensagem conforme explicado anteriormente. A base para esse processo de decisão está na presença ou não de informação na tabela de roteamento e na variável *NZones*.

As mensagens `RESPONSE` são encaminhadas aos clientes que originaram a requisição. No ambiente IP este processo é feito pelo protocolo de roteamento ativo. No ambiente Ethernet é utilizada a informação de requisições pendentes. No tratamento da mensagem `RESPONSE` pode ocorrer o *caching* do conteúdo nos CRs, em função do parâmetro *Caching Threshold*. Esse parâmetro é aplicado na função de probabilidade a fim de não sobrecarregar o cache pois o mesmo tem capacidade limitada.

Não existe nenhum tipo de sinalização entre os CRs, ou seja, eles não funcionam de forma cooperada com relação ao *caching*. Um mesmo conteúdo pode estar armazenado probabilisticamente em todos os CRs no caminho feito por uma mensagem `RESPONSE`.

Os CRs podem ser especializados (Super CR) na função de *caching* como forma de atender a requisitos de escalabilidade evitando inundação de mensagens. Este tipo de CR cria hierarquia e conceito de plano de controle pela troca de mensagem específica (`ANNOUNCE_SCR`). Assim, um CR que não tem o conteúdo no cache local e não conhece a rota, tenta o Super CR antes de recorrer ao *flooding*.

4. Implementação

O CR é composto por um módulo de controle que trata da comunicação e um módulo que trata da manipulação das mensagens. O módulo de controle é especializado para atender aos ambiente IP e Ethernet; intercepta mensagens em trânsito, inspeciona o cabeçalho e faz recepção e envio de mensagens. O módulo de manipulação das mensagens verifica e armazena conteúdos na tabela de *caching*, verifica e armazena informação na tabela de rotas e responde às requisições relativas aos conteúdos por ele armazenado.

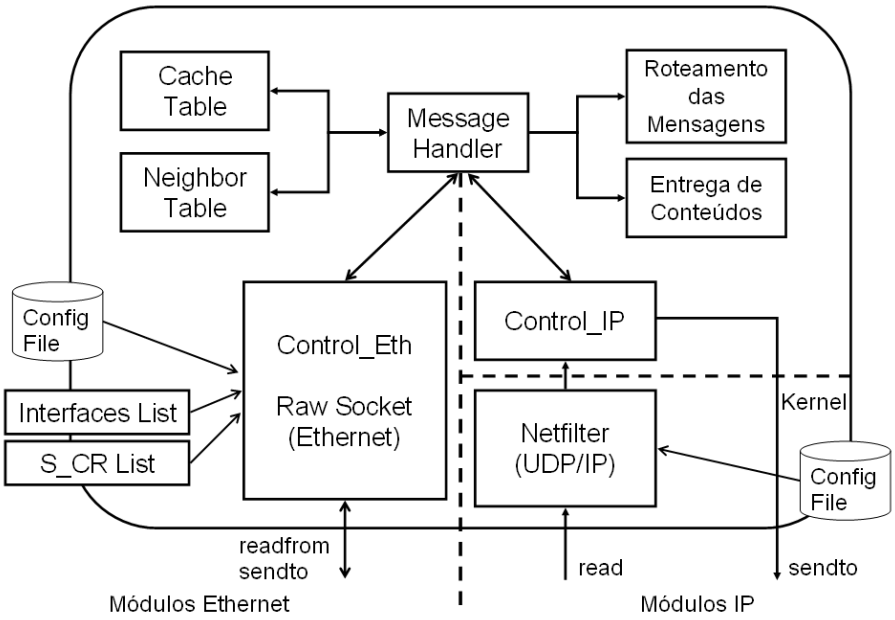


Figura 1. Módulos do CR

O CR mantém duas estruturas de dados importantes: *Cache Table* e *Neighbor Table* (ou tabela de rotas). As duas tabelas são indexadas pelo identificador do conteúdo, *chunkId*, têm capacidade limitada e estratégia de substituição simples (FIFO). A Figura 1 dá uma visão geral dos módulos.

No ambiente Ethernet o CR mantém a lista das interfaces disponíveis (Interfaces List) e a lista de Super CRs (`S_CR List`). Cada interface mantém as requisições pendentes (como a PIT no CCN [Jacobson et al. 2009]) utilizando estrutura de Bloom Filter tipo contador, assim o CR evita que a requisição por um mesmo *chunkId* não seja re-encaminhada e minimiza a quantidade de mensagens gerada na rede. Ao receber uma resposta o CR encaminha o dado a todas as interfaces que tem o *chunkId* pendente e remove a requisição pendente.

O protótipo foi desenvolvido em linguagem C e implementa as funções dos elementos CR, Cliente e Servidor. A implementação atual não inclui: mecanismo de resolução de nomes e modelo de segurança. O Cliente tem disponível o metadata dos conteúdos para iniciar as requisições. No ambiente IP não se tem o anúncio de conteúdos.

Basicamente, o funcionamento dos elementos do protótipo é conforme o seguinte processo: o *cliente* requisita conteúdos (*chunkId*) através das mensagens `REQUEST` e aguarda os dados pela mensagem `RESPONSE`; re-envia a requisição no caso de *timeout*. O *servidor* espera por solicitação de conteúdos e responde com dados através da mensagem `RESPONSE` quando tem o conteúdo. Os CRs interceptam as mensagens na rede e processam conforme as informações presentes nas tabelas de roteamento e *caching* e configuração do `NZones`.

5. Metodologia Experimental

Este trabalho utiliza método de pesquisa quantitativa e procedimento do tipo experimento [Creswell 2009]. Participam do experimento todos os elementos da rede: clientes, servi-

dores e roteadores de conteúdo (CRs). Todos os participantes são designados por conveniência, ou seja, não existe nenhum mecanismo aleatório na seleção dos participantes.

Os elementos da rede são um conjunto de máquinas virtuais criadas utilizando a ferramenta XEN [1] que permite que elas atuem como roteadores. Todos os elementos são executados em ambiente Debian GNU/Linux. Para os testes da rede em modo *overlay*/IP, os roteadores utilizam a pilha de protocolos *open-source* Quagga [2] mediante o protocolo OSPF (*Open Shortest Path First*).

Como variáveis independentes (as que causam ou influenciam nos resultados) temos as seguintes:

- *caching*: probabilidade de armazenamento dos roteadores da rede.
- *neighbor zones*: busca de conteúdo em caminho conhecido/aprendido até determinado limite de *hops* (profundidade).
- *flooding*: possibilidade de uso de mecanismo de inundação da requisição até encontrar um conteúdo não conhecido (somente CR no ambiente Ethernet).

Além destas variáveis, colaboram na consequência dos resultados a dimensão das tabelas de *caching* e de roteamento.

Variáveis dependentes ou de resultado (as consequencias):

- utilização dos servidores e roteadores (quantidade de mensagens tratadas)
- distância da resolução em número de *hops*
- tempo de transferência do conteúdo

Uma topologia de 12 CRs, 4 servidores e 8 clientes é utilizada para os testes da rede de CRs nos ambientes IP ou Ethernet (Figura 2). Para fins deste artigo somente os resultados de testes a partir de um único cliente são apresentados para facilitar entendimento das variáveis envolvidas. Testes com variação da topologia foram executados e apresentaram comportamento comparáveis.

O protótipo é usado como instrumento no experimento. Todos os elementos geram arquivos com registro de eventos (arquivos de LOG) e arquivos com contadores no formato CSV (*comma separated value*). Esses arquivos são gerados quando o elemento de rede recebe mensagem de controle ao final de uma solicitação de um conteúdo e ao final do teste (seqüência de 30 solicitações). As mensagens de controle têm função operacional (ex: LOG, STOP) sem relação com o protocolo da rede. Como a topologia e as interfaces de cada CR são conhecidas, o envio destas mensagens é feito de forma direcionada para interfaces pré-definidas em cada CR garantindo que todos os elementos recebam a mensagem sem necessidade de recorrer ao *flooding*.

5.1. Procedimento

Um teste é composto de uma seqüência de 30 solicitações de conteúdos seguindo uma distribuição *power law*: 10 arquivos são solicitados sendo que cada um deles tem um grau de popularidade, ou seja, alguns vão ser solicitados com maior freqüência. A distribuição de popularidade por conteúdo é resultado de programa que gera número aleatório e o *script* de execução do teste. A Tabela 1 mostra a lista dos arquivos com respectiva quantidade de *chunks* e quantas vezes cada arquivo é solicitado no teste.

[1] http://www.xensource.com
[2] http://www.quagga.org

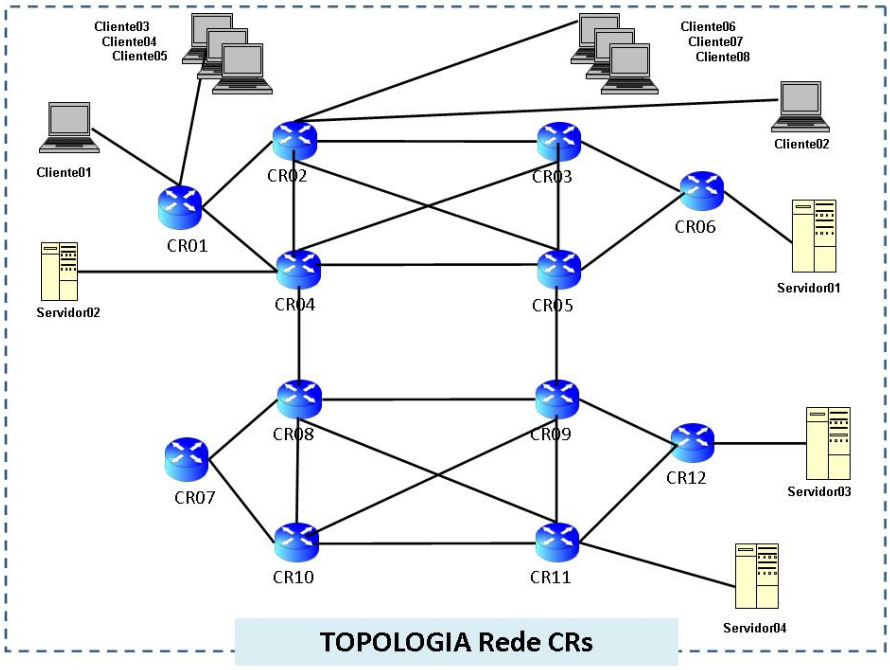


Figura 2. Participantes do experimento - Topologia

O teste é iniciado por um elemento cliente que faz o papel do solicitante. Ao final do teste os arquivos de LOG e CSV são coletados nos diversos elementos. Os testes são repetidos com variação de valores das variáveis independentes descritas anteriormente. Os valores são atribuídos através de arquivo de configuração.

5.2. Medidas

As medidas são coletadas através de mensagens de controle enviadas pelo elemento que faz a requisição. Os roteadores e servidores ao receberem uma mensagem de LOG registram os contadores (ex: total de mensagens recebidas, enviadas, cache hit etc). Tais medidas são armazenadas em arquivos CSV e então são traduzidas para tabelas no aplicativo Excel para cálculo de média entre testes do mesmo cenário, análise entre diferentes cenários, geração de gráficos e comparações.

Tabela 1. Arquivos de conteúdo do procedimento de testes.

#	Nome do arquivo	Qtde. chunks	# Requisições/teste
1	aliancas.jpg	1059	0
2	biblioteca.jpg	563	1
3	DistributedCachingAlgoritms.pdf	175	2
4	EnhanceContentBroadcast.pdf	118	1
5	GatewayControlledContentCaching.pdf	286	2
6	PacktpubMoodleSecurity.pdf	3406	16
7	SantaClausIsComingToTown.mp3	5099	2
8	TeachYourselfFreeBSD.pdf	7176	1
9	TikosGrooveFeatGosha.mp3	6590	1
10	towerEiffel.jpg	411	4
	TOTAL	**24888**	**30**

6. Resultados

Após a execução dos testes e consolidação dos resultados foram gerados gráficos para análise e avaliação. Nestes cenários, o requisitante é o $Cliente01$ conforme Figura 2.

6.1. Ambiente IP/Overlay

Os gráficos na Figura 3(a) e na Figura 3(b) mostram o comportamento da rede orientada a conteúdo para ambiente IP/overlay[3] nos diferentes cenários: probabilidade de *caching* (0%, 50%, 70%) e valor de *neighbor zones NZ* (0, 3, 4, 6). É possível observar uma complementariedade entre a resolução nos servidores e nos CRs.

Nos cenários (1) sem disponibilidade de cache nos CRs, a resolução das requisições acontece sempre nos servidores. O valor de NZ não colabora no processo pois a requisição é encaminhada sempre para o endereço do servidor.

Para cache habilitado observar que a resolução acontece em grande parte nos CRs, diminuindo a carga nos servidores, porém sem muitas vantagens no aumento da probabilidade de 50% para 70%. A busca na vizinhança (NZ) também colabora para aumentar a resolução na rede.

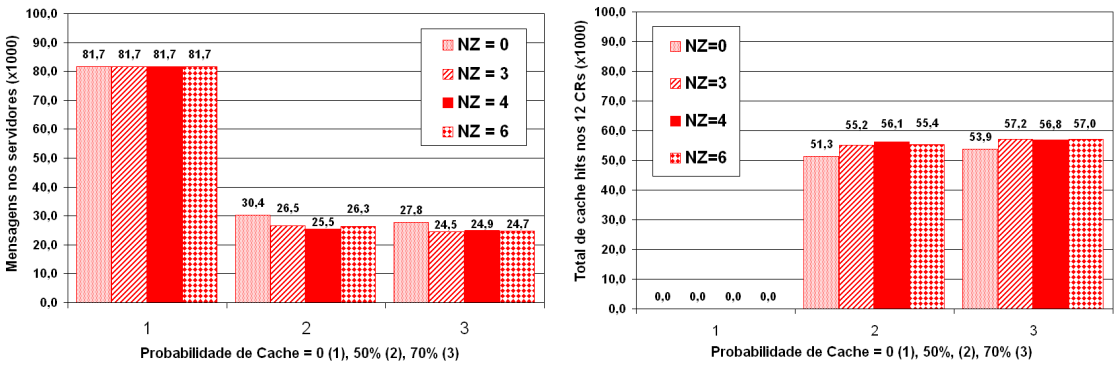
As Figuras 4(a) e 4(b) mostram, no caso do IP, a tendência da resolução das requisições no $CR1$, mais próximo do $Cliente01$ que fez as requisições, quando $NZ = 6$. Alguns CRs não participam do tratamento de mensagens (CRs 3,7 e 10) pois o OSPF não fez a opção pela rota que passa por eles.

6.2. Ambiente Ethernet

Os gráficos na Figura 5(a) e na Figura 5(b) mostram o comportamento da rede orientada a conteúdo para a proposta no nível de enlace (ambiente Ethernet) para os diferentes cenários sob avaliação (Probabilidade de *caching*, *# Neighbor Zones – NZ*).

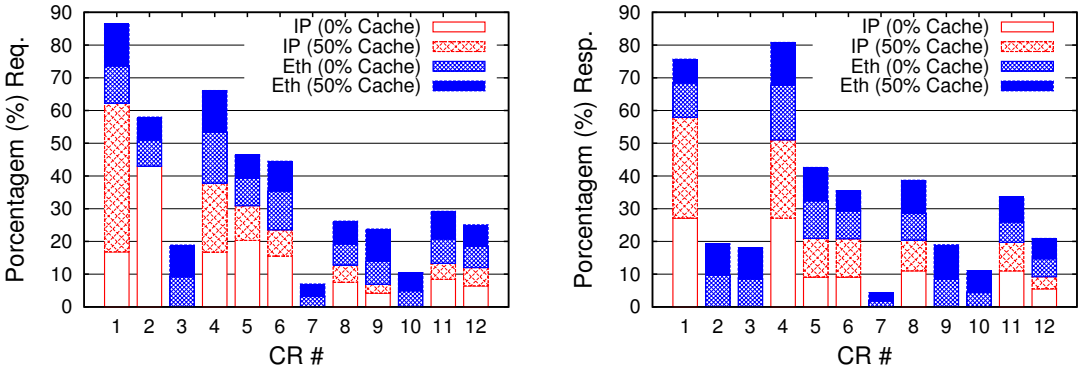
Nos cenários sem cache (1) onde a resolução das requisições acontece nos servidores o valor de NZ tem impactos diversos: $NZ = 0$: não faz busca, pára, conforme esperado. $NZ = 3$: gera menor número de mensagens; resolução no servidor mais

[3]Vale a pena notar que não são contabilizadas as mensagens do protocolo OSPF (ex: Hello, LSA, LSU).



(a) Requisições atendidas pelos servidores (b) Requisições atendidas pelos CRs (cache hit)

Figura 3. Tratamento de requisições (IP)



(a) Mensagens `REQUEST` processadas (b) Mensagens `RESPONSE` processadas

Figura 4. Distribuição de mensagens nos CRs (IP vs. Ethernet).

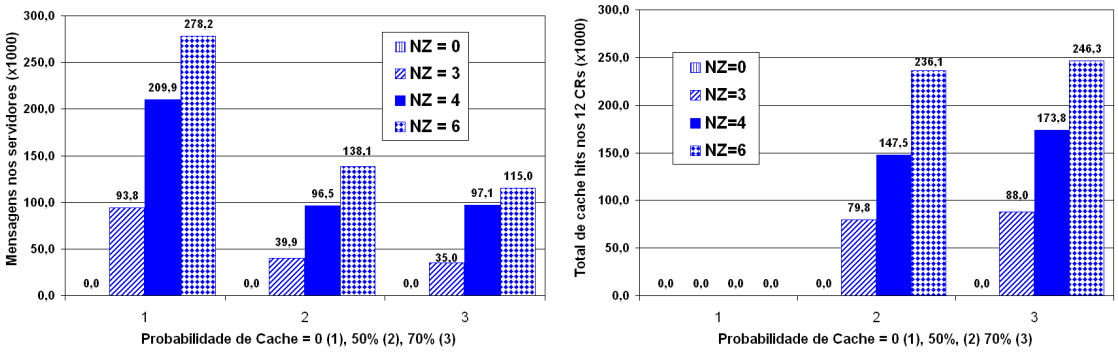
próximo ($Servidor02$). $NZ = 4$ ou 6 : aumenta o número de mensagens geradas na rede; resolução distribuída nos demais servidores.

Para cache habilitado verificar que a resolução acontece nos CRs, diminuindo a carga nos servidores, porém sem muitas vantagens no aumento da probabilidade de 50% para 70% como no caso do IP. A busca na vizinhança (NZ) colabora para aumentar o número de mensagens na rede.

As Figuras 4(a) e 4(b) mostram a distribuição das requisições nos CRs quando $NZ = 6$. Nesta condição todos os CRs respondem às requisições apesar de nem todas serem "utilizadas": o *flooding* causa várias respostas para uma mesma requisição; estas respostas serão ignoradas pelos CRs que não têm a requisição pendente.

6.3. Ambiente IP vs. Ethernet

A topologia do tipo Internet utilizada nos testes coloca a solução no nível de enlace em desvantagem comparada à solução IP/overlay pois aumenta o número de pontos de inundação e aumenta a quantidade de mensagens. Nas figuras 3 e 5 é possivel observar que apesar das quantidades de mensagens geradas na rede serem diferentes (Ethernet precisando de mais mensagens que o IP), o comportamento dos dois ambientes tende a ser o mesmo. As Figuras 4(a) e 4(b) mostram que a rede Ethernet distribui melhor o



(a) Requisições atendidas pelos servidores (b) Requisições atendidas pelos CRs (cache hit)

Figura 5. Tratamento de requisições (Ethernet)

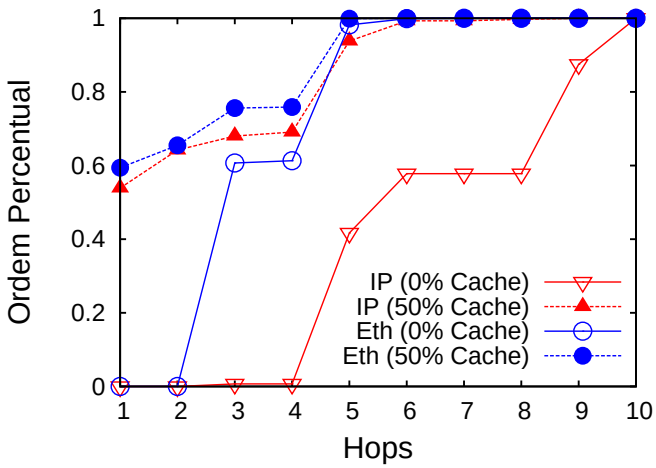


Figura 6. IP vs. Ethernet: Distância em saltos para resolução das requisições

processamento entre os diferentes CR sem sobrecarregar nenhum CR em particular.

Já o gráfico na Figura 6 faz uma comparação dos dois ambientes em termos de número de *hops* desde a fonte do conteúdo (cache ou servidor) até o cliente consumidor. Vale a pena destacar que o ambiente Ethernet consegue caminhos mais curtos (60% das vezes com 4 ou menos $hops$) que o IP (60% das vezes com 6 ou menos $hops$) e atinge resultados quase comparáveis aos da rede IP com cache ativado (concentração entre 3 e 4 *hops*). Observar que em alguns casos a distância percorrida é maior que NZones definido para o teste ($NZ = 6$). Isso ocorre porque houve tentativa de busca na vizinhança sem sucesso (roteamento no conteúdo) e ocorre um acréscimo de *hops* além da distância normal até o servidor (roteamento no IP).

Finalmente, a Tabela 6.3 compara a requisição de um mesmo arquivo. No *testbed* com os elementos virtualizados em uma máquina física o tempo não é um resultado que possa ser considerado de forma absoluta pelos efeitos da virtualização e o consumo concorrente de CPU, mas apresenta certa coerência. Os números mostram que a rede Ethernet tem um custo maior para a resolução da primeira requisição devido a busca por *flooding*, porém, após o conhecimento da rota, nas requisições subsequentes as duas redes se comportam de forma equivalente. O tempo se refere ao tempo de recepção do arquivo (3406 chunks) pelo $Cliente01$ medido desde o tempo de envio do primeiro `REQUEST`.

Tabela 2. IP vs. Ethernet: Tempo de transferência em segundos de um arquivo de 3406 chunks para $NZ = 4$ e $Cache = 50\%$.

Requisição #	1^a	2^a	3^a	4^a	5^a
IP	24,55	5,62	3,50	2,86	2,56
Ethernet	42,67	4,92	3,68	2,96	2,77

7. Conclusões e Trabalhos Futuros

O objetivo da arquitetura proposta consiste em uma abordagem de roteamento diferente das baseadas na propagação de mensagens de alcançabilidade como no caso do CCN. O artigo discute duas arquiteturas diferentes: a primeira é efetivamente uma arquitetura

híbrida com roteamento no conteúdo e no IP e a segunda é completamente limitada ao nível de enlace com roteamento no conteúdo. Com os resultados obtidos até o momento algumas conclusões já podem ser apresentadas. A principal vantagem desta proposta é dispensar o roteamento baseado em localização e operar diretamente na camada de enlace através do roteamento baseado no conteúdo. A contrapartida da abordagem é um maior número de mensagens trocadas na rede. Porém, considerando o desempenho apresentado pela proposta uma vez aprendidos os caminhos até o conteúdo, vale a pena considerar o *trade-off* de sobrecarga em tráfego de controle pela simplicidade de uma arquitetura *plug-and-play* no nível de enlace. Como na arquitetura CCN, a nossa proposta não precisa do IP e faz roteamento com base no conteúdo, porém o CCN não tem uma proposta específica para o nível de enlace.

Abordagens no espírito da proposta deste trabalho podem ser uma opção para redes domiciliares (HomeNets) [Velloso et al. 2004] ou outros tipos de redes menores onde não existe necessidade de endereçamento IP ou configurações mais complexas. Com o aumento de dispositivos do tipo *tablets*, *laptops*, *smartphones* e aparelhos de segurança para monitoramento é esperado adição de roteadores nestas redes. O uso de *gateways* não IP como o Super CR é uma opção para aumentar a complexidade da rede sem aumento da complexidade de operação da rede.

Entre trabalhos relacionados, destacamos a arquitetura CONET [Detti et al. 2011] por apresentar uma abordagem no nível de enlace além de IP/overlay e IP integrado. Similar à proposta dos CRs no aprendizado das rotas, a arquitetura CONET limita o tamanho da tabela mas introduz um elemento centralizado com informações de rotas que serve todos os demais elementos em um sub-sistema. A idéia de especialização dos nós é semelhante à nossa proposta de Super CR que pode atuar como *gateway* para integrar a rede Ethernet com a rede IP.

Como trabalhos futuros temos identificadas as seguintes áreas de atuação a serem implementadas e testadas com o protótipo:

Arquitetura: o Super CR pode ter suas funções ampliadas e funcionar como: (i) ponto de *rendezvous* para suporte ao modelo *publish/subscribe* como PSIRP [Tarkoma et al. 2009], (ii) ponto de acesso inter-domínio ou *gateway* para compor um ambiente misto IP/Ethernet. Outros testes e análises serão feitos para verificar comportamento com diferentes opções de *flooding* e o uso das mensagens de anúncio de conteúdo na rede Ethernet, avaliando o impacto da inserção de um elemento especializado como Super CR.

Otimização de mecanismos: as tabelas de roteamento e *caching* podem utilizar outras estratégias além da utilizada atualmente (FIFO); a tabela de roteamento mantém somente o melhor caminho e não uma lista de opções como na proposta original da arquitetura CCN [Jacobson et al. 2009] e o processamento não é instanciado por interface mas um único processo atende todas as interfaces do CR; o *flooding* por todas as interfaces pode ser evitado pelo aprendizado e identificação das melhores interfaces (cf. [Yi et al. 2012]), seja pelo tempo ou percentual de respostas; inserção do mecanismo de resolução de nomes e obtenção do *metadata* e uso de um *proxy* para aplicações legadas; inserção de políticas de *caching*; otimizações nas estruturas dos Bloom filters.

Aplicação em cenários de *HomeNets*: instanciar uma versão do protótipo numa rede domiciliar combinando ambiente sem fio e cabeado onde o Super CR atua como *gateway*

para o mundo IP e onde qualquer dispositivo pode atuar como CR provendo ampla capacidade de *caching* e colaborativamente oferecendo um sistema de arquivos distribuído para os usuários da HomeNet.

Referências

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. (2012). A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36.

Ahlgren et al. (2008). Design considerations for a network of information. In *ACM CoNEXT '08*, pages 1–6, Madrid, Spain.

Choi et al. (2011). A Survey on content-oriented networking for efficient content delivery. *IEEE Communications Magazine*, 49(3):121–127.

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage, Los Angeles, third edit edition.

de Brito, G. M., Velloso, P. B., and Moraes, I. M. (2012). Redes Orientadas a Conteúdo: Um Novo Paradigma para a Internet. In *SBRC 2012*.

Detti, A., Blefari Melazzi, N., Salsano, S., and Pomposini, M. (2011). CONET: a content centric inter-networking architecture. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking - ICN '11*, pages 50–55.

Ghodsi, A., Shenker, S., Koponen, T., Singla, A., Raghavan, B., and Wilcox, J. (2011). Information-centric networking: Seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, pages 1–6, Cambridge, USA.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *CoNEXT'09*, pages 1–12, Rome, Italy.

Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. (2007). A data-oriented (and beyond) network architecture. *ACM SIGCOMM CCR*, pages 181–192.

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, New York, USA. ACM Press.

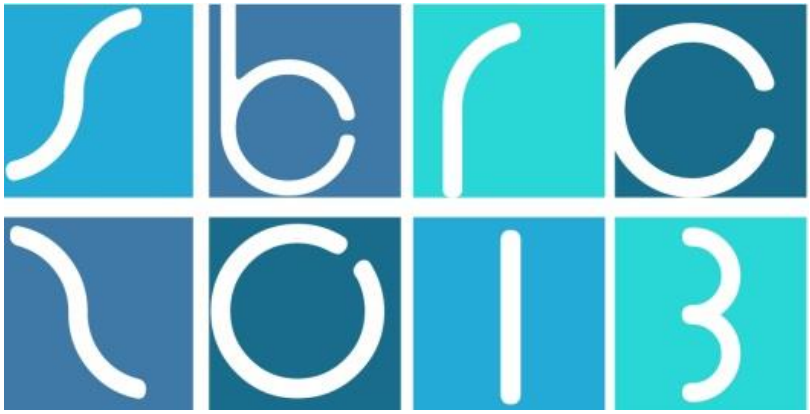
Tarkoma, S., Ain, M., and Visala, K. (2009). The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture. In *Towards the Future Internet - A European Research Perspective, 2009*, pages 102–111.

Tarkoma, S., Rothenberg, C. E., and Lagerspetz, E. (2012). Theory and Practice of Bloom Filters for Distributed Systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155.

Velloso, P., Cunha, D., Amodei Jr, A., Rubinstein, M., and Duarte, O. (2004). Redes domiciliares: Princípios e desafios das tecnologias sem novos fios. *Minicursos SBRC2004*, pages 221–268.

Wong, W., Giraldi, M., Magalhaes, M. F., and Kangasharju, J. (2011). Content Routers: Fetching Data on Network Path. In *IEEE ICC*, pages 1–6. IEEE.

Yi, C., Afanasyev, A., Wang, L., Zhang, B., and Zhang, L. (2012). Adaptive forwarding in named data networking. *ACM SIGCOMM CCR*, 42(3):62–67.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 18

Computação nas Nuvens: Avaliação de Serviços

Cloud Crawler: Um Ambiente Programável para Avaliar o Desempenho de Aplicações em Nuvens de Infraestrutura

Matheus Cunha, Nabor Mendonça, Américo Sampaio

[1]Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321, Edson Queiroz, CEP 60811-905 Fortaleza, CE

mathcunha@gmail.com, nabor@unifor.br, americo.sampaio@unifor.br

Resumo. *Um dos principais desafios enfrentados pelos usuários de nuvens que oferecem infraestrutura-como-serviço (IaaS) é a dificuldade para dimensionar adequadamente os recursos virtuais necessários às suas aplicações. Embora muitos provedores de nuvem permitam adquirir e liberar recursos virtuais rapidamente, é importante que os usuários entendam o impacto que cada tipo de recurso oferecido pelo provedor pode exercer no desempenho das aplicações. Este trabalho apresenta um novo ambiente programável para apoiar os usuários na execução automática de testes de desempenho de aplicações em nuvens IaaS. O ambiente, denominado* Cloud Crawler*, é composto pela linguagem declarativa* Crawl*, usada para descrever diversos cenários de avaliação de desempenho de uma aplicação na nuvem, e o motor de execução* Crawler*, que executa os cenários descritos em* Crawl *e coleta os resultados das avaliações. O uso do ambiente é ilustrado através da avaliação do desempenho de uma aplicação de rede social, utilizando vários perfis de recursos virtuais sob diferentes níveis de demanda, em dois provedores IaaS comerciais.*

Abstract. *One of the main challenges faced by users of infrastructure-as-a-service (IaaS) clouds is the difficulty to adequately estimate the virtual resources necessary to their applications. Although many cloud providers offer fast ways to acquire and release resources, it is important that users have a prior understanding of the impact that each virtual resource type offered by the provider may impose on the performance of their applications. This work presents a new programmable environment to support users in automatically executing application performance tests in IaaS clouds. The environment, called* Cloud Crawler*, includes the* Crawl *declarative language, used to describe different performance evaluation scenarios for a given cloud application, and the* Crawler *execution engine, which executes the evaluation scenarios described in* Crawl *and collects their results. The use of the environment is illustrated through an evaluation of the performance of a social network application, using several virtual resources types under varying demand levels, in two commercial IaaS providers.*

1. Introdução

A computação em nuvem está revolucionando a indústria da Tecnologia da Informação, tendo sido cada vez mais adotada como uma alternativa mais econômica, escalável e robusta à aquisição de infraestrutura computacional própria por parte das

organizações [Armbrust et al. 2010]. Nesse contexto, tem havido um crescente interesse por nuvens do tipo IaaS (*infrastructure-as-a-service*), que oferecem recursos de infraestrutura (como máquinas virtuais, redes e espaço de armazenamento) na forma de serviço [Zhang et al. 2010]. Esse sucesso deve-se principalmente ao atrativo modelo de negócios oferecido pelos provedores desse tipo de nuvem, onde o preço cobrado pelos recursos é proporcional ao tempo de utilização, e ao fato de oferecerem poucas restrições a seus usuários quanto às tecnologias e aplicações que podem ser implantadas na infraestrutura virtual da nuvem [Rhoton 2009].

Por outro lado, o aumento do número de provedores de nuvens IaaS,[1] com cada provedor oferecendo uma ampla variedade de recursos virtuais com diferentes preços e perfis de configuração, traz à tona um novo desafio aos usuários interessados em disponibilizar suas aplicações na nuvem: como identificar os provedores e os perfis de recursos virtuais com o melhor custo/benefício para uma determinada aplicação? Para dimensionar os recursos virtuais a serem adquiridos de um provedor de nuvem, é necessário que o usuário, dono da aplicação, conheça bem não apenas as características (por exemplo, configuração, preço e localização geográfica) de cada perfil de recurso virtual oferecido pelo provedor, mas também o impacto que cada um desses perfis poderá ter no desempenho de sua aplicação. Se não houver um planejamento adequado por parte do usuário, a aplicação poderá vir a ser implantada na nuvem com recursos *aquém* ou *além* do necessário. Ambos os casos podem resultar em prejuízos para o usuário, embora por motivos distintos. No primeiro caso, a aplicação disporá de menos recursos do que de fato necessita, podendo afetar negativamente o seu desempenho ou até mesmo inviabilizar a sua execução; já no segundo, haverá desperdício de recursos, com o usuário tendo que pagar por algo que não irá efetivamente utilizar [Armbrust et al. 2010].

Embora vários trabalhos tenham sido publicados com foco na avaliação do desempenho de serviços e aplicações na nuvem (por exemplo, [Sobel et al. 2008, Ostermann et al. 2010, Malkowski et al. 2010, Li et al. 2010, Jayasinghe et al. 2011, Cunha et al. 2011], relativamente pouca atenção tem sido dada até o momento ao suporte ferramental necessário para facilitar a execução automática de diferentes testes de desempenho de aplicações por parte dos próprios usuários [Jayasinghe et al. 2012]. Este trabalho apresenta um novo ambiente programável para apoiar os usuários de nuvens IaaS na realização de testes automáticos de desempenho de aplicações na nuvem. As principais contribuições do ambiente são: a linguagem declarativa *Crawl*, com a qual os usuários podem especificar, através de uma notação simples e de alto nível de abstração, uma grande variedade de cenários de avaliação de desempenho de uma aplicação na nuvem; e o motor de execução *Crawler*, que automaticamente executa e coleta os resultados dos cenários descritos em *Crawl* em um ou mais provedores. Essas duas ferramentas, denominadas conjuntamente de *Cloud Crawler*, serão descritas em mais detalhes na Seção 2.

Para ilustrar as facilidades oferecidas pelo novo ambiente, foram realizados dois experimentos de avaliação de desempenho de uma aplicação de rede social em dois provedores IaaS comerciais, cujos cenários envolveram diferentes níveis de demanda pela aplicação e diferentes configurações de máquinas virtuais de cada provedor. Os resultados desses experimentos, descritos na Seção 3, mostram que o ambiente proposto, ao

[1] O sítio `cloudorado.com`, que oferece um serviço de busca e comparação de nuvens IaaS, lista, na data de elaboração do artigo, mais de 20 provedores localizados em diferentes partes do mundo.

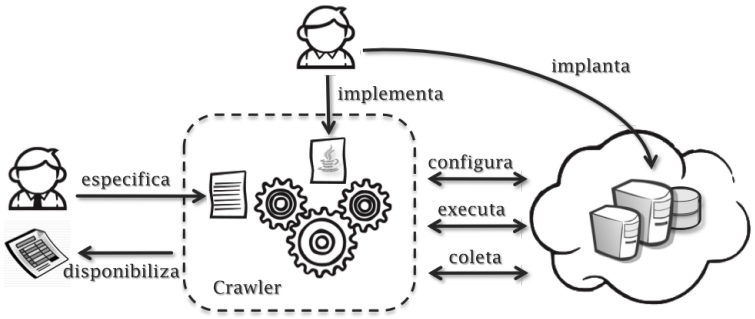


Figura 1. Visão geral do ambiente *Cloud Crawler*.

facilitar a especificação e a automação de diversos testes de desempenho da aplicação, pode ser uma importante ferramenta de suporte ao planejamento e à alocação de recursos virtuais em nuvens IaaS.

A última parte do artigo contém uma discussão mais pontual das contribuições do trabalho à luz de outros trabalhos relacionados (Seção 4), seguida das conclusões e sugestões para trabalhos futuros (Seção 5).

2. Ambiente *Cloud Crawler*

O ambiente *Cloud Crawler* é composto pela linguagem de descrição de cenários, *Crawl*, e o motor de execução dos cenários descritos em *Crawl*, de nome *Crawler*. O uso do ambiente envolve a execução de uma série de atividades por parte dos usuários, conforme ilustra a Figura 1.

Inicialmente, o usuário implanta os componentes da aplicação a ser avaliada (como sevidor web, servidor de banco de dados, etc.) na infraestrutura virtual do provedor de nuvem. Essa atividade é necessária haja visto que o ambiente ainda não oferece suporte à implantação automática de aplicações. Em seguida, o usuário implementa a interface a ser utilizada pelo *Crawler* para se comunicar com os componentes da aplicação na nuvem. Essa interface contém métodos básicos para interação com os recursos previamente implantados na nuvem, os quais devem ser customizados pelo usuário para atender as características específicas de sua aplicação. Após isso, o usuário especifica os cenários de avaliação da aplicação, utilizando a linguagem *Crawl*, e os repassa ao *Crawler* para que sejam executados na nuvem. De posse da especificação dos cenários em *Crawl*, e da implementação da interface de comunicação, o *Crawler* configura os recursos virtuais da nuvem de acordo com as necessidades de cada cenário, passando então a executá-los e a coletar seus resultados. Ao final, o *Crawler* consolida os resultados obtidos e os disponibiliza ao usuário na forma de planilhas, facilitando, assim, a sua posterior análise e visualização.

Vale ressaltar que a implantação dos componentes na nuvem e a implementação da interface de comunicação só precisam ser executas uma única vez por aplicação. Já a especificação e a execução dos cenários podem ser realizadas repetidas vezes, a critério do usuário. Dessa forma, é possível imaginar duas categorias (ou papéis) de usuários para o ambiente: especialistas no ambiente, que ficariam responsáveis pela sua extensão e customização para diferentes aplicações e provedores de nuvem; e especialista nas aplicações, que ficariam responsáveis por especificar os cenários de avaliação de desem-

penho e analisar e validar os seus resultados. Obviamente, nada impede que um mesmo usuário possa exercer ambos os papéis.

2.1. A Linguagem *Crawl*

Crawl[2] é uma linguagem declarativa criada com o objetivo de permitir a especificação das informações necessárias à execução automática de testes de desempenho em nuvens IaaS — tais como os componentes da aplicação e seus parâmetros de configuração, o provedor da nuvem, a quantidade e o perfil dos recursos virtuais a serem alocados a cada componente, e a carga de trabalho à qual a aplicação será submetida durante os testes —, através de um modelo sintático flexível e de alto nível de abstração.

Algumas características inerentes às nuvens IaaS tornam a especificação de cenários de avaliação para esse tipo de infraestrutura particularmente desafiadora. Uma delas é a necessidade de configurar certos componentes da aplicação com informações sobre os recursos virtuais da nuvem que só são conhecidos em tempo de execução. Por exemplo, em aplicações web é comum configurar os componentes da camada de negócio com o IP do servidor de banco de dados. Porém, na maioria dos provedores de nuvem IaaS, o IP de uma máquina virtual só é conhecido após a máquina ter sido efetivamente criada e iniciada, o que impede que os parâmetros de configuração dos componentes da aplicação possam ser totalmente definidos *a priori* (isto é, em tempo de especificação dos cenários de avaliação).

Outro desafio é a dinamicidade relacionada ao perfil dos recursos virtuais alocados à aplicação. Isso porque um mesmo componente da aplicação pode ter que ser configurado de diferentes maneiras, dependendo do perfil do recurso virtual que lhe for alocado. Por exemplo, o número adequado de *workers* (ou *threads*) que devem ser criados em um servidor web vai depender fortemente das características de hardware (como a quantidade de CPUs e o tamanho da memória RAM) da máquina virtual onde o servidor será executado. Para dar suporte a esse caráter dinâmico das aplicações na nuvem, *Crawl* oferece mecanismos que permitem (re)configurar, em tempo de execução dos cenários, diversos aspectos relacionados à utilização dos recursos da nuvem e de seus componentes de software.

2.1.1. Modelo de Entidades

Crawl provê um conjunto de dez tipos de entidades, conforme ilustrado na Figura 2, com as quais o usuário do ambiente pode especificar uma ampla variedade de cenários de avaliação de desempenho em nuvens IaaS. Estes tipos constituem o modelo de entidades da linguagem e estão descritos na Tabela 1. Para definir as entidades de *Crawl* que comporão os cenários de avaliação da aplicação, o usuário do ambiente faz uso da representação textual apresentada a seguir.

2.1.2. Representação Textual

Em princípio, qualquer notação textual estruturada poderia ser adotada para representar as entidades de *Crawl*. Nesse sentido, a linguagem de marcação XML surge

[2]O nome da linguagem é um acrônimo para **C***loud* **r***esource* **a***pplication and* **w***orkload* **l***anguage*.

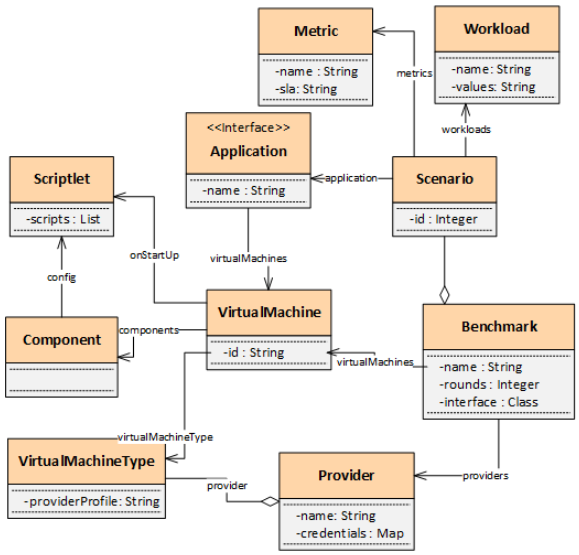


Figura 2. Modelo de entidades de *Crawl*.

como uma candidata natural, devido à sua universalidade e ao seu amplo suporte ferramental. Recentemente, porém, XML tem perdido espaço como formato de representação de dados para outras notações textuais mais simples e compactas, como JSON [JSON 2011] e Yaml [Ben-Kiki et al. 2001], particularmente no ambiente da web [Fonseca and Simões 2007]. Yaml, em especial, tem se destacado pela sua sintaxe extremamente simples e flexível, além de ser um superconjunto funcional de JSON [Fonseca and Simões 2007]. Por essas razões, Yaml foi escolhida como base para a representação textual da linguagem *Crawl*.

Yaml usa indicadores estruturais limpos e minimalistas, baseados fortemente na identação de elementos textuais encadeados [Ben-Kiki et al. 2001]. A Tabela 2 apresenta os principais indicadores de Yaml e suas respectivas funções sintáticas dentro da linguagem. Além da simplicidade, outra vantagem de Yaml é a sua flexibilidade, permitindo que novas entidades sejam definidas a qualquer momento e referenciadas sempre que necessário. Essas características se refletem em uma grande vantagem para o usuário da linguagem *Crawl*, que passa a ter em mãos uma simples mas poderosa notação para especificar os componentes de sua aplicação e os cenários de avaliação de seu interesse.

A Figura 3 mostra um exemplo de um cenário de avaliação especificado em *Crawl*. Nesse exemplo, o primeiro bloco da especificação (linhas 1-4) contém a definição de uma entidade do tipo *Benchmark* e três de seus atributos, os quais definem, respectivamente, o nome do *Benchmark* (linha 2), o número de vezes que este deverá executado pelo *Crawler* (linha 3), e o intervalo de tempo, em milisegundos, a ser aguardado entre cada uma de suas execuções (linha 4).

Em seguida, nas linhas linhas 7-16, é definido o provedor de nuvem a ser utilizado (no caso, Rackspace), cujos atributos incluem as credencias de acesso ao provedor, tal como fornecidas pelo usuário (linha 9), e dois perfis de máquinas virtuais, de nomes *flavor1* (linhas 11-13) e *flavor2* (linhas 14-16). As linhas 19-37, por sua vez, contêm a definição de duas máquinas virtuais do provedor Rackspace. A primeira, do perfil *flavor1*, está configurada com uma instância do servidor web Apache (linha 24), enquanto

Tipo	Descrição
Benchmark	Representa a entidade maior do modelo. Contém o contexto global de uma avaliação de desempenho especificada em *Crawl*. Nela, são definidos os provedores de nuvem, os cenários de avaliação e demais atributos de escopo global.
Provider	Representa um provedor de nuvem. Possui como atributos as credenciais de acesso à nuvem, tal como fornecidas pelo usuário, e uma lista com os diferentes perfis de máquinas virtuais fornecidos pelo provedor.
VirtualMachineType	Representa um perfil de máquina virtual oferecido pelo provedor de nuvem.
VirtualMachine	Representa uma máquina virtual a ser utilizada para executar um ou mais componentes de uma aplicação na nuvem. A associação entre a entidade declarada em *Crawl* e a máquina virtual previamente criada no provedor é feita através do atributo *ID*, cujo valor deve ser idêntico ao ID da máquina virtual tal como atribuído pelo provedor. Pode ser declarada com escopo global, associada à entidade *Benchmark* (nesse caso, a máquina virtual será ligada no início da execução dos cenários e desligada após o último cenário ter sido executado), ou local, associada à entidade *Application* (nesse caso, a máquina virtual será ligada e desligada apenas no contexto da execução do cenário onde foi declarada).
Component	Entidade que permite descrever os componentes da aplicação (por exemplo, servidor web, servidor de banco de dados, gerador de carga) que serão executados nas respectivas máquinas virtuais.
Scriptlet	Representa um *script* que será executado em uma máquina virtual como parte da configuração inicial de um componente ou serviço alocado a essa máquina.
Application	Representa a aplicação cujo desempenho será avaliado na nuvem. Através dessa entidade é possível definir a arquitetura de implantação da aplicação em termos de suas máquinas virtuais, que podem ser tanto de escopo local quanto global.
Metric	Representa uma métrica de desempenho (por exemplo, tempo médio de resposta, tempo total de execução, etc.) que será utilizada como critério de avaliação em um ou mais cenários.
Workload	Representa uma carga de trabalho a ser submetida à aplicação. O mecanismo de geração de carga é definido pelo próprio usuário, como parte da implementação da interface de comunicação com a nuvem utilizada pelo *Crawler*.
Scenario	Representa um cenário de avaliação de desempenho que será executado na nuvem pelo *Crawler*. Cada cenário é especificado envolvendo uma única aplicação. Possui como atributos um conjunto de métricas e um conjunto de cargas de trabalho, cujos valores definem como o *Crawler* irá avaliar o desempenho da aplicação na nuvem.

Tabela 1. Descrição das entidades de *Crawl*.

Símbolo	Função
:	Atribui o valor especificado no lado direito à chave declarada no lado esquerdo
–	Declara um elemento de uma sequência
!	Denota um tipo de entidade
&	Denota uma entidade referenciável
*	Referencia uma entidade previamente declarada

Tabela 2. Principais indicadores sintáticos de Yaml.

a segunda, do perfil *flavor2*, está configurada com uma instância do servidor de banco de dados Postgres (linha 36). Note que as duas máquinas virtuais estão associadas aos seus respectivos recursos virtuais na nuvem, os quais devem ter sido previamente criados junto ao provedor, através dos identificadores 7HJ38K (linha 21) e YH838K (linha 33). Note, ainda, que foi definida uma entidade do tipo *Scriptlet* como atributo da máquina virtual configurada com o servidor Apache (linhas 27-30). Essa entidade contém os dados do *script* que será executado pelo *Crawler* como parte do procedimento de inicialização dessa máquina virtual, e que tem como finalidade configurar o servidor web com o IP da máquina virtual onde será executado o servidor de banco de dados.

A última parte da especificação (linhas 40-52) contém a definição do cenário propriamente dito. Note que o cenário inclui a definição de uma carga de trabalho, cujos valores variam entre 100 e 400, representando o número de usuários concorrentes da aplicação (linhas 43-46) e de uma métrica de desempenho baseada no tempo de resposta da aplicação, com um acordo de nível de serviço (SLA) de 90%[3] (linhas 47-49). Por

[3]Um SLA de 90% indica que pelo menos 90% das requisições enviadas à aplicação pelo gerador de carga devem ser respondidas em um tempo igual ou inferior ao tempo máximo estabelecido pelo usuário.

```
1  !benchmark
2  name: bench_1
3  rounds: 3
4  interval: 5000
5
6  # definição do provedor
7  providers:
8   -&rackspace !provider
9     credentialPath: Rackspace.txt
10    -virtualMachineTypes:
11       -&flavor1 !virtualMachineType
12          providerProfile: 1
13          provider: *rackspace
14       -&flavor2 !virtualMachineType
15          providerProfile: 2
16          provider: *rackspace
17
18 # definição das máquinas virtuais
19 virtualMachines:
20  - &web_vm !virtualMachine
21    id: 7HJ38K
22    type: *flavor1
23    components:
24     - &apache !component
25        id: 1
26        onStartup:
27         -&apacheConfig !scriptlet
28           id: 1
29           scripts:
30            db.sh ${db.IP}
31
32  - &database_vm !virtualMachine
33    id: YH838K
34    type: *flavor2
35    components:
36       - &postgres !component
37          id: 2
38
39 # definição do cenário
40 scenarios:
41   - !scenario
42     id: 1
43     workloads:
44      - &load !workload
45         id: 1
46         values: [100,200,300,400]
47     metrics:
48      - &responseTime !metric
49        sla: 90
50     web_app: !application
51       db_layer: *database_vm
52       web_layer: *web_vm
```

Figura 3. Exemplo da especificação de um cenário de avaliação em *Crawl*.

fim, as linhas 50-52 definem a arquitetura de implantação da aplicação, a qual é composta pelas duas máquinas virtuais configuradas com os servidores Apache e Postgres, respectivamente.

De modo a facilitar ainda mais a especificação dos cenários de avaliação por parte dos usuários, *Crawl* oferece duas extensões à sintaxe original de Yaml. A primeira permite a inclusão de entidades definidas em arquivos externos, o que torna as especificações em *Crawl* mais modulares, reutilizáveis e fáceis de compreender, particularmente para cenários envolvendo um grande número de entidades. A segunda extensão permite a definição de laços de interação, através dos quais é possível definir, de uma forma simples e compacta, um conjunto de entidades similares que compartilham vários elementos em comum. Essas duas extensões são ilustradas na Figura 4, que mostra um exemplo de uma especificação em *Crawl* com as definições dos conjuntos de perfis de máquinas virtuais fornecidos pelos provedores Amazon EC2 e Rackspace. Note que a definição dos perfis do provedor Amazon EC2 é incluída diretamente de um arquivo externo, disponível no caminho /lib/providers/ec2/vm_profiles.crawl, através da definição da entidade especial include (linhas 2-3).

Já a definição dos perfis do provedor Rackspace é feita utilizando um laço de iteração, através da declaração da entidade especial foreach (linhas 6-14). Essa entidade possui quatro atributos: list, que define a lista com os valores a serem iterados no laço; count, que corresponde ao contador de iterações do laço, cujo valor inicial é definido pelo usuário e incrementado de uma unidade cada nova iteração; var, que identifica o nome da variável que receberá cada item da lista de valores do laço; e statement, que contém a definição de todos elementos que fazem parte do escopo do laço. No escopo do laço é possível usar um segundo delimitador de variável, $[], através do qual pode-se referenciar os atributos count e var, os quais serão substituídos por seus respectivos valores a cada iteração do laço. Dessa forma, a cada iteração são geradas novas definições para os elementos contidos no escopo do laço, com cada definição diferindo das anteriores apenas nos trechos que referenciam os valores dos atributos count e var.

No caso do exemplo da Figura 4, note que a lista de valores do laço (linha 8) corresponde exatamente à sequência de inteiros que identificam cada um dos perfis disponibilizados pelo provedor Rackspace. Note também que apenas uma entidade do tipo *VirtualMachineType* está sendo explicitamente definida dentro do escopo do laço (linhas 12-14), e que tanto o nome dessa entidade (linha 12) quanto o valor de seu atributo de nome providerProfile (linha 13) fazem referência à variável do laço, no caso, flavor. Durante o

```
1  # definição dos perfis do provedor Amazon EC2
2  !include
3    file: /lib/providers/ec2/vm_profiles.crawl
4
5  # definição dos perfis do provedor Rackspace
6  virtualMachineTypes:
7  - !foreach
8    list: [1, 2, 3, 4, 5, 6, 7, 8, 9]
9    count: 0
10   var: flavor
11   statement:
12 - &flavor${flavor} !virtualMachineType
13     providerProfile: ${flavor}
14     provider: *rackspace
```

Figura 4. Especificação de múltiplos perfis de máquinas virtuais em *Crawl*.

processamento da especificação *Crawl*, realizado pelo motor *Crawler*, essa variável será substituída por cada um dos valores da lista em cada iteração do laço (no caso, 1, 2, 3 e assim sucessivamente), resultando na definição de nove novas entidades do tipo *VirtualMachineType*, diferentes entre si apenas nos trechos que foram substituídos pelos valores das variáveis do laço.

2.2. O Motor *Crawler*

O motor *Crawler* é o mecanismo encarregado de validar, executar e coletar os resultados dos cenários de avaliação descritos em *Crawl*. Também fazem parte de suas funções gerenciar o ciclo de vida das máquinas virtuais na nuvem e configurar os componentes da aplicação de acordo com as necessidades de cada cenário. A versão atual do motor foi desenvolvida inteiramente em Java, na forma de um serviço RESTful que pode ser executado tanto em uma infraestrutura local quanto em algum provedor de nuvem.

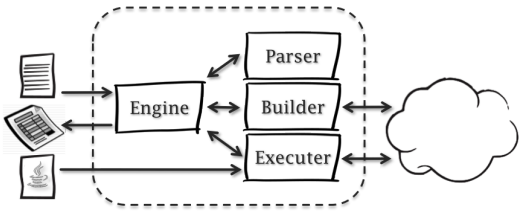


Figura 5. Arquitetura do *Crawler*.

A arquitetura do *Crawler* é composta de quatro módulos: *Engine*, *Parser*, *Builder* e *Executer* (ver Figura 5). *Engine* é o módulo principal da ferramenta, e tem como função responder aos comandos de invocação do usuário e coordenar as ações dos demais módulos. O *Parser* tem a função de interpretar e validar a especificação dos cenários descritos em *Crawl*, também sendo responsável pela resolução das variáveis declaras dinamicamente na linguagem. A função do *Builder* é preparar os recursos virtuais na nuvem e configurar os componentes da aplicação que serão executados nesses recursos. Para a preparação dos recursos, o *Builder* utiliza a biblioteca de código aberto *jclouds*,[4] e para a configuração dos componentes na nuvem é utilizada a ferramenta *JSch*.[5] O módulo *Executer* se encarrega de executar os cenários na nuvem com os recursos preparados pelo *Builder* e de coletar os resultados das avaliações. A interação do *Executer* com os componentes da aplicação durante a execução dos cenários é feita exclusivamente através da interface de comunicação com a nuvem, cuja implementação é fornecida pelo usuário. Ao final, o módulo *Engine* consolida os dados coletados em cada cenário pelo *Executer* e em seguida os disponibiliza ao usuário na forma de planilhas.

[4] http://www.jclouds.org/
[5] http://www.jcraft.com/jsch/

Uma característica importante do *Crawler* é a forma como ele gerencia o ciclo de vida das máquinas virtuais na nuvem. Em vez de solicitar a criação de novas máquinas virtuais ao provedor de nuvem, o *Crawler* reutiliza as máquinas virtuais previamente criadas pelo usuário, apenas alterando os seus perfis, caso necessário. Essa estratégia tem como principal vantagem a diminuição do custo de execução dos cenários na nuvem, uma vez que, como a maioria dos provedores IaaS atuais adota um modelo de precificação onde é cobrado um mesmo valor por hora ou fração de hora utilizada, dependendo do tempo de execução dos cenários, reutilizar uma máquina virtual existente pode sair muito mais barato do que solicitar a criação de uma nova máquina ao provedor.

3. Experimentos

Esta seção descreve dois experimentos realizados com o intuito de ilustrar o uso e as facilidades propiciadas pelo ambiente *Cloud Crawler*.

3.1. Metodologia e Especificação dos Cenários

Os dois experimentos envolveram a avaliação do desempenho de uma mesma aplicação de rede social de código aberto, Olio,[6] em dois provedores IaaS comercias, Amazon EC2 e Rackspace, os mais utilizados atualmente.[7] O objetivo era investigar quais configurações de recursos virtuais de cada provedor ofereceriam a melhor relação custo/desempenho para a aplicação, considerando diferentes níveis de demanda.

A Olio foi originalmente desenvolvida no âmbito do projeto Cloudstone [Sobel et al. 2008], com parte de um novo *benchmark* para nuvens IaaS. A aplicação está disponível em duas versões, PHP e Ruby on Rails (RoR), sendo que apenas essa última foi utilizada nos experimentos. Ambas as versões adotam uma arquitetura web tradicional, composta de três camadas: apresentação, aplicação e dados. No caso da versão RoR, os componentes utilizados em cada camada são, respectivamente, o servidor web Nginx,[8] o servidor de aplicação Thin,[9] e o servidor de banco de dados MySQL.[10] Em princípio, cada um desses componentes pode ser executado em uma máquina diferente, embora a arquitetura de implantação típica da Olio seja executar os servidores Nginx e Thin concorrentemente, em uma mesma máquina, com o Nginx servindo de *proxy* reverso e balanceador de carga para as múltiplas instâncias do Thin, deixando o servidor de banco de dados em uma máquina separada [Sobel et al. 2008]. Essa mesma arquitetura foi utilizada nos experimentos.

Para simular os diferentes níveis de demanda a que a Olio seria submetida nos diferentes cenários, foi escolhida a ferramenta de geração de carga Faban,[11] também utilizada no projeto Cloudstone. Visando limitar o número total de configurações de recursos da nuvem a serem investigados em cada provedor, decidiu-se implantar a aplicação fixando-se os perfis das máquinas virtuais onde seriam executados o gerador de carga e o servidor de banco de dados. No caso da Amazon, utilizou-se o perfil *c1.xlarge*, o mesmo utilizado no Cloudstone; já no caso do Rackspace, utilizou-se o perfil *flavor 5*,

[6] http://incubator.apache.org/projects/olio.html
[7] http://gigaom.com/cloud/amazon-is-no-1-whos-next-in-cloud-computing/
[8] http://nginx.org/en/
[9] http://code.macournoyer.com/thin/
[10] http://www.mysql.com/
[11] http://java.net/projects/faban

que possui uma configuração de hardware similar ao referido perfil da Amazon. Dessa forma, as variações concentraram-se no perfil e na quantidade das máquinas virtuais utilizadas para executar os servidores Nginx e Thin, bem como nos valores dos parâmetros de configuração específicos de cada um desses dois servidores (no caso, o número de *workers* do Nginx e o número de instâncias do Thin). Essa estratégia de implantação permitiu avaliar tanto a escalabilidade horizontal da aplicação, alterando-se o número de máquinas virtuais alocadas aos servidores Nginx e Thin, quanto a sua escalabilidade vertical, alterando-se o perfil das máquinas alocadas a esses dois componentes e ajustando-se os seus respectivos parâmetros de configuração de acordo com o perfil escolhido.

Em todos os cenários investigados, a aplicação foi submetida a duas cargas de trabalho, definidas em termos do número de usuários concorrentes que seriam simulados pelo gerador de carga. Na primeira carga, denominada *demanda baixa*, o número de usuários concorrentes variava entre 25 e 150, enquanto na segunda, denominada *demanda moderada*, esse número variava entre 200 e 800. Durante os experimentos, cada cenário foi executado três vez para cada número de usuários de cada uma das cargas de trabalho. Essa medida visava capturar eventuais flutuações no desempenho dos recursos virtuais da nuvem, decorrentes da possível concorrência de outros usuários compartilhando a mesma infraestrutura física.

Como métrica de desempenho, foi utilizado o tempo de resposta da aplicação. Cada execução de um cenário para cada nível de demanda era considerada bem sucedida se pelo menos 90% das respostas recebidas pelo gerador de carga estivessem dentro do limite máximo estabelecido para cada operação da aplicação. Nos experimentos, esses limites foram definidos com os mesmos valores utilizados no projeto Cloudstone [Sobel et al. 2008].

A Figura 6 mostra a especificação em *Crawl* dos cenários de avaliação da aplicação Olio na nuvem Amazon EC2.[12] A especificação para a nuvem Rackspace é feita de forma análoga, mudando-se apenas as definições relativas aos perfis de máquinas virtuais de cada provedor, e foi omitida por limitações de espaço.

A primeira parte da especificação (linhas 1-41) define os atributos globais do *Benchmak*, incluindo: um conjunto de perfis de máquinas virtuais (linhas 8-20), utilizados como perfis das máquinas virtuais onde serão executados os servidores Nginx e Thin; duas máquinas virtuais (linhas 22-34), onde serão executados o gerador de carga (Faban) e o servidor de banco de dados (MySQL), respectivamente; e duas cargas de trabalho (linhas 36-41). Note que a entidade especial include é utilizada, logo no início da especificação (linhas 4-5), para importar as definições relativas aos provedores de nuvem de um arquivo externo de nome providers.crawl. Note ainda que cada perfil de máquina virtual foi definido com uma lista de propriedades contendo dois atributos, de nomes workers e thinServers. Através desses dois atributos o usuário pode definir parâmetros de configuração específicos da aplicação, os quais serão utilizados pelo *Crawler* durante a execução dos cenários para configurar o número de *threads* do sevidor Nginx e o número de instâncias do servidor Thin, respectivamente, de acordo com as características de hardware das máquinas virtuais onde cada um será executado.

A segunda e última parte da especificação (linhas 43-101) define os cenários de

[12] A especificação foi ligeiramente modificada no artigo para facilitar a sua visualização.

```
1  !benchmark
2  name: OlioBenchmark
3  rounds: 3
4  - !include
5    file: providers.crawl
6
7  properties:
8   - &low_prof !virtualMachineType
9     providerProfile: t1.micro
10    provider: *ec2
11    properties:
12     - workers: 2
13     - thinServers: 2
14  ...
15  - &high_prof !virtualMachineType
16    providerProfile: c1.xlarge
17    provider: *ec2
18    properties:
19     - workers: 6
20     - thinServers: 18
21
22 virtualMachines:
23  - &faban !virtualMachine
24     id: 20408853
25     type: *high_prof
26     name: DRIVER
27     components:
28      - *fabanApp
29  - &mysql !virtualMachine
30     id: 20406221
31     type: *high_prof
32     name: MYSQL
33     components:
34      - *mysqlApp
35
36 workloads:
37  - &low !workload
38    values: [25,50,75,100,125,150]
39  - &moderate !workload
40    values: [200,300,400,500,600,
41                700,800]
42
43 scenarios:
44  # implantação vertical
45  - !foreach
46    list: [*low_prof, ..,
47           *high_prof]
48    var: profile
49    statement:
50    - !scenario
51      workloads:
52       - *low
53       - *moderate
54      metrics:
55       - *responseTime
56      application: !OlioApp
57        database: *mysql
58        driver: *faban
59        web:
60         - !virtualMachine
61           id: 20400657
62           name: THIN SERVER
63           type: $[profile]
64           components:
65            - *railsApp
66            - *tomcatApp
67            - *nginxApp
68            - !component
69              config:
70               - &db_script !scriptlet
71                 scripts:
72                   - sudo
73                     /home/db.sh ${IP}
74
75 # implantação horizontal
76 - !scenario
77   workloads:
78    - *low
79    - *moderate
80   metrics:
81    - *responseTime
82   application: !OlioApp
83     database: *mysql
84     driver: *faban
85     web:
86      - !foreach
87        list: [239999, ..,
88               9989989]
89        var: machine_id
90        statement:
91         - !virtualMachine
92           id: $[machine_id]
93           name: THIN SERVER
94           type: *high_profile
95           components:
96            - *railsApp
97            - *tomcatApp
98            - *nginxApp
99            - !component
100             config:
101              - *db_script
```

Figura 6. Especificação dos cenários de avaliação da Olio na Amazon EC2.

avaliação da Olio considerando duas arquiteturas de implantação: uma arquitetura vertical, na qual os servidores Nginx e Thin são implantados em uma única máquina virtual com diferentes perfis (linhas 45-73); e uma arquitetura horizontal, na qual os servidores Nginx e Thin são implantados em múltiplas máquinas virtuais de mesmo perfil (linhas 76-101). Note que a entidade especial foreach é utilizada em ambas arquiteturas de implantação para definir as máquinas virtuais de cada cenário variando apenas seus perfis (linhas 45-49) ou seus IDs (linhas 86-90). Note também que as máquinas virtuais da camada web em ambas arquiteturas são definidas reutilizando um mesmo *script* de configuração (o qual é definido nas linhas 70-73 e referenciado na linha 101), cujo papel é configurar os componentes da camada web com o endereço IP da máquina virtual onde será executado o servidor de banco de dados.

Deve-se enfatizar aqui o modo através do qual foram definidas as máquinas virtuais utilizadas na avaliação da Olio. As máquinas virtuais referentes ao balanceador de carga e ao servidor de banco de dados foram definidas como sendo de escopo global, o que significa que essas máquinas serão ligadas no início do processo de avaliação e desligadas apenas ao seu final. Essa decisão justifica-se pelo fato de que essas duas máquinas serão utilizadas na execução de todos os cenários de avaliação definidos para a Olio. Já as máquinas virtuais referentes aos componentes da camada web foram definidas como sendo de escopo local, o que significa que elas serão ligadas e desligadas no contexto específico de seus respectivos cenários. Dessa forma, máquinas virtuais que não forem realmente necessárias em um cenário específico permanecerão desligadas durante a execução desse cenário pelo *Crawler*, resultando em uma significativa economia de recursos para o usuário da nuvem.

Devido a restrições de espaço, também foram omitidos deste artigo os aspectos referentes à implantação dos componentes da Olio nos dois provedores e à customização do *Crawler* através da implementação de sua interface de comunicação. Uma descrição mais detalhada desses e de outros aspectos do ambiente pode ser encontrado em [Cunha 2012].

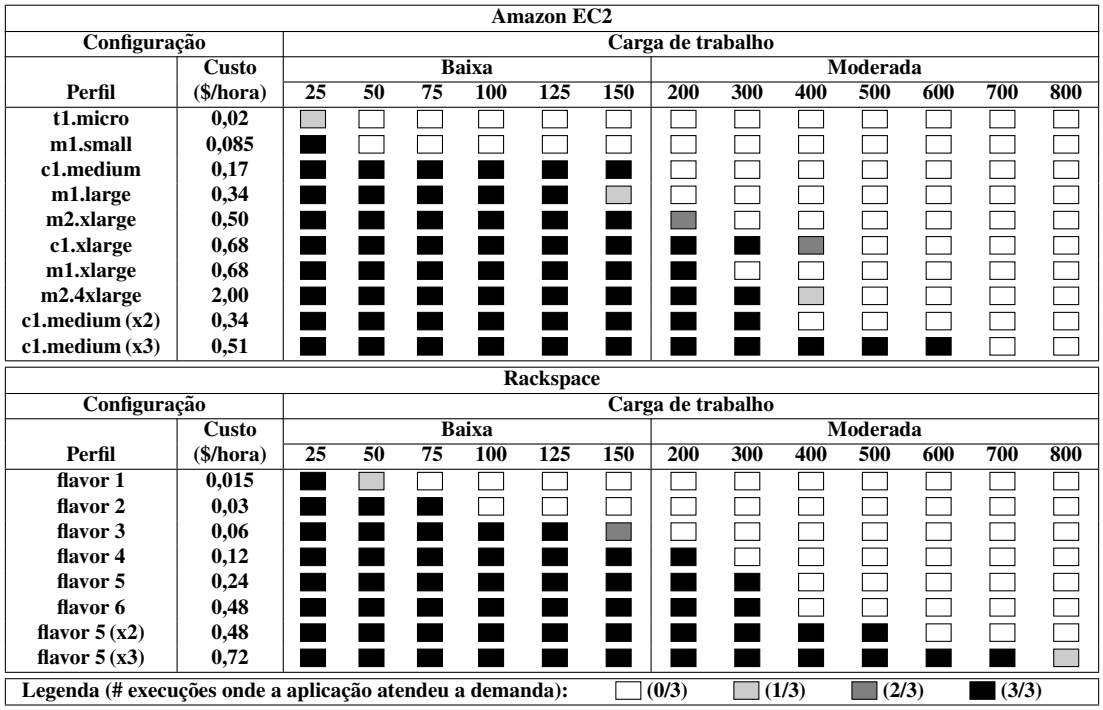


Tabela 3. Desempenho da aplicação Olio nas nuvens Amazon EC2 e Rackspace.

3.2. Resultados

A Tabela 3 mostra uma visão consolidada dos resultados coletados pelo *Crawler* após a execução dos diversos cenários de avaliação da Olio nos dois provedores.

Pelos resultados mostrados na Tabela 3, é possível observar que há diferenças significativas entre as diversas configurações de máquinas virtuais avaliadas de cada provedor, tanto em termos de desempenho quanto de custo. Por exemplo, no provedor Amazon EC2, a configuração com uma máquina virtual de perfil *c1.medium* apresenta um desempenho comparável (conseguindo ser até ligeiramente superior) ao da configuração com uma máquina virtual de perfil *m1.large* para níveis baixos de demanda, embora a segunda tenha o dobro do custo da primeira. Outra constatação com respeito a esse provedor é que configurações com preços similares podem apresentar desempenho bastante diferentes, como é o caso das configurações formadas pelas máquinas virtuais de perfil *c1.xlarge* e *m1.xlarge*, respectivamente, onde a primeira apresenta um desempenho nitidamente superior ao da segunda para níveis moderados de demanda. No caso do provedor Rackspace, as diferenças de desempenho entre as configurações foram, até certo ponto, proporcionais às suas diferenças em termos de custo, com configurações mais caras oferecendo desempenhos gradualmente superiores ao das configurações mais baratas. A exceção foi a configuração com uma máquina virtual de perfil *flavor 6*, que manteve o nível de desempenho da configuração anterior, embora custando o dobro do preço.

O uso de escalabilidade horizontal mostrou-se uma alternativa de implantação bastante atraente em ambos os provedores. No provedor Amazon EC2, por exemplo, a configuração com três máquinas virtuais de perfil *c1.medium* consegue atender uma demanda de até 600 usuários concorrentes, o que não é possível com nenhuma outra configuração, a um custo muito mais baixo do que o da maioria das configurações avali-

adas. Um resultado parecido foi observado no provedor Rackspace, onde a configuração com três máquinas virtuais de perfil *flavor 5* consegue atender uma demanda de até 800 usuários concorrentes, representando um ganho de quase 170% sobre a demanda máxima atendida pela configuração de maior preço formada por uma única máquina virtual.

4. Trabalhos Relacionados

Até o momento, poucos trabalhos foram publicados descrevendo soluções automatizadas para apoiar os usuários de nuvens IaaS na realização de seus próprios testes de desempenho. Em [Snellman et al. 2011], os autores propõem um gerador de carga que usa recursos de nuvens IaaS para a realização de testes de desempenho de aplicações web. Esse trabalho, porém, não oferece suporte automatizado para variar a arquitetura de implantação das aplicações sendo avaliadas. Um outro trabalho recente foi descrito em [Jayasinghe et al. 2012], onde os autores propõem um ambiente para execução automática de testes de desempenho em nuvens IaaS. Nesse ambiente, chamado Expertus, os usuários implementam testes de desempenho através da geração e customização de *scripts* Shell a partir de *templates* especificados na forma de documentos XML [Jayasinghe et al. 2012]. A solução adotada no Expertus, por envolver o uso de linguagens imperativas, ainda exige um alto grau de conhecimento sobre comandos básicos de configuração de sistemas por parte dos usuário. Além disso, a especificação dos testes em uma sintaxe baseada em *scripts* Shell pode acarretar problemas de portabilidade, dificultando a sua execução e reutilização em máquinas virtuais configuradas com diferentes linguagens de comando ou diferentes sistemas operacionais.

Para facilitar a especificação, execução e reutilização de testes de desempenho em diferentes tipos de máquinas virtuais e sistemas operacionais, este trabalho propõe uma nova linguagem declarativa para a descrição, em um alto nível de abstração, das entidades (componentes da aplicação, parâmetros de configuração, níveis de demanda, métricas de desempenho, perfis de recursos virtuais, etc.) que compõem um cenário de avaliação de desempenho na nuvem. Dessa forma, cabe ao motor de execução da linguagem interpretar, configurar e executar os cenários descritos pelos usuários na infraestrutura virtual do provedor de nuvem, contribuindo, assim, para uma drástica redução da complexidade e do esforço tradicionalmente envolvidos neste tipo de atividade.

5. Conclusão

Este trabalho apresentou o ambiente *Cloud Crawler*, que tem com objetivo facilitar a execução de testes de desempenho de aplicações em provedores de nuvens IaaS. O ambiente oferece um conjunto de tecnologias que permitem especificar, através de uma notação de alto nível de abstração, diferentes cenários de avaliação de desempenho na nuvem, os quais são automaticamente executados pelo ambiente. Conforme foi demonstrado através dos dois experimentos descritos no artigo, o ambiente constitui-se em uma importante ferramenta de planejamento e alocação de recursos em nuvens IaaS, podendo contribuir de forma significativa para um melhor aproveitamento dos recursos da nuvem por parte dos usuários, não apenas em termos de desempenho mas também de custo.

Atualmente, há uma série de linhas de pesquisa sendo consideradas visando melhorar e estender as funcionalidade do ambiente. Entre elas, destacam-se: o suporte à implantação automática dos componentes da aplicação na nuvem; a inclusão de meca-

nismos para permitir o controle e o monitoramento do fluxo de trabalho do motor de execução; e a implementação de uma ferramenta gráfica de edição e geração de cenários.

Agradecimentos

Este trabalho é parcialmente financiado pela Fundação Edson Queiroz, Universidade de Fortaleza, através do Projeto OW2.

Referências

Armbrust, M. et al. (2010). A view of cloud computing. *CACM*, 53(4):50–58.

Ben-Kiki, O. et al. (2001). YAML Ain't Markup Language (YAML) Version 1.1. 11.

Cunha, M. (2012). Um Ambiente Programável para Avaliar o Desempenho de Aplicações em Nuvens de Infraestrutura. Master's thesis, Universidade de Fortaleza.

Cunha, M. et al. (2011). Investigating the Impact of Deployment Configuration and User Demand on a Social Network Application in the Amazon EC2 Cloud. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 746–751. IEEE Computer Society.

Fonseca, R. and Simões, A. (2007). Alternativas ao XML: YAML e JSON. In *5ª Conferência Nacional em XML, Aplicações e Tecnologias Aplicadas (XATA 2007)*.

Jayasinghe, D. et al. (2011). Variations in Performance and Scalability When Migrating n-Tier Applications to Different Clouds. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 73–80. IEEE Computer Society.

Jayasinghe, D. et al. (2012). Expertus: A Generator Approach to Automate Performance Testing in IaaS Clouds. In *Cloud Computing (CLOUD), 2012 IEEE International Conference on*, pages 73–80.

JSON (2011). JavaScript Object Notation. `http://www.json.org/`.

Li, A. et al. (2010). CloudCmp: Comparing Public Cloud Providers. In *Internet Measurement Conference*.

Malkowski, S. et al. (2010). CloudXplor: A tool for configuration planning in clouds based on empirical data. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 391–398. ACM.

Ostermann, S. et al. (2010). A performance analysis of EC2 cloud computing services for scientific computing. *Cloud Computing*, pages 115–131.

Rhoton, J. (2009). *Cloud Computing Explained: Handbook for Enterprise Implementation*. Recursive Ltd.

Snellman, N., Ashraf, A., and Porres, I. (2011). Towards Automatic Performance and Scalability Testing of Rich Internet Applications in the Cloud. In *Software Engineering and Advanced Applications, 2011 37th EUROMICRO Conference on*.

Sobel, W. et al. (2008). Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0. In *Proc. of CCA*. Citeseer.

Zhang, Q. et al. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.

IoNCloud: uma abordagem não entrópica orientada a tráfego para reserva e isolamento de recursos em nuvens

Miguel C Neves[1], Daniel S Marcon[1], Rodrigo R Oliveira[1], Leonardo R Bays[1], Luciano P Gaspary[1], Marinho P Barcellos[1]

[1]Universidade Federal do Rio Grande do Sul (UFRGS)
Instituto de Informática

{mcneves, daniel.stefani, ruas.oliveira, lrbays, paschoal, marinho}@inf.ufrgs.br

Resumo. *Na computação em nuvem, a interferência entre tráfegos de naturezas diferentes gera imprevisibilidade no desempenho das aplicações. Propostas recentes, baseadas na formação de* clusters *virtuais isolando aplicações/locatários, resultam em subutilização de recursos ou aumento no custo de implementação. Neste artigo, propomos IoNCloud, um modelo de reserva e isolamento de recursos baseado nas características de tráfego das aplicações. Em IoNCloud, aplicações são atraídas ou repelidas de um mesmo* cluster *de forma a minimizar a interferência entre tráfegos. Experimentos demonstram que, em comparação com abordagens anteriores, IoNCloud acarreta baixa interferência entre tráfegos e reduz a subutilização dos recursos da nuvem. Ademais, IoNCloud não requer mudanças significativas na infraestrutura atual.*

Abstract. *The heterogeneity of traffic in intra-cloud networks leads to network performance variability (interference among traffic flows), which hinders application performance. Recent work addresses this issue by isolating applications/tenants in virtual networks, resulting in underutilization of resources or increased implementation costs. In this paper, we introduce IoNCloud, a traffic-based resource reservation and isolation model. In IoNCloud, an application is attracted to or repelled from a cluster according to the traffic interference it causes in other applications. Experiments show that, in comparison to previous work, IoNCloud leads to lower network performance interference and reduces the underutilization of resources. Moreover, IoNCloud does not require significant changes to current cloud datacenter infrastructures.*

1. Introdução

Utilização sob demanda e elástica de recursos e pagamento apenas pelo uso são alguns dos fatores que tornaram a computação em nuvem um paradigma atraente e em ampla difusão. Nesse contexto, benefícios como a redução dos custos com operação e manutenção da infraestrutura computacional fazem com que novos locatários migrem suas aplicações para os grandes *data centers* de nuvem [Armbrust et al. 2010]. Com o intuito de reduzir custos operacionais e viabilizar o modelo comercial proposto pelo paradigma, provedores de nuvem costumam empregar técnicas de multiplexação de recursos. Tal prática leva ao compartilhamento de recursos físicos por um elevado número de aplicações provenientes de diferentes locatários [Genez et al. 2012].

Recentemente, o desempenho baixo e imprevisível das aplicações, causado principalmente pela multiplexação dos recursos da rede interna da nuvem, tornou-se uma crescente preocupação [Ballani et al. 2011]. O fenômeno ocorre quando aplicações com tráfegos de naturezas diferentes compartilham o mesmo conjunto de dispositivos físicos [Benson et al. 2010]. Como alternativa, a literatura recente propõe o uso de virtualização de redes para isolar os locatários, de modo a oferecer garantia de recursos às aplicações e isolamento entre tráfegos [Guo et al. 2010, Ballani et al. 2011, Xie et al. 2012]. Tais estratégias levam à instanciação de uma rede virtual (ou *cluster* virtual) por aplicação ou por locatário. Embora reduza interferências, esse tipo de abordagem pode acarretar subutilização ou sobrecarga de gerenciamento dos recursos de rede da nuvem. Ademais, essas limitações são agravadas pela natureza variável dos requisitos de rede da maioria das aplicações hospedadas nos *data centers* de nuvem [Benson et al. 2010].

Este artigo propõe IoNCloud (*Isolation of Networks in the Cloud*), um modelo de reserva e isolamento de recursos de rede, como solução eficiente às questões de interferência entre tráfegos e imprevisibilidade de desempenho das aplicações. Em primeiro lugar, IoNCloud analisa os padrões de tráfego gerados pelas máquinas virtuais (*Virtual Machines* – VMs) das aplicações da nuvem, de forma a quantificar a necessidade de isolamento dos recursos de rede entre pares de aplicações. Em seguida, IoNCloud agrupa instâncias de aplicações e suas respectivas VMs em *Clusters* Virtuais Orientados a Tráfego (*Traffic-Oriented Virtual Clusters* – TOVCs), de acordo com as necessidades de isolamento. Esse procedimento leva à criação de grupos de aplicações de tal forma que a interferência entre tráfegos alocados a um mesmo TOVC seja a menor possível.

As principais contribuições deste artigo são resumidas a seguir:

- **Criação de métrica para expressar a necessidade de isolamento (Seção 3).** Definimos *entropia* como a variação em relação à média da demanda de determinado tráfego. Sendo assim, a métrica proposta quantifica quais aplicações apresentam melhor comportamento médio ao serem agrupadas sobre os mesmos recursos de rede.
- **Definição de um novo modelo para isolar tráfegos de rede em *data centers* de nuvem (Seção 4).** Amparado na métrica de entropia, IoNCloud permite melhor eficiência na utilização de recursos e redução na interferência entre os tráfegos das aplicações da nuvem. Além disso, o modelo independe da topologia de rede, de algoritmos de posicionamento de VMs ou de tipos de tráfego, tornando-se ortogonal à implementação de novas tecnologias nesse contexto.
- **Análise quantitativa do modelo proposto (Seção 5).** A eficiência de IoNCloud é analisada em cenários com base em um estudo de medição sobre tráfego em *data centers* [Benson et al. 2010]. Ademais, a abordagem é comparada com propostas anteriores, as quais não diferenciam aplicações ou tipos de tráfego.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta os principais aspectos relacionados às características de tráfegos em *data centers* de nuvem. A Seção 3 define uma métrica para calcular a *entropia* da rede. A Seção 4 descreve o modelo de formação de TOVCs e agrupamento de aplicações, enquanto a Seção 5 faz uma análise deste modelo e compara-o com propostas da literatura. Os principais trabalhos relacionados são discutidos na Seção 6 e as considerações finais, apresentadas na Seção 7.

2. Tráfego e Causas de Interferência na Rede da Nuvem

Diversos fluxos de naturezas diferentes coabitam uma rede de *data center* de nuvem. Em geral, esses fluxos seguem uma classificação bimodal, onde a primeira classe representa aplicações com grandes volumes de dados, enquanto a segunda representa aplicações orientadas à interação com o usuário [Kandula et al. 2009]. Entretanto, a divisão dos fluxos em apenas duas classes é simplista, visto que dentro de cada grande classe existe uma variação significativa quanto ao comportamento dos fluxos [Benson et al. 2010].

Com o uso de mecanismos de multiplexação de recursos, aplicações com diferentes requisitos compartilham os dispositivos de rede. Em consequência, a rede interna da nuvem opera de maneira imprevisível, uma vez que tráfegos de naturezas distintas podem interferir uns nos outros [Abts and Felderman 2012]. Recentemente, a literatura vem identificando problemas particulares nos *data centers* de nuvem. Especificamente, tais problemas estão relacionados às características topológicas das redes de grande porte utilizadas nessas infraestruturas [Ballani et al. 2011].

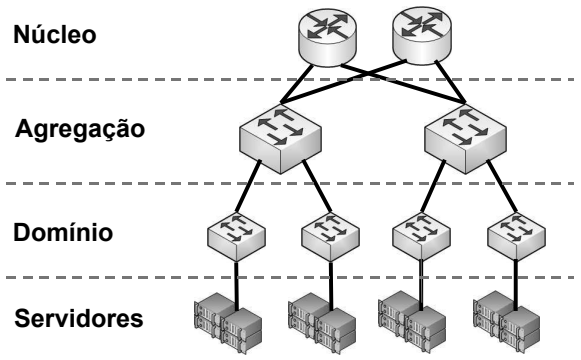


Figura 1. Topologia canônica de uma rede de *data center* de nuvem, adaptada de [Benson et al. 2010].

Conforme demostrado na Figura 1, a topologia dessas redes está organizada em forma de árvore, com os servidores dispostos nas folhas e os demais dispositivos organizados de forma hierárquica. Como os dispositivos estão fisicamente muito próximos, o atraso de propagação é muito pequeno. Com isso, a maioria do tráfego em um *data center* pode ser caracterizado por um modelo *on/off* cujos períodos são de curta duração. Relacionado a isso, medições em [Benson et al. 2010] apontam que a utilização da maioria dos enlaces é pequena em todos os níveis da árvore exceto na raiz.

No entanto, apesar da baixa utilização, quando uma aplicação envia um fluxo de duração relativamente maior, o número de RTTs na conexão é suficiente para expandir a janela de transmissão do TCP, ao contrário das conexões de vida curta. Logo, o restante do tráfego nesse enlace acaba sendo prejudicado ao competir com a conexão mais longa. Mais precisamente, os fluxos mais curtos acabam sendo terminados antecipadamente ou geram retransmissões em demasia, agravando a situação [Abts and Felderman 2012]. Em última instância, atrasos e interrupções na propagação dos dados acabam por intervir na computação das VMs, prejudicando o desempenho geral das aplicações [Ballani et al. 2011].

A imprevisibilidade do tráfego das aplicações acarreta o uso ineficiente dos recursos das redes de *data centers* de nuvem. Dessa forma, é útil a definição de uma métrica que expresse a real necessidade de isolamento dos recursos de rede das aplicações hospedadas nesses ambientes. A próxima seção propõe uma métrica que expressa essa necessidade por meio da quantificação da interferência entre tráfegos.

3. Quantificando a Entropia na Rede da Nuvem

Neste artigo, denominamos de *entropia* ($\mathcal{E}$) a variação em relação à média da demanda de determinado tráfego em uma rede de *data center* de nuvem. O objetivo desta seção é definir uma métrica, entropia, que quantifique a influência gerada por diversas instâncias de tráfego sobre a previsibilidade do tráfego agregado. Tais tráfegos são originados de aplicações distintas que, por uma circunstância do estado atual do posicionamento de VMs no *data center*, compartilham e competem por recursos de uma parte da infraestrutura. Considera-se que a interferência entre tráfegos é dada pela largura de banda excedente em relação à média do tráfego agregado. Sendo assim, uma contribuição chave deste artigo é a análise da entropia sobre o tráfego agregado, de forma a expressar com precisão a necessidade de isolamento das aplicações hospedadas na nuvem.

Conceitualmente, verifica-se qual o comportamento médio da demanda de tráfego resultante, ao serem agrupadas diversas aplicações sobre os mesmos recursos de rede. O cálculo do valor de $\mathcal{E}$ é composto por três etapas: geração do tráfego agregado com base no histórico de demandas; discretização do tempo em N amostras para obtenção do desvio médio do tráfego agregado; e realização de procedimento de convolução, de forma a capturar variações temporais A formalização de cada etapa é descrita a seguir.

Supondo Π como o conjunto de todos os tráfegos compartilhando os mesmos recursos de rede no data center, onde $\pi \in \Pi$ representa o tráfego entre um par de VMs quaisquer. Sendo assim, o histórico da demanda por largura de banda de um tráfego qualquer $\pi \in \Pi$ em determinado intervalo $\Delta\tau = [t_i, t_f]$ é dado pela função $\mathfrak{h}_\pi(t) : t \in \Delta\tau$. Com isso, a demanda agregada $\mathcal{H}(t)$ para um grupo de tráfegos compartilhando determinado enlace é dada pela soma de toda a demanda ao longo do tempo $\mathcal{H}(.) = \sum_\pi^\Pi \mathfrak{h}_\pi(.)$.

Cada histórico de demanda $\mathfrak{h}_\pi(.)$ expressa a largura de banda consumida em um instante t ao longo do intervalo $\Delta\tau$. Pragmaticamente, isso implica a coleta de N amostras com intervalo de amostragem δt_A constante e pequeno o suficiente para expressar a precisão desejada[1]. Essa discretização do espaço amostral resulta em $|\Pi|$ vetores na forma $h_\pi[.] = \{h_\pi[1],\ h_\pi[2],\ \ldots,\ h_\pi[N]\}$. Logo, o vetor $H[.] = \sum_\pi^\Pi h_\pi[.]$ expressa a demanda agregada por largura de banda ao longo do intervalo de tempo discretizado $\Delta\tau$. A seguir, são calculados a média M e o desvio médio $D = \sum_{i=1}^{N} |H[i] - M|/N$ das amostras, expressando a variação do valor (agregado) ao longo do tempo.

Além de calcular a variação na demanda agregada, é preciso considerar que muitas vezes aplicações na nuvem não possuem um comportamento constante. Contudo, cada demanda de forma individual segue uma distribuição probabilística bem definida, indicando que os padrões de tráfego repetem-se com certa frequência [Benson et al. 2010]. Sendo assim, utiliza-se um procedimento de convolução com o intuito de agregar à métrica possíveis variações temporais no comportamento da demanda.

[1]Esse histórico de amostras pode ser obtido por meio de ferramentas de monitoramento como, por exemplo, o Amazon CloudWatch: `http://aws.amazon.com/cloudwatch`

O procedimento de convolução consiste em mover um dos tráfegos sobre todos os outros, de forma a capturar comportamentos deslocados ao longo do tempo. Para isso, inicialmente, todos os tráfegos são concatenados para gerar uma situação onde nenhum deles se intercepta. Essa concatenação resulta em uma janela de tempo de tamanho Δt_W. A seguir, para cada $K = |\Pi| - 1$ iterações do procedimento, seleciona-se o último tráfego e desloca-o ao longo da janela de tempo Δt_W. A cada etapa de deslocamento, o cálculo do desvio médio é refeito e a demanda do tráfego selecionado é adiantada em δt_C unidades, resultando $\Delta t_W / \delta t_C$ operações. Ao final de cada etapa, o tráfego selecionado é descartado e Δt_W é atualizado. Com isso, seja Δt_{W_i} o tamanho da janela na iteração $i \in K$ e $D_{i,j}$ o desvio médio calculado na etapa de deslocamento j da iteração $i \in K$, a entropia é dada pela *média dos desvios médios* ao longo das K iterações do procedimento de convolução, conforme a seguinte equação:

$$\mathcal{E} = \frac{\sum_{i=1}^{K} \sum_{j=1}^{\Delta t_{W_i}/\delta t_C} D_{i,j} / (\Delta t_{W_i} / \delta t_C)}{K}.$$

Essa métrica pode ser utilizada pelos sistemas de alocação e gerenciamento dos recursos da nuvem de forma a otimizar a reserva e o isolamento dos mesmos. Por esse motivo, ela forma o núcleo do modelo de alocação de recursos em *data centers* de nuvem proposto neste artigo, o qual será definido a seguir.

4. IoNCloud

Conforme discutido anteriormente, o modelo proposto neste artigo visa prover isolamento de tráfego às aplicações de *data center* sem sacrificar a multiplexação de recursos da nuvem. Para alcançar esse objetivo, IoNCloud (*Isolation of Networks in the Cloud*) agrupa aplicações cujos tráfegos possuem pouca ou nenhuma interferência entre si, reduzindo a imprevisibilidade da nuvem. IoNCloud inspira-se em processos naturais (Figura 2), onde VMs (átomos) com alta interferência entre tráfegos aumentam a imprevisibilidade do ambiente, enquanto VMs com baixa interferência entre tráfegos são consequência de um ambiente estável, caracterizado por uma maior previsibilidade. Durante esse processo [Figura 2(a)], aplicações com grande entropia, ou seja, cujos tráfegos interfeririam um no outro, são repelidas, enquanto as demais são atraídas. O sistema torna-se estável [Figura 2(b)] quando a menor entropia global é atingida.

Mais especificamente, IoNCloud agrupa aplicações com baixa interferência entre si em um mesmo *cluster virtual orientado a tráfego* (definido a seguir, na Seção 4.1). A abordagem descrita neste trabalho (Seção 4.2) é complementar ao processo de alocação de recursos na infraestrutura da nuvem. Esse último visa instanciar VMs e/ou topologias de rede virtuais em recursos físicos, assumindo (i) que todas as aplicações devem ser alocadas no mesmo substrato; ou (ii) que estas devem ter seus recursos completamente isolados. Em contraste, este artigo apresenta o problema de alocação em nível de aplicações ao invés de VMs ou redes virtuais.

Sendo assim, IoNCloud torna-se uma etapa predecessora utilizada para definir quais aplicações deverão ser alocadas em quais *clusters* virtuais. A abordagem independe do algoritmo posterior utilizado para realizar o posicionamento de VMs dentro de um *data center* [Ballani et al. 2011, Bodík et al. 2012] ou o mapeamento de redes virtuais em substratos de rede [Chowdhury et al. 2009, Alkmim et al. 2011].

(a) VMs dispostas de forma desordenada, compartilhando os recursos de um mesmo ambiente sem isolamento. Tráfegos de diferentes naturezas podem causar comportamento imprevisível.

(b) Agrupamentos de VMs que geram menor entropia (desvio em relação ao comportamento médio) são preferidos. Logo, VMs são atraídas ou repelidas de um mesmo *cluster* virtual de forma a fazer o sistema atingir menor entropia global.

Figura 2. Desordem na rede da nuvem, uma analogia aos processos naturais.

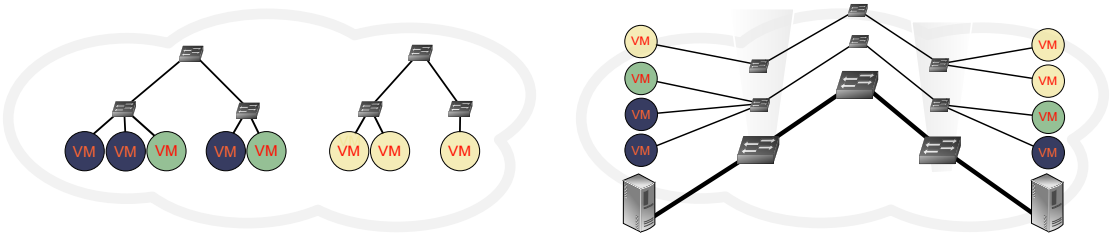
4.1. *Clusters* Virtuais Orientados a Tráfego

Um *Cluster* Virtual (*Virtual Cluster* – VC) consiste em um conjunto de VMs interligadas através de uma rede virtual, conforme definido em [Xie et al. 2012, Ballani et al. 2011]. Nesse modelo de abstração, cada rede virtual – e, consequentemente, cada VC – possui uma parcela de recursos de rede reservados e completamente isolados. Com isso, as VMs de um VC não podem interagir com VMs de qualquer outro VC[2].

Similarmente aos *clusters* virtuais da literatura, um *Cluster* Virtual Orientado a Tráfego (*Traffic-Oriented Virtual Cluster* – TOVC) é formado por uma topologia de rede virtual genérica (ou seja, não necessariamente limitada a topologias de árvore) conectando um número arbitrário de VMs. A topologia desses *clusters* virtuais é descrita por um grafo, onde os nós virtuais representam máquinas ou *switches* virtuais e as arestas representam enlaces virtuais. Contudo, apesar dos TOVCs de IoNCloud serem fundamentados em propostas anteriores, eles se diferenciam dos tradicionais por uma série de razões, conforme segue:

1. VMs pertencentes a diferentes aplicações e/ou locatários podem compartilhar um mesmo *cluster*. A principal motivação por trás desse compartilhamento é prover uma melhor granularidade na utilização dos recursos de rede.
2. A reserva dos recursos de rede para um determinado TOVC deve garantir a *média agregada dos padrões de tráfego* das VMs das aplicações alocadas. Tal valor pode ser obtido através do histórico da demanda de tráfego das VMs. Essa medida busca prover garantias de tráfego que evitem o desperdício de recursos.
3. O tamanho de cada TOVC (número de aplicações e quantidade de recursos virtuais) varia de acordo com a entropia (ou seja, conforme o grau de interferência entre os tráfego das aplicações). Em situações limite, o modelo reduz-se às abordagens atuais, onde (i) todas as aplicações são alocadas em um mesmo TOVC ou (ii) cada aplicação é alocada a um TOVC distinto. No primeiro caso não existe isolamento, acarretando imprevisibilidade de desempenho. Em contrapartida, o segundo caso reduz a interferência entre tráfegos ao custo de uma menor utilização de recursos.

[2]VCs protegem apenas o tráfego local, inter-VMs. Existem mecanismos para transpassar o isolamento provido pelos VCs, por exemplo, com o uso de IPs globais (utilizando a Internet). No entanto, tais mecanismos afetam o tráfego externo à rede do *data center* e estão fora do escopo deste trabalho.



(a) Dois *clusters* virtuais alocando VMs de aplicações distintas.

(b) Possível mapeamento de recursos para os *clusters* virtuais representados na figura ao lado.

Figura 3. *Clusters* virtuais do modelo IoNCloud.

A Figura 3(a) apresenta dois possíveis *clusters* virtuais no modelo IoNCloud. No primeiro *cluster* (esquerda), aplicações distintas compartilham os recursos de rede reservados, enquanto no segundo *cluster* (direita) uma única aplicação utiliza os recursos. A Figura 3(b) demonstra um exemplo simplificado de mapeamento para os recursos virtuais dos *clusters* sobre os dispositivos da rede física. Apesar dos recursos serem alocados sobre os mesmos dispositivos, não existe interferência entre tráfegos de VMs pertencentes a TOVCs distintos.

4.2. Formalização do modelo

Esta subseção formaliza o modelo de agrupamento ótimo de aplicações em TOVCs. Para facilitar a compreensão, a Tabela 1 resume todos os símbolos utilizados na formulação.

O ambiente em consideração é composto por um conjunto de aplicações $a \in \mathcal{A}$ e suas respectivas máquinas virtuais $vm \in \mathcal{VM}_a : a \in \mathcal{A}$. Além disso, considera-se que cada par de VMs $vm_1, vm_2 \in \mathcal{VM}_a : a \in \mathcal{A}$ possui uma necessidade de comunicação expressa por $c(vm_1, vm_2) \in \mathbb{B}$, representando um tráfego $\pi \in \Pi_a : a \in \mathcal{A}$. Caso duas VMs precisem se comunicar [$c(vm_1, vm_2) = 1$], a largura de banda média necessária será expressa por $b(vm_1, vm_2) \in \mathbb{R}^+$ (ou $b(\pi)$, onde π é o tráfego agregado entre as mesmas). Nesse contexto, o problema de distribuição de aplicações em TOVCs consiste em um problema de agrupamento ótimo de aplicações $\mathcal{A}$ sobre um conjunto de TOVCs $\mathcal{Z}$ que possua entropia $\mathcal{E}$ mínima, onde $\mathcal{E}(\Pi)$ denota a entropia de um conjunto de tráfegos.

Objetivo. Nesse modelo, a variável $\alpha_{a,z} \in \mathbb{B}$ indica se a aplicação $a \in \mathcal{A}$ será alocada ao TOVC $z \in \mathcal{Z}$. Dessa forma, a seguinte função expressa o objetivo de otimização da formulação:

$$\min \sum_{z}^{Z} \sum_{a_1,a_2}^{\mathcal{A}:a_1 \neq a_2} \alpha_{a_1,z} * \alpha_{a_2,z} * \mathcal{E}(\Pi_{a_1} \cup \Pi_{a_2}). \quad (1)$$

A primeira parte da Equação (1) ($\sum_{z}^{Z} \sum_{a_1,a_2}^{\mathcal{A}:a_1 \neq a_2} \alpha_{a_1,z} * \alpha_{a_2,z}$) indica se as aplicações estão alocadas no mesmo TOVC, ou seja, a expressão resulta em 1 se, e somente se, ambas as aplicações são alocadas no mesmo *cluster*. Tal processo é executado para todas as distribuições possíveis de pares de aplicações em TOVCs. A segunda parte da equação [$\mathcal{E}(\Pi_{a_1} \cup \Pi_{a_2})$] calcula a entropia dos tráfegos. Observe que, em virtude da multiplicação entre os dois termos, a entropia só precisa ser calculada para aplicações dentro de um mesmo *cluster*.

Tabela 1. Lista de símbolos utilizados na formalização do modelo.

Símbolo	Descrição
$\mathcal{A}$	Conjunto de aplicações.
$\mathcal{VM}_a$	Conjunto de máquinas virtuais de uma aplicação $a \in \mathcal{A}$.
Π, Π_a	Respectivamente, conjunto de tráfegos quaisquer (Π) e conjunto de tráfegos pertencentes à uma aplicação $a \in \mathcal{A}$ (Π_a).
$\mathcal{Z}$	Conjunto de TOVCs.
$\mathcal{E}(\Pi) \in \mathbb{R}^+$	Entropia entre tráfegos pertencentes a um conjunto Π.
$c(vm_1, vm_2) \in \mathbb{B}$	Entrada expressando a necessidade de comunicação entre duas VMs quaisquer, onde $vm_1, vm_2 \in \mathcal{VM}_a : a \in \mathcal{A}$.
$b(vm_1, vm_2) \in \mathbb{R}^+$	Entrada expressando a demanda de banda média necessária para comunicação entre duas VMs, onde $vm_1, vm_2 \in \mathcal{VM}_a : a \in \mathcal{A}$.
$b(\pi) \in \mathbb{R}^+$	Entrada expressando a demanda de banda média necessária para o tráfego $\pi \in \Pi_a : a \in \mathcal{A}$.
$\lambda \in \mathbb{R}^+$	Constante definindo um valor máximo para a entropia de um TOVC.
$\alpha_{a,z} \in \mathbb{B}$	Variável binária indicando se a aplicação $a \in \mathcal{A}$ será alocada ao TOVC $z \in \mathcal{Z}$.
$\delta_{vm,z} \in \mathbb{B}$	Variável binária indicando se a máquina virtual $vm \in \mathcal{VM}_a : a \in \mathcal{A}$ será alocada ao TOVC $z \in \mathcal{Z}$.
$\gamma^{e^v,\pi} \in \mathbb{B}$	Variável binária indicando se determinado tráfego π está utilizando um enlace virtual e^v.
$b(e^v) \in \mathbb{R}^+$	Variável indicando a demanda que deve ser reservada para determinado enlace virtual e^v.

Restrições. O processo anterior precisa satisfazer uma série de restrições de forma a tornar o problema válido.

- VMs de uma mesma aplicação precisam ser alocadas no mesmo TOVC, ou seja, se duas VMs quaisquer necessitam se comunicar, então elas devem estar no mesmo *cluster* $z \in \mathcal{Z}$. Portanto, seja $\delta_{vm,z} \in \mathbb{B}$ uma variável binária representando a pertinência da máquina virtual $vm \in \mathcal{VM}_a : a \in \mathcal{A}$ ao *cluster* $z \in \mathcal{Z}$, a Equação (2) precisa ser satisfeita:

$$\delta_{vm,z} = \alpha_{a,z} \qquad \forall vm \in \mathcal{VM}_a, \forall a \in \mathcal{A}, z \in \mathcal{Z}. \tag{2}$$

- Aplicações devem ser alocadas a, exatamente, um TOVC. Dessa forma, não pode haver mais do que uma instância de $\alpha_{a,z}$ com valor 1 *para a mesma aplicação* $a \in \mathcal{A}$ [Eq. (3)]. No entanto, a recíproca não é verdadeira, uma vez que pode haver diversas aplicações em um mesmo *cluster* $z \in \mathcal{Z}$ [Eq. (4)].

$$\sum_{z}^{\mathcal{Z}} \alpha_{a,z} = 1 \qquad \forall a \in \mathcal{A}; \tag{3}$$

$$\sum_{a}^{\mathcal{A}} \alpha_{a,z} \geq 1 \qquad \forall z \in \mathcal{Z}. \tag{4}$$

- A entropia entre dois tráfegos de VMs em um mesmo *cluster* $z \in \mathcal{Z}$ que não necessitam se comunicar $[c(vm_1, vm_2) = 0]$ não deve ser maior do que um limite λ previamente especificado. A constante $\lambda \in \mathbb{R}^+$, nesse caso, é um coeficiente não negativo definido pelo provedor de infraestrutura de modo a evitar degradação significativa no desempenho dessas máquinas ao longo do seu período de execução [Eq. (5)].

$$\sum_{a_1,a_2}^{\mathcal{A}} \alpha_{a_1,z} * \alpha_{a_2,z} * [1 - c(vm_1, vm_2)] * \mathcal{E}(\Pi_{a_1} \cup \Pi_{a_2}) \leq \lambda \qquad \forall z \in \mathcal{Z}. \quad (5)$$

- Por fim, a função objetivo [Eq. (1)] busca obter a menor entropia possível dentro dos TOVCs, o que levaria no extremo à criação de um TOVC por aplicação. Esse efeito é indesejável, uma vez que acarretaria em baixa utilização de recursos. Portanto, são impostos dois limites, inferior ($\mathcal{Z}^{inf}$) e superior ($\mathcal{Z}^{sup}$), ao número de TOVCs necessários para obter uma solução. Na prática, os valores de $\mathcal{Z}^{inf}$ e $\mathcal{Z}^{sup}$ podem ser configurados para os valores limites do modelo (conforme descritos na subseção anterior), ou seja, o valor unitário representa um cenário sem isolamento, enquanto o número máximo de aplicações ($|\mathcal{A}|$) representa isolamento completo. Dessa forma, a primeira tentativa de resolução do problema parte de $\mathcal{Z}^{inf}$ TOVCs. Caso alguma das restrições anteriores não seja satisfeita, o número de TOVCs é incrementado iterativamente até o limite superior $\mathcal{Z}^{sup}$.

Após a distribuição das aplicações em *clusters* virtuais, devem ser utilizados algoritmos de alocação de recursos bem estabelecidos para (i) o posicionamento de VMs [Ballani et al. 2011, Bodík et al. 2012]; e (ii) o mapeamento de redes virtuais em substratos de rede física [Chowdhury et al. 2009, Alkmim et al. 2011]. A única restrição a esse processo deve ser dada na capacidade de transmissão dos enlaces virtuais. Cada enlace virtual e^v de um cluster $z \in \mathcal{Z}$ deve possuir uma largura de banda $b(e^v)$ tal que acomode a soma das demandas médias de todos os seus tráfegos. Dessa forma, seja $\gamma^{e^v,\pi} \in \mathbb{B}$ a variável que indica se determinado tráfego utiliza o referido enlace virtual. Nesse caso, a seguinte restrição deve ser satisfeita:

$$b(e^v) = \sum_{a}^{\mathcal{A} \,:\, \alpha_{a,z}=1} \sum_{\pi}^{\Pi_a} b(\pi) * \gamma^{e^v,\pi} \qquad \forall e^v \in z, \forall z \in \mathcal{Z}. \quad (6)$$

5. Experimentos

A principal contribuição deste artigo é IoNCloud, um modelo amparado nas características de tráfego para oferecer isolamento de recursos às aplicações. Esta seção avalia a abordagem proposta através da análise da *(i)* interferência entre tráfegos; *(ii)* fração de utilização dos recursos disponíveis; e *(iii)* quantidade de banda reservada para cada aplicação e/ou grupo de aplicações. Todos os resultados apresentados utilizam a média de 30 repetições, com intervalo de confiança em 95%.

5.1. Cenário de avaliação

Os experimentos realizados seguem um modelo simplificado, baseado no pior caso, onde todas as aplicações de um dado *cluster* virtual compartilham o mesmo enlace virtual.

Seguindo essa premissa, a entropia total do sistema é calculada por meio da análise de um par de VMs de cada aplicação. Ademais, conforme definido na seção anterior, a largura de banda reservada em um enlace virtual é dada pela demanda média do tráfego agregado. Com isso, a interferência é calculada como toda a demanda que ultrapassa essa reserva. Essa situação pode ocorrer mesmo que exista somente uma aplicação por *cluster* virtual.

Para obter a carga de trabalho, desenvolvemos um gerador de tráfegos sintéticos, os quais simulam o comportamento de aplicações típicas de um *data center* (p.ex., HTTP, MapReduce, LDAP etc). Esses tráfegos seguem distribuições com parâmetros conhecidos, conforme definidos por Benson *et. al.* (2010). Ademais, foram utilizadas 12 aplicações simultâneas, igualmente divididas entre 3 tipos de tráfego e agrupadas de acordo com a entropia do tráfego agregado do grupo.

Por fim, o agrupamento de aplicações é restrito às combinações que geram o mesmo número de aplicações por *cluster* virtual. Esse procedimento é necessário pois o cálculo de todas as possíveis combinações torna o processo computacionalmente custoso. Embora tal simplificação acarrete um resultado sub-ótimo, essa primeira análise demonstra que IoNCloud provê isolamento competitivo em relação a abordagens da literatura, a um custo menor na subutilização de recursos.

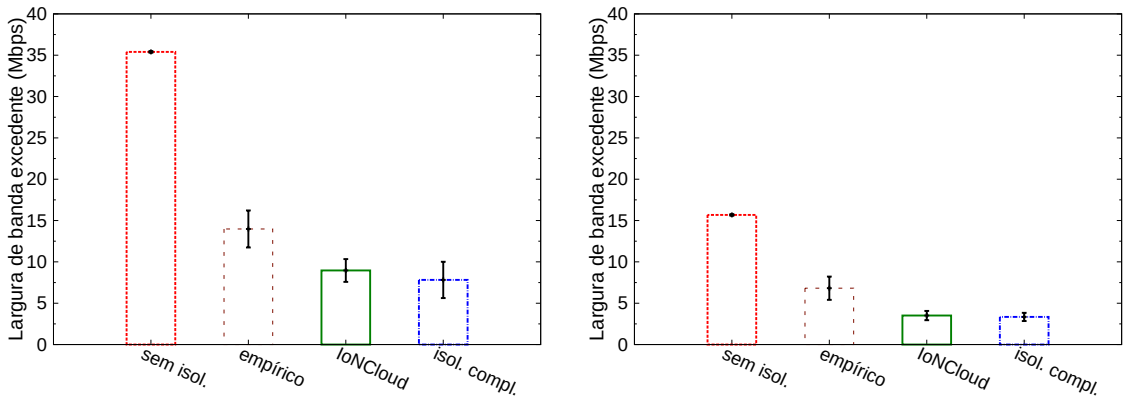
5.2. Método de comparação

Em arquiteturas atuais de nuvem pública, sem isolamento de recursos, todos os locatários compartilham os mesmos recursos físicos e estão sujeitos a interferência entre tráfegos. Em resposta, propostas recentes utilizam um *cluster* virtual (ou rede virtual) por locatário [Ballani et al. 2011, Xie et al. 2012] para prover isolamento e garantias de largura banda. Essas abordagens foram denominadas de *sem isolamento*, *isolamento completo*, respectivamente. Em contraste, IoNCloud oferece isolamento compartilhado, permitindo a coexistência das aplicações em um mesmo *cluster* virtual.

A fim de reduzir a interferência, IoNCloud utiliza a entropia (ou seja, desvio em relação à média) como métrica para definir quais aplicações serão agrupadas ou isoladas. Uma variação dessa abordagem consiste em agrupar aplicações em recursos físicos de acordo com métricas variadas, as quais ignoram padrões de tráfego (p.ex, número máximo de VMs compartilhando o mesmo enlace físico [Webb et al. 2011]). Para representar esse tipo de abordagem, aplicações são selecionadas ao acaso para serem agrupadas em *clusters* virtuais. Nos experimentos, esse procedimento foi denominado de *empírico*.

5.3. Resultados

A abordagem de isolamento de IoNCloud é competitiva com propostas de isolamento completo. A Figura 4 demonstra a demanda de tráfego máxima e média excedente à reservada. Quando comparado ao caso sem isolamento, IoNCloud consegue reduzir a interferência máxima em aproximadamente 75% (78% na interferência média). Além disso, é possível detectar um ganho no agrupamento baseado na entropia, uma vez que o procedimento empírico comporta-se 36% pior do que IoNCloud para a interferência máxima (49% para a interferência média). Em contraste, quando comparado com o isolamento completo (1 aplicação por TOVC) a diferença para IoNCloud é de apenas 12% na interferência máxima (aproximadamente 5% na interferência média). Esse valor pode tornar-se desprezível quando comparado ao ganho na utilização de recursos, conforme demostrado na próxima avaliação.



(a) Interferência máxima. (b) Interferência média.

Figura 4. Interferência entre tráfegos, medida pela banda excedente à reservada.

O agrupamento por menor entropia ($\mathcal{E}$) reduz a subutilização absoluta de recursos. A Figura 5 demostra a média de recursos ociosos para cada estratégia de isolamento. Como esperado, a subutilização é menor quando não há isolamento. Entretanto, quando comparado com as abordagens que provêm isolamento, a subutilização é menor em IoNCloud, chegando a representar apenas 30% do valor do isolamento completo. Tal fenômeno ocorre porque o agrupamento de tráfegos com menor $\mathcal{E}$ resulta na menor dispersão em relação à média (vide métrica definida na Seção 3). Logo, a métrica influencia o modelo ao agrupar aplicações cujo tráfego agregado possui poucos picos (valores acima da média – interferência) e vales (valores abaixo da média – subutilização).

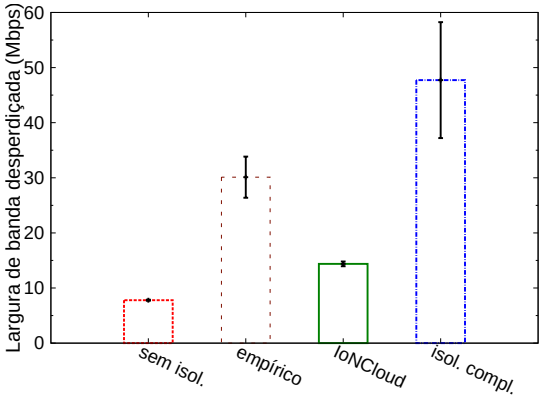
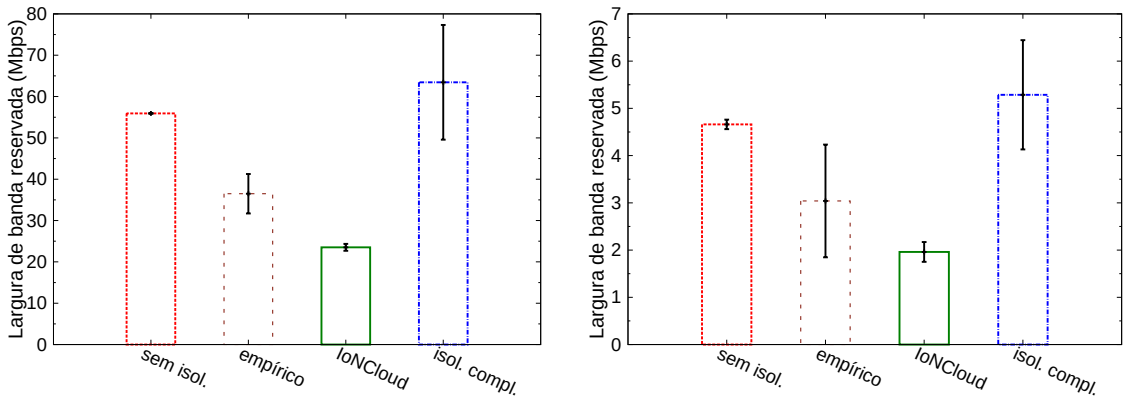


Figura 5. Subutilização média de recursos, medida pela quantidade de recursos ociosos (abaixo do reservado).

IoNCloud acarreta menor demanda por reserva de recursos. A Figura 6 indica a largura de banda total reservada no *data center* da nuvem [Figura. 6(a)] e a largura de banda média por aplicação [Figura 6(b)], ou seja, o total reservado para o *cluster* virtual dividido pelo número de aplicações alocadas no *cluster* virtual. Em ambos os casos, IoNCloud reserva uma quantia menor de recursos. Intuitivamente, seria possível inferir que aplicações em IoNCloud são privadas de largura de banda. No entanto, conforme demonstrado nos experimentos, esses recursos acabam tendo uso ineficiente, seja pela interferência dada pela falta de isolamento [Figura 4], ou pelo desperdício [Figura 5], como no caso do isolamento completo. Logo, potenciais benefícios obtidos pela reserva de uma quantidade maior de recursos são apenas aparentes.



(a) Banda total reservada. (b) Banda reservada por aplicação.

Figura 6. Largura de banda reservada na nuvem.

6. Trabalhos Relacionados

Diversas propostas presentes na literatura buscam oferecer garantias de desempenho às aplicações dos locatários da nuvem. Esta seção discute tanto trabalhos que incluem algum nível de controle de tráfego, quanto trabalhos que efetivamente provêm isolamento de recursos de rede. Alguns mecanismos fundamentais como a limitação da banda de um fluxo (*rate limiting*) [Raghavan et al. 2007] e o posicionamento de VMs (*VM placement*) [Jiang et al. 2012] permeiam a maioria das estratégias de reserva e/ou isolamento presentes nessas propostas, dando sustentação a modelos mais precisos e efetivos.

Virtualização de redes. Algumas estratégias recentes fazem uso da virtualização de redes nos *data centers* de nuvem para mitigar a interferência entre tráfegos. Guo *et. al.* (2010), Ballani *et. al.* (2011) e Rodrigues *et. al.* (2011) provêm garantias de banda fixas às comunicações entre pares ou grupos de VMs pertencentes a um mesmo locatário e isoladas em uma rede virtual. Xie *et. al.* (2012), por sua vez, exploram a possibilidade das demandas de rede das aplicações variarem ao longo do tempo e propõem um mecanismo de reserva de banda também variante no tempo às VMs isoladas nas redes virtuais. Apesar de oferecerem certas garantias de desempenho, tais abordagens consideram os requisitos de cada locatário individualmente, normalmente resultando em baixa utilização de recursos. Em contraste, Marcon *et. al.* (2012) propõem uma estratégia de alocação de recursos baseada no agrupamento de locatários mutuamente confiáveis em domínios virtuais isolados. Tal trabalho é ortogonal ao IoNCloud, uma vez que os autores utilizam confiança mútua como métrica para alocação.

Compartilhamento proporcional. Outro grande grupo de propostas envolve o compartilhamento proporcional dos recursos de rede entre as VMs dos locatários. Lam *et. al.* (2012) adotam a atribuição de pesos específicos para cada locatário da nuvem para, então, alocar os recursos de enlaces congestionados com base nesses pesos. Por sua vez, Popa *et. al.* (2012) propõem mecanismos eficientes para a utilização proporcional dos recursos de rede na presença de garantias mínimas de banda. Essas propostas, apesar de facilitarem a previsibilidade de desempenho das aplicações, possuem alto custo de gerenciamento de recursos de rede.

Limitação no uso de recursos. Algumas alternativas envolvem a limitação do número de VMs e locatários que compartilham os recursos de um determinado enlace ou porção da rede. A Amazon (2012), por exemplo, oferece serviços que reservam uma porção dos recursos da rede para a comunicação exclusiva entre as VMs de um locatário. De forma mais relaxada, a estratégia de Webb *et. al.* (2011) permite que os locatários especifiquem o número de tarefas (VMs) que compartilham um determinado enlace. Esse grupo de estratégias normalmente aumenta os custos dos locatários e diminui a eficiência na utilização dos recursos dos *data centers*, uma vez que o tráfego da maioria das aplicações de nuvem é de natureza intermitente.

Escalonamento de tarefas. Lago *et. al.* (2012) propõem o escalonamento de tarefas ciente do consumo de energia nas máquinas virtuais da nuvem. Apesar dos benefícios relacionados ao controle de energia, a proposta não trata da interferência de desempenho causada pela heterogeneidade do tráfego presente na rede interna da nuvem.

7. Considerações Finais

Embora a computação em nuvem traga diversas vantagens, o compartilhamento dos recursos de rede entre um elevado número de aplicações, potencialmente com requisitos de largura de banda distintos, acarreta uma variabilidade significativa no desempenho de rede das aplicações. Consequentemente, o tempo de execução dessas aplicações é negativamente afetado e o custo dos locatários torna-se imprevisível.

Neste artigo, foi proposto um modelo de reserva e isolamento de recursos de rede da nuvem (IoNCloud). Nesse modelo, instâncias de aplicações são agrupadas em *clusters* virtuais orientados a tráfego (TOVCs), com o objetivo de isolar aplicações cujo compartilhamento dos recursos de rede ocasiona imprevisibilidade de desempenho. Resultados demonstram que IoNCloud provê dois benefícios chave. Por um lado, locatários obtém melhor previsibilidade de desempenho, uma vez que IoNCloud é competitivo em relação a estratégias de isolamento completo. Por outro, provedores de infraestrutura obtém melhor multiplexação de recursos, visto que IoNCloud apresenta subutilização de recursos 70% menor do que o isolamento completo.

Nos trabalhos em andamento, estamos desenvolvendo heurísticas para resolver o problema de forma eficiente, viabilizando a aplicação da proposta em nuvens de larga escala. Em trabalhos futuros, vislumbramos novos algoritmos de posicionamento de VMs que possam explorar melhor as características dos TOVCs. Também estuda-se a possibilidade de agregar maior dinamicidade ao ambiente, permitindo que aplicações e *clusters* sejam migrados para atender novas demandas de tráfego.

Referências

[Abts and Felderman 2012] Abts, D. and Felderman, B. (2012). A guided tour of data-center networking. *Commun. ACM*, 55(6):44–51.

[Alkmim et al. 2011] Alkmim, G., Batista, D. M., and da Fonseca, N. L. S. (2011). Mapeamento de redes virtuais em substratos de rede. In *Anais do XXIX SBRC*.

[Amazon 2012] Amazon (2012). High performance computing (hpc) on aws. Disponível em `https://aws.amazon.com/hpc-applications/`.

[Armbrust et al. 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

[Ballani et al. 2011] Ballani, H., Costa, P., Karagiannis, T., and Rowstron, A. (2011). Towards predictable datacenter networks. In *Proceedings of the ACM SIGCOMM '11 conference*.

[Benson et al. 2010] Benson, T., Akella, A., and Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM IMC conference*.

[Bodík et al. 2012] Bodík, P., Menache, I., Chowdhury, M., Mani, P., Maltz, D. A., and Stoica, I. (2012). Surviving failures in bandwidth-constrained datacenters. In *Proceedings of the ACM SIGCOMM '12 conference*.

[Chowdhury et al. 2009] Chowdhury, N., Rahman, M., and Boutaba, R. (2009). Virtual network embedding with coordinated node and link mapping. In *Proceedings of the 28th IEEE INFOCOM conference*.

[Genez et al. 2012] Genez, T. A. L., Bittencourt, L. F., and Madeira, E. R. M. (2012). Discretização do tempo na utilização de programação linear para o problema de escalonamento de workflows em múltiplos provedores de nuvem. In *Anais do XXX SBRC*.

[Guo et al. 2010] 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 the 6th ACM Co-NEXT conference*.

[Jiang et al. 2012] Jiang, J., Lan, T., Ha, S., Chen, M., and Chiang, M. (2012). Joint vm placement and routing for data center traffic engineering. In *Proceedings of the 31st IEEE INFOCOM conference*.

[Kandula et al. 2009] Kandula, S., Sengupta, S., Greenberg, A., Patel, P., and Chaiken, R. (2009). The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM IMC conference*.

[Lago et al. 2012] Lago, D. G., Madeira, E. R. M., and Bittencourt, L. F. (2012). Escalonamento com prioridade na alocação ciente de energia de máquinas virtuais em nuvens. In *Anais do XXX SBRC*.

[Lam et al. 2012] Lam, V. T., Radhakrishnan, S., Pan, R., Vahdat, A., and Varghese, G. (2012). Netshare and stochastic netshare: predictable bandwidth allocation for data centers. *ACM SIGCOMM Comput. Commun. Rev.*, 42(3):5–11.

[Marcon et al. 2012] Marcon, D. S., Neves, M. C., Oliveira, R. R., Buriol, L. S., Gaspary, L. P., and Barcellos, M. P. (2012). Mitigando ataques de egoísmo e negação de serviço em nuvens via agrupamento de aplicações. In *Anais do XII SBSeg*.

[Popa et al. 2012] Popa, L., Kumar, G., Chowdhury, M., Krishnamurthy, A., Ratnasamy, S., and Stoica, I. (2012). Faircloud: sharing the network in cloud computing. In *Proceedings of the ACM SIGCOMM '12 conference*.

[Raghavan et al. 2007] Raghavan, B., Vishwanath, K., Ramabhadran, S., Yocum, K., and Snoeren, A. C. (2007). Cloud control with distributed rate limiting. In *Proceedings of the ACM SIGCOMM '07 conference*.

[Rodrigues et al. 2011] 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 the 3rd USENIX WIOV workshop*.

[Webb et al. 2011] Webb, K. C., Snoeren, A. C., and Yocum, K. (2011). Topology switching for data center networks. In *Proceedings of the 11th USENIX Hot-ICE conference*.

[Xie et al. 2012] 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 the ACM SIGCOMM '12 conference*.

Avaliando o Aprisionamento entre Várias Plataformas de Computação em Nuvem

Arthur E. C. da Silva Souza[1], José A. Medeiros de Lima[1], Renato Gondim[1], Thomás Diniz[2], Nélio Cacho[1], Frederico Lopes[2] , Thais Batista[1]

[1]Departamento de Informática e Matemática Aplicada – Universidade Federal do Rio Grande do Norte (UFRN)
Natal – RN – Brasil

arthurcassio@yahoo.com.br, .j.alex.medeiros@gmail.com, renato@dimap.ufrn.br, neliocacho@dimap.ufrn.br, thaisbatista@gmail.com

[2]Escola de Ciências e Tecnologia – Universidade Federal do Rio Grande do Norte (UFRN)
Natal – RN – Brasil

thomasfdsdiniz@gmail.com, fred.lopes@gmail.com

Abstract. *Despite cloud lock-in being one of the main drawbacks of cloud computing, there are no studies that systematically evaluate the compatibility among different cloud platforms. The goal of this paper is twofold: (i) to propose a generic process based on compatibility tests to evaluate the lock-in among cloud platforms; (ii) to apply the proposed process to evaluate the compatibility among a public platform (Amazon EC2) and three private platforms (OpenStack, Eucalyptus, and OpenNebula). This paper shows a compatibility evaluation and investigates if the existing compatibility is enough to avoid cloud lock-in.*

Resumo. *Apesar do cloud lock-in (aprisionamento) ser um dos principais problemas da computação em nuvem, não há estudos que avaliem sistematicamente a compatibilidade entre diferentes plataformas de nuvem. Esse artigo tem dois objetivos principais: (i) propor um processo genérico para avaliação do aprisionamento entre plataformas de nuvem através de uma abordagem de teste de compatibilidade; (ii) aplicar o processo proposto para avaliar o aprisionamento entre uma plataforma pública (Amazon EC2) e três plataformas privadas (OpenStack, Eucalyptus e OpenNebula). Esse trabalho mostra uma avaliação da compatibilidade e investiga se a compatibilidade existente é suficiente para evitar o problema de aprisionamento.*

1. Introdução

Computação em Nuvem surgiu como um modelo de computação distribuída onde recursos computacionais (*hardware*, armazenamento, plataformas de desenvolvimento, e comunicação) são virtualizados e disponibilizados como serviços (re)configuráveis, pagos pela sua utilização, sendo suportados por grandes centros de dados na Internet [Armburst et al. 2010, Foster et al. 2008, Vaquero et al. 2009]. As nuvens podem ser organizadas basicamente em três modelos [Armburst et al. 2010]: nuvem pública, nuvem privada e nuvem híbrida. Os serviços de nuvem pública são caracterizados pela

sua disponibilidade para clientes através de um modelo do tipo pagamento por uso. Por sua vez, no serviço de nuvem privada o centro de dados (*datacenters*) é interno a empresa ou outro tipo de organização, não estando disponíveis publicamente [Armbrust et al. 2010]. A nuvem privada oferece muitos dos benefícios de um ambiente de Computação em Nuvem pública, como a elasticidade, ainda que limitada, e o modelo baseado em serviços. Finalmente, a nuvem híbrida materializa-se quando uma nuvem privada é suplementada com características presentes em nuvens públicas (i.e. informações de negócio que sejam não críticas e operações de processamento podem ser mantidas na nuvem pública, enquanto serviços e dados de negócio que sejam considerados críticos podem ser mantidos sob o controle dos usuários no escopo da nuvem privada). Dessa forma, há uma utilização mista e integrada desses dois modelos de nuvem. Tais plataformas permitem que dados e processos sejam gerenciados internamente a uma organização sem restrições de banda de rede, exposição à segurança e requisitos legais que o uso de serviços de nuvem pública pode requerer. De fato, boa parte das restrições de uso imposta pelas nuvens públicas leva ao *cloud lock-in* [Buyya, Broberg e Goscinski 2011, Armbrust et al. 2010].

O *cloud lock-in* pode ser visto como a dependência entre uma aplicação e os recursos de um determinado provedor de computação em nuvem [Buyya, Broberg e Goscinski 2011]. Esse problema geralmente ocorre quando o provedor implementa um conjunto próprio de APIs (*Application Programming Interface*), e ao migrar a aplicação de um provedor para outro, é necessário modificá-la substancialmente para usar a API do novo provedor. Em termos práticos isto aprisiona a aplicação à nuvem, uma vez que a mudança de provedor é uma tarefa complexa e de custo elevado [Buyya, Broberg e Goscinski 2011]. Deste ponto em diante, o termo *cloud lock-in* será traduzido como *aprisionamento*.

Apesar do aprisionamento ser considerado um dos principais desafios da computação em nuvem [Armbrust et al. 2010], não há estudos que avaliem sistematicamente a compatibilidade entre diferentes plataformas de nuvem. Assim, este artigo busca avaliar o aprisionamento que uma organização teria que superar para migrar uma aplicação de uma nuvem pública para uma privada/híbrida, ou vice-versa. Tal avaliação é relevante uma vez que pode: (i) nortear os usuários de nuvens na tomada de decisões sobre migrações entre plataformas de nuvens públicas e privadas, além de (ii) fornecer embasamento a uma possível interoperabilidade entre plataformas de nuvens.

Esse trabalho tem dois objetivos principais: (i) propor um processo genérico para avaliação do aprisionamento entre plataformas de nuvem através de uma abordagem de teste de compatibilidade; (ii) aplicar o processo proposto para avaliar o aprisionamento entre uma plataforma pública (Amazon EC2) e três plataformas privadas (OpenStack, Eucalyptus e OpenNebula). A justificativa para escolha da Amazon EC2 como base é que em termos de ambiente de nuvem comercial, a Amazon é líder de mercado e, por esse motivo, algumas plataformas de nuvem *open-source*, como Eucalyptus, OpenStack e OpenNebula, preconizam que disponibilizam acesso a seus serviços via a API da Amazon EC2. No entanto, na literatura não há relato sobre a existência de uma abordagem sistemática utilizada para a avaliação da compatibilidade dessas nuvens com a EC2. Esse trabalho mostra uma avaliação da compatibilidade e investiga se a compatibilidade existente é suficiente para evitar o problema de aprisionamento.

O artigo está estruturado em cinco seções. A Seção 2 apresenta trabalhos que realizam comparações entre plataformas de nuvens. A Seção 3 apresenta o processo definido nesse trabalho para avaliar o aprisionamento entre plataformas de nuvens. A Seção 4 descreve um estudo de caso que aplica o processo avaliativo proposto e os resultados obtidos em cada fase. Por fim a Seção 5, contém as conclusões.

2. Trabalhos Relacionados

Existem vários estudos que buscam comparar e avaliar plataformas de nuvem privadas e híbridas, de modo que potenciais usuários, desenvolvedores e administradores possam ter um conjunto de dados para ajudá-los a decidir qual seria a plataforma mais adequada para suas demandas. No trabalho desenvolvido por [Peng et al 2009] são comparadas as plataformas Eucalyptus, OpenNebula, entre outras. Na descrição da compatibilidade é afirmado que a Eucalyptus tem suporte a API EC2. Apesar de tal conclusão, o estudo não deixa claro como tal resultado foi obtido. Outro exemplo é encontrado em [Mahjoub et al. 2011] no qual as plataformas Eucalyptus e OpenNebula, dentre as avaliadas, são ditas compatíveis com Amazon EC2 enquanto que a OpenStack não. De forma semelhante ao artigo citado anteriormente, não há clareza na sistemática que levou a essa conclusão. O trabalho realizado por [von Laszewski, et. al 2012] afirma que a API EC2 é um padrão "de fato" e que as três plataformas objetos de estudo neste trabalho são apresentadas como compatíveis a API EC2. Embora seja ressaltado que essa compatibilidade corresponde às funcionalidades básicas como: registro e *upload* de imagens e execução, finalização e descrição de máquinas virtuais, mais uma vez a abordagem metodológica não é salientada.

Algumas soluções de código aberto para computação na nuvem são comparadas em [Endo, et al 2010] e são apresentados os desafios que os desenvolvedores enfrentam para trabalharem com soluções usando plataformas de nuvens. Esse trabalho analisou a capacidade da API de trabalhar com diferentes tipos de sistemas operacionais e diversos tipos de máquinas virtuais. Além disso, esse trabalho analisou as restrições que a referida API impõe aos programadores, assim como os acessos destes na solução.

O presente trabalho diferencia-se dos demais nos seguintes aspectos: (i) apresentação de um processo genérico de avaliação do aprisionamento com sistemática bem definida; (ii) avaliação específica do aprisionamento em termos da API fornecida pelas plataformas, não abordando outros problemas de aprisionamento como os relacionados a imagens e virtualização e (iii) uso de teste de compatibilidade para avaliar o processo proposto através de um estudo de caso que compara o aprisionamento entre plataforma pública (Amazon EC2) e três plataformas privadas (OpenStack, Eucalyptus e OpenNebula).

3. Processo de Avaliação de Aprisionamento

Esta seção descreve a metodologia definida nesse trabalho para avaliar o aprisionamento entre plataformas de nuvem através de uma abordagem de teste de compatibilidade. O teste de compatibilidade [Lewis, W. E. 2000] busca identificar incompatibilidades entre componentes. Neste caso, o teste de compatibilidade será utilizado para avaliar se as interfaces fornecidas pelas plataformas de nuvem são compatíveis ou não com uma interface específica. O processo de avaliação aqui proposto foi inspirado em [Mariani, L. et al. 2007; Flores e Polo 2009] e é formado por três etapas, como descritas na Figura 1.

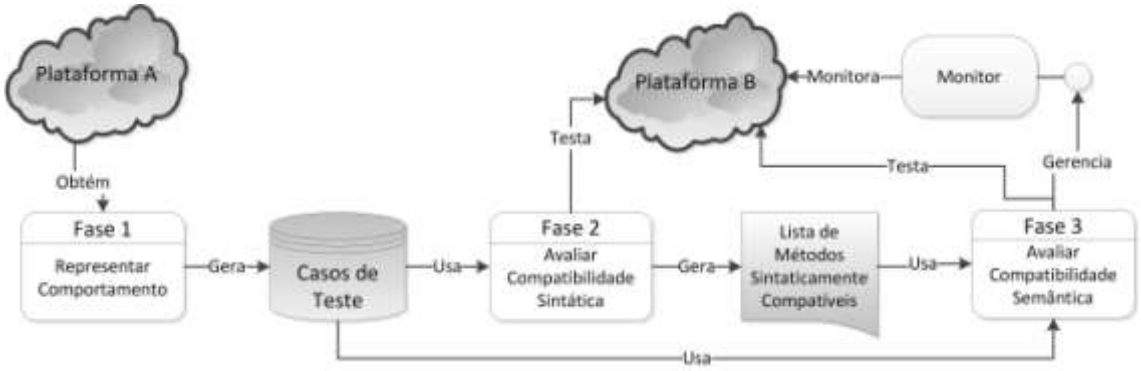


Figura 1. Processo para avaliação de Nível de Aprisionamento

A primeira fase do processo de avaliação de aprisionamento é responsável por analisar o comportamento de uma das plataformas envolvidas no processo de avaliação. A Plataforma A é geralmente representada pela plataforma onde uma determinada aplicação esta implantada. Nesta fase são gerados casos de teste unitários (CT), isto é casos de teste da menor unidade, no caso métodos, que buscam descrever diferentes aspectos das funcionalidades fornecidas pela Plataforma A. Note que o objetivo dos casos gerados nesta fase não é procurar defeitos na Plataforma A, mas representar seu comportamento. Dado que os CTs são executados com sucesso na Plataforma A, espera-se resultado similar durante a execução dos CTs na Plataforma B, para que a mesma seja compatível com a Plataforma A. Maiores detalhes sobre esta fase serão descritos na Seção 4.2.

Por sua vez, a segunda fase do processo de avaliação consiste em avaliar se a interface fornecida pela Plataforma B é sintaticamente compatível com a interface fornecida pela Plataforma A. A Plataforma B é geralmente representada por uma plataforma alvo de uma possível migração. A compatibilidade sintática avalia possíveis diferenças entre nomes e parâmetros dos elementos da interface. Tal compatibilidade é testada através de execução dos casos de teste gerados na fase anterior (CTs) na Plataforma B. Se o caso de teste passar, o referido método testado é inserido em uma lista de métodos com provável compatibilidade. Por outro lado, se o caso de teste falhar, o método é marcado como incompatível.

A última fase do processo proposto avalia se o comportamento da Plataforma B é o mesmo da Plataforma A para os métodos avaliados. Isto implica em executar os casos de teste na Plataforma B e avaliar através de monitores se o comportamento observado é o mesmo esperado pela Plataforma A. Para reduzir o numero de monitores gerados, este processo monitora apenas os métodos que compõem a lista de métodos gerada na fase anterior. Maiores detalhes sobre esta fase serão descritos na Seção 4.3.

A geração de monitores não é uma tarefa trivial uma vez que exige o monitoramento assíncrono de vários atributos da nuvem avaliada. Por este motivo, o processo foi dividido entre compatibilidade sintática (fase 2) e semântica (fase 3), visando com isso reduzir o número de monitores a serem gerados.

4. Estudo de Caso

Como descrito na Seção 2, vários artigos afirmam que as plataformas OpenStack, Eucalyptus e OpenNebula são compatíveis com a API Amazon EC2. No entanto, nenhum deles quantifica tal compatibilidade. Diante disto, esta seção irá detalhar como

o processo proposto pode ser utilizado para quantificar o aprisionamento entre uma plataforma pública (Amazon EC2) e três plataformas privadas (OpenStack, Eucalyptus e OpenNebula).

Para efeito de ilustração, a Amazon EC2 será considerada a plataforma original (que hospeda uma aplicação fictícia A), enquanto que as plataformas privadas serão tratadas como plataformas candidatas a hospedarem tal aplicação A. Neste caso, o objetivo do estudo de caso é quantificar a compatibilidade entre a Amazon EC2 e as outras plataformas avaliadas. Quanto maior a compatibilidade, menor será o aprisionamento.

4.1. Plataformas Avaliadas e Ambiente de Execução

Para este estudo de caso, a Amazon *Elastic Compute Cloud* (EC2) foi escolhida como plataforma original por ser, atualmente, a nuvem comercial mais popular, estável e rica em recursos [Juve et al. 2009], mantendo a liderança do mercado e sendo utilizada em várias soluções [Juve et al. 2009, Juve et al. 2010, Turcu, Foster e Nesterov 2009, Lenk et al. 2011], de forma que sua API de acesso a AWS EC2 é considerada um padrão "de fato" [von Laszewski, et. al 2012]. Por sua vez, as plataformas OpenStack, Eucalyptus e OpenNebula foram escolhidas para compor a avaliação por serem consideradas as principais plataformas que suportam a implantação de nuvens Privada e Hibridas [Khan, Ylitalo e Ahmed 2011]. As seções seguintes descrevem suscintamente cada uma das plataformas avaliadas.

4.1.1. Amazon Web Services (AWS)

O *Amazon Web Services* (AWS) são serviços de nuvem providos pela *Amazon*. Estes serviços oferecem poder computacional, facilidades de armazenamento e várias outras funcionalidades que permitem que empresas implantem aplicações e serviços com suporte a escalabilidade e confiabilidade. O AWS é formado por vários serviços: (i) Amazon *Elastic Compute Cloud* (EC2), Amazon *Simple Storage Service* (S3), Amazon *Relational Database Service* (RDS), Amazon *SimpleDB*, entre outros serviços.

Para este estudo de caso, iremos restringir a avaliação apenas ao A*mazon EC2* dado que este é o único serviço suportado por todas as outras plataformas avaliadas. O *Amazon EC2 é* um serviço que oferece capacidade de computação redimensionável na nuvem. Esse serviço apresenta-se como um verdadeiro ambiente de computação virtual, permitindo aos usuários, através de uma interface Web, criar, usar e gerenciar máquinas virtuais com sistemas operacionais Windows e Linux, ou mesmo iniciar tais máquinas de acordo com as necessidades das aplicações.

4.1.2. OpenStack

OpenStack [OpenStack 2012] é uma coleção de projetos de software *open-source* que podem ser utilizados por empresas ou provedores de serviços para implantar uma infraestrutura de nuvem privada ou híbrida. O OpenStack é constituído por três serviços principais: *OpenStack Compute Infrastructure* (*Nova*), *OpenStack Storage Infrastructure (Swift*) e *OpenStack Imaging Service* (*Glance*).

Para efeito deste estudo de caso, o principal serviço utilizado será o *OpenStack Compute Infrastructure* (*Nova*). O *Nova* implementa um serviço de computação virtual,

incluindo a criação, iniciação e gerência de instâncias de máquinas virtuais, o gerenciamento da infraestrutura de rede e o controle de acesso de usuários aos recursos da nuvem. Seu desenvolvimento teve como base o serviço Nebula, desenvolvido pela NASA. É semelhante em escopo ao serviço EC2 da Amazon e expõe suas funcionalidades para usuários através de suas APIs [OpenStack 2012].

4.1.3. Eucalyptus

Eucalytpus [Eucalyptus 2012] é um *framework open-source* de computação em nuvem que permite a criação de *clusters* privados em *datacenters*. O Eucalyptus é constituído pelos seguintes serviços: *Cloud Controller* (CLC), *Walrus Storage Controller* (WS3), *Cluster Controller* (CC), *Node Controller* (*NC*) e *Elastic Block Storage Controller* (EBS).

Neste estudo de caso, o principal serviço utilizado será o *Cloud Controller* (CLC). O CLC implementa o ponto de entrada na nuvem para usuários e administradores, oferecendo um conjunto de APIs para facilitar sua administração. O CLC é responsável por coletar informações dos controladores de nós de execução e tomar decisões de escalonamento repassadas aos controladores de clusters [Eucalyptus 2012].

4.1.4. OpenNebula

O OpenNebula [OpenNebula 2012] é um projeto *open-source* que compreende um conjunto de ferramentas que viabilizam a gerência de *datacenters* e a criação de nuvens privadas ou híbridas virtualizadas segundo o modelo de infraestrutura como serviço. O OpenNebula é composto por cinco componentes principais: (i) interfaces e APIs, que permitem a interação de um usuário com uma infraestrutura estabelecida com o OpenNebula; (ii) grupos e usuários, responsáveis pela criação de contas e grupo e pelos mecanismos de autenticação e autorização que o sistema fornece; (iii) controle de rede, subsistemas responsáveis gerenciar a integração com outras redes ou *datacenter*, (iv) *host* e virtualizações, que suporta gerenciadores de virtualização para gerência do ciclo de vida; e (v) monitoramento das máquinas virtuais e o componente para armazenamento que pode ser configurado para suportar um sistema de arquivos compartilhado ou não [OpenNebula 2012].

Em nosso estudo de caso, o principal serviço avaliado será o *OpenNebula EC2 Query*. O EC2 *Query* API é um serviço Web que permite o gerenciamento de máquinas virtuais em uma nuvem OpenNebula [OpenNebula 2012].

4.1.5. Ambiente de Execução

Para esta avaliação foram utilizadas as versões Diablo do OpenStack, Eucalyptus 3.1 e OpenNebula 3.4. Tais plataformas foram instaladas em um servidor HP Proliant ML350 G6, 24 CPUs de 2.4GHz, 32 GB de memória e 2 TB de disco. Foi utilizado ainda o gerenciador de máquina virtual Xen Server na versão 5.6 e a distribuição GNU/Linux Ubuntu Server 11.04 com kernel atualizado para versão 3.2.0-23. A configuração *single node*, onde todos os componentes são instalados em um mesmo servidor, foi utilizada para todas as plataformas. Tal configuração foi utilizada para garantir que todas as plataformas analisadas usassem a mesma arquitetura de hardware e software.

4.2. Primeira Fase: Representação do Comportamento da EC2

O objetivo desta primeira fase da avaliação é gerar um conjunto de casos de teste (CTs) unitários que representem o comportamento da plataforma original (Amazon EC2), de modo que todas as suas funcionalidades sejam exercitadas.

A documentação e os métodos que compõem a API da plataforma avaliada são pontos que devem ser considerados durante a geração dos CTs. Por exemplo, as funcionalidades da Amazon estão descritas por meio de documentação e de uma API disponíveis através de kits de desenvolvimentos de aplicativos (SDK). Para este estudo de caso, foi utilizado o SDK Java (release 1.2.15), em virtude do mesmo ser amplamente utilizado e independente de sistema operacional.

Os casos de teste foram gerados com base nos métodos fornecidos pela classe "*com.amazonaws.services.ec2.AmazonEC2Client*". Tal classe disponibiliza um conjunto de 113 métodos que permitem gerenciar as funcionalidades da Amazon EC2.

```
01  public class BehaviorTestCase extends TestCase {
02    static boolean monitorFlag = false;
03
04    public static AmazonEC2 ec2;
05   public static void init(String endPoint, String accessKey, String secretKey) throws Exception {
06       BasicSessionCredentials credentials = new BasicSessionCredentials(accessKey, secretKey,"");
07       ec2 = new AmazonEC2Client(credentials);
08       ec2.setEndpoint(endPoint);
09    }
10    public void testDescribeImages(){
11       ec2.describeImages();
12       int statusCodeReturn = TestCaseLogger.getstatusCode("DescribeImages", monitorFlag);
13       assertEquals("Compativel", true, (statusCodeReturn >= 200 && statusCodeReturn <= 206));
14    }
15  }
```

Figura 2. Código de Exemplo do caso de teste DescribeImages.

A Figura 2 mostra um exemplo de caso de teste utilizado para testar o método *describeImages()*. O método *init* (Linhas 4-8) é responsável por criar uma instância da classe *AmazonEC2Client* e recebe como parâmetro três variáveis: um *endpoint*, uma *accesskey* e uma *secretkey*. O *endpoint* representa o endereço de rede do servidor alvo da solicitação. Por sua vez, a variável *accesskey* e *secretKey* representam a identificação alfanumérica única do usuário e sua senha, respectivamente.

O método *testDescribeImages()* representa o caso de teste unitário, onde uma invocação ao método *describeImages()* da API da Amazon EC2 é realizada (linha 10). Cada invocação à API da Amazon EC2 envia mensagens através do protocolo *Hypertext Transfer Protocol (HTTP)* contendo os detalhes da solicitação. Ao tratar tal mensagem, a plataforma alvo retorna uma mensagem de resposta HTTP contendo o *status code* [HTTP/1.1]. Através de uma analise do código fonte do SDK da Amazon (Classe *com.amazonaws.http.AmazonHttpClient*), foi observado que toda mensagem de resposta com *status code* entre valor 200 a 206 é considerada como requisição bem sucedida. O *status code* de cada mensagem resposta é obtido através de uma análise no log gerado pelo próprio SDK da Amazon.

Para realizar tal análise, foi implementada a classe *TestCaseLogger* que relaciona cada invocação de método com sua respectiva mensagem de resposta (*status code*). A Figura 3 descreve a relação entre os casos de teste e a classe *TestCaseLogger*. O *status code* da invocação do método *describeImages* é obtido através da invocação do

método *getstatusCode*. Tal método recebe como parâmetro o nome do método cuja *status code* deve ser retornado. O segundo parâmetro será detalhado na Seção 4.4.

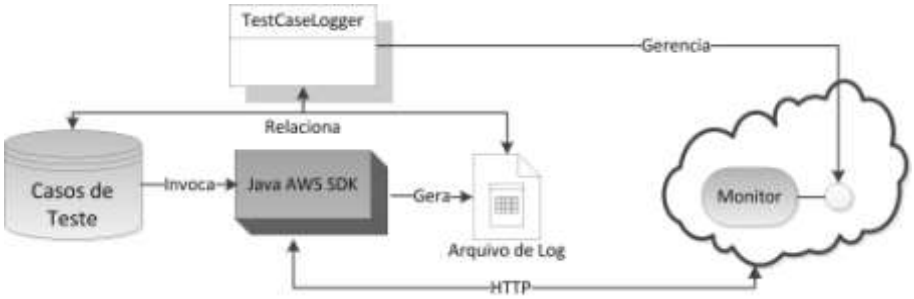


Figura 3. Arquitetura dos casos de teste gerados na Fase 1.

Baseado na arquitetura mostrada na Figura 3, foi utilizada a ferramenta IDE Eclipse [Eclipse 2012] em conjunto com o *framework* JUnit [JUnit 2012] para a geração de um caso de teste para cada método da API disponível no AWS SDK Java. A execução dos casos de teste foi organizada de modo que respeitasse a sequência exibida na Tabela 1. Tal sequência permite criar um conjunto de elementos (*Gateways*, DHCPs, *Images*, etc), realizar operações de modificação sobre estes elementos, consultar se tais mudanças foram realizadas e finalmente remover os elementos criados.

Tabela 1. Sequência de execução dos casos de teste.

Ord.	Operação	Métodos
1	Criar	*CreateCustomerGateway, CreateDhcpOptions, CreateImage, CreateInternetGateway, CreateKeyPair[12], CreateNetworkAcl, CreateNetworkAclEntry, CreatePlacementGroup, CreateRoute, CreateRouteTable, CreateSecurityGroup[12], CreateSnapshot[12], CreateSpotDatafeedSubscription, CreateSubnet, CreateTags, CreateVolume[12], CreateVpc, CreateVpnConnection, CreateVpnGateway*
2	Modificar	*AllocateAddress[123], AssociateAddress[3], AssociateDhcpOptions, AssociateRouteTable, AttachInternetGateway, AttachVolume[2], AttachVpnGateway, AuthorizeSecurityGroupEgress, AuthorizeSecurityGroupIngress[12], BundleInstance, CancelBundleTask, CancelSpotInstanceRequests, DeregisterImage[2], DetachInternetGateway, DetachVolume[2], DetachVpnGateway, DisassociateAddress[3], DisassociateRouteTable, ImportKeyPair[2], ModifyImageAttribute, ModifyInstanceAttribute, ModifySnapshotAttribute, MonitorInstances, PurchaseReservedInstancesOffering, RebootInstances[12], RegisterImage[2], ReleaseAddress[3], ReplaceNetworkAclAssociation, ReplaceNetworkAclEntry, ReplaceRoute, ReplaceRouteTableAssociation, RequestSpotInstances, ResetImageAttribute, ResetInstanceAttribute, ResetSnapshotAttribute, RevokeSecurityGroupEgress, RevokeSecurityGroupIngress[12], RunInstances[123], StartInstances[12], StopInstances[12], TerminateInstances[123], UnmonitorInstances*
3	Consultar	*ConfirmProductInstance[2], DescribeAddresses[123], DescribeAvailabilityZones[12], DescribeBundleTasks, DescribeConversionTasks, DescribeCustomerGateways, DescribeDhcpOptions, DescribeImageAttribute, DescribeImages[123], DescribeInstanceAttribute, DescribeInstances[123], DescribeInstanceStatus, DescribeInternetGateways, DescribeKeyPairs[12], DescribeNetworkAcls, DescribePlacementGroups, DescribeRegions[12], DescribeReservedInstances[2], DescribeReservedInstancesOfferings[2], DescribeRouteTables, DescribeSecurityGroups[12], DescribeSnapshotAttribute, DescribeSnapshots[12], DescribeSpotDatafeedSubscription, DescribeSpotInstanceRequests, DescribeSpotPriceHistory, DescribeSubnets, DescribeTags, DescribeVolumes[123], DescribeVpcs, DescribeVpnConnections, DescribeVpnGateways, GetConsoleOutput[12], GetPasswordData*
4	Deletar	*DeleteCustomerGateway, DeleteDhcpOptions, DeleteInternetGateway, DeleteKeyPair[12], DeleteNetworkAcl, DeleteNetworkAclEntry, DeletePlacementGroup, DeleteRoute, DeleteRouteTable, DeleteSecurityGroup[12], DeleteSnapshot[12], DeleteSpotDatafeedSubscription, DeleteSubnet, DeleteTags, DeleteVolume[12], DeleteVpc, DeleteVpnConnection, DeleteVpnGateway*

A partir da geração dos casos de teste, foi observado que dos 125 métodos da API, 12 métodos (mostrados na Tabela 2) não são implementados pelo SDK fornecido pela Amazon. Ao se consultar a documentação da Amazon, constatou-se que esta é uma limitação intrínseca da versão do SDK utilizada.

Tabela 2. Métodos não avaliados.

Métodos da EC2 API não implementados pelo AWS SDK Java		
AttachNetworkInterface	DescribeNetworkInterfaceAttribute	ImportVolume
CancelConversionTask	DescribeNetworkInterfaces	ModifyNetworkInterfaceAttribute
CreateNetworkInterface	DetachNetworkInterface	ReportInstanceStatus,
DeleteNetworkInterface	ImportInstance	ResetNetworkInterfaceAttribute

4.3. Segunda Fase: Compatibilidade sintática

Na avaliação da compatibilidade sintática, o objetivo é verificar se a implementação da interface AWS EC2 disponibilizada pela Plataforma B (OpenStack, Eucalyptus e OpenNebula) é compatível com sintaxe dos métodos fornecidos pela Plataforma A (Amazon EC2). Assim, considerando que os casos de teste (CTs) gerados na fase anterior representam as funcionalidades da API EC2, na Amazon, o objetivo desta fase é avaliar se o mesmo conjunto de CTs são executados, com sucesso, para cada Plataforma B avaliada. Se o caso de teste for executado com sucesso (obter um *status code* entre valor 200 a 206), então tal método testado é considerado sintaticamente compatível com a API EC2.

Cada requisição a API EC2 baseia-se no protocolo HTTP. Dessa forma, a ordem dos parâmetros não interfere no código de retorno. Caso haja a exclusão de algum parâmetro é retornado um código de erro e considera-se não compatível. Para observar esse comportamento foi necessário manipular o código do AWS SDK. Dado que ao utilizar apenas o AWS SDK tal comportamento é encapsulado em objetos de retorno. Salientamos que foi preciso diminuir o nível de segurança das requisições submetidas às nuvens OpenStack e Eucalyptus, uma vez que por padrão o AWS SDK utiliza a versão "V2" do algoritmo "*HmacSHA1*", tendo sido necessário fazer uso da versão "V1" nessas duas plataformas.

Com a avaliação da plataforma da Amazon EC2, realizada na fase 1, foram identificados 113 métodos que tiveram suas chamadas implementadas em casos de testes. Os resultados obtidos na segunda fase revelaram um comportamento singular da plataforma OpenNebula, uma vez que todos os métodos foram compatíveis sintaticamente. Na Eucalyptus, 34 foram compatíveis e na OpenStack, 26. Os resultados comparativos da Fase 1 e Fase 2 são apresentados na Tabela 3.

Tabela 3. Resultados das Fases 1 e 2 do processo avaliativo.

Plataformas Avaliadas	Número de Métodos compatíveis	
	Fase 1	Fase 2
Amazon EC2	113	n/a
OpenStack	n/a	26
Eucalyptus	n/a	35
OpenNebula	n/a	113

4.4 Terceira Fase: Compatibilidade semântica

A terceira fase objetiva monitorar o comportamento na plataforma B (OpenStack, Eucalyptus e OpenNebula) durante a execução dos casos de testes (CTs) selecionados na fase 2. O monitoramento irá garantir se a ação solicitada pelo caso de teste foi realizada como esperado na plataforma avaliada. Para isso, deve ser implementado um monitor para cada caso de teste que foi executado com sucesso na fase anterior. Considerando que, em muitos casos, a implementação do monitor não é uma tarefa

trivial, a divisão da avaliação em compatibilidade sintática e semântica permitiu reduzir o número de monitores implementados para as plataformas OpenStack e Eucalyptus.

Um cenário que exemplifica a necessidade do monitoramento para comprovar a compatibilidade semântica entre plataformas é representado pelo comportamento do método *RunInstances*. Esse método inicia a execução de uma ou mais instância(s) na nuvem, tal operação não é realizada de forma síncrona. Isto ocorre, pois para efetuar tal ação a plataforma deve carregar as informações da imagem para que o *hypervisor* (gerenciador de máquinas virtuais) responsável possa iniciar a execução da máquina virtual que representará a instância. Todo esse processo demanda certo intervalo de tempo, de forma que uma chamada ao método *RunInstances* apenas garante que o processo foi iniciado. Assim, a comprovação da eficácia da chamada efetuada, isto é, se a instância está de fato executando, deve ser realizada por um monitor de comportamento.

Como mostrado na Figura 1, os mesmos casos de teste gerados na fase 1 e que foram sintaticamente compatíveis na fase 2 são utilizados na fase 3. No caso de teste ilustrado na Figura 2, para realizar o monitoramento das ações invocadas é necessário apenas mudar o valor da variável *monitorFlag* de *false* para *true*. Tal mudança fará com que a classe *TestCaseLogger* procure por um monitor previamente registrado para monitorar tal ação. Além disso, quando o segundo parâmetro do método *getstatusCode* é informado como *true,* este método fica bloqueado até que o monitor retorne se a operação foi bem sucedida ou não.

```
01  public class RunInstanceResource extends ServerResource implements RunInstace {
02     @Get
03     public int monitore(String instanceId) {
04        return monitoreTentativa(0, instanceId);
05     }
06     private int monitoreTentativa(int tentativa, String instanceId){
07        int retorno = 400;
08        try {
09           if(OpenStackAPIClient.getStatusInstance(instanceId).equals("running")){
10              return 200;
11           }else{
12              if(tentativa <= 9){
13                 Thread.sleep(30000);
14                 return monitoreTentativa(++tentativa, instanceId);
15              }
16           }
17        } catch (InterruptedException e) {
18           e.printStackTrace();
19        }
20           return retorno;
21        }
22  }
```

Figura 4. **Exemplo de Implementação do Monitor.**

Um exemplo de monitor implementado em Java, com o Restlet *Framework* [Restlet 2012], é ilustrado na Figura 4. Neste caso, o comportamento do *RunInstances* é monitorado, através do método *monitore*. Tal método recebe como parâmetro o *instanceId* (linha 03) que representa a identificação única da instância iniciada via o caso de teste *testRunInstances*. Através desse parâmetro o monitor requisita informações sobre a instância que está sendo iniciada. Dentre os atributos retornados, o *instanceState* define o estado da instância, que pode ser: *pending, running, shutting-down, terminated, stopping* ou *stopped*. Para isso, o monitor efetua 10 requisições (linha

12-14) à plataforma, sendo uma a cada 30 segundos (linha 13), consultando se a instância daquele *instanceId* já possuí um *instanceState* de valor *running* (linha 09), caso isso ocorra, o monitor retorna para o *TestCaseLogger* um *status code* com valor *200* (linha 20), indicando que a operação foi bem sucedida. Note que o monitor descrito na Figura 4 está executando em uma instancia da nuvem OpenStack e usa a própria API do OpenStack para realizar tal monitoramento.

Após a execução dos casos de teste e dos seus respectivos monitores, os resultados obtidos mostram que, dos 113 métodos avaliados: na OpenStack, 26 são compatíveis e 87 não são compatíveis; na Eucalyptus, 34 são compatíveis e 79 não; e na OpenNebula, 10 são compatíveis, enquanto 103 não são. A Tabela 4 resume o número de métodos compatíveis para cada fase do processo de avaliação. A lista de todos os métodos compatíveis pode ser obtida na Tabela 1. Os métodos que possuem um número são métodos que obtiveram compatibilidade semântica com alguma plataforma. A presença do número 1 indica que o método é compatível com a plataforma OpenStack, enquanto que 2 representa compatibilidade com a plataforma Eucalyptus e 3 indica a compatibilidade com OpenNebula. Por exemplo, o método *RunInstances*[123] é compatível com todas as plataformas avaliadas, enquanto que *DetachVolume*[2] é compatível apenas com a plataforma Eucalyptus.

Tabela 4. Resultados por Fases (1,2 e 3) do processo avaliativo.

Plataformas Avaliadas	Número de Métodos compatíveis		
	Fase 1	Fase 2	Fase 3
Amazon EC2	113	n/a	n/a
OpenStack	n/	26	26
Eucalyptus	n/a	34	34
OpenNebula	n/a	113	10

Quanto aos métodos compatíveis na terceira fase, as plataformas Eucalyptus e OpenStack obtiveram o mesmo resultado da fase anterior. Por outro lado, a plataforma OpenNebula apresentou apenas 10 métodos compatíveis quando na fase anterior tinha sido a melhor avaliada. Esse comportamento ocorre devido a implementação da API EC2 dessa plataforma possuir um código que retorna um *status code* de sucesso (200) para qualquer chamada efetuada a ela, mesmo essa chamada não estando propriamente implementada.

Considerando apenas os 113 métodos testados, a compatibilidade das plataformas avaliadas foi de 23,01% na OpenStack, 30,09% na Eucalyptus e 8,85% na OpenNebula. Considerando que quanto menor for a compatibilidade entre plataformas, maior será o aprisionamento. Concluímos que o aprisionamento nas plataformas objetos da avaliação foi de 76,99% (100% - 23,01%), 69,61% (100% - 30,09%) e 91,15% (100% - 8,85%) para a OpenStack, Eucalyptus e OpenNebula, respectivamente.

Quando analisamos a compatibilidade por funcionalidade, observamos que algumas funcionalidades chegam a ter 100% de compatiblidade, entretanto a grande maioria ainda possui 0% de compatibilidade. Na Tabela 5 apresentamos os resultados agrupados por funcionalidade e adotamos as abreviaturas **M** para número total de métodos da funcionalidade, **C** para quantidade de compatíveis, **N** para não compatíveis e **%C** para a porcentagem da compatibilidade.

Tabela 5. Compatibilidade por Função EC2 API.

Amazon EC2 API		Plataformas Avaliadas								
		OpenStack			Eucalyptus			OpenNebula		
Funcionalidade	M	C	NC	%C	C	NC	%C	C	NC	%C
Amazon DevPay	1	0	1	0 %	1	0	100 %	0	1	0 %
AMIs	7	1	6	14,29 %	3	4	42,86 %	1	6	14,29 %
Availability Zones and Regions	2	2	0	100 %	2	0	100 %	0	2	0 %
Customer Gateways	3	0	3	0 %	0	3	0 %	0	3	0 %
DHCP Options	4	0	4	0 %	0	4	0 %	0	4	0 %
Elastic Block Store	11	6	5	54,55 %	8	3	72,73 %	0	11	0 %
Elastic IP Addresses	5	2	3	40 %	2	3	40 %	5	0	100 %
General	1	1	0	100 %	1	0	100 %	0	1	0 %
Instances	10	6	4	60 %	6	4	60 %	4	6	40 %
Internet Gateways	5	0	5	0 %	0	5	0 %	0	5	0 %
Key Pairs	4	3	1	75 %	4	0	100 %	0	4	0 %
Monitoring	2	0	2	0 %	0	2	0 %	0	2	0 %
Network ACLs	7	0	7	0 %	0	7	0 %	0	7	0 %
Placement Groups	3	0	3	0 %	0	3	0 %	0	3	0 %
Reserved Instances	3	0	3	0 %	2	1	66,67 %	0	3	0 %
Route Tables	9	0	9	0 %	0	9	0 %	0	9	0 %
Security Groups	7	5	2	71,43 %	5	2	71,43 %	0	7	0 %
Spot Instances	7	0	7	0 %	0	7	0 %	0	7	0 %
Subnets	3	0	3	0 %	0	3	0 %	0	3	0 %
Tags	3	0	3	0 %	0	3	0 %	0	3	0 %
VM Import	1	0	1	0 %	0	1	0 %	0	1	0 %
VPCs	3	0	3	0 %	0	3	0 %	0	3	0 %
VPN Connections	3	0	3	0 %	0	3	0 %	0	3	0 %
Virtual Private Gateways	5	0	5	0 %	0	5	0 %	0	5	0 %
Windows	4	0	4	0 %	0	4	0 %	0	4	0 %
Totais	**113**	**26**	**87**	**23,01 %**	**34**	**79**	**30,09 %**	**10**	**103**	**8,85 %**

Conforme os resultados apresentados, a plataforma Eucalyptus, que foi a de maior compatibilidade, teve sua compatibilidade igual a apenas 30,09%. Comprovando que a compatibilidade com a Amazon EC2 API não ocorre em nenhuma das plataformas avaliadas. Caso seja considerada a análise apenas das funcionalidades principais, como afirma [von Laszewski, et al. 2012], *AMIs* (gerência de imagens) e *Instances* (gerência do ciclo de vidas das instâncias), a plataforma Eucalyptus novamente destaca-se como melhor avaliada (52,94%), a OpenStack fica com 41,18% e a OpenNebula com 29,41%.

Como justificativas para a não compatibilidade foram elencados três motivos: (i) a plataforma não possuía a funcionalidade avaliada, (ii) a plataforma possui tal funcionalidade, mas ela é incompatível com a EC2 e (iii) a plataforma possui tal funcionalidade, mas o módulo EC2 da plataforma não implementa tal chamada. Os resultados dessa análise estão exibidos na Tabela 6 que segue a notação **I** para o primeiro motivo, **II** para o segundo e **III** para o terceiro.

Tabela 6. Motivos para Não Compatibilidade.

Plataformas Avaliadas	Motivos para Não compatibilidade		
	I	II	III
OpenStack	69	6	11
Eucalyptus	61	1	9
OpenNebula	74	0	29

As funcionalidades que expõem serviços próprios da Amazon foram as principais responsáveis pelos resultados de não compatibilidade, como exemplo destacam-se: *Network ACLs*, *Placement Groups*, *Virtual Private Gateways*, entre outras. E mesmo uma das funcionalidades principais como *Instances*, não obteve 100% de compatibilidade em nenhuma das plataformas.

5. Conclusão

Este artigo propôs um processo que avalia o aprisionamento entre plataformas de nuvens. O processo é baseado em teste de compatibilidade entre diferentes plataformas. Assim, quanto maior a compatibilidade entre as plataformas menor será o aprisionamento. O processo foi instanciado através de uma avaliação do aprisionamento entre a API da Amazon EC2 e as plataformas OpenStack, Eucalyptus e OpenNebula.

Apesar de vários artigos [Peng et al 2009; Mahjoub et al. 2011; von Laszewski, et. al 2012] concluírem que as plataformas OpenStack, Eucalyptus e OpenNebula fornecem compatibilidade com a API da Amazon EC2, neste artigo observamos, de forma sistemática, que a compatibilidade foi de apenas 23,01% na OpenStack, 30,09% na Eucalyptus e 8,85% na OpenNebula. Esses resultados indicam que aplicações que usam extensivamente a API EC2 sofrem aprisionamento (*cloud lock-in)* na Amazon, dado que a compatibilidade das plataformas avaliadas com a API EC2 ainda é reduzida. Abordagens como [OCCI 2013] e [JClouds 2013] foram propostas para reduzir o aprisionamento, constituindo-se em uma possível solução aos questionamentos que surgem pelos resultados apresentados. Como trabalhos futuros pretendemos selecionar outras plataformas de nuvem livres que disponibilizem acesso via EC2 API aplicando sobre elas estudo similar.

Agradecimentos

Este trabalho é parcialmente apoiado pelo Instituto Nacional de Ciência e Tecnologia para Engenharia de Software (INES), financiado pelo CNPq e pela FAPERN, com os projetos 573964/2008-4 e PPP-III-013/ 2009.

Referências

Armbrust, M., Fox, A., Griffith, R., et al. (2010). “A View of Cloud Computing”. Communications of The Acm, Vol 53, No 4, p. 50-58.

AWS EC2API (2012). Amazon Elastic Compute Cloud. “API Reference Version 15/12/2011”. http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/Welcome.html?r=8186, Fevereiro.

AWS JAVA (2012). Amazon Web Service (2012). “AWS SDK for Java”. http://aws.amazon.com/pt/sdkforjava/, Fevereiro.

Buyya R.; Broberg J.; Goscinski A. (2011). “Cloud Computing Principles e Paradigms.”, John Wiley & Sons ltd., England.

EC2 (2012). Amazon Web Services. “Amazon Elastic Compute Cloud (Amazon EC2)”, http://aws.amazon.com/pt/ec2/, Fevereiro.

Eclipse (2012). Eclipse Project, http://www.eclipse.org/, Março.

Endo, P. T.; Estácio, G. G.; Judith, K.; Sadok, D. (2010). “A Survey on Open-source Cloud Computing Solutions.” VIII Workshop em Clouds, Grids e Aplicações.

Eucalyptus (2012). “Eucalyptus open-source cloud computing infrastructure – an overview”. Technical report, Eucalyptus, Inc., August 2009.

Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing

360-degree compared. GCE'08,p. 1–10. IEEE.

Flores, A., and Polo, M.. 2009. Testing-based Process for Evaluating Component Replaceability. *Electron. Notes Theor. Comput. Sci.* 236 (April 2009), 101-115.

HTTP/1.1 (2012). Hypertext Transfer Protocol -- HTTP/1.1. http://www.ietf.org/rfc/rfc2616.txt, Março.

JClouds (2013). Cloud interfaces, simplified. http://www.jclouds.org/

JUnit (2012). Welcome to the JUnit wiki, https://github.com/kentbeck/junit/wiki, Março.

Juve, G., Deelman, E., Vahi, K., Mehta, G., et al (2009). Scientific Workflow Applications on Amazon EC2. e-Science 2009, Oxford, UK.

Juve, G., Deelman, E., Vahi, K., Mehta, G., et al (2010). Data Sharing Options for Scientific Workflows on Amazon EC2. Proceedings of the 2010 ACM/IEEE SC10

Khan, R.H., Ylitalo, J., Ahmed, A.S., (2011). OpenID Authentication As A Service in OpenStack. 7th IAS, p. 372-377.

Lenk, M., Menzel, M., Lipsky, J. et al (2011). What are you paying for? Performance benchmarking for Infrastructure-as-a-Service offerings. 2011 4th IEEE Cloud.

Lewis, W. E. Software Testing and Continuous Quality Improvement. CRC Press LLC, Apr. 2000.

Mariani, L., Papagiannakis, S., and Pezze, M. 2007. Compatibility and Regression Testing of COTS-Component-Based Software. ICSE '07, 85-95.

Mahjoub, M., Mdhaffar A., et al (2011). A comparative study of the current Cloud Computing technologies and offers. 2011 IEEE NCCA. IEEE. p. 131 a 134.

OCCI (2013). Open Cloud Computing Interface. http://occi-wg.org.

OpenStack (2012). OpenStack Compute Stater Guide.

http://docs.openstack.org/diablo/openstack-compute/starter/openstack-starter-guide-diablo.pdf

OpenNebula (2012). .http://opennebula.org/. Novembro.

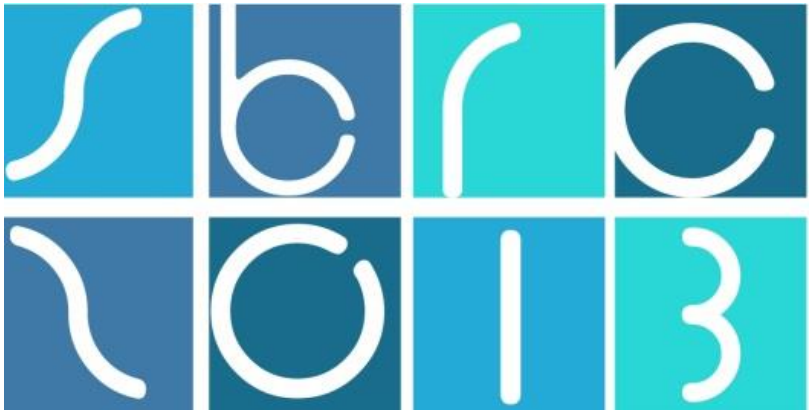
Peng, J.; Zhang, X., Lei, Z., Zhang, B., Zhang, W., Li, Q. (2009). "Comparison of Several Cloud Computing Platforms". ISISE '09, p. 23-27.

Restlet (2012). RestFul web framework for Java . http://www.restlet.org/. Novembro.

Turcu, G., Foster, I., Nesterov, S. (2010). Reshaping text data for efficient processing on Amazon EC2. Proceedings of the 19th ACM HPDC.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2009). "A Break in the Clouds: Towards a Cloud Definition". ACM SIGCOMM Computer Communication Review, Vol. 39(1), p.50–55, Janeiro de 2009.

von Laszewski, G.; Diaz, J.; Fugang Wang; Fox, G.C. (2012). "Qualitative Comparison of Multiple Cloud Frameworks". Proceedings: 2012 IEEE 5th international conference on cloud computing, p. 734-741.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 19

Segurança

Detecção de Alertas de Segurança em Redes de Computadores Usando Redes Sociais

Luiz Arthur F. Santos[1,2], Rodrigo Campiolo[1,2], Marco A. Gerosa[2], Daniel M. Batista[2]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão – PR – Brasil

[2]Departamento de Ciência da Computação – Universidade de São Paulo (USP)
São Paulo – SP – Brasil

{luizsantos,rcampiolo}@utfpr.edu.br, {gerosa,batista}@ime.usp.br

Resumo. *Neste artigo é apresentado um método de extração de notificações de segurança a partir de mensagens postadas em redes sociais. Nossa metodologia consiste em coletar e processar as mensagens de segurança postadas no Twitter usando técnicas de mineração de dados, em específico, agrupamento e classificação. Desenvolvemos um método para extrair e evidenciar alertas e problemas de segurança em ambientes de redes de computadores, otimizado com heurísticas de aumento de relevância. Como resultado, verificou-se que cerca de 92% das mensagens recuperadas abordam tópicos relacionados com segurança de redes de computadores e que mais de 50% representam potenciais alertas.*

Abstract. *This paper presents a method of extracting security notifications based on messages posted on social networks. We collect and process security messages posted on Twitter using data mining techniques, in particular, clustering and classification. We developed a method to extract and to show alerts and security problems in computing environments, with optimized heuristics to increase relevance. As a result, it was found that about 92% of retrieved messages address topics relating to network security, and more than 50% represent potential alerts.*

1. Introdução

O acesso rápido a informações sobre ameaças que afetam ambientes computacionais é fundamental para implantação de medidas preventivas. Sítios especializados e listas de e-mail são formas tradicionais de alerta sobre vulnerabilidades ou ameaças à segurança, entretanto nem sempre são eficazes em publicar rapidamente ameaças recentes [Frei et al. 2006]. Sistemas de detecção de intrusão e antivírus, por exemplo, sofrem com problemas de falsos negativos, causados pela ausência de assinaturas das ameaças em suas bases [Akyazi and Uyar 2010, Yuan and Zou 2011]. Como as redes sociais proveem uma fonte de dados rica e heterogênea, além de disseminarem informações de forma rápida e colaborativa, elas podem ser exploradas como formas de propagação de notificações de segurança.

[Santos et al. 2012] comprovaram que há mensagens no Twitter que alertam ou notificam sobre ameaças ou problemas de segurança. Entretanto, as mensagens devem

ser classificadas e evidenciadas para serem efetivamente exploradas por administradores de redes e usuários preocupados com a segurança. Esse processamento é necessário por conta de mensagens de segurança relacionadas a outros assuntos, como por exemplo, mensagens de *spams* e *phishing*, mensagens de segurança física e política, mensagens mal formadas, entre outras.

Neste artigo é proposto um método para extrair notificações de segurança em mensagens postadas no *microblog* Twitter. As mensagens com conteúdos irrelevantes são removidas de uma lista de interesse por meio do uso de técnicas heurísticas e de aprendizagem não supervisionada. Além disso, são apresentadas técnicas para agrupar e aumentar a relevância de mensagens de segurança baseada na sua propagação na rede social.

Considerando as características de heterogeneidade e colaboração dos usuários de redes sociais e, em específico, a característica de alta taxa de disseminação e propagação das mensagens no Twitter [Cha et al. 2012, Lerman and Ghosh 2010], tem-se como contribuição um mecanismo que viabiliza o acesso rápido a importantes notificações de segurança. Este artigo difere-se dos encontrados na literatura porque explora as redes sociais como fontes de informação sobre alertas de redes de computadores.

O artigo discute na Seção 2 os trabalhos relacionados, a Seção 3 descreve os procedimentos e questões de pesquisa que nortearam o desenvolvimento do trabalho, a Seção 4 apresenta os resultados, a análise e discussões dos procedimentos e resultados, e a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

As redes sociais possibilitam a colaboração e a difusão rápida de informações entre os usuários. Muitos trabalhos [Lerman and Ghosh 2010, Ye and Wu 2010, Kwak et al. 2010] verificaram que as mensagens propagadas no Twitter mantêm taxa de propagação constante e podem atingir uma alta taxa de disseminação em um curto período de tempo. Essa característica de redes sociais é útil para a área de segurança.

Ao contrário dos trabalhos que analisam as redes sociais como formas de propagação de *malware* e *spams* [Brown et al. 2008, Luo et al. 2009, Gao et al. 2010, Grier et al. 2010], as redes sociais podem ser exploradas como uma fonte valiosa de informações para prevenir ataques. O monitoramento e detecção de eventos em redes sociais já tem sido explorado por muitas áreas [Bonchi et al. 2011, Lee 2012].

[Sakaki et al. 2010] apresentaram um método para monitorar e correlacionar eventos de terremotos, usando um classificador baseado nas palavras-chaves, número de palavras e contexto de mensagens postadas no Twitter. [Sayyadi et al. 2009] partiram da premissa que documentos que descrevem o mesmo evento contém conjunto de palavras-chaves similares. [Aggarwal and Subbian 2012] realizaram experimentos combinando informações de conteúdo e fluxo de mensagens, e alcançaram melhores resultados que os métodos baseados apenas em texto.

No mesmo contexto de nosso trabalho, [Phuvipadawat and Murata 2010] propuseram um método para agrupar e monitorar notícias de última hora no Twitter baseado na similaridade definida pelo esquema TF-IDF (*Term Frequency-Inverse Document Frequency*), aperfeiçoado por um fator de identificação de substantivos importantes nas men-

sagens, o que possibilita obter melhores resultados no agrupamento de mensagens com pouco texto.

Outros trabalhos usam informações de tempo e espaço para detectar novos eventos em redes sociais [Petrović et al. 2010, Mathioudakis and Koudas 2010]. Destaca-se o uso de termos que aparecem em alta taxa durante um período para elevar a importância de um tópico de interesse. Nosso trabalho também explora as informações temporais como um fator para elevar a importância de uma notificação.

Na área de segurança em redes de computadores os sistemas de detecção de intrusão são notórios na identificação de invasões e/ou geração de alertas. Para isso, um fator determinante é obter informações confiáveis sobre atividades maliciosas em *hosts* ou redes. Entretanto, coletar dados confiáveis é uma atividade complexa [Kemmerer and Vigna 2002] por isso algumas abordagens de detecção de intrusão têm focado em colaboração [Locasto et al. 2005]. [Apel et al. 2009, Flegel et al. 2010] propõem a troca de informação entre organizações como forma de prevenção e detecção antecipada de ameaças à segurança. [Grobauer et al. 2006] apresentam uma infraestrutura e um arcabouço organizacional para compartilhar, correlacionar e analisar cooperativamente dados e alertas de diferentes organizações. Todas as abordagens apresentadas são direcionadas à colaboração entre organizações, não tratando informações e as ameaças a usuários finais. Também dependem de um contrato prévio entre as organizações para a geração de alertas.

A nossa abordagem explora técnicas fundamentadas por trabalhos de recuperação de informação, mas possui adaptações para explorar notificações de segurança. O conceito de colaboração e disseminação nas redes sociais é explorado para obter alertas antecipados em detrimento às abordagens clássicas de colaboração e propagação de alertas.

3. Metodologia Proposta

Nesta pesquisa propomos um método para extrair e evidenciar alertas a respeito de problemas de segurança em redes de computadores por meio de mensagens postadas em redes sociais. A ideia fundamental é explorar a disseminação rápida de informações das redes sociais para disponibilizar notificações de segurança em tempo hábil para reagir antecipadamente a problemas de segurança de computadores. Objetivamos também amenizar o problema de falta de informação sobre problemas de segurança recentes, tal como vulnerabilidades dia zero.

Nossos trabalhos anteriores [Santos et al. 2012, Campiolo et al. 2013] comprovaram que há mensagens sobre ameaças computacionais postadas no Twitter. Entretanto, foram identificados alguns desafios para usar efetivamente a informação de mensagens do Twitter: (a) muitas mensagens de segurança; (b) mensagens similares; (c) diferentes níveis de importância como notificação de segurança; e (d) mensagens irrelevantes para segurança computacional.

Os itens (a) e (b) implicam na dificuldade de processar e ler todas as mensagens com conteúdo relevante, pois é oneroso lidar com essa atividade sem auxílio de ferramentas específicas e alto poder computacional. Os itens (c) e (d) implicam na existência de mensagens relacionadas à segurança porém fora do contexto de segurança em redes de computadores, como por exemplo, propagandas de ferramentas de segurança, mensagens

informando a respeito de guerra entre povos, mensagens mal formadas e que não trazem informação suficiente para caracterizar alertas, mensagens contendo boatos sobre ataques e novas ameaças, e mensagens capturadas erroneamente como se fossem de segurança de computadores, por possuírem alguns termos relacionados.

Neste contexto, definimos duas questões de pesquisa para extrair efetivamente notificações de segurança postadas em redes sociais:

Q1 É possível minimizar o impacto negativo das mensagens que não são notificações de segurança computacional?

Q2 Dentre os inúmeros problemas de segurança postados em redes sociais, é possível evidenciar quais são os problemas com maior relevância e que estão sendo mais comentados neste tipo de ambiente?

Para extrair os alertas de segurança das redes sociais e responder as questões citadas anteriormente propomos a arquitetura ilustrada na Figura 1.

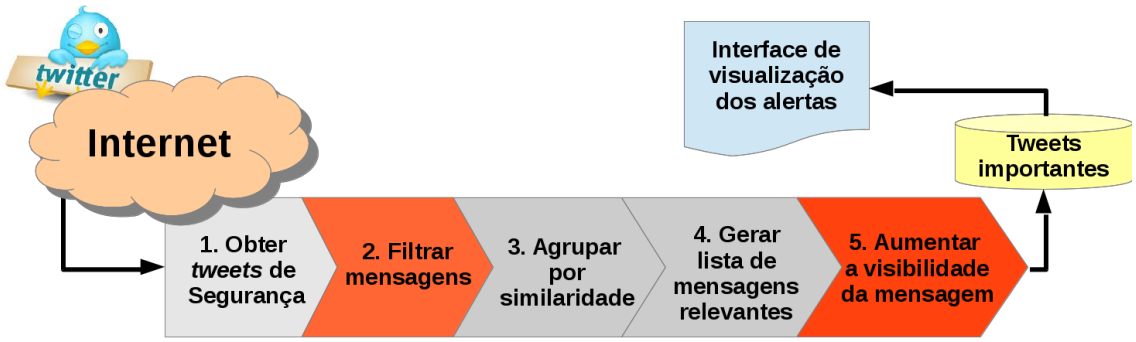


Figura 1. Método para extrair mensagens de segurança do Twitter.

O método proposto na Figura 1 é composto de cinco etapas:

1. Obter *tweets* de segurança: capturar mensagens de segurança postadas no Twitter.
2. Filtrar mensagens: remover os *tweets* que não representam mensagens de segurança da base de dados de *tweets* capturados.
3. Agrupar por similaridade: relacionar *tweets* que relatam um mesmo assunto.
4. Gerar lista de mensagens mais relevantes: selecionar mensagens que informam sobre os principais problemas de segurança comentados nas redes sociais.
5. Aumentar a visibilidade da mensagem: aumentar a importância das mensagens de segurança segundo as características do grupo de *tweets*.

As etapas do método são detalhadas nas subseções seguintes.

3.1. Obter *tweets* de segurança

A coleta de mensagens do Twitter foi realizada utilizando um software desenvolvido pelos autores a partir de uma API (*Application Programming Interface*) do Twitter[1]. Pelo fato das consultas no Twitter serem restritas e não devolverem *tweets* de mais do que algumas semanas anterior à data corrente, optou-se por realizar consultas em intervalos periódicos

[1]foi utilizada a API Twitter4j - http://twitter4j.org

de 1 minuto. O software implementado realiza a busca, descarta os *tweets* repetidos e armazena os novos *tweets* em uma base de dados.

As buscas por *tweets* de segurança foram realizadas por meio de uma consulta com os termos: *security*, *virus*, *worm*, *attack*, *intrusion*, *invasion*, *ddos*, *hacker*, *cracker*, *exploit* e *malware*. Basicamente os termos de busca obrigam que todos os *tweets* capturados tenham um ou mais termos comuns a área de segurança em redes de computadores.

A coleta por *tweets* de segurança foi realizada entre 28/04/2012 e 05/12/2012, ou seja, 222 dias. No processo de coleta foram obtidos 155.631 *tweets*, que foram postados por 74.809 usuários. Aproximadamente 84,9% dos *tweets* continham URLs, 37,0% continham *hashtags* (#) e 43% mencionavam (@) outros usuários no corpo da mensagem.

3.2. Filtrar mensagens

As mensagens coletadas na etapa anterior precisam ser analisadas e filtradas para não gerarem altos índices de falsos positivos. A Figura 2 apresenta o fluxograma do processo de filtragem.

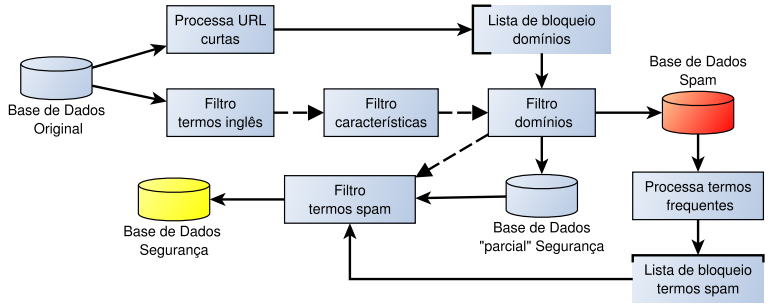


Figura 2. Fluxograma do processo de filtragem.

O *Filtro termos inglês* remove mensagens que não são da língua inglesa. Pois, mesmo utilizando termos de busca em inglês, foram coletadas muitas mensagens em outras línguas. Como critério de filtragem, adotamos a política de remover da base todos os *tweets* com o código de linguagem diferente do inglês.

O *Filtro características* remove mensagens com as seguintes características: menos de quatro palavras, tamanho inferior a quarenta caracteres, mais de três URLs, menção a usuários e número de hashtags superior a metade do número de termos na mensagem. Essas características são importantes para remover mensagens mal formadas.

O *Filtro domínios* remove as mensagens irrelevantes como alertas considerando a URL presente nas mensagens dos *tweets*. Aproximadamente, 84,9% dos *tweets* coletados possuem URL, pois *tweets* de notificação geralmente contém referência para uma descrição mais detalhada. Para esta etapa, as mensagens com a mesma URL curta são agrupadas e, em seguida, submetidas a serviços de tradução para URLs longas. Para toda URL longa, foi extraído o endereço do domínio. Os domínios com mais citações nas mensagens foram analisados e uma lista de bloqueio com 160 domínios foi gerada. A lista de bloqueio continha endereços de sítios que geravam notícias de segurança - mas que não são de segurança de redes de computadores, como por exemplo, cnn.com, washingtonpost.com, amazon.com, forbes.com.

Após aplicar o *Filtro domínios*, foram geradas duas bases: base de dados parcial "segurança" e a base de dados "spams". Processando os termos mais frequentes na base de *spams*, foi gerada uma lista de palavras comuns em mensagens irrelevantes. Os termos foram selecionados manualmente. São exemplos de termos: *lybia*, *benghazi*, *obama*, *bomb*, *iran*, *officials*, *consulate* e *killed*. O *Filtro termos spam* remove da base de dados "segurança" os *tweets* com os termos na *Lista de bloqueio termos spam*. Como resultado, temos a base de dados "Segurança" contendo potenciais alertas.

3.3. Agrupar por similaridade

Nesta etapa temos em sua maioria mensagens relacionadas à segurança de redes de computadores. Essas mensagens são submetidas ao Apache Lucene[2], uma biblioteca de indexação e busca de texto. Ao submetermos uma consulta ao motor de busca do Lucene, são devolvidos os textos/documentos encontrados seguidos por um escore. Escores elevados representam alto grau de similaridade e o escores baixos representam pouca similaridade dos documentos em relação à busca. O Apache Lucene emprega o cálculo de similaridade de cosseno simplificado e variáveis ajustáveis para calcular o escore, tal como, atribuir pesos para termos utilizados na consulta, amenizar parágrafos repetidos, entre outros.

A estratégia de agrupamento consiste em comparar cada *tweet* na base com todos os outros, ou seja, o termo de busca é a mensagem de cada *tweet*. Dessa forma, a partir de um determinado grau de similaridade é possível considerar que os *tweets* abordem um mesmo assunto ou similar [Manning et al. 2008].

No processo de agrupamento do Lucene temos dois fatores muito importantes: grau de similaridade e o número de *tweets* similares. O grau de similaridade é um número que indica o quão similar é a mensagem de um *tweet* em relação a outro, esse número é calculado pelo Apache Lucene. Usamos o grau de similaridade para definir se os *tweets* abordam um mesmo assunto, por exemplo, caso a comparação entre dois *tweets* retorne um grau de similaridade 0,5 ou superior pode-se considerar que esses *tweets* são tão parecidos que podem ser *retweets* ou que relatam um mesmo assunto. Em nossa metodologia o grau de similaridade indica o nível de agrupamento que desejamos. Se ajustado para um valor alto tende a produzir mais grupos e aumenta a probabilidade de grupos distintos abordando o mesmo assunto. Se ajustado para um valor baixo tende a produzir menos grupos e aumenta a probabilidade de mensagens de contextos diferentes no mesmo grupo.

Inicialmente utilizávamos um grau de similaridade fixo, normalmente 0,5 ou 0,75, mas depois de experimentos preliminares, chegou-se a conclusão que usar um grau de similaridade fixo não produzia bons resultados devido ao tamanho variável do texto dos *tweets*. O problema é que quando submetemos uma mensagem pequena - menos que 70 caracteres, por exemplo - há uma grande chance dessa combinar com algumas palavras de outras mensagens maiores, assim o *tweet* pequeno parece falsamente ter um alto grau de similaridade com a mensagem maior e por consequência é dado como importante, sendo que na maioria das vezes não é importante. Para resolver este problema calculamos o grau de similaridade pela Equação 1, que permite que o valor seja calculado em função do tamanho do *tweet*. Na Equação 1 tem-se:

[2]http://lucene.apache.org

- δ representa o grau de similaridade mínima aplicada a uma mensagem de tamanho máximo. O tamanho máximo de uma mensagem é 160 e não 140 que seria o padrão. O δ pode ser por exemplo 0,75;
- x é o número de caracteres da mensagem que está sendo analisada;
- α é usado como um fator extra de crescimento para o grau de similaridade, que pode ser por exemplo 2 ou 8.

$$GrauSimilaridadeExigido = ((\delta - \frac{x * \delta}{160}) * \alpha) + \delta \quad (1)$$

Quanto menor o texto submetido para consulta, mais rígidas são as regras de agrupamento aplicadas pela Equação 1, pois o grau de similaridade exigido aumenta, o que valoriza a importância da informação e evita que assuntos de contextos diferentes sejam agrupados.

3.4. Gerar lista de *tweets* mais relevantes

A etapa de agrupamento por similaridade produz uma lista que contém o número de *tweets* similares em cada grupo e uma mensagem que representa esse grupo. A mensagem que representa o grupo é o texto do *tweet* que gerou a busca no Apache Lucene. Devemos apontar dentre esses grupos quais tem mensagens que são relevantes.

Para determinar se um grupo dessa lista é importante utilizamos uma premissa básica entre os usuários do Twitter, que é: uma mensagem importante tem a tendência de ser retransmitida entre os usuários do Twitter, ou seja, sofre um *retweet* [Morris et al. 2012]. Mas não olhamos somente para os *retweets* e sim para mensagens similares. Então, usamos o fator número de *tweets* similares para determinar se um grupo é importante ou não. Desta forma, quanto maior o número de *tweets* similares mais importante é a informação.

Um assunto postado no Twitter foi considerado relevante caso gerasse um grupo com 10 ou mais elementos, isso significa que pelo menos 10 pessoas acharam a informação relevante e decidiram repassá-la para seus seguidores, sem considerar as pessoas que podem ter lido a mensagem sem ser seguidor.

Após alguns experimentos preliminares, notamos que normalmente mensagens de *spam* continham um grande número de *tweets* similares e comprometiam os resultados na busca por notificações de segurança. Entretanto, também notamos que essas mensagens de *spam* vinham de alguns poucos usuários do Twitter, então passamos a considerar um grupo de mensagem importante não pelo número de mensagens que este continha, mas sim pelo número de usuários que postaram essas mensagens.

Ao final, temos uma lista com as mensagens mais relevantes. O Algoritmo 1 apresenta o pseudocódigo das etapas apresentadas nas Subseções 3.3 e 3.4.

3.5. Aumentar a visibilidade das mensagens

Durante o processo de agrupamento de mensagens vários *tweets* são descartados por não serem similares a nenhum outro. Este processo ajuda a determinar quais mensagens são mais relevantes. Porém, mesmo com o processo de filtragem podemos ter, ainda, uma lista com centenas ou milhares de mensagens consideradas relevantes o que pode inviabilizar a

Algoritmo 1: Agrupamento por similaridade

Entrada: Um vetor de *tweets* - $todosTweets$
Saída: Mensagens consideradas importantes por serem postadas por mais de 10 usuários

```
1  ApacheLucene.indexa(todosTweets)
2  α ← 2
3  δ ← 0,75
4  enquanto todosTweets ≠ φ faça
5      tweet ← todosTweets.proximo()
6      x ← tweet.textoMensagem.tamanho()
7      grauSimilaridadeExigido ← ((δ − x∗δ/160) ∗ α) + δ
8      tweetsLucene ← ApacheLucene.obterTweetsSimilares(tweet.textoMensagem)
9      para cada tweetsLucene.proximo() faça
10         se tweetsLucene.score ≥ grauSimilaridadeExigido então
11             tweetsSimilares.adiciona(tweetsLucene.tweet)
12     numeroUsuarios ← removerTweetsComUsuariosRepetidos(tweetsSimilares).tamanho()
13     se numeroUsuarios ≥ 10 então
14         mensagensImportantes.adiciona(numeroUsuarios,tweet.textoMensagem)
15         todosTweets.remover(tweetsSimilares)
16 retorna mensagensImportantes
```

leitura de todas essas mensagens. É necessário então algum mecanismo que aponte quais mensagens são mais importantes dentre as mensagens selecionadas anteriormente.

Existem inúmeras técnicas utilizadas para indicar a importância ou até mesmo a credibilidade de mensagens do Twitter [Lee 2012]. Essas técnicas vão desde análise de palavras contidas no *tweet* até a investigação do usuário que postou a mensagem.

Em nosso trabalho, o fator primordial para dizer se uma mensagem é importante ou não, é o número de usuários que postaram essa mensagem. Então, esse número é o ponto de partida para decidir a importância da mensagem. Nesta etapa incrementamos ou decrementamos a importância de uma mensagem por meio da análise de algumas peculiaridades da mensagem e do grupo de *tweets* ao qual essa mensagem pertence.

Um método muito utilizado para aumentar a importância de uma mensagem é a análise da frequência das palavras da mensagem. Podemos analisar a mensagem palavra por palavra e verificar se o uso de uma palavra aumentou de um dia para outro. Assim, caso exista uma palavra que apresente um grande aumento na frequência de seu uso, podemos considerar que foi muito comentada na data em que o *tweet* foi postado e pode ter um conteúdo importante. O conceito é muito parecido com a técnica de *trend topics* do Twitter.

Outra técnica é observar a ocorrência de palavras raras nas mensagens dos grupos. Caso exista uma palavra rara há uma grande chance desse *tweet* estar abordando um assunto novo e portanto importante. Para decidir se uma palavra é rara criamos uma lista formada pela tupla, palavra e número de ocorrências em todos *tweets*. Assim, se uma mensagem apresentar uma palavra que não ocorreu ou quase não apareceu na lista ela é considerada rara.

Durante nossos experimentos observamos que *tweets* que apresentam várias palavras raras, são em sua grande maioria, mensagens fora do contexto de segurança de computadores. Assim, caso a mensagem apresente uma ou duas palavras raras nós aumentamos a importância dessa mensagem. Se a mensagem apresentar mais de três pala-

vras raras diminuímos sua importância. O mesmo ocorre para o aumento da frequência do uso de palavras, portanto o mesmo procedimento foi aplicado.

Tweets que relatam assuntos da atualidade, tal como a ocorrência de uma catástrofe, possuem o comportamento de registrar mensagens sobre o assunto no primeiro dia, ter uma explosão no número de *tweets* no segundo dia e diminuir a sua intensidade gradativamente nos dias posteriores. Assim, analisamos o comportamento das mensagens de um grupo e se essas apresentarem esse comportamento de propagação aumentamos a sua importância. Também aumentamos a importância de um grupo se ele apresentar um dia de pico muito elevado, ou seja, se o número de *tweets* crescer muito de um dia para o outro e depois diminuir. Esses dois fatores são importantes, pois por exemplo *spams*, não apresentam dia de pico e a sua propagação é constante. Em trabalhos anteriores [Campiolo et al. 2013, Santos et al. 2012] verificamos que mensagens de segurança tem uma média de propagação de aproximadamente 12 dias, já *spams* são propagados por meses, por isso também aumentamos a importância de mensagens que são disseminadas durante um período de 10 a 14 dias.

4. Análise dos Resultados

Apresenta-se nesta seção a caracterização dos resultados da filtragem e do método de extração de notificações de segurança (sem e com filtragem), bem como, a avaliação dos procedimentos e a avaliação das questões de pesquisa.

4.1. Caracterização dos resultados

A filtragem das mensagens coletadas do Twitter (Seção 3.2) foi realizada para melhorar o resultado do agrupamento de mensagens relevantes.

O procedimento de filtrar por URL resultou na seleção de sítios Web que divulgam informações de segurança, mas não são sítios especializados. Por outro lado, identificou muitos sítios e *blogs* que propagam informações de alertas. As tabelas 1 e 2 apresentam, respectivamente, exemplos de domínios descartados por não tratarem de segurança de redes de computadores e domínios considerados úteis por relatarem ameaças a ambientes de redes de computadores.

Tabela 1. Domínios descartados.

Ocorrências	Domínio
1117	washingtonpost.com
484	forbes.com
428	amazon.com
405	computerworld.com
356	pcworld.com
27	infowars.com
26	inquisitr.com

Tabela 2. Domínios considerados.

Ocorrências	Domínio
1369	net-security.org
1311	zdnet.com
1266	sophos.com
1144	thehackernews.com
917	h-online.com
98	blog.sucuri.net
34	blog.webroot.com

O conteúdo da base de mensagens de "spam" foi indexado e a frequência dos termos mais comuns dessas mensagens foi calculada usando o Lucene. Após a inspeção dos termos mais frequentes, foi gerada uma lista de palavras-chaves que indicam mensagens fora do contexto. A Tabela 3 apresenta exemplos de termos selecionados. Foram inspecionados todos os termos com mais de 10 ocorrências.

Tabela 3. Exemplos de termos comuns em mensagens irrelevantes

Ocorrências	Termos
4164	libya
3304	benghazi
2291	obama
1848	iran
984	killed
708	nuclear
435	ambassador
429	dead
422	terrorism
345	free

Tabela 4. Exemplos de termos comuns em mensagens relevantes

Ocorrências	Termo
7334	exploit
3233	ddos
1919	zero(day)
1850	vulnerab(ility)
866	trojan
856	bots ou botnet
850	bug
628	hijack
558	inject
282	backdoor

A partir dos termos que indicam potencias mensagens a serem descartadas, foi realizada uma filtragem na base de dados parcial "segurança" para remover *tweets* que apresentem esses termos. O filtro por palavras-chaves continha 120 termos.

O conteúdo da base de dados "segurança" filtrado foi indexado e a lista de termos mais frequentes foi calculada. Um conjunto de 300 palavras-chaves que caracterizam mensagens de segurança computacional foi selecionado baseado no número de ocorrências. O critério adotado durante a inspeção foi a seleção de termos que são específicos a área de segurança em redes de computadores. Dessa forma, termos como *security* e *attack* foram ignorados. A Tabela 4 apresenta exemplos de termos selecionados.

A Tabela 5 apresenta os resultados com o uso do Filtro de aumento de visibilidade de mensagens. Também aplicamos métodos para aumentar a importância da mensagem, identificada pelo campo *Brst*, que é o número de usuários que postaram a mensagem acrescido ou decrescido segundo as técnicas mencionadas na subseção 3.5.

Todas as mensagens apresentadas na Tabela 5 tiveram sua importância aumentada já que representam mensagens de segurança, conforme os valores apresentados pelos campos *Usu* e *Brst*. Ao comparar as mensagens da Tabela 5 com mensagens recuperadas sem o filtro de aumento de visibilidade notamos que todas as mensagens da Tabela 5 informam sobre problemas de segurança em rede de computadores, enquanto na lista de mensagens sem filtragem haviam 3 mensagens não relacionadas com segurança em redes de computadores (a lista de mensagens sem filtragem não é apresentada por falta de espaço).

4.2. Avaliação dos Procedimentos

Duas etapas importantes para a extração efetiva de notificações foram as de filtro de mensagem e aumento de relevância das mensagens. Os experimentos foram realizados em uma base de dados estática de 157 MiB, contendo mensagens coletadas durante 7 meses. Para a avaliação dos procedimentos de filtragem consideramos na análise os impactos na base de dados.

O *Filtro termos inglês* reduziu a base de 157 MiB para 144,4 MiB, ou seja, uma redução de aproximadamente 8%. Em uma amostra de 1.000 *tweets* na base reduzida, foi encontrado apenas uma mensagem em francês, mas que possuía alguns termos em inglês. Foram encontrados apenas 3 *tweets* em inglês em uma amostra de 100 *tweets*

Tabela 5. Lista de *tweets* relevantes com *Filtro de spam* e aumento de importância

Mês	Msgs	Usu	Brst	Texto da mensagem
Mai	88	85	116	Microsoft boots Chinese firm for leaking Windows exploit..http://t.co/0QMqcQAl
	78	78	97	Android has made malware for Linux a reality. http://t.co/Qi5PxGcM
	72	70	88	...The Pirate Bay returns Anonymous hater takes credit for DDoS http://t.co/FQ0OrdTj
Jun	318	312	474	...How Flame has changed everything for online security firms...
	271	260	360	Scientists crack RSA SecurID 800 tokens steal cryptographic keys...
	108	106	133	...Thousands of office printers hit by ”gibberish”malware http://t.co/GgzSIFcx
Jul	151	144	217	Serial hacker says latest Android will be ”pretty hard”to exploit...
	147	141	195	More malware found hosted in Google’s official Android market...
	155	150	189	Security firm: Android malware pandemic by year?s end...
Ago	156	141	195	...30K workstations fell victim to cyber attack http://t.co/5dcvvKyS
	119	119	163	Attack against Microsoft scheme puts hundreds of crypto apps at risk...
	118	118	161	...MS-CHAPv2 puts hundreds of crypto apps at risk http://t.co/TQPsRkpN
Set	89	86	117	Android security suffers as malware explodes by 700%...
	80	79	117	GoDaddy attack likely a psyop to discredit Anonymous while pushing cyber security...
	84	81	111	...Android security sucks hard as malware explodes...
Out	208	194	243	Hacker Steals Millions of...Social Security Numbers...
	167	166	208	DSL modem hack used to infect millions with banking fraud malware...
	107	107	147	...Hacker Scores $60 000 From Google For Discovering Security Issue In Chrome...
Nov	133	133	183	...Mac OS X has more security holes than windows and they are easier to exploit too.
	98	95	130	New Linux rootkit injects mal HTML into Web servers...
	97	94	129	New Linux rookit injects malicious HTML into Web servers via ars technica...

removidos. Os 3% de *tweets* removidos não prejudica os resultados, pois as mensagens muito importantes são postadas por centenas de usuários.

Os demais filtros reduziram o tamanho da base da seguinte forma: *características*: removeu 292 mensagens irrelevantes, *domínios*: reduziu a base em 29,2%, *termos spam*: reduziu a base em 29,7%.

Concluímos duas importantes observações por meio das medições. Primeiro, a análise do comportamento de propagação da mensagem, número de dias que a mensagem é propagada e dia de pico das mensagens ajudam a identificar mensagens que não são *spam*. A análise da frequência de palavras e de palavras raras na fase de aumento de relevância identificou mensagens importantes e fora de contexto. Segundo, por meio de listas de termos específicos para aumentar ou diminuir a importância de uma mensagem, conseguimos filtrar mensagens relevantes e irrelevantes como alertas. Em um experimento, selecionamos 10 mensagens na base “segurança” e aplicamos +1 para termos de segurança e -1 para termos *spam*. Como resultado, verificamos que mensagens com poucas citações, mas interessantes como alertas, acabaram sendo notadas, enquanto mensagens de segurança, mas pouco importantes, perderam notoriedade.

4.3. Avaliação das Questões de Pesquisa

As questões de pesquisa foram avaliadas considerando somente os meses de junho e setembro. Os meses foram selecionados aleatoriamente. Não apresentamos informações detalhadas sobre os outros meses por limite de espaço.

Para responder a questão Q1, o gráfico apresentado na Figura 3 ilustra os resultados do método de extração de notificações nos meses de junho e setembro. Observa-se que o uso de filtro reduziu consideravelmente o número de mensagens fora do contexto de segurança de redes de computadores, pois em junho houve queda de 18% para 6% e em setembro de 38% para 6%. Logo, podemos afirmar que é possível eliminar ou pelo

menos suavizar o impacto das mensagens que não são notificações sobre problemas em redes de computadores.

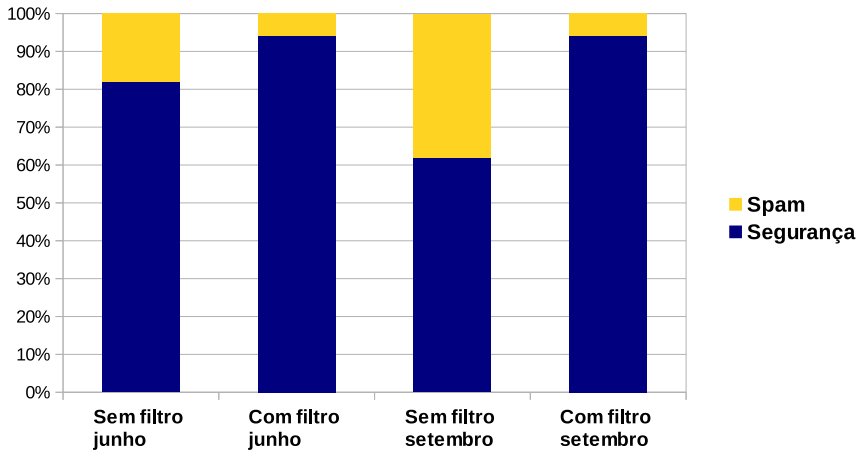


Figura 3. Caracterização das mensagens do mês de junho e setembro após aplicação de filtros.

Para responder a questão Q2, analisamos manualmente as mensagens devolvidas como importantes nos meses de junho e setembro. Das 134 mensagens extraídas como mensagens de segurança importantes deste mês, 65,7%, são alertas reais de segurança para ambientes de redes de computadores. Já o mês de setembro apresentou 56,9% de alertas, dentre as mensagens de segurança consideradas importantes. A Tabela 6 apresenta três mensagens de alerta de cada mês analisado.

Tabela 6. Exemplo de alertas de junho e setembro

Mês	Msgs	Usu	Brst	Texto da mensagem
Jun	108	106	133	RT @ZDNet: Thousands of office printers hit by ”gibberish”malware http://t.co/GgzSIFcx
	39	38	47	A virus specialized for AutoCAD a perfect cyber espionage tool — The Hacker News http://t.co/UhaHeXGn #THN #Security
	29	29	35	ALERT - A 0day exploit taking advantage of an unpatched Windows / Office vuln is being exploited in the wild http://t.co/BovMm9Nd
Set	91	89	111	Emergency security patch issued by Microsoft to squash Internet Explorer zero day #Exploit... http://t.co/D1FXvkO7 @SecurityPhresh
	73	70	88	#hackernews CRIME : New SSL/TLS attack for Hijacking HTTPS Sessions: Two security researchers claim to have deve... http://t.co/J0IOy0Hj
	35	34	50	Attack Easily Cracks Oracle Database Passwords: Oracle?s software update for the flaw doesn?t protect http://t.co/o1ibSeuw #infosec

Os alertas apresentados da Tabela 6 demonstram o potencial das redes sociais quanto a propagação de alertas. Por exemplo, a primeira mensagem do mês de junho alerta sobre um *malware* que afeta impressoras, através desse alerta o administrador pode tomar medidas, tal como, a atualização do antivírus e regras de IDS (*Intrusion Detection System*) para tentar impedir este problema. A segunda mensagem de junho alerta sobre um vírus para o software AutoCAD, tal vírus envia via e-mail projetos desenvolvidos no software, sabendo desse problema um administrador de rede pode implementar regras de *firewall*, *proxy* e/ou ferramentas anti-*spam* que impeçam o envio desses arquivos, ou simplesmente verificar se o antivírus consegue eliminar esse problema. Em casos mais

radicais os administradores podem retirar o acesso à Internet dessas máquinas, já que este vírus visa espionagem industrial.

5. Conclusões

O uso de mecanismos de filtragem, agrupamento e aumento de notoriedade de mensagens de segurança postadas no Twitter permitiram elaborar um método capaz de extrair alertas de segurança de mensagens postadas em redes sociais. O método foi eficiente em trazer apenas notícias associadas à segurança de redes. Os resultados produzidos por filtros e agrupamentos mostraram a dificuldade em lidar com a semântica de mensagens postadas em *microblogs*, mesmo quando essas abordam em sua maioria assuntos de segurança. Em trabalhos futuros pretendemos utilizar técnicas de aprendizagem supervisionada para melhorar a classificação das mensagens. Com a otimização do método, pretendemos também implementar um software para coleta e distribuição de alertas postados em redes sociais.

Referências

Aggarwal, C. and Subbian, K. (2012). Event detection in social streams. In *Proceedings of the Twelfth SIAM International Conference on Data Mining*, pages 624–635.

Akyazi, U. and Uyar, A. c. (2010). Detection of ddos attacks via an artificial immune system-inspired multiobjective evolutionary algorithm. In *Proceedings of the EvoCOMNET'10*, pages 1–10. Springer-Verlag.

Apel, M., Biskup, J., Flegel, U., and Meier, M. (2009). Towards early warning systems - challenges, technologies and architecture. In *CRITIS*, pages 151–164.

Bonchi, F., Castillo, C., Gionis, A., and Jaimes, A. (2011). Social network analysis and mining for business applications. *ACM Trans. Intell. Syst. Technol.*, 2(3):22:1–22:37.

Brown, G., Howe, T., Ihbe, M., Prakash, A., and Borders, K. (2008). Social networks and context-aware spam. In *Proceedings of the ACM CSCW '08*, pages 403–412.

Campiolo, R., Santos, L. A. F., Gerosa, M. A., and Batista, D. M. (2013). Evaluating the utilization of twitter messages as a source of security alerts. In *Proceedings SAC 2013*. ACM. To be published.

Cha, M., Benevenuto, F., Haddadi, H., and Gummadi, K. (2012). The world of connections and information flow in twitter. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, PP(99):1 –8.

Flegel, U., Hoffmann, J., and Meier, M. (2010). Cooperation enablement for centralistic early warning systems. In *Proceedings of the ACM SAC '10*, pages 2001–2008.

Frei, S., May, M., Fiedler, U., and Plattner, B. (2006). Large-scale vulnerability analysis. In *Proceedings of SIGCOMM Workshop on LSAD '06*, pages 131–138.

Gao, H., Hu, J., Wilson, C., Li, Z., Chen, Y., and Zhao, B. Y. (2010). Detecting and characterizing social spam campaigns. In *Proceedings of the IMC '10*, pages 35–47.

Grier, C., Thomas, K., Paxson, V., and Zhang, M. (2010). @spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM CCS*, pages 27–37.

Grobauer, B., Mehlau, J. I., and Sander, J. (2006). Carmentis: A co-operative approach towards situation awareness and early warning for the internet. In *IMF*, pages 55–66.

Kemmerer, R. and Vigna, G. (2002). Intrusion detection: a brief history and overview. *Computer*, 35(4):27 –30.

Kwak, H., Lee, C., Park, H., and Moon, S. (2010). What is twitter, a social network or a news media? In *Proceedings of the 19th WWW*, pages 591–600. ACM.

Lee, C.-H. (2012). Mining spatio-temporal information on microblogging streams using a density-based online clustering method. *Expert Systems with Applications: An International Journal*, 39(10):9623–9641.

Lerman, K. and Ghosh, R. (2010). Information contagion: An empirical study of the spread of news on digg and twitter social networks. In *Proceedings of 4th ICWSM*.

Locasto, M., Parekh, J., Keromytis, A., and Stolfo, S. (2005). Towards collaborative security and p2p intrusion detection. In *Proceedings of IEEE SMC*, pages 333 – 339.

Luo, W., Liu, J., Liu, J., and Fan, C. (2009). An analysis of security in social networks. In *Proceedings of the IEEE DASC '09*, pages 648–651.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

Mathioudakis, M. and Koudas, N. (2010). Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD*, pages 1155–1158.

Morris, M. R., Counts, S., Roseway, A., Hoff, A., and Schwarz, J. (2012). Tweeting is believing?: understanding microblog credibility perceptions. In *Proceedings of the ACM CSCW*, pages 441–450.

Petrović, S., Osborne, M., and Lavrenko, V. (2010). Streaming first story detection with application to twitter. In *Proceedings of the HLT '10.*, pages 181–189. ACL.

Phuvipadawat, S. and Murata, T. (2010). Breaking news detection and tracking in twitter. In *Proc. of the IEEE/WIC/ACM WI-IAT*, volume 3, pages 120 –123.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th WWW*, pages 851–860. ACM.

Santos, L. A. F., Campiolo, R., Gerosa, M. A., and Batista, D. M. (2012). Analysis of security messages posted on twitter. In *Proceedings of Brazilian Symposium on Collaborative Systems*, pages 20–28. IEEE.

Sayyadi, H., Hurst, M., and Maykov, A. (2009). Event Detection and Tracking in Social Streams. In *Proceedings of the ICWSM'09*. AAAI.

Ye, S. and Wu, S. F. (2010). Measuring message propagation and social influence on twitter.com. In *Proceedings of the Second SocInfo*, pages 216–231. Springer-Verlag.

Yuan, S. and Zou, C. (2011). The security operations center based on correlation analysis. In *Proceedings of the 2011 IEEE ICCSN*, pages 334 –337.

Gerenciamento de Identidades Tolerante a Intrusões

Luciano Barreto[1], Frank Siqueira[1], Joni da Silva Fraga[1], Eduardo Feitosa[2]

[1]Universidade Federal de Santa Catarina
Caixa Postal 476 – 88040-900 – Florianópolis – Santa Catarina – Brasil

[2]Instituto de Computação – Universidade Federal do Amazonas

{lucianobarreto, fraga}@das.ufsc.br, frank@inf.ufsc.br,
efeitosa@icomp.ufam.edu.br

Resumo. *O gerenciamento de identidades é um ponto central na segurança de grandes aplicações e sistemas distribuídos. Os provedores de identidade (IdPs) são elementos que concentram informações críticas de usuários. Estas informações são armazenadas com cuidados especiais nestes provedores e invasões não necessariamente resultam em violações de segurança. Mas, intrusões com implantes de comportamentos maliciosos podem modificar a ação destes elementos de autenticação. Indivíduos não autorizados podem ser aceitos no sistema e usuários legítimos podem ter seus acessos negados. Neste artigo introduzimos uma abordagem de tolerância a intrusões que visa garantir o comportamento correto na autenticação em grandes sistemas.*

Abstract. *Identity management is a central point to the security of large applications and distributed systems. The identity providers are elements that concentrate critical information of users. These information are stored with special care in these providers and intrusions do not necessarily result in security violations. But intrusions may implant malicious behaviors which modify the action of these providers. In this way, unauthorized accesses may be accepted into the system and legitimate users may have their accesses denied. In this paper we introduce an approach to intrusion tolerance for ensuring the correct behavior in authentications of large systems.*

1. Introdução

Em sistemas distribuídos, a aplicação de políticas e a segurança de serviços do sistema dependem, principalmente, dos controles de autenticação e autorização. A abordagem convencional empregada na concretização destes controles é baseada na implementação dos mesmos sobre provedores de serviços (SPs). O usuário deve se autenticar junto a um serviço e fica sujeito à política de autorização do mesmo. A complexidade das aplicações e sistemas distribuídos atuais torna este modelo limitado. Vários modelos e infraestruturas nos últimos anos têm sido introduzidos, onde estes controles tomam forma a partir de um conceito mais amplo que é o de gerenciamento de identidades.

No caso de sistemas distribuídos de larga escala como *clouds*, grades e redes colaborativas, o gerenciamento de identidades deve se basear em uma infraestrutura que integre políticas e tecnologias, permitindo às diferentes organizações que participam destes sistemas e aplicações ultrapassar seus domínios locais (domínios administrativos ou de políticas). Diante disto, usuários e aplicações podem ter acesso, de maneira segura, a recursos remotos, desde que possuam as credenciais exigidas pelas políticas

destes recursos. A infraestrutura necessária para o gerenciamento de identidades envolve muitas vezes um sistema de autenticação (usualmente em sistemas de larga escala corresponde a uma rede de autoridades de autenticação) e um sistema de gerenciamento de atributos (serviço que fornece informações adicionais sobre usuários). Estes atributos, individual ou coletivamente, podem identificar o usuário (autenticação) e fornecer as informações necessárias (atributos do usuário) para a realização dos controles de autorização (controle baseado em atributos) permitindo assim, a execução das operações solicitadas. Em muitas propostas e infraestruturas, as autoridades de autenticação também concentram o gerenciamento de atributos e são identificados como provedores de identidades (*Identity Providers - IdP*).

Com isto, em vários modelos de gerenciamento de identidades (dentre os quais destacamos o centralizado, identidades federadas e *user-centric* [Jøsang et al. 2005]), a autenticação não é mais realizada nos SPs, mas sim em autoridades de autenticação (ou *IdPs*). A autorização continua sendo concretizada nos SPs porque estes conhecem seus recursos e políticas (de aplicação) e devem zelar pelos mesmos. Nesta situação, usuários e SPs não mais precisam manter sob sigilo listas muitas vezes enormes de credenciais (*senhas*, certificados, atributos de usuário, etc.). Estes provedores de identidades (autoridades de autenticação) passam a ser os pontos centrais na aplicação de políticas de segurança em sistemas distribuídos. Mas, também por serem serviços que ficam disponíveis via Internet, estes *IdPs* estão sujeitos a ataques que podem resultar em intrusões, o que seria catastrófico para a segurança das informações e recursos do sistema.

O trabalho que apresentamos neste texto é parte do projeto *SecFuNet*[1], que se propõe a desenvolver, dentre vários serviços, um *framework* de segurança para *clouds*, introduzindo funções de autenticação e autorização para este tipo de ambiente. Neste sentido, apresentamos neste artigo a proposta de um modelo para tornar os *IdPs* do *SecFuNet* tolerantes a intrusão. O gerenciamento de identidades do *SecFuNet* é fortemente dependente de componentes de *hardware* protegido. Os *logins* de usuários são feitos diretamente entre s*martcards* (de posse do usuário) e processadores seguros dispostos em um servidor de identidades (servidor de autenticação). No gerenciamento de identidades do *SecFuNet*, as relações de confiança entre *browsers* (usuários), *IdPs* e *SPs* fazem uso dos protocolos do *framework OpenID* [OpenID 2007].

Embora sejam empregados componentes seguros e protocolos que não deixam as informações de usuários disponíveis em um *IdP* invadido, estes cuidados não impedem que *IdPs* invadidos possam ter sido alvo de "implantações" de comportamentos maliciosos. Atuando segundo estes comportamentos, um *IdP* malicioso pode certificar indivíduos (intrusos) não autorizados por políticas, assim como também pode não autenticar usuários reconhecidos pelas mesmas políticas. As soluções para tolerância a intrusões ou a faltas bizantinas BFT ("*Byzantine Fault Tolerance"*) são normalmente baseadas em replicação Máquina de Estados (ME) [Schneider 1990]. Vários trabalhos presentes na literatura apresentam soluções práticas para tolerância a faltas maliciosas (intrusões), sendo os mais conhecidos o *PBFT* [Castro and Liskov 2002] e o *Zyzziva* [Kotla et al. 2007]. Estes trabalhos assumem que em suas replicações, enquanto os limites de faltas maliciosas (definido como *f*) não forem atingidos, a segurança do sistema é garantida. Um dos problemas destas abordagens está no fato das mesmas usarem grande

[1] *SecFuNet Project*: "*Security for Future Networks*" Edital Brasil/Europa, MCT/CNPq número 66/2010, processo CNPq nº 590047/2011-6.

número de réplicas físicas para execução do protocolo. No *PBFT*, o número de réplicas necessárias para execução do protocolo é definido como *3f + 1*. Estes protocolos, se usados para a tolerância a intrusões de *IdPs*, devido aos altos custos em termos de mensagens, tornaria mais custoso o desempenho do procedimento de *login*.

O modelo de tolerância a intrusões que usamos neste texto é baseado na tecnologia de virtualização e não envolve replicação física o que é benéfico para o nosso sistema onde os *IdPs* fazem uso de *hardware* especializado (processadores seguros) e cuja replicação aumentaria os custos financeiros e poderiam também trazer novos problemas como o aumento da disponibilidade de informações sigilosas na rede. Nossa abordagem de replicação faz uso de memória compartilhada e é uma adaptação do modelo apresentado em [Lau et al. 2012a][Lau et al. 2012b]. As adaptações neste modelo foram produzidas para adequá-lo às características do *IdP* e aos protocolos do *framework OpenID*.

O artigo na seção 2 apresenta as características e premissas assumidas no modelo de sistema usado. Questões do esquema de tolerância a malícia são discutidas na seção 3. Na sequência, a seção 4 descreve a infraestrutura que permite a construção de provedores de identidades tolerantes a intrusões. Os aspectos de implementação de um protótipo e a apresentação de resultados de testes são focadas na seção 5. Na seção 6 são recuperados e discutidos os principais trabalhos relacionados. Por fim, na seção 7 são apresentadas as conclusões finais do artigo.

2. Caracterização do Ambiente de Autenticação

O processo de autenticação de usuários em sistemas distribuídos envolve, em um primeiro passo, um procedimento de *login* onde um usuário, através de trocas com um provedor de identidade (usando técnicas como *userid*/*password*, *userid*/*certificate*, *Challange*/*Response*, etc.), tem a sua identidade validada diante do sistema. Este procedimento, normalmente termina com o usuário recebendo do seu *IdP* um *token* ou asserção de autenticação que serão usados para atestar sua autenticidade diante de outras entidades do sistema em subsequentes interações.

Em sistemas de computação em nuvens, serviços de gerenciamento de identidades são usualmente implementados em *clouds* privadas ou em servidores com *hardware* especial. A não colocação de um *IdP* em *clouds* públicas se justifica pela criticidade das informações. O projeto *SecFuNet*, por enfatizar o uso de processadores seguros no processo de autenticação de usuários, implica em limitações de escala. Então o gerenciamento de identidades deve envolver vários *IdPs*, cada um com o seu próprio domínio de política. Estes provedores de identidades fornecem asserções de autenticação, seguindo suas políticas, para usuários e provedores de serviço (SPs) de seus domínios de política. Nestes domínios, portanto, o gerenciamento de identidades segue uma abordagem centralizada onde o usuário de posse de um cartão inteligente (*smartcard*), fornece informações que são verificadas em um processador seguro no *IdP* do seu domínio. O *IdP* desempenha o papel de uma terceira parte confiável nas interações entre usuário e SPs.

No entanto, para sistemas complexos como *clouds*, que normalmente se estendem além destes domínios locais, este modelo de intermediação simples é limitado. É necessário expandir esse modelo para um sistema de gerenciamento de

identidade com base em redes de confiança, envolvendo vários destes provedores locais de identidade. Ou seja, é necessário adotar abordagens de gerenciamento onde ou os *IdPs* de diversos domínio possuem relações de confiança entre si (abordagem de "identidades federadas"), ou os *SPs* possuem listas de *IdPs* em quem confiam (abordagem centrada em *SPs*). Para a aplicação destas abordagens citadas, é necessário estender a escala das relações de confiança e isto é feito com o uso de Infraestruturas de Autenticação e Autorização. Exemplos notórios destas infraestruturas são o *Shibboleth* [Lewis 2008] e o *Liberty Alliance* [Liberty 2003] para abordagens de identidades federadas e *OpenID* como abordagem centrada em SPs. No projeto *SecFuNet*, seguimos a abordagem do *OpenID*, onde cada provedor de serviços mantém uma lista de *IdPs* em que confia e que seus usuários devem ter identidades em pelo menos um destes *IdPs*.

3. O Modelo de Tolerância a Intrusões IT-VM

O uso da tecnologia de virtualização torna possível a implementação de novos protocolos e suportes de tolerância a intrusão. É o caso do algoritmo IT-VM (*Intrusion Tolerance by Virtual Machines*) [Lau et al. 2012a], [Lau et al. 2012b] que define réplicas de servidores implementadas em máquinas virtuais (VMs) isoladas e que se comunicam via memória compartilhada na efetivação de uma replicação Máquina de Estados (ME). As réplicas e suas VMs são mapeadas sobre somente um servidor físico neste modelo. Como o protocolo ZZ [Wood et al. 2011] o IT-VM apresenta funções para alteração dinâmica da replicação do modelo, ou seja, réplicas maliciosas quando detectadas, são removidas e substituídas de forma eficiente e sem deixar que a replicação durante seu ciclo de vida fique limitada a um número fixo de falhas no sistema.

As soluções IT-VM e ZZ operam em situações sem falhas de réplicas com replicação mínima: $f + 1$ réplicas ativas na configuração. Mas, no protocolo ZZ, as VMs de réplicas são executadas em um conjunto de máquinas físicas. Com isto, no ZZ, as VMs não interagem via memória compartilhada mesmo quando estão dispostas na mesma máquina física. Estas interações entre VMs são realizadas através de trocas de mensagens via protocolos de rede, implicando em custos adicionais no desempenho deste último protocolo. O ZZ também não trabalha com a ideia de elementos confiáveis como o modelo IT-VM e, portanto, não permite a separação de faltas *crash* e de intrusões. Como queremos desempenho e tratar somente com intrusões, o modelo IT-VM se mostram como uma base adequada para o uso nos *IdPs* do projeto *SecFuNet*.

O algoritmo IT-VM descreve diferentes componentes: i) As réplicas isoladas em diferentes VMs que são os únicos elementos visíveis via rede; ii) memória compartilhada que é usada pelos componentes em diferentes níveis para se comunicarem; iii) *Agreement Service* (*AS*), componente confiável que se mantém isolado da rede externa e das VMs e agrega funções de verificação do comportamento das réplicas assim como assinatura das respostas enviadas aos clientes. O *Agreement Service*, por executar operações vitais, é assumido como "elemento confiável" no algoritmo IT-VM. A garantia de confiabilidade do *Agreement Service* é baseada em premissas de isolamento e simplicidade deste componente, que podem facilitar a verificação e testes sobre o código do mesmo. O isolamento do *AS* e da memória compartilhada é concretizado através do *hypervisor* de virtualização.

4. Provedor de Identidades Tolerante a Intrusões (*IdP-IT*)

O modelo proposto para a tolerância a Intrusões do *IdP* (*IdP-IT*) no *SecFuNet* envolve a estratificação de camadas apresentada na Figura 1. O *IdP-IT* é explicitado pelo ambiente virtualizado da figura e serve de *front-end* a um servidor de autenticação (*SA*). Este servidor de autenticação é construído sobre um *grid* de processadores seguros onde cada um destes processadores mantém dados e controla as decisões de autenticação de um conjunto de usuários. O *login* é concretizado com o uso de *userid*/*certificate* onde estes certificados podem ser de uma *PKI* oficial ou formados com as chaves públicas dos usuários sendo assinadas pelo servidor de autenticação de seus domínios. As interações de *login* são realizadas com a liberação do certificado diretamente do *smartcard* do usuário. Uma vez verificada as informações de *login* de um usuário, é estabelecido um canal EAP-TLS entre o *smartcard* e o processador seguro correspondente para a troca de atributos que devem vir do cartão e ficarem disponíveis neste último durante toda a sessão de computação do usuário no sistema.

O servidor de autenticação é isolado da rede externa e dos níveis onde o nosso modelo de tolerância a intrusões atua, através de uma interface (Interface com o *SA* na Figura 1). Esta interface possui a operação *verifyCertificate*() que é usada na verificação de certificados de usuários. A operação *createSign*() é também parte desta interface e retorna uma assinatura feita com a chave privada do *SA* sobre dados passados como parâmetros. Outras operações ligadas à criação e remoção de contas de usuário, criação e remoção de certificados, introdução e remoção de atributos, etc, são visíveis apenas para administradores do servidor de autenticação.

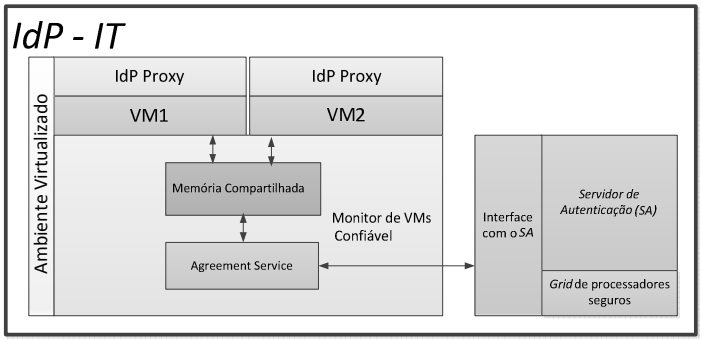


Figura 1 – *IdP* Tolerante a Intrusões

O trabalho descrito neste texto se fixa na parte do ambiente virtualizado da Figura 1. No nível mais alto do modelo *IdP-TI*, estão as VMs de *IdP proxies*, constituindo as réplicas do algoritmo IT-VM citado na seção anterior. As VMs devem executar os *IdPs* do *framework* definido para as relações de confiança no *SecFuNet* (no caso o *OpenID*). Estes *IdPs* na verdade não controlam as contas dos usuários, mas passam as informações obtidas dos usuários, para o servidor de autenticação que é mantido isolado do ambiente de rede (por isso identificados como proxy).

As VMs que executam os *IdPs proxies* do *OpenID* atuam como provedores diferentes com endereços próprios. Externamente são vistos como diferentes provedores de identidades *OpenID*. Suas atuações serão consideradas como corretas ou falhas diante de uma mesma requisição de *login* de um usuário. Máquinas virtuais corretas mantém a execução do protocolo segundo as especificações, enquanto máquinas virtuais faltosas podem apresentar qualquer comportamento arbitrário, tal como: adulterar mensagens; omitir seus recebimentos; omitir envios de asserções e certificar indivíduos não autori-

zados. No nível subsequente abaixo dos *IdPs proxies*, está o monitor de máquinas virtuais (VMM) que implementa a abstração de memória compartilhada, usada para a comunicação dos *IdPs proxies* e o *Agreement Service* (*AS*). O isolamento do *AS*, que é um dos requisitos para a correção do algoritmo IT-VM é garantido pela implementação do mesmo também neste nível. Ou seja, o *Agreement Service* não pode ser comprometido através de intrusões via rede. Na verdade as intrusões devem se limitar as VMs dos *IdPs proxies* que são os únicos componentes do modelo visíveis externamente.

O *AS* desempenha as seguintes funções na arquitetura da Figura 1: verificação das requisições (*OpenID)* dos clientes recebidas pelos *IdPs proxies*; comunicação via interface indicada com o servidor de autenticação, criação e assinatura da asserção *OpenID* que deve refletir o resultado do *login*. Estas asserções são mantidas no *AS* durante as seções de computação dos usuários e liberadas sob demanda a provedores de serviços segundo os protocolos do *OpenID*. Em situações normais de execução o *AS* mantém a configuração de f +1 VMs de *proxies*. Na detecção de comportamento malicioso, o *AS* ativa novas f VMs p*roxies* e, uma vez terminado o processamento da requisição do *login*, desativa as VMs suspeitas de comportamento malicioso, mantendo a replicação com o quórum de execução normal (f +1 VMs de *proxies*).

4.1 Integração do *IdP* tolerante a intrusões com protocolos *OpenID*

OpenID é um framework para gerenciamento de identidades que segue a abordagem centralizada. Neste *framework*, o protocolo de identificação inicia com um usuário se apresentando a um provedor de serviços (chamado de *Relying Party* nas especificações *OpenID*) através de um identificador *OpenID*. Este identificador *OpenID* é usado em vários provedores de serviço (*Relying Parties*) que confiam no *IdP OpenID* gerador deste identificador. Esta confiança é necessária porque os provedores de serviços (*RPs*) devem delegar a autenticação do usuário indicado pelo identificador ao provedor de identidades (*OpenID Provider*) correspondente. Estes identificadores de usuários podem ser URLs ou ainda, documentos XRI (*Extensible Resource Identifier*) que forneçem informações adicionais como URLs de provedores de identidades onde as credenciais dos usuários possam ser aceitas.

4.1.1 Descrição do protocolo de autenticação *OpenID*

O protocolo de autenticação *OpenID* é representado pela Figura 2. Este protocolo inicia com o usuário, através de seu navegador (*user-agent*), acessando o provedor de serviços (RP) (Passo1). O RP então apresenta no navegador do usuário um formulário solicitando o identificador *OpenID* do cliente (Passo2). Após a apresentação do identificador do usuário (Passo 3), é feita a normalização [Berners-Lee et al. 2005] deste identificador que tem por objetivo obter o endereço do *IdP* do usuário.

Depois de normalizado o identificador, é executado um protocolo de descoberta (Passo 4). A descoberta é um processo executado pelo RP com o objetivo de obter informações necessárias para o início das trocas de mensagem com o provedor de identidades (*IdP OpenID*). O processo de descoberta pode ser realizado através de documentos XRI ou através da URL apresentada como *ID* do usuário. O RP faz uso do protocolo *Yadis* [Miller 2006] quando a descoberta deve tratar com documentos XRI. Caso seja uma URL, então métodos HTTP são usados nesta descoberta. De posse das informações necessárias, o RP pode então estabelecer associação direta ou indireta com o *IdP* (Passo 5). A associação direta é opcional e usa troca de segredos entre as partes segundo o

protocolo *Diffie-Hellman Key Agreement*. No Passo 6, o *browser* do usuário é redirecionado para o *IdP OpenID* de modo que esse faça o seu *login*. A autenticação do usuário pode ser feita baseada no envio de sua senha (ou outro tipo de credencial) em interação com o *IdP*. Caso o usuário já tenha se identificado junto ao seu *IdP*, o RP (provedor de serviço) simplesmente solicita diretamente ao *IdP* a asserção de autenticação correspondente. O Passo 8 descreve o envio da asserção de autenticação pelo *IdP* para o RP, com informações sobre o processo de *login*. Esta asserção pode ser positiva quando o *IdP* reconhece o usuário e sua autenticação ou negativa, em caso contrário. Um dos campos importantes da asserção é a assinatura do *IdP* que garante a legitimidade do processo para o RP. No final da identificação positiva, o usuário tem o seu *browser* redirecionado para o RP, para solicitar os seus acessos seguindo as políticas de autorização do RP.

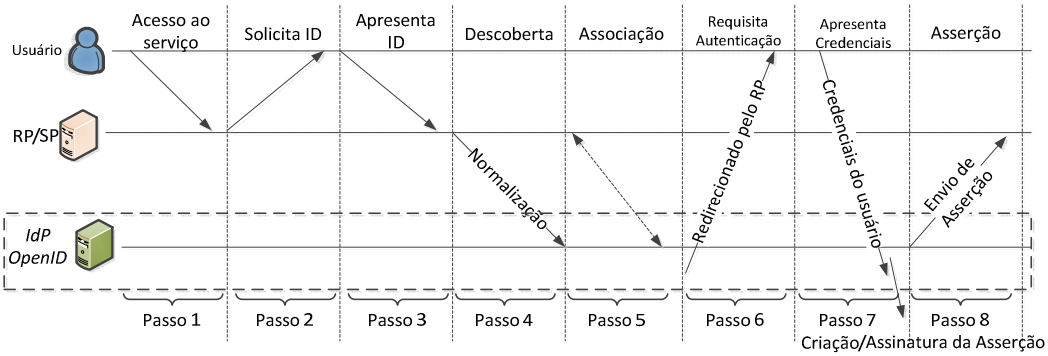


Figura 2 – Protocolo *OpenID*

4.1.2 Integração de Mecanismos de TI aos Protocolos *OpenID*

Esta seção trata da integração dos protocolos e mecanismos *OpenID* e com o *IdP* tolerante a intrusões (*IdP-IT*). Esta integração é desenvolvida de modo a não resultar em modificações nos protocolos *OpenID* e de deixar transparente tanto para usuários como para provedores de serviços (RPs) a replicação usada neste *IdP*. A Figura 3 ilustra os passos do protocolo *OpenID* adaptado para a tolerância a intrusões no modelo *IdP-IT*.

O protocolo funciona de maneira similar ao protocolo *OpenID* apresentado na seção 4.1.1 até o Passo 4. Como a abordagem de replicação conta sempre com mais de um servidor (pelo menos *f+1* servidores), o passo de descoberta deve ser feito obrigatoriamente utilizando o protocolo de descoberta *Yadis*, que retorna para RPs descritores indicando provedores de identidade. Quando um cliente apresenta um identificador *Yadis* a um provedor de serviços, este identificador indica para o RP uma URL onde pode ser liberado um *Yadis Document* com descritores para *IdPs*. Geralmente são encontrados neste documento todos os provedores de identidade que podem ser utilizados para autenticar um usuário. Esta lista contendo os descritores pode definir prioridade de uso dos recursos (no nosso caso *IdPs*) indicados por estes descritores.

No protocolo proposto nesta seção, é importante sempre que sejam retornados todos os descritores dos *IdP proxies* (os *IdPs OpenID* replicados) presentes no modelo *IdP-IT*. Para não modificar o protocolo *OpenID* nas interações *browser* / *IdPs*, foi necessário adaptar o modelo IT-VM que inicialmente previa replicação ativa, para replicação passiva (sem estado). Nesta nova situação, o *IdP proxy* no papel de primário deve interagir com o *browser* e o SP. E caso este *proxy* apresente comportamento malicioso, outra réplica ativa (um *IdP proxy* no papel de *backup*) passa a assumir a sua

função de primário. Na lista de descritores retornada ao RP na descoberta com o *Yadis*, os diferentes *proxies* da replicação de *IdPs* deverão ser sinalizados com diferentes prioridades indicando os seus papéis (primário ou *backups*). Assumindo então diferentes papéis, o SP deve estabelecer a associação do Passo 5 com o *proxy* primário.

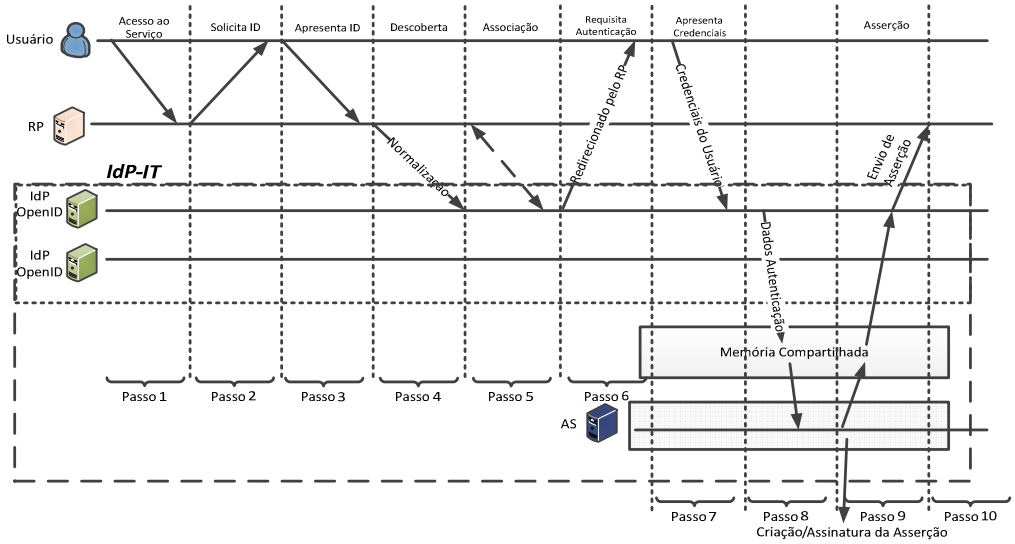


Figura 3– Protocolo *OpenID* estendido para tolerância a intrusões

O *proxy* primário recebe então a credencial do usuário (certificado) em mensagem *OpenID* (Passo 7), extrai então esta credencial e escreve a mesma na memória compartilhada para que seja lida pelo *Agreement Service* (passo 8). O *Agreement Service* deve então submeter o certificado do usuário às verificações correspondentes, usando a operação *verifyCertificate*() da interface do servidor de autenticação subjacente (seção 4 e Figura 1). Se o retorno desta operação indicar *true*, a credencial está correta e o *Agreement Service* pode criar a asserção de autenticação correspondente ao *login* do usuário (Passo 9, Figura 3).

A asserção tem informações como a identidade do usuário, a URL para a qual o *IdP proxy* no papel de primário deverá redirecionar o *browser* do usuário, um *handle* da associação *RP-IdP* (*proxy* primário), o *nonce* e a assinatura do servidor de autenticação sobre todos estes campos. A assinatura destas informações é conseguida pelo *Agreement Service* usando a operação *createSign*() da interface do servidor de autenticação. Para completar o Passo 9, o *Agreement Service* deve passar esta asserção para o *proxy* primário para que este, por sua vez, envie através do protocolo *OpenID* esta asserção ao provedor de serviço (RP). A chave pública do servidor de autenticação, na forma de um certificado auto assinado (ou assinado por uma chave disponível hierarquicamente superior dentro de uma *PKI*) tem que estar disponível para os RPs participantes das interações. Feitas as verificações (assinaturas e demais informações) sobre a asserção pelo RP, este pode conceder o acesso requisitado ao usuário, que ficará sujeito também a verificações de políticas de autorização locais. Vale ressaltar que no protocolo *IdP-IT*, mesmo envolvendo a inclusão de replicação (*IdP proxies*), nenhuma modificação nas mensagens *OpenID* foi necessária.

4.1.3 Execuções anormais do protocolo *IdP-IT*

Porém, apesar de todo o isolamento das entidades críticas, ataques aos *IdPs proxies* não podem ser evitados. Na abordagem proposta, quando as réplicas são compro-

metidas por intrusões, não há uma grande preocupação quanto a possíveis modificações de asserções ou mesmo o envio de credenciais falsas, uma vez que estas informações estão sujeitas a assinaturas ou a verificações envolvendo a parte confiável do *IdP-IT* que é o servidor de autenticação (isolado e livre de intrusões). Porém, os atacantes podem corromper réplicas de *IdP proxies* de modo que estas não executem corretamente o protocolo *OpenID*. Os mecanismos usados em conjunto com replicação de *proxies* devem detectar possíveis ações maliciosas e remover o *proxy* comprometido do *IdP-IT*.

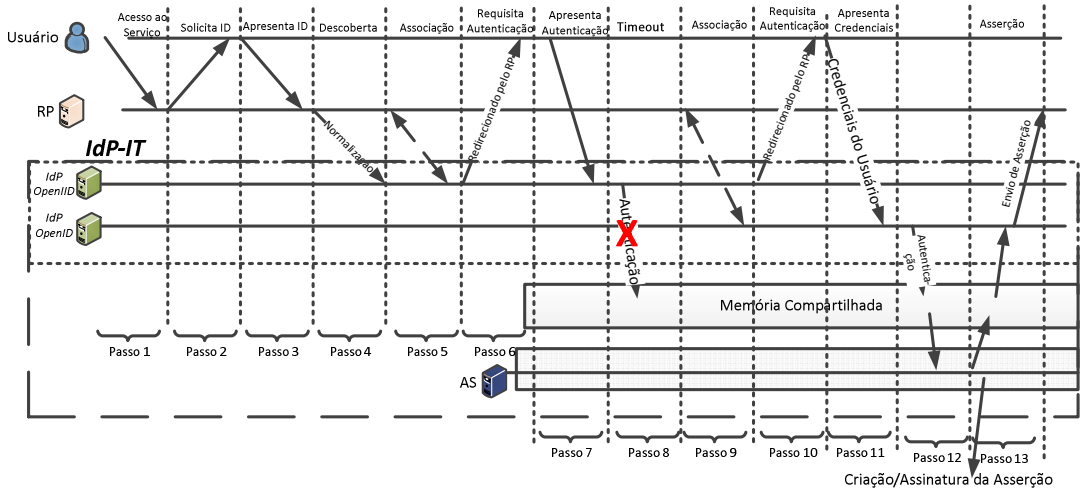


Figura 4 – Protocolo *OpenID* replicado na presença de intrusões

A Figura 4 ilustra o comportamento do protocolo *IdP-IT* estendido em presença de malícia. Os passos 1 a 7 ocorrem da mesma forma como apresentado na seção 4.1.2. Lembrando que no passo 4, onde é executado o protocolo de descoberta *Yadis*, é retornada para o RP a lista de *proxies* do *IdP-IT* que este pode fazer uso. Porém, na instância de protocolo apresentada na Figura 4, é mostrado o comportamento errôneo do *proxy* primário da replicação que se recusa a inserir a credencial (certificado) enviada pelo usuário na memória compartilhada, impedindo que o protocolo possa prosseguir na sua execução normal.

A detecção desta anomalia é conseguida com *timeout* no RP que solicitou a autenticação do usuário. Dessa forma, o RP ativa seu temporizador aguardando uma asserção de autenticação como resposta quando iniciou qualquer processo de troca de mensagem com o *IdP proxy* primário. Se estas não chegam dentro de prazo determinado, o RP utiliza-se da lista de descritores retornados pelo *Yadis*, para tentar conexão com outro *proxy* do *IdP-IT*. Na Figura 4, com o temporizador do RP excedido, este redireciona o pedido de autenticação do usuário para o *proxy backup* (Passo 9) no sentido que apresente sua credencial (passo 10). Este processo pode se repetir *f* vezes até que se atinja uma réplica correta (com *f* substituições de *proxies* no papel de primário) e então o usuário é autenticado. Todas as situações de comportamentos maliciosos (recusa de recebimento e envio de dados e associações, modificações em asserções, etc) são detectadas a partir do RP (provedor de serviços) com temporizações e verificações locais.

Além das considerações citadas, a configuração *f + 1* réplicas *proxies* é sempre restaurada, de modo que o *IdP-IT* mantém sempre no seu ciclo de vida o mesmo número de réplicas. As réplicas maliciosas são sempre removidas e substituídas por novas réplicas *proxies*. Estas substituições são muito simples porque os *proxies* não mantêm informações de estado. As informações de estado (informações de conexões, asserções e

demandas de autenticação) estão contidas no *Agreement Service* que é mantido protegido, isolado da rede externa. Mas, as reconfigurações geram algumas implicações: o documento retornado pelo protocolo *Yadis* deve ser modificado para a próxima demanda de autenticação de um *RP*. Ou seja, os novos *proxies* deverão possuir descritores e também prioridades indicando seus papéis nesta nova configuração no documento *Yadis*. O *Agreement Service* é o agente destas reconfigurações tratando de ativações e remoções de *proxies*. É também este serviço que prepara o documento *Yadis* com as alterações de configuração no *IdP-IT*.

5. Implementação de Protótipo e Análise de Resultados

De forma geral, a estrutura utilizada em nosso protótipo é representada pela Figura 5. Para a replicação dos *IdP proxies* foi utilizado o software de virtualização XEN, para a execução das máquinas virtuais. Além disso, o uso do *hypervisor* XEN oferece a facilidade de implementação da memória compartilhada para a comunicação entre as VMs *proxies* e o *Agreement Service*. A memória compartilhada e o *Agreement Service* foram implementados no nível do *hypervisor*, o que permitiu o isolamento destas abstrações, de forma que ambas possam ser consideradas componentes confiáveis, que é premissa básica da proposta. Nas VMs de *proxies* foram utilizadas diferentes versões do sistema operacional Linux, com o objetivo de manter certa independência de vulnerabilidades já que estas máquinas são os únicos componentes do *IdP-IT* visíveis externamente (via rede).

Como provedor de identidades (*OpenID*) foi utilizada a implementação em código livre *JOID*, que foi alterada para que o protocolo fosse executado diante das características de replicação e uso de componentes confiáveis, como descritas nas seções anteriores. Vale salientar que foram removidos da implementação *JOID* os módulos de gerenciamento de conta de usuários. O provedor de identidades *OpenID* implementado é executado sobre o servidor de aplicação Apache *Tomcat*. O *Agreement Service* e outras entidades introduzidas pelo modelo *IdP-IT* foram desenvolvidos com a linguagem de programação Java.

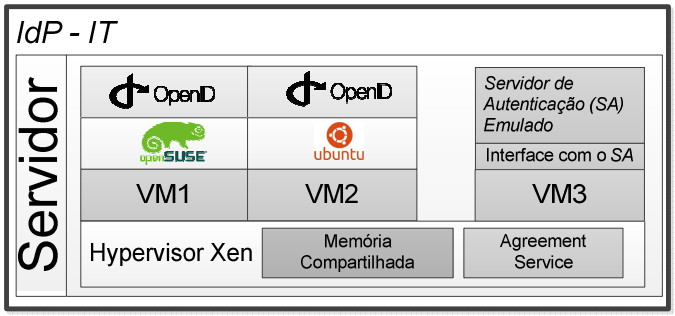


Figura 5 – Estrutura do sistema

O provedor de serviços (*Relying Party*) desenvolvido não precisou de extensões especiais em relação ao protocolo *OpenID*, mas este tem que saber lidar com a indisponibilidade momentânea de um IdP *proxy*. Diante disto, este RP deve contatar outro *proxy* (como alternativa do *IdP-IT*). Fizemos uso da tecnologia AJAX para o desenvolvimento do provedor de serviços. O uso do AJAX nos permitiu que houvesse um maior controle sobre a aplicação e suas trocas de mensagens com o *IdP-IT*. Ainda que ocorram trocas de provedor de identidades (na verdade trocas de *IdPs proxies*), toda a interação é feita pela aplicação executada no navegador do cliente. O usuário não precisa começar

todas as trocas de novo a partir do RP. Na substituição de *IdP* primário, a página de *login* do *OpenID* é reativada em seu *browser*.

O servidor de autenticação, mantido completamente isolado da rede, é quem realmente gerencia as contas de usuários e está sendo desenvolvido no *SecFuNet* usando um *grid* de microcontroladores seguros[2], cada um desses com *software* embutido. A estrutura geral do servidor de autenticação é concretizada sob um sistema operacional dedicado. Como o foco deste texto é o ambiente virtualizado que deve tolerar intrusões e interagir com o servidor de autenticação, emulamos o comportamento deste servidor a partir de uma máquina virtual isolada. O protótipo final do projeto *SecFuNet* prevê o *IdP-IT* e todos os seus componentes (incluindo também o servidor de autenticação), compartilhando um mesmo servidor físico em seu domínio de políticas.

5.1 Experimentação do Protótipo

Com o objetivo de testar nossas proposições o protótipo foi executado em um servidor com processador Intel Core i7 3.4 *Ghz*, com 8 GB de memória conectado com seus clientes através de uma rede local Ethernet funcionando a 100 *Mbps*. Em nossos testes usamos o envio automático de dados de identificação a partir de uma máquina cliente. Definimos o limite de réplicas maliciosas como sendo o valor de $f = 1$. Desta forma, como a arquitetura dispõe de $f + 1$ réplicas *proxies* executando o *IdP OpenID*, o total de réplicas usado no protocolo foi 2. O valor escolhido para f não tem muito significado em relação à robustez do modelo porque, imediatamente a detecção, esta réplica corrompida é removida e ativada outra em seu lugar. Os reflexos no desempenho não são muito sentidos porque a iniciação de novos *proxies* não envolve transferências de estado.

Realizamos testes envolvendo três rodadas de 1000 operações de autenticação. Estes testes foram divididos em duas classes: i) execução de autenticações sem falhas maliciosas ii) execução de autenticações com falhas maliciosas nos *proxies*. Na primeira classe de testes (sem falhas) as simulações não levaram em conta a presença de intrusões no sistema, de forma que o serviço funcionasse sempre sem a necessidade da troca do *proxy* primário. Na segunda classe, foram incluídas modificações no sistema de forma que intrusões fossem simuladas. Nessas simulações a classe de faltas (maliciosas) foi no domínio de valores. Ou seja, os proxies maliciosos alteravam asserções de forma que quando checadas pelo RP fossem verificadas como inválidas.

Como resultado dos testes, foi calculada a média dos tempos de resposta nas identificações. Visto que em nossa abordagem a replicação é renovada, montamos casos de testes com diferentes distribuições de réplicas maliciosas. As distribuições são apresentadas na Tabela 1. A coluna $\mathbf{f}_{\%}$ apresenta, dentro do conjunto de 1000 requisições, qual a porcentagem de falhas maliciosas que ocorrem no conjunto. A coluna $\mathbf{N}_{f}$ apresenta o número de vezes que as falhas ocorrerão. Na Figura 6 é apresentado o tempo médio de autenticação nas duas classes de testes citadas acima. No caso de teste onde não existem *proxies* maliciosos, o tempo médio foi de 7 *ms*. Nos demais testes, onde as intrusões foram simuladas, este tempo foi crescente em relação ao aumento do número de intrusões. Este crescimento está relacionado ao tempo de renovação da replicação no

[2] As placas com o *grid* de microcontroladores seguros formam um produto de propriedade intelectual das empresas Implementa, EtherTrust e Telecom ParisTech.

algoritmo. No nosso esquema de testes temos 2 réplicas participando da configuração de *IdP proxies* (*f = 1*). Também para evitar os custos de criação de novas réplicas (em torno 1 segundo), mantemos 8 réplica (VMs com os seus proxies) em espera fora da configuração. A remoção de uma faltosa envolve a ativação destas réplicas em espera. Vale a pena salientar que o teste com 100% de falhas reflete que, a cada requisição de autenticação ocorra uma intrusão.

Tabela 1 – Distribuição das faltas

$f_{\%}$	N_f
0,1%	1
1,00%	10
2,50%	25
5,00%	50
10,00%	100
12,50%	125
16,60%	166
25,00%	250
50,00%	500
100,00%	1000

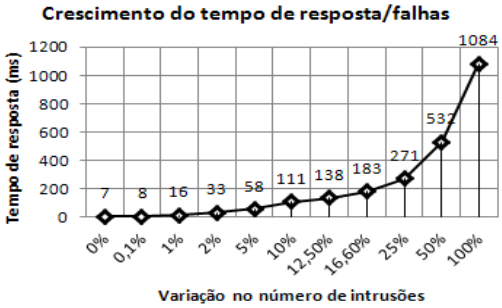


Figura 6 – Tempo de resposta das autenticações

Apesar de não haverem comparações com outras abordagens de forma a atestar a eficiência do algoritmo proposto, os testes serviram para demonstrar que a abordagem é factível e como esta se comporta em um ambiente hostil onde intrusões são frequentes.

6. Trabalhos relacionados

Até o presente momento não foram encontradas na literatura abordagens que tratassem de tolerância a intrusões em provedores de identidades. Porém, o uso de provedores de identidade vem sendo crescente em aplicações e sistemas distribuídos de larga escala como computação em nuvem, organizações virtuais, computação em grade, etc. Alguns trabalhos citados nesta seção são confrontados ou mesmo usados como mecanismo de apoio no sentido de evidenciar resultados de nossa proposta.

Em [Coppola et al. 2012] é descrito um infraestrutura para federação de provedores de computação em nuvem chamada *Contrail*. Esta infraestrutura é responsável pelo gerenciamento dos recursos oferecidos por nuvens federadas. Dessa forma, quando o cliente deseja implantar sua aplicação em nuvem, o *framework* citado seleciona a nuvem que melhor possa atender aos requisitos de *SLA* (*Service Level Agreement*) negociados com o cliente. Do ponto de vista do gerenciamento de identidades, ainda que as aplicações de um cliente estejam sendo executadas em várias nuvens distintas, este não deve necessariamente se apresentar com uma identidade diferente em cada um dos provedores das mesmas. Desta forma, na abordagem *Contrail* é definido um nível de gerenciamento de identidades que estabelece relações de confiança entre os provedores de identidade usados nas diferentes nuvens. Estas relações de confiança definem então uma federação de nuvens. Para tanto, foi utilizado nesta infraestrutura o protocolo *OAuth2*.0 [Microsoft 2012] para a geração de *tokens* de autenticação e de autorização e documentos SAML (*Security Assertion Markup Language*) para troca de certificados entre os *IdPs* de diferentes nuvens. Os *IdPs* desta proposta são implementados nas próprias nuvens, o que certamente pode ser um obstáculo significativo para se garantir a segurança dos mesmos. Dentre os trabalhos futuros propostos pelos autores está a integração do protocolo *Shibboleth* para identificação dos usuários.

Outro trabalho que propõe o uso de federação a partir de *IdPs* de diferentes nuvens é apresentado em [Tusa et al. 2012]. Neste trabalho são discutidos principalmente detalhes da implementação de autenticação única (*Single Sign On*) através da integração do protocolo *Shibboleth*, SASL (*Simple Authentication and Security Layer*) e SAML (*Security Assertion Markup Language*). Na abordagem apresentada, quando um provedor de computação em nuvem (chamado de Origem) deseja utilizar recursos de outra nuvem (chamado de Destino) este inicia o processo com trocas de mensagens através do protocolo XMPP. Esta proposta também implementa os provedores de identidades nas correspondentes nuvens e, portanto, sem os cuidados de segurança necessários.

Em [Leandro et al. 2012] é proposta uma arquitetura de federação de provedores de identidades para o uso em computação em nuvem, utilizando o protocolo *Shibboleth*. Neste trabalho é apresentada ainda uma implementação da arquitetura proposta. Nessa abordagem, os clientes possuem seus provedores de identidades fora da nuvem. Cada provedor de identidades define um domínio de políticas com os seus clientes e provedores de serviços (provedores de computação em nuvem). A associação de confiança que garante a transposição de uma asserção SAML de autenticação de um domínio para o outro é estabelecida pelo *Shibboleth*.

Os trabalhos citados acima tratam o gerenciamento de identidade de maneira similar, com o uso do conceito de federação. Somente a última experiência citada é que constrói os provedores de identidades fora das nuvens. A nossa experiência descrita neste texto, não usa o conceito de identidades federadas. O nosso modelo é o centralizado em provedores de serviço (que na verdade é a abordagem da ferramenta *OpenID*), ou seja, cada serviço (RP) mantém uma lista de provedores que este confia e localiza em qual destes o usuário está registrado usando o identificador *OpenID* do usuário. Uma das vantagens do *OpenID* é que este não requer nenuma um *pilha* de protocolos especiais do lado do cliente. O que não é o caso do *Shibboleth* que requer um *stack* de protocolos no lado do cliente, limitando com isto o seu uso em determinados dispositivos móveis. Em relação ao modelo *IdP*-TI acreditamos ser uma experiência única neste nível. Possíveis intrusões que resultem em comprometimento de *IdPs* (*proxies* no nosso modelo), não resultam na quebra de confidencialidade de dados de usuários e muito menos na interrupção do serviço de autenticação de usuários.

7. Conclusão

Este artigo descreveu a nossa experiência no desenvolvimento de provedor de identidades *OpenID* tolerante a intrusões. A arquitetura que implementamos faz uso da tecnologia de virtualização e não envolve replicação física do *hardware* especializado (processadores seguros) que é usado nos *IdP*. As informações de usuários são mantidas em compartimentos separados e qualquer intrusão no modelo se traduz em ações inócuas. A separação dos compartimentos é feita através do isolamento das máquinas virtuais e componentes confiáveis, que se comunicam exclusivamente através de uma memória compartilhada.

O uso da virtualização permite a recuperação da replicação definida já a partir da detecção do comportamento malicioso. Um protótipo foi implementado do modelo desenvolvido e os testes realizados com os mesmos nos dão a certeza da viabilidade das nossas propostas. Este trabalho nos parece singular porque não encontramos experiências correlatas na literatura. As propostas e desenvolvimentos apresentados neste texto fazem parte do Projeto *SecFuNet*.

Agradecimentos

Este trabalho foi financiado pelo CNPq através dos processos 590047/2011-6 (Projeto SecFuNet) e também pelo processo 307588/2010-6. Agradecemos ainda a CAPES pelo apoio financeiro com bolsa de doutorado.

Referências

Berners-Lee, T., Fielding, R. and Masinter, L. (2005). [RFC 3986] Uniform Resource Identifier (URI).

Castro, M. and Liskov, B. (2002). Practical Byzantine Fault Tolerance and Proactive Recovery. ACM Transactions on Computer Systems, v. 20, n. 4, p. 398–461.

Coppola, M., Dazzi, P., Lazouski, A., et al. (2012). The Contrail approach to cloud federations. The Internation Symposium on Grids and Clouds (ISGC) 2012.

Jøsang, A., Fabre, J., Hay, B. and Pope, S. (2005). Trust requirements in identity management. Proc. Australasian workshop on Grid computing and e-research, p. 99–108.

Kotla, R., Alvisi, L., Dahlin, M., Clement, A. and Wong, E. (2007). Zyzzyva: speculative byzantine Fault Tolerance. ACM SIGOPS Operating Systems Review, v. 41, n. 6, p. 45–58.

Lau, J., Barreto, L. and Fraga, J. (2012a). An Infrastructure based on Virtualization for Intrusion Tolerant Services. 19th IEEE International Conference on Web Services, p. 170 - 177.

Lau, J., Barreto, L. and Fraga, J. (2012b). Infraestrutura Baseada em Virtualização para Serviços Tolerantes a Intrusões. XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, p. 522 - 535.

Leandro, M. A. P., Nascimento, T. J., Santos, D. R. Dos, Westphall, C. M. and Westphall, C. B. (2012). Multi-Tenancy Authorization System with Federated Identity for Cloud-Based Environments Using Shibboleth. ICN 2012, The Eleventh International Conference on Networks, n. c, p. 88–93.

Lewis, J. A. (2008). Authentication 2.0 - new opportunities for online identification.

Liberty (2003). Introduction to the Liberty Alliance Identity Architecture.

Microsoft (2012). [RFC 6749] The OAuth 2.0 Authorization Framework.

Miller, J. (2006). Yadis Specification 1.0.

OpenID (2007). OpenID authentication 2.0.

Schneider, F. B. (1990). Implementing Fault-Tolerant Approach: A Tutorial Services Using the State Machine. Computing, v. 22, n. 4.

Tusa, F., Celesti, A., Villari, M. and Puliafito, A. (2012). Federation Between CLEVER Clouds Through SASL/Shibboleth Authentication. The Fourth International Conference on Evolving Internet, n. 4, p. 12–17.

Wood, T., Singh, R., Venkataramani, A., Shenoy, P. and Cecchet, E. (2011). ZZ and the Art of Practical BFT Execution. Proceedings of the sixth conference on Computer Systems EuroSys ’11, p. 123–138.

Análise do tráfego de spam coletado ao redor do mundo

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

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

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

{pedro.lascasas,dorgival,meira}@dcc.ufmg.br

{cristine,klaus,mhp}@cert.br

{osvaldo.morais,elverton,rubens}@dcc.ufmg.br

Abstract. *Several efforts have been pursued to create a comprehensive view of spam traffic. However, observations at isolated points of the Internet are always limited by factors of spatial locality. This work aims to add a dimension to this analysis by contrasting samples of spam traffic collected simultaneously at different points. Our analyses indicate that factors such as location and connectivity have significant impact on the observed traffic, but certain features, such as profiles of messages sent by different protocols, source addresses and test patterns from spammers repeat themselves around the world.*

Resumo. *Diversos esforços têm sido feitos para se criar uma visão abrangente do tráfego de spam. Entretanto, observações em pontos isolados da Internet estão sempre limitadas por fatores de localidade espacial. Este trabalho pretende acrescentar uma dimensão a essa análise ao contrastar amostras de tráfego de spam coletadas simultaneamente em diferentes pontos. Nossas análises indicam que fatores como localização e conectividade têm impacto sensível sobre o tráfego observado, porém certas características, como perfis das mensagens enviadas por diferentes protocolos, endereços de origem e padrões de teste dos spammers se repetem ao redor do mundo.*

1. Introdução

Ainda hoje, *spam* é um dos grandes problemas presentes na Internet. Além do caráter indesejado, *spams* são muitas vezes relacionados com o envio de *phishing*, além da propagação de *malwares*, como cavalos de tróia, vírus e *worms* [Newman et al. 2002], tornando-os ainda mais nocivos para a rede e seus usuários. Por tudo isso, estudos mostram que o abuso causado pelos *spams* acarretam vários bilhões de dólares de prejuízo às empresas e à sociedade em geral [Sipior et al. 2004].

O combate ao *spam* é caracterizado pela constante evolução das técnicas de detecção dessas mensagens, uma vez que as ferramentas utilizadas pelos *spammers* se mostram cada vez mais sofisticadas, reduzindo a eficácia dos filtros anti-*spam* e a rastreabilidade dos *spammers* [Goodman et al. 2007]. Um exemplo disso é o crescimento na

utilização de máquinas infectadas por *malwares*, como os bots, para o envio de *spam* e *phishing* [Xie et al. 2008], permitindo que o *spammer* permaneça no anonimato.

Mesmo com o desenvolvimento de técnicas de combate, é necessário um esforço contínuo para entender a forma de atuação dos *spammers* ao enviar mensagens indesejadas pela Internet, devido à sua natureza evolutiva. Essa evolução acontece tanto no modo como *spammers* disseminam suas mensagens pela rede, buscando maximizar o volume de mensagens enquanto mantêm sua identidade oculta, quanto na forma como compõem o conteúdo das mesmas [Pu 2006]. Logo, analisar o comportamento dos *spammers* ao longo do tempo permite identificar padrões e características que podem ajudar na evolução das ferramentas de combate ao *spam*. Por exemplo, o tráfego gerado por máquinas participantes de *botnets* tende a apresentar características similares. Sendo assim, o entendimento desses padrões pode ser útil para identificar *bots* e as redes de que fazem parte.

Para entender o comportamento dos *spammers* na rede, realizamos neste trabalho a caracterização do tráfego de *spam* utilizando dados coletados por *honeypots* de baixa interatividade localizados em redes intermediárias. A grande vantagem da análise aqui realizada está no fato dos dados serem coletados em diversos pontos distintos da Internet, fornecendo assim visões diferentes do problema e não apenas uma visão singular, proveniente de apenas uma fonte de dados, como visto em diversos trabalhos anteriores. Com isso, pretendemos adicionar uma dimensão às análises existentes, ao contrastar amostras de um grande volume de tráfego, oriundos de diversas localidades, reduzindo distorções comuns em coletas restritas a um único local.

Os resultados indicam que algumas características, como perfis das mensagens enviadas por diferentes protocolos, endereços de origem e padrões de teste dos spammers, se repetem nas diversas localidades observadas. Além disso, mostram que o comportamento das campanhas de *spam* que aparecem nos *honeypots* possuem grande influência no tráfego gerado. Por fim, os resultados evidenciam o comportamento dos *spammers* ao descobrirem um novo *honeypot* na rede, que começam a enviar mensagens utilizando o protocolo SOCKS indiscriminadamente, enquanto o início do envio de mensagens utilizando SMTP possui restrições.

2. Trabalhos Relacionados

Diversos trabalhos avaliaram as características apresentadas pelo tráfego de *spam* coletado nos servidores de correio de destino. Kim *et al.* caracterizaram o tráfego de *spam* a partir de dados da camada de aplicação coletados em servidores de correio eletrônico de destino [Kim e Choi 2008]. Gomes *et al.* analisaram uma carga de trabalho de mensagens de usuários de uma universidade brasileira e destacaram uma série de características capazes de diferenciar *spams* de mensagens legítimas [Gomes et al. 2007]. Esses trabalhos, entretanto, focaram apenas tráfego em um ponto específico da rede, os servidores de correio de destino, e consideraram a análise de elementos do conteúdo das mensagens.

Outros consideraram a origem do *spam* na rede como, por exemplo, Las-Casas *et al.*, que propõem uma ferramenta de detecção de *spammers* que analisa o tráfego de saída de um provedor [Las-Casas et al. 2011, Las-Casas et al. 2013]. Nossas observações sobre tráfego em pontos intermediários da rede permite que administradores entendam a natureza do tráfego que passa por suas redes, quando o controle direto sobre a origem do mesmo não é possível.

Com relação ao tráfego de *spam* gerado por *botnets*, relatórios recentes mostram que 88% do total de *spams* enviados são provenientes dessas redes [Symantec 2011]. John *et al.* mostram que apenas 6 *botnets* são responsáveis pela maior parte dos *spams* recebidos [John et al. 2009]. Já Stone-Gross *et al.* destacam o gerenciamento de campanhas de *spam* enviadas por *botnets*, do ponto de vista do *botmaster*. Observando o envio de *spam* do ponto de vista do espaço de endereços IP, eles concluiram que as atividades das *botnets* estão mais dispersas no espaço de endereços IP, o que dificulta a filtragem de *botnets* baseada em análise de endereços [Kokkodis e Faloutsos 2009]. Outros investigaram características de tráfego coletado da camada de rede que seriam comuns a *spammers* [Ramachandran e Feamster 2006], mas se basearam em uma visão local do tráfego.

Neste trabalho realizamos uma caracterização do tráfego de *spam* que, diferente de trabalhos anteriores, possui uma visão global, evitando assim possíveis distorções por se considerar um ponto único da rede. Acreditamos que este trabalho pode ser útil para a evolução dos trabalhos que consideram o combate ao *spam*, uma vez que nele é mostrado como o tráfego de *spam* se comporta em diferentes locais e quais padrões e características são importantes para a contínua identificação das mensagens indesejadas.

3. Metodologia

Os dados foram coletados através de um conjunto de oito *honeypots* de baixa interatividade instalados ao redor do mundo. Esses *honeypots* foram configurados de modo a simular computadores com *proxies* e *mail relays* abertos, que frequentemente são abusados para o envio de *spam* e para outras atividades maliciosas.

Dentre os oito *honeypots* utilizados, dois se encontram em redes brasileiras (BR-01 e BR-02), enquanto os restantes se localizam em diferentes *country codes*: AU-01 (Austrália), AT-01 (Áustria), EC-01 (Equador), NL-01 (Holanda), TW-01 (Taiwan) e UY-01 (Uruguai). Como mencionado anteriormente, a distribuição dos *honeypots* por diferentes pontos da rede mundial teve por objetivo permitir uma visão ampla do tráfego de *spam* ao redor da Internet para, com isso, reduzir distorções comuns em coletas restritas a uma única localidade.

A captura de mensagens foi feita pelos sistemas *spamsinkd* e *spamtestd*, desenvolvido por alguns dos autores, para capturar spams e mensagens de teste, respectivamente [CERT.br 2013]. Quando uma máquina se conecta à porta 25 de um dos *honeypots*, tem a impressão de interagir com um servidor SMTP operando como *open relay*, pronto para repassar mensagens. Máquinas que se conectam a portas tradicionais utilizadas por mecanismos de *proxy*, como as associadas aos protocolos HTTP e SOCKS, são levadas a crer que suas tentativas de conexão para outros servidores SMTP através de tais *proxies* são bem-sucedidas. Em ambos os casos, porém, nenhum *spam* é efetivamente entregue. Todas as transações são armazenadas com informações como data e hora, origem (endereço IP, prefixo de rede e AS), protocolo que foi abusado no *honeypot* e, finalmente, o conteúdo completo de cada mensagem que teria sido enviada.

Neste trabalho consideramos o período de 84 dias entre 09/05/2012 e 31/07/2012. Esse período foi escolhido por ser o maior intervalo recente em que a coleta ocorreu sem interrupções. Ao todo, durante aquele período, foram coletados quase 815 milhões de mensagens, enviadas por 53 mil endereços distintos. Mais detalhes sobre o tráfego coletado são apresentados na seção 4.

4. Resultados

Nesta seção mostramos os resultados mais interessantes encontrados na análise do tráfego de *spam*, além da discussão a respeito destes. Inicialmente apresentamos uma visão geral dos dados, seguido pela comparação e análise detalhada dos diferentes *honeypots*. Observações interessantes encontradas são mostradas posteriormente, além da discussão sobre o redescobrimento de um *honeypot* na rede e o comportamento do tráfego neste.

Tabela 1. Visão Geral

	SMTP(%)	SOCKS(%)	HTTP(%)	Total
Mensagens (milhões)	143,87 (17,7%)	557,73 (68,4%)	113,26 (13,9%)	814,87
Endereços IP	50.348 (93,9%)	3.146 (5,9%)	644 (1,2%)	53.608
Prefixos de rede	4.477 (86,0%)	921 (17,7%)	74 (1,4%)	5.207
Sistemas Autônomos (AS)	1.551 (93,7%)	268 (16,2%)	25 (1,5%)	1.655
***Country Codes* (CC)**	131 (97,0%)	63 (46,7%)	9 (6,7%)	135
Volume de tráfego (TB)	0,51 (15,0%)	2,91 (56,3%)	0,95 (28,0%)	3,39

A tabela 1 oferece uma visão geral dos dados coletados pelos oito *honeypots*. No período de 84 dias, quase 815 milhões de mensagens foram coletadas, oriundas de endereços associados a 135 *country codes* distintos, cerca de 55% dos *country codes* do mundo. Do total de mensagens, 68,44% foram enviadas utilizando o protocolo SOCKS, 17,65% usando SMTP e apenas 13,89% usando HTTP. É interessante notar que, das aproximadamente 113 milhões de mensagens utilizando o protocolo HTTP, mais de 82 milhões foram provenientes do *honeypot* TW-01 que, conforme será discutido posteriormente, possui características bastante distintas dos demais.

Tabela 2. Top 10 Country Codes

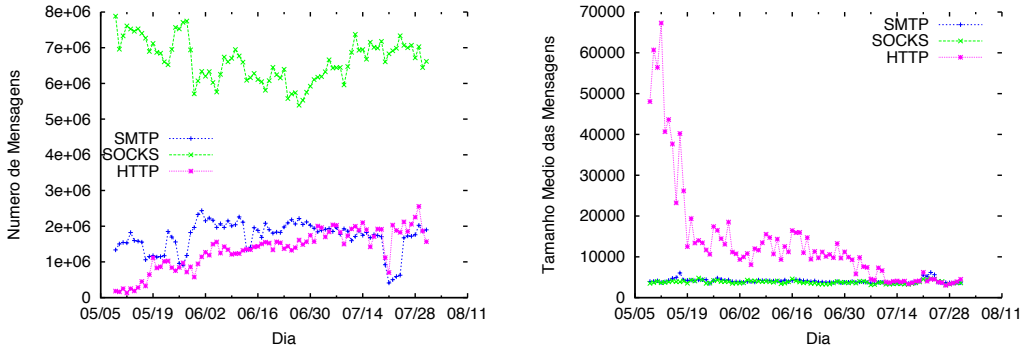
	Mensagens	Endereços IP	Prefixos's	AS's
US	479.118.779	1.533	769	278
PH	119.506.411	97	23	7
CN	47.049.075	18.508	915	51
BR	27.398.959	946	601	99
TW	21.898.881	28.183	130	21
JP	13.340.731	70	39	17
RU	9.097.963	493	387	246
KR	6.430.220	190	127	35
IN	6.271.617	265	209	40
HK	5.856.073	59	55	23

Apesar de mensagens terem sido observadas originando-se de 135 *country codes* distintos, alguns poucos CCs foram responsáveis pela grande maioria de todo o *spam* recebido. A tabela 2 mostra aqueles que originaram mais mensagens indesejadas ao longo do período. Como é possível verificar, os 10 *country codes* que mais enviaram mensagens são responsáveis por mais de 90% do total de *spam* recebido. Quase 60% das mensagens recebidas foi oriunda de endereços associados ao CC US. Estes resultados sugerem que o envio de *spams* está concentrado em alguns poucos *country codes*.

4.1. Comportamentos por protocolo

Através das porcentagens na tabela 1, já é possível observar alguns elementos interessantes do comportamento dos *spammers*: (i)a maior parte das mensagens é enviada através de *proxies* SOCKS, porém por um número relativamente pequeno de máquinas; (ii)apesar de responder por uma fração pequena do total de mensagens (pouco menos de 14%), o protocolo HTTP responde por aproximadamente 28% do tráfego; (iii)o número de endereços IP que usam cada protocolo e o número de mensagens enviadas por cada um variam bastante entre protocolos.

Uma análise temporal dos dados agregados por protocolo indicam comportamentos relativamente estáveis em termos de mensagens enviadas por cada protocolo, conforme pode ser visto na figura 1(a). O volume total enviado usando-se cada protocolo também segue um comportamento bastante semelhante (não apresentado por limitações de espaço). Já a figura 1(b) mostra que o tamanho médio das mensagens enviadas por SOCKS e SMTP permaneceu relativamente estável durante todos os dias, enquanto as mensagens enviadas por HTTP variaram significativamente ao longo do tempo, com algumas de suas mensagens bem maiores que as demais. Inspecionamos essas mensagens maiores e observamos que elas foram enviadas por apenas 5 endereços distintos e todos provenientes do *country code* CN, e foram recebidas por todos os *honeypots*, com exceção do BR-01 e TW-01[1]. Logo, o aparecimento destas pode estar relacionado a algumas campanhas específicas, originadas em CN.



(a) Mensagens por protocolo (b) Tamanho das mensagens por protocolo

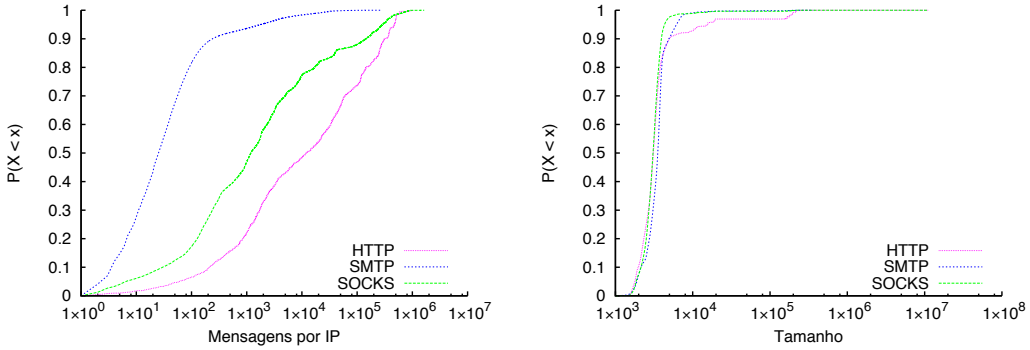
Figura 1. Dados agregados por protocolo ao longo do tempo

A figura 2 apresenta dados sobre a distribuição de mensagens por endereços de origem e por tamanho para cada protocolo, ressaltando as diferenças entre eles. A figura 2(a) mostra que os endereços que utilizam protocolos diferentes possuem comportamentos distintos com relação ao número de mensagens enviadas. Enquanto quase 90% dos endereços IP utilizando SMTP enviaram menos de 100 mensagens ao longo dos 84 dias, apenas 17% dos transmissores SOCKS e 8% dos HTTP enviam menos que isso. Por outro lado, uma fração desprezível dos transmissores SMTP enviou mais que cem mil mensagens, enquanto 11% dos SOCKS e 25% dos HTTP ultrapassaram esse valor.

Com relação ao tamanho das mensagens enviadas usando cada protocolo (fig. 2(b)), o perfil de 80% das mensagens enviadas por cada protocolo são bastante semelhantes, sendo as mensagens enviadas por SOCKS apenas ligeiramente maiores. Entretanto, apenas 2% das mensagens enviadas usando SMTP tinha mais de 5 KB, quanto eram 10% das mensagens enviadas por SOCKS e HTTP. Já no caso de HTTP, cerca de 3% das mensagens enviadas com aquele protocolo eram maiores que 200 KB.

Além disso, apenas uma fração pequena das máquinas utilizou mais de um protocolo para enviar *spam* durante o período (a soma dos números de endereços IP distintos que foram vistos usando cada protocolo é menos de 1% superior ao total de endereços distintos). Esses fatores nos permitem afirmar que as máquinas que utilizam cada tipo de

[1]Naquele período, BR-01 havia trocado de endereço recentemente e ainda não estava sendo abusado pelos *spammers* em geral; TW-01, como discutido posteriormente, tem um comportamento diferenciado com relação a tráfego oriundo de CN.



(a) Núm. de mensagens por end. origem (b) Tamanho das mensagens

Figura 2. Distribuições acumuladas para mensagens por protocolo

protocolo têm comportamentos diferentes, indicando origens (*spammers*) diferentes: (i) máquinas que usam *proxies* abertos (HTTP e SOCKS) enviam mensagens em grande volume, o que sugere o uso de uma infraestrutura especializada; (ii) máquinas que enviam *spam* usando SMTP tendem a enviar bem menos mensagens, mas seu conjunto ainda é responsável por uma parcela significativa do tráfego, o que sugere a formação de *botnets*.

4.2. Comparação entre os diversos locais de coleta

A visão geral dos dados fornece informações interessantes sobre comportamentos que se mantém entre os *honeypots*. Entretanto, analisar cada um dos *honeypots* individualmente nos permite identificar características não possíveis de serem observadas anteriormente, visualizando os dados como um todo. Dado que os *honeypots* estão em localizações distintas, as características dos dados coletados em cada ponto pode apresentar variações.

Distribuição dos endereços IP de origem entre os honeypots

A tabela 3 apresenta o número de endereços IP em comum entre cada par de *honeypots*. Analisando-a, percebemos que alguns *honeypots* possuem muitos endereços em comum, como AU-01 e UY-01: 91% dos endereços presentes em UY-01 também estão presentes no AU-01, correspondendo a 58% dos endereços ali observados (AU-01 foi contactado por mais máquinas). Outro exemplo é entre os *honeypots* AT-01 e BR-01, que possuem 15.417 endereços em comum, que representam 85% de AT-01. Este alto número de origens em comum entre os *honeypots* indica que *spammers* tendem a distribuir seu tráfego por uma grande variedade de pontos intermediários, independente de localidade. Com isso, algumas das características do tráfego presente em diferentes pontos tendem a ser similares, como observado anteriormente, uma vez que parte dos transmissores são comuns e, em consequência, as mensagens de *spam* observadas tendem a ser semelhantes entre eles.

Ainda com relação à tabela 3, podemos notar que o *honeypot* TW-01 é bastante distinto dos demais com relação aos endereços IP que abusaram dele. Além de registrar um número muito menor de endereços IP de origem (apenas 2.194), estes endereços não aparecem em altas proporções nos demais *honeypots*. O *honeypot* que apresenta maior número de endereços em comum com o TW-01 é o BR-02, com 66% dos 2.194 endereços do primeiro sendo observados no segundo também. Porém, em média, cerca de 51% dos endereços observados em TW-01 aparecem nos demais *honeypots*, valor baixo quando

comparado aos valores dos demais. Por conta dessa diferença, TW-01 será tratado separadamente mais à frente.

Tabela 3. Percentual de endereços IP em comum entre os *honeypots*

	AT-01	AU-01	BR-01	BR-02	EC-01	NL-01	TW-01	UY-01	Total
AT-01	-	86%	85%	77%	50%	71%	6%	58%	18.138
AU-01	56%	-	79%	78%	54%	68%	3%	52%	27.636
BR-01	41%	58%	-	74%	57%	49%	3%	35%	37.563
BR-02	41%	63%	82%	-	68%	58%	4%	37%	33.920
EC-01	27%	44%	62%	67%	-	37%	3%	25%	34.253
NL-01	57%	84%	83%	88%	56%	-	6%	53%	22.284
TW-01	49%	43%	46%	66%	51%	57%	-	49%	2.194
UY-01	66%	91%	83%	80%	53%	74%	7%	-	15.917

Distribuição dos *Country Codes* de origem

Como os *honeypots* se localizam em diferentes posições do globo, torna-se interessante observar a origem dos transmissores dos *spams* recebidos por cada um deles, para verificar se a localização influencia no recebimento de mensagens de diferentes origens. A tabela 4 mostra os 5 *country codes* que mais enviaram mensagens para cada um dos *honeypots*. Em todos os *honeypots*, os *country codes* que mais abusaram são basicamente os mesmos, alterando pouco entre eles e, como seria de se esperar, estão todos entre os mais frequentes no global (tab. 2).

Tabela 4. Percentual do total de mensagens por *country codes* em cada *honeypots*

AT-01	AU-01	BR-01	BR-02	EC-01	NL-01	TW-01	UY-01
US (72,88%)	CN (22,98%)	BR (13,79%)	US (51,20%)	CN (23,72%)	US (84,73%)	US (67,83%)	US (70,82%)
PH (15,64%)	BR (11,29%)	CN (12,06%)	PH (23,92%)	TW (16,18%)	PH (9,90%)	PH (25,08%)	PH (13,64%)
CN (2,50%)	US (8,04%)	US (9,38%)	CN (6,53%)	BR (14,45%)	JP (1,57%)	TW (2,94%)	CN (3,97%)
JP (1,19%)	RU (5,56%)	RU (6,12%)	BR (3,29%)	US (10,11%)	CN (1,25%)	JP (1,93%)	BR (1,71%)
BR (1,15%)	IN (3,34%)	TW (3,76%)	JP (2,96%)	IT (3,24%)	TW (0,55%)	TH (0,22%)	JP (0,90%)

Entretanto, a proporção de mensagens enviadas por cada *country code* se altera em cada um. Dos oito *honeypots*, em cinco deles mais de 50% das mensagens recebidas são provenientes de US. Já nos outros três *honeypots*, a origem dos *spams* se distribui entre os vários *country codes*. Além disso, naqueles três, o número de mensagens concentrado nos cinco principais CCs é significativamente inferior ao dos demais, indicando uma maior distribuição da origem dos *spammers* nesses casos. Outro ponto interessante é que, embora os dois *honeypots* brasileiros estejam próximos geograficamente, a distribuição dos *country codes* originários das mensagens recebidas por eles é muito distinta. No BR-02, US representa mais de 50% dos *spams*, enquanto no BR-01 representa apenas 9,38%. Já as mensagens provenientes do próprio Brasil equivalem a mais de 11% dos *spams* recebidos pelo BR-01, sendo o segundo *country code* a mais abusar aquela máquina. Por outro lado, no BR-02 o *country code* BR é apenas o quarto, com cerca de 3% do total de mensagens recebidas.

Essa diferença de perfil chama a atenção pelo fato dos *honeypots* BR-01 e EC-01 estarem instalados em redes de "pior qualidade", segundo relatos dos membros da equipe que acompanham essas máquinas ao longo do tempo. Apesar de todos os coletores terem uma limitação de banda de entrada de 1 Mbps, a capacidade máxima dos links de acesso às redes onde estão aquelas máquinas é mais baixa que a do restante, além de serem conexões que ao longo do tempo se mostraram mais instáveis, com maiores taxas de perda, etc. Isso sugere que a semelhança entre essas máquinas (e sua diferença para as demais) pode ser

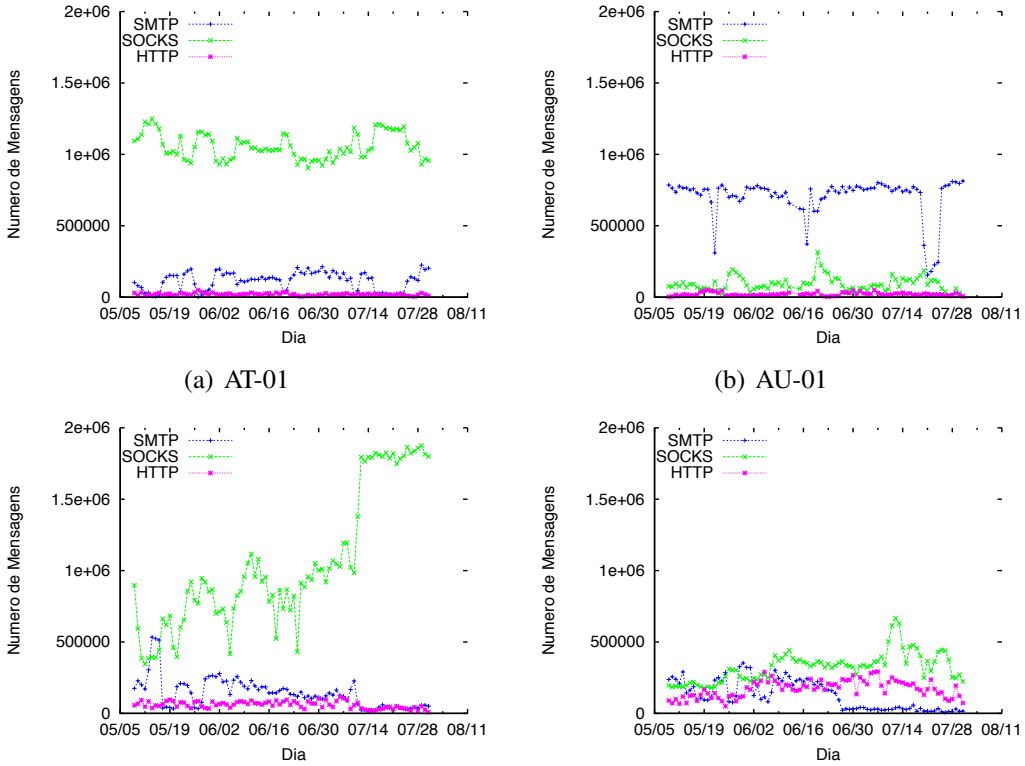
mais devida à sua conectividade que à sua posição na rede. Nesse sentido, TW-01 também se destaca, por ser o coletor na rede considerada de maior banda e qualidade.

Entretanto, fatores geográficos ainda podem desempenhar um papel na escolha dos *spammers*. Chama a atenção, por exemplo, o fato de que o único *honeypot* que não recebeu um grande volume de tráfego originado de CN (China) seja exatamente TW-01 (Taiwan). A razão para essa diferença não está clara e exigiria uma análise mais abrangente das condições (inclusive políticas) daquela região.

Tráfego de *spam* por protocolo

O tráfego apresentado por todos os *honeypots* apresenta variações ao longo do tempo. Entretanto, mesmo havendo variação, é possível perceber características gerais em cada *honeypot*, como mostrado nas figuras 3 e 4. Observando o AT-01, por exemplo, percebemos que o tráfego majoritário presente é causado por SOCKS, bem como no BR-02. Já o *honeypot* AU-01 possui maior parte do tráfego causado por SMTP, enquanto EC-01 possui um equilíbrio no tráfego gerado por cada protocolo, com uma queda no número de mensagens no final do período analisado. Considerando a discussão anterior, AT-01 e BR-02 estão em redes acadêmicas de boa qualidade, enquanto EC-01 está em uma rede mais limitada.

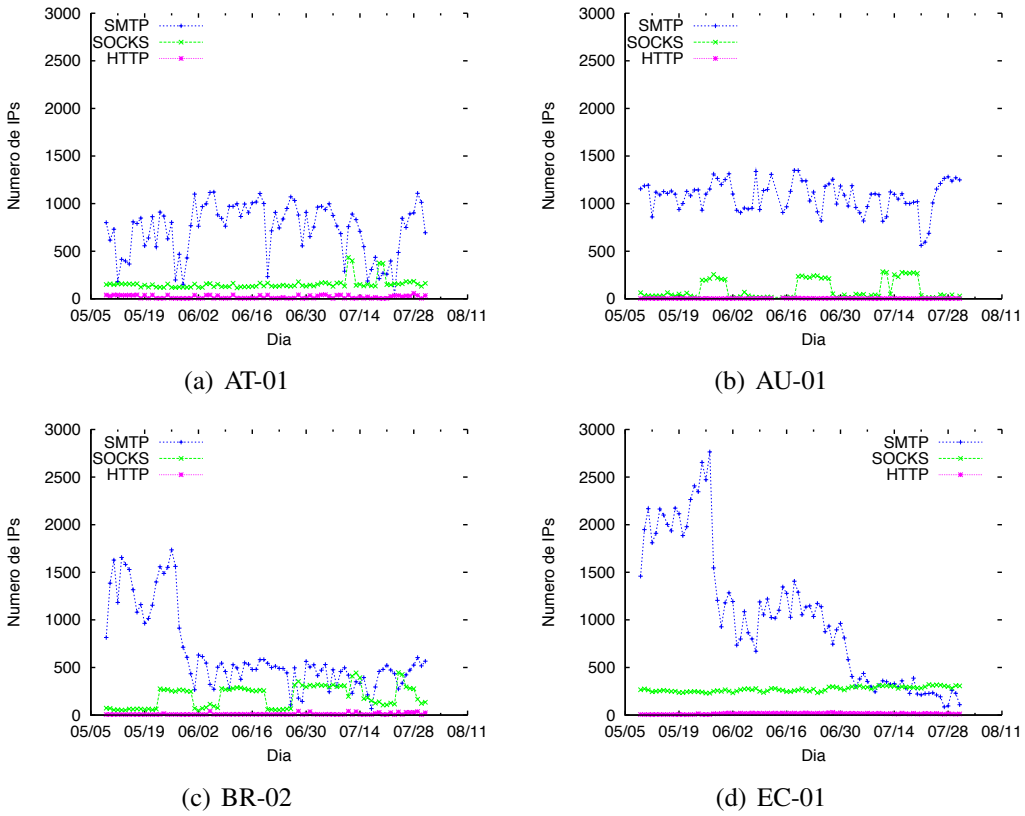
Com relação ao número de endereços de origem, todos eles apresentam número pequeno de transmissores utilizando SOCKS que, conforme observado anteriormente, enviam um número alto de mensagens. Além disso, possuem um número alto de transmissores explorando SMTP e um número quase irrisório de endereços de origem abusando a máquina através de HTTP.



(a) AT-01 (b) AU-01

(c) BR-02 (d) EC-01

Figura 3. Número de mensagens por protocolo



(a) AT-01 (b) AU-01

(c) BR-02 (d) EC-01

Figura 4. Número de endereços IP por protocolo

Mensagens de teste recebidas pelos *honeypots*

Uma característica interessante encontrada em todos os 8 *honeypots* é o padrão de mensagens de teste recebidas. Essas mensagens são geradas pelos *spammers* periodicamente e contêm em seu corpo a identificação da máquina que está aparentemente sendo utilizada para entregar o *spam* (nesse caso, um dos *honeypots*). Elas são identificadas no tráfego coletado e recebem tratamento especial, sendo as únicas mensagens que o *honeypot* realmente entrega, periodicamente, para garantir que o *spammer* ache que seu objetivo está sendo alcançado e continue a utilizar a nossa infraestrutura. Apesar de não serem apresentadas por questões de espaço, as curvas que representam o número de endereços IP que enviaram mensagens de teste são muito semelhantes entre todos os coletores, com correlações extremamente elevadas (muito próximas de 1,0, para a maioria dos casos). Além disso, o número de endereços de origem em comum é elevado, sendo maior que 90% em grande parte dos casos, comparando os *honeypots* par-a-par.

Foram encontrados 1.608 endereços IP enviando mensagens de teste em todos os *honeypots*. Desse total, 324 (20,2%) enviaram apenas mensagens de teste no período observado, enquanto os 79,9% restantes enviaram tanto mensagens de teste quanto *spams*. Outra característica desses endereços é que a maioria (77,5%) é proveniente de redes domésticas, de acordo com dados de *black lists* consultadas durante a coleta. Com base ainda nessas *black lists*, 32,5% são, garantidamente, máquinas infectadas por algum tipo de *malware*[2]. Além disso, 230 máquinas enviaram mensagens de teste para todos os

[2]www.spamhaus.org

honeypots e, dessas, 97% enviaram tanto mensagens de teste quando *spams* e 73,91% são máquinas infectadas. Esses fatores indicam que, normalmente os testadores são máquinas infectadas presentes em redes domésticas e que, além de testarem o funcionamento da máquina abusada, continuam enviando *spams* em paralelo a essas mensagens.

4.3. Análise baseada em campanhas

Em algumas das análises utilizamos o conceito de campanhas de *spam*. Uma campanha é um conjunto de mensagens que possuem um objetivo comum e uma mesma estratégia de disseminação [Guerra et al. 2008]. Neste trabalho utilizamos o algoritmo de agrupamento FPCluster para agrupar as mensagens com base em diversos de seus atributos e identificar as estratégias de ofuscação utilizadas. Esse algoritmo constrói uma árvore de padrões frequentes que é usada para extrair os padrões de agrupamento das mensagens [Totti et al. 2012, Guerra et al. 2008]. O mecanismo desenvolvido anteriormente foi estendido para realizar a identificação de campanhas existentes por vários dias, a partir do agrupamento de campanhas com mesmos atributos encontradas em dias consecutivos. Ao se realizar esse agrupamento, 448.123 campanhas de um dia foram agrupadas em 155.696 campanhas de maior duração.

A tabela 5 apresenta o percentual de campanhas em comum entre cada par de *honeypots* em relação ao total de campanhas do *honeypot* da linha. Os *honeypots* AT-01 e NL-01 possuem 23.367 campanhas em comum, o que representa 81,33% do total das campanhas do AT-01. TW-01 possui um número pequeno de campanhas em comum com os demais *honeypots*, e, em consequência, possui características muito distintas das demais máquinas. Essas indicações nos levam a crer que *honeypots* com muitas campanhas em comum possuem muitas similaridades, o que não acontece nos casos em que o número de campanhas em comum é baixa.

Tabela 5. Percentual de campanhas em comum entre os *honeypots*

	AT-01	AU-01	BR-01	BR-02	EC-01	NL-01	TW-01	UY-01	Total
AT-01	-	5,04%	9,06%	28,59%	0,82%	81,33%	1,95%	4,31%	28.731
AU-01	31,05%	-	52,43%	30,92%	21,30%	14,65%	7,98%	22,00%	4.663
BR-01	50,28%	47,25%	-	22,76%	15,48%	38,84%	5,80%	14,45%	5.175
BR-02	33,06%	5,80%	4,74%	-	6,02%	69,84%	2,50%	2,65%	24.844
EC-01	5,97%	25,23%	20,35%	37,98%	-	8,23%	6,22%	6,05%	3.936
NL-01	40,95%	1,20%	3,52%	30,40%	0,57%	-	1,71%	1,69%	57.068
TW-01	0,74%	0,49%	0,39%	0,82%	0,32%	1,28%	-	15,69%	76.215
UY-01	6,46%	5,36%	3,90%	3,44%	1,24%	5,05%	62,42%	-	19.159

Mais um fator interessante com relação ao impacto da rede em que o *honeypot* se encontra, é que aqueles localizados nas redes com maior qualidade possuem maior número de campanhas, enquanto os dois identificados anteriormente como estando em redes de baixa qualidade (BR-01 e EC-01) possuem um número muito menor de campanhas distintas. As campanha observadas em alguns momentos têm impacto direto sobre o perfil de tráfego observado, à medida que campanhas se iniciam ou terminam.

Aumento do número de endereços IP utilizando SOCKS

Nos gráficos referentes aos transmissores encontrados nos *honeypots* AT-01 e BR-02, (figs. 4(a) e 4(c)), são perceptíveis períodos em que há um aumento no número de endereços IP que enviam mensagens utilizando SOCKS. Nesses períodos é visível um degrau na curva, que sobe no início do intervalo e volta ao seu valor mais baixo após alguns dias. Observando as curvas para o número de mensagens observadas (figs. 3(a)

e 3(c)), podemos observar variações no tráfego SOCKS no mesmo período.

Para tentar explicar este comportamento, focamos aqui no primeiro aumento ocorrido no *honeypot* BR-02, por volta do dia 23/05. Identificamos as principais campanhas do período e os endereços das máquinas que dela participavam. O resultado pode ser visto na tabela 6. Inicialmente, aquelas campanhas são praticamente inexistentes, com um número muito pequeno de transmissores; no dia em que ocorre o aumento do tráfego elas passam a se originar de um grande número de endereços. Há claramente uma relação com o surgimento dessas novas campanhas, a participação de novos transmissores e o aumento do tráfego SOCKS.

Tabela 6. Número de endereços IP das 5 maiores campanhas do dia 23/05 ao dia 31/05

	22/05	23/05	...	31/05	01/06
Campanha 1	2	182	...	175	8
Campanha 2	2	170	...	174	9
Campanha 3	-	137	...	128	-
Campanha 4	-	137	...	130	-
Campanha 5	-	137	...	127	-

Queda do número de transmissores utilizando SMTP

Como pode ser visto nas figuras 4(c), e 4(d), durante o período de 26/05 e 01/06, houve uma queda visível no número de endereços IP que enviaram mensagens utilizando o protocolo SMTP em alguns *honeypots*. Para entender o motivo da queda, observamos as principais campanhas envolvidas neste período. A tabela 7 contém as maiores campanhas no dia 26/05, dia em que se iniciou a queda, e nos dias seguintes. Como podemos observar, essas campanhas foram perdendo força, algumas chegando até a serem encerradas. Isso sugere que a queda se deveu ao término dessas campanhas.

Tabela 7. Número de endereços IP das 5 maiores campanhas do dia 26/05

	26/05	27/05	28/05	29/05	30/05	31/05	01/06
Campanha 1	469	476	29	31	20	9	4
Campanha 2	256	180	0	0	0	0	0
Campanha 3	237	217	123	63	1	28	6
Campanha 4	236	11	0	0	0	0	0
Campanha 5	220	144	0	0	0	0	0

Outras análises semelhantes foram realizadas para outros pontos onde observamos mudanças bruscas no perfil do tráfego. Através dessas análises, concluimos que eventos externos, como o início ou término de uma campanha, têm forte impacto sobre os detalhes do perfil de tráfego observado, devendo ser considerado em análises de tráfego desse tipo.

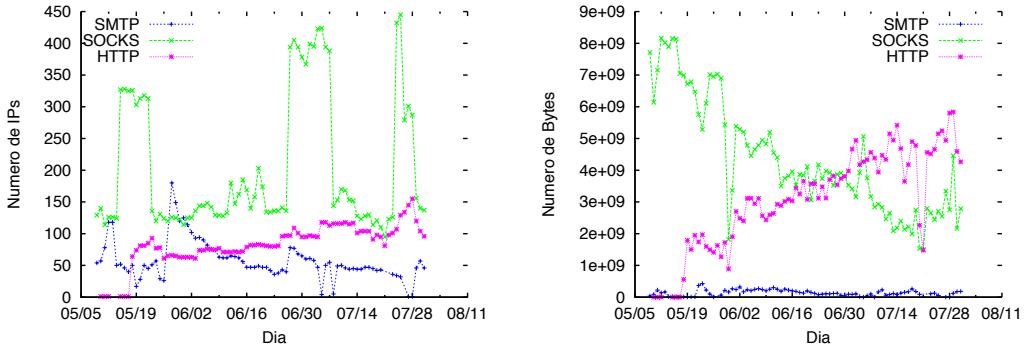
4.4. O *honeypot* TW-01

Conforme mencionado anteriormente, o perfil que mais difere dos demais foi o do *honeypot* TW-01. Apesar de apresentar um número muito pequeno de transmissores (2.194), o número de mensagens é muito alto, com mais de 207 milhões de mensagens, sendo maior que o de todos os demais *honeypots* individualmente.

Outro aspecto que diferencia TW-01 dos demais é o tráfego HTTP. Enquanto os outros 7 *honeypots* enviaram poucas mensagens utilizando esse protocolo ao longo do período, no TW-01 houve mais de 82 milhões de *spams* enviados dessa forma. Esse número representa quase 73% do total de mensagens utilizando HTTP recebidas por todos os *honeypots*. Todas essas mensagens recebidas pelo TW-01 foram enviadas a partir de

apenas 345 endereços IP.

Ainda com relação ao número de *spams* recebidos, o tráfego SMTP foi muito pequeno, representando apenas 1,4% do total. Já o tráfego SOCKS é bastante significativo, sendo utilizado por mais de 1.500 endereços IP e totalizando mais de 122 milhões de mensagens. Outro fator interessante desse *honeypot* está no número de campanhas presente em seu tráfego. Durante o período de 84 dias, mais de 76 mil campanhas distintas foram observadas no TW-01, o que representa quase 49% do total de campanhas presentes em todos os *honeypots*.



(a) Endereços IP por protocolo (b) Bytes por protocolo

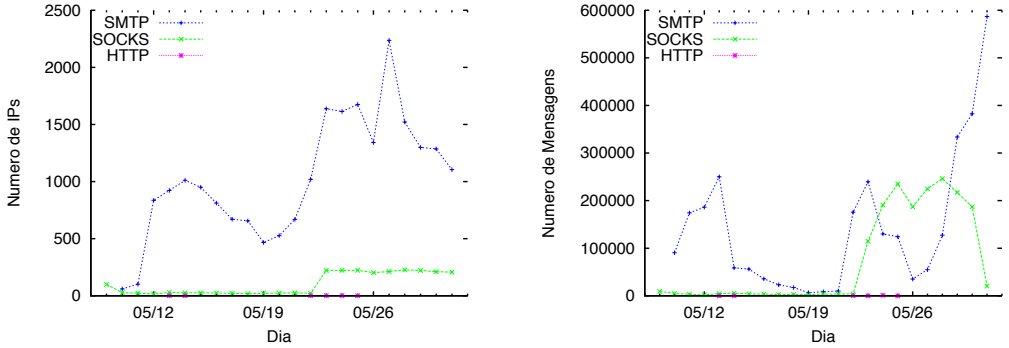
Figura 5. Características do *honeypot* TW-01

Como mencionado anteriormente, TW-01 está localizado em uma rede de alta velocidade e de qualidade superior à de todos os demais coletores. Esse perfil de rede parece beneficiar *spammers* que usam transmissão "por atacado", talvez porque esses transmissores tendem a permanecer ativos por mais tempo e se beneficiam da estabilização do algoritmo de controle de congestionamento de TCP. Dessa forma, eles teriam uma condição privilegiada para aproveitar a banda disponível, dificultando o acesso a máquinas de menos capacidade que enviam apenas poucas mensagens periodicamente. Provavelmente pelo mesmo motivo, aumentos significativos no número de transmissores usando SOCKS observados durante período em três momentos não tiveram grande impacto sobre o tráfego observado, pois os novos transmissores se viam em desvantagem ao concorrer pela banda disponível com os transmissores pesados já existentes.

4.5. Início do ataque a uma máquina vulnerável na rede

Por fatores externos ao projeto, o endereço IP de um dos *honeypots* foi alterado no início do nosso período de coleta. Apesar dessa máquina já estar em atividade há um longo período, a troca de seu endereço acabou simulando o aparecimento de uma nova máquina na rede. Com isso, foi possível analisar o processo do descobrimento de *proxies* e *mail relays* em uma máquina vulnerável por um *spammer*. Os gráficos da figura 6 mostram o comportamento do tráfego recebido por aquele *honeypot* logo após o seu ressurgimento na rede, usando o novo endereço pela primeira vez.

Logo no primeiro dia após o surgimento da máquina, cerca de 100 endereços IP já começaram a abusar da mesma, enviando cerca de 9 mil mensagens. Todos esses transmissores abusaram da máquina através do protocolo SOCKS. Uma possível explicação para o seu surgimento é que *spammers* que utilizam SOCKS vasculham a rede à procura de *proxies* abertos e quando encontram já começam a enviar *spams* através dos mesmos. Já os transmissores que utilizam SMTP apareceram somente no segundo dia após o apa-



(a) Endereços IP por protocolo (b) Mensagens por protocolo

Figura 6. Tráfego após surgimento do *honeypot* BR-01

recimento do *honeypot*, coincidindo com o aparecimento da primeira mensagem de teste enviada por um *spammers*. Esse fato leva a crer que os *spammers* que utilizam SMTP se valem mais de mensagens de teste para confirmar se a máquina a ser abusada realmente entrega as mensagens.

Após o primeiro dia, o número de transmissores enviando *spams* com SOCKS se estabiliza, com apenas alguns picos, em que o número de endereços aumenta consideravelmente (fato discutido na seção 4.3), assim como o número de mensagens enviadas por eles. Com relação aos endereços utilizando SMTP, a partir do segundo dia o número de mensagens de teste aumenta, coincidindo com o aumento do número de transmissores e do número de mensagens enviadas. A maioria dos transmissores que passaram a usar a máquina por SMTP nunca chegaram a fazer uma varredura da máquina antes do primeiro envio, nem usaram mensagens de teste, o que sugere que foram informados da existência do *honeypot* por algum canal externo (provavelmente através dos canais de comando e controle de uma *botnet*).

5. Conclusões e Trabalhos Futuros

O combate ao *spam* é uma tarefa contínua, onde se busca sempre entender melhor a evolução dos *spammers* e suas técnicas de disseminação de mensagens. Este trabalho buscou estender o entendimento sobre padrões de comportamento usado por esses transmissores com o objetivo de enviar seu tráfego mantendo-se ocultos atrás de máquinas intermediárias. Para isso, utilizamos uma análise do tráfego de *spam* no nível da rede.

Diferentemente de outros trabalhos anteriores na área, utilizamos um conjunto de máquinas geograficamente dispersas pelo mundo, a fim de coletar tráfego de *spam* em diferentes pontos da Internet. Isso nos permitiu observar que diversas características do tráfego, como tamanho das mensagens, endereços de origem e o uso de mensagens de teste, se repetem em diferentes locais. Por outro lado, certos detalhes como volume de tráfego instantâneo observado e a distribuição do tráfego entre protocolos e origens pode ser mais afetado por características da conectividade das máquinas atacadas e pelo padrão de ocorrência de campanhas que por fatores de localização. Acreditamos que essas observações serão úteis para outros pesquisadores que busquem mais informações sobre a origem do *spam*, a fim de viabilizar novas pesquisas na área.

Como trabalhos futuros, pretendemos avançar nas análises com a observação do

conteúdo específico das mensagens, para melhor entender a relação com localidades específicas, bem como realizar uma análise mais profunda dos padrões de tráfego de rede para confirmar as observações sobre o impacto da qualidade da conexão da rede atacada ao restante da Internet.

Agradecimentos

Esta pesquisa foi parcialmente financiada pelo Instituto Nacional de Ciência e Tecnologia para a Web - InWeb (MCT/CNPq 573871/2008-6), NIC.br, CNPq, Capes e FAPEMIG.

Referências

CERT.br (2013). SpamPots Project. *http://honeytarg.cert.br/spampots*.

Gomes, L. H., Cazita, C., Almeida, J. M., Almeida, V., e Jr., W. M. (2007). Workload Models of Spam and Legitimate E-mails. *Performance Evaluation*, 64(7-8):690–714.

Goodman, J., Cormack, G. V., e Heckerman, D. (2007). Spam and the ongoing battle for the inbox. *Communications ACM*, 50:24–33.

Guerra, P. H. C., Guedes, D., Meira Jr., W., Hoepers, C., e Steding-Jessen, K. (2008). Caracterização de estratégias de disseminação de spams. Em *SBRC 2008*, Rio de Janeiro, Brasil.

John, J., Moshchuk, A., Gribble, S. D., e Krishnamurthy, A. (2009). Studying Spamming Botnets Using Botlab. Em *6th USENIX Symp. on Networked Systems Design and Implementation*, Boston, EUA.

Kim, J. e Choi, H. (2008). Spam Traffic Characterization. Em *Int'l Technical Conference on Circuits/Systems, Computers and Communications*, Shimonoseki City, Japão.

Kokkodis, M. e Faloutsos, M. (2009). Spamming botnets: Are we losing the war? Em *Proceedings of the 6th Conference on e-mail and anti-spam (CEAS)*.

Las-Casas, P. H., Guedes, D., Almeida, J. M., Ziviani, A., e Marques-Neto, H. T. (2013). Spades: Detecting spammers at the source network. *Computer Networks*, 57(2):526 – 539. Botnet Activity: Analysis, Detection and Shutdown.

Las-Casas, P. H. B., Guedes, D., Almeida, J. M., Ziviani, A., e Marques-Neto, H. T. (2011). Detecção de Spammers na Rede de Origem. Em *SBRC 2011*, Campo Grande, Brasil.

Newman, M. E. J., Forrest, S., e Balthrop, J. (2002). Email Networks and the Spread of Computer Viruses. *Physical Review E*, 66(3):035101.

Pu, C. (2006). Observed trends in spam construction techniques: A case study of spam evolution. Em *In Third Conference on Email and Anti-Spam (CEAS)*.

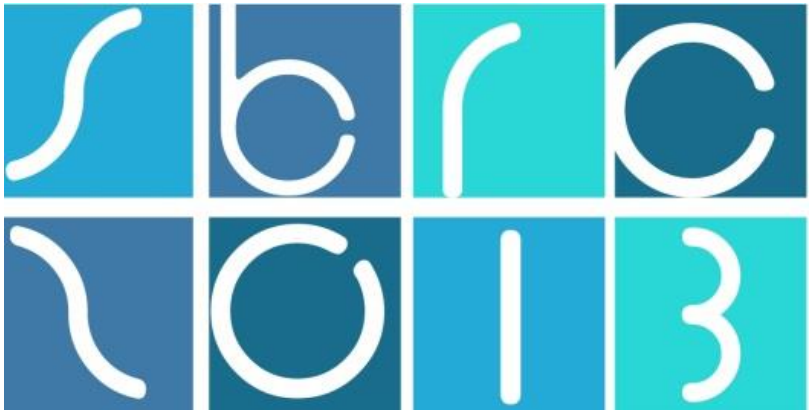
Ramachandran, A. e Feamster, N. (2006). Understanding the Network-Level Behavior of Spammers. *SIGCOMM Computer Communication Review*, 36(4):291–302.

Sipior, J. C., Ward, B. T., e Bonner, P. G. (2004). Should spam be on the menu? *Communications ACM*, 47(6):59–63.

Symantec (2011). Internet Security Threat Report, Volume 17. *http://www.symantec.com/threatreport*.

Totti, L. C., Moreira, R. E. A., Fazzion, E., Fonseca, O., Meira Jr., W., Guedes, D., Hoepers, C., Steding-Jessen, K., e Chaves, M. H. P. (2012). Impacto da Evolução Temporal na Detecção de Spammers na Rede de Origem. Em *SBRC 2012*, Ouro Preto, Brasil.

Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., e Osipkov, I. (2008). Spamming botnets: signatures and characteristics. *SIGCOMM Computer Communication Review*, 38(4):171–182.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 20

Redes Veiculares 1

Certificados Sociais para Segurança em Redes Veiculares Tolerantes a Interrupções

Thiago R. Oliveira[1,2], Sérgio de Oliveira[2], Daniel F. Macedo[1], José Marcos S. Nogueira[1]

[1] Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627, Pampulha - Belo Horizonte, MG - Brasil

[2] Universidade Federal de São João Del Rei
Campus Alto Paraopeba - Ouro Branco, MG - Brasil

thiagool@ufsj.edu.br, sergiool@ufsj.edu.br, damacedo@dcc.ufmg.br, jmarcos@dcc.ufmg.br

Abstract *– Disruption tolerant vehicular networks appear due to the search of information by drivers, which build mobile ad hoc networks that may suffer from long interruptions. This paper proposes a security model that employs self-signed certificates, enabling cars to share keys through direct contacts between two acquaintances, that warrant their identity, so they sign the reciprocal certificate. One certificate signed by a third party can be validated if its public key is available to the other party. Further, the reputation mechanism can still identify certificates of trusted users. The evaluation shows the behavior of a vehicular network that uses social certificates as a cryptographic security mechanism.*

Resumo *– Redes veiculares tolerantes a interrupções surgem pela busca de informações dos motoristas, que podem formar redes ad hoc móveis onde a comunicação pode sofrer longas interrupções. Este artigo propõe um modelo com certificados auto-assinados para possibilitar o compartilhamento de chaves entre duas pessoas que se conhecem e garantem sua identidade, que então assinam o certificado do outro pelo contato direto. Um certificado assinado por um terceiro pode ser validado se sua chave pública estiver disponível à outra parte. Um mecanismo de reputação ainda pode identificar certificados de usuários confiáveis. As avaliações realizadas mostram o comportamento de uma rede veicular com utilização de certificados sociais como mecanismos de segurança criptográficos.*

1. Introdução

Redes veiculares tolerantes a atrasos e interrupções visam disponibilizar novas tecnologias e aplicações, que podem tornar mais seguro o trânsito nas ruas e avenidas por meio de maior comunicação entre os usuários. Em regiões urbanas ou rodovias, geralmente não existe uma infraestrutura de rede ou a mesma não atinge áreas contíguas.

As redes veiculares são heterogêneas, aproveitando a infraestrutura existente nas cidades como as redes metropolitanas e os dispositivos dos próprios usuários, dada a popularização de dispositivos móveis como *notebooks*, *tablets* e *smartphones* com mais recursos disponíveis, que podem ser utilizados por motoristas e autoridades de trânsito. Alguns equipamentos podem estar conectados à Internet móvel, com cobertura em expansão nas áreas urbanas, o que pode fornecer diversas informações sobre o trânsito. Esse artigo aborda a questão de segurança para a meta de comunicação para aplicações

de Sistemas Inteligentes de Transporte do projeto (CIA)[2] da RNP, que envolve várias universidades federais por 2 anos com foco no desenvolvimento de cidades inteligentes.

Uma VANET (*Vehicle Ad Hoc NETworks*) é uma rede móvel *ad hoc* (MANET) [6] com algumas características importantes, como restrições a leis de circulação e vias específicas. Os veículos podem ter velocidades altas, a topologia da rede formada é bastante dinâmica e o conhecimento dos vizinhos alcançáveis pelo veículo muda com frequência. O tempo de contato entre os veículos pode não ser suficiente para estabelecer uma conexão e transferir dados, o que pode fragmentar a rede.

Redes veiculares são formadas por veículos e por equipamentos fixos localizados às margens das ruas e estradas, para permitir a comunicação entre veículos e de veículos com algum ponto de acesso como mostrado na Figura 1. Para tanto, deve-se considerar que as redes formadas por esses elementos possuem várias limitações, pois necessitam de certas condições que nem sempre são satisfeitas. Na comunicação entre veículos, a conectividade fim-a-fim é altamente suscetível à interrupção de comunicação.

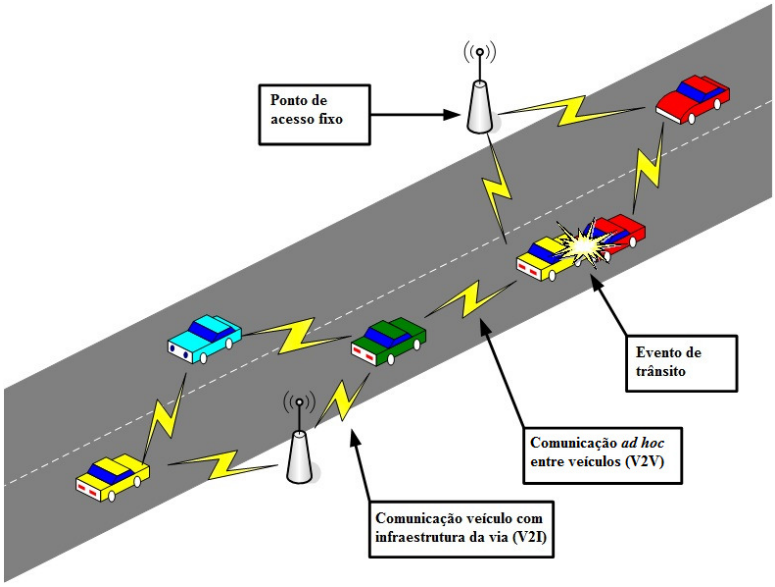


Figura 1 - Comunicação nas redes veiculares

Os dispositivos móveis podem ser utilizados pelos motoristas para obtenção e divulgação de informações, possibilitando alterar as rotas de trânsito utilizadas. Alguma empresa, por exemplo posto de combustível, pode tentar interferir nessas rotas. Nesse contexto, observa-se a necessidade de aumentar a segurança entre esses recursos de forma que troquem mensagens confiáveis, mesmo sem conexão no momento do envio.

A utilização da arquitetura de rede DTN (*Disruption Tolerant Networks*) [1] é necessária para se trabalhar em redes com conectividade intermitente ou com atrasos longos. Uma DTN utiliza um esquema de comunicação assíncrona e não há necessidade de um caminho fim-a-fim. Os nós seguem o paradigma guardar-carregar-repassar mensagens (*store-carry-foward*), podendo mantê-las em um *buffer* caso o nó não tenha conexão direta com o destino.

A segurança torna-se mais desafiadora em redes DTN devido às suas características específicas, como mobilidade imprevisível e latência variável. É necessário eliminar mensagens expiradas e evitar vazamento de informações, devido à conectividade esporádica e grande possibilidade de atraso em transmissão de mensagens. Esses

aspectos devem ser considerados para utilizar redes veiculares tolerantes a interrupções com objetivo de incentivar a participação de todos os veículos em uma rede social.

É necessária uma alternativa que torne viável a autenticação das mensagens, pois o gerenciamento de chaves com tolerância a desconexões ainda é uma questão em aberto. O estabelecimento de chaves entre elementos da rede permite a autenticação no enlace. Na criptografia de chave pública, por exemplo, cada usuário tem um par de chaves privada e pública. Um esquema de distribuição de chaves seguro e eficiente possibilita a autenticação dos nós da rede. Entretanto, nenhum esquema de gerenciamento de chaves ainda é reconhecido como adequado para redes tolerantes a interrupções [6].

Um certificado é um arquivo, assinado digitalmente por uma autoridade certificadora confiável, confirmando a identidade do usuário e contendo uma cópia confirmada da chave pública do usuário. A certificação digital pode garantir alguns requisitos de segurança essenciais na rede. Um certificado auto-assinado ou assinado por um terceiro, não reconhecido como autoridade certificadora oficial [7], pode ser validado se sua chave pública estiver disponível à outra parte.

Este trabalho investiga o compartilhamento de chaves entre elementos da rede por meio do uso de certificados em redes veiculares tolerantes a interrupções, sendo tratadas como redes sociais. Propomos o uso de certificados sociais para troca de material criptográfico em relacionamentos cotidianos, como encontro com amigos ou ajuda a outro veículo, para estabelecer um grau de confiança entre os elementos da rede. Associado a esse material, pode-se utilizar um mecanismo de reputação de forma que os usuários beneficiados possam também emitir uma assinatura para o certificado de quem o ajudou. O princípio de reputação é uma recompensa para usuários que se comportam bem na divulgação de informações do trânsito e repassam informações verdadeiras.

O conteúdo deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados, enquanto a Seção 3 discorre sobre o modelo de rede. A Seção 4 define os mecanismos de segurança considerados para redes veiculares tolerantes a interrupções. O modelo de certificação proposto é descrito na Seção 5 e sua avaliação é apresentada na Seção 6. Na Seção 7, as conclusões e trabalhos futuros são apresentados.

2. Trabalhos Relacionados

Veículos que recebem informações de outros veículos ou entidades da rede precisam saber a confiabilidade de quem gerou aquela informação. O nível de privacidade pode ser ajustado pelo usuário, sendo ainda aberta a área de anonimicidade e privacidade adaptativa, onde usuários poderiam selecionar a privacidade que desejam ter [3]. Tais questões são abordadas neste trabalho tratando usuários numa rede social.

As redes veiculares possuem desafios específicos em relação à segurança [4]. Pouca atenção tem sido dedicada à segurança em redes veiculares, que é um desafio crucial e um elemento chave nesse aspecto é a confiança [5], devido ao grande número de nós independentes envolvidos e a presença de fatores humanos na rede que podem aumentar a tendência de mau comportamento. O conhecimento prévio existente entre os usuários da rede é utilizado para definir a confiabilidade deste trabalho.

A maioria das propostas para redes veiculares trata de problemas específicos, como a autenticação e a privacidade dos nós da rede. Um mecanismo de reputação para redes veiculares considerando tolerância a atrasos e desconexões é proposto em [7], mas assume que todos os veículos são certificados. Algumas propostas existentes descrevem

arquiteturas de segurança mais genéricas [9]. Este trabalho se diferencia por atender os vários requisitos de segurança das redes veiculares e pela tolerância a interrupções.

Entre as propostas que têm sido publicadas relativas à segurança em redes DTN, em [6] foram apresentadas algumas idéias preliminares sobre a distribuição e gerenciamento de chaves para DTN, mas percebe-se que ainda são questões abertas. Um modelo para gerenciamento de segurança considerando aspectos das redes DTN foi proposto em [8]. Tais propostas não foram reconhecidas como soluções definitivas ou dependem da existência de alternativas para gerenciamento de chaves, como apresentado neste artigo.

Iniciativas existentes para redes veiculares são direcionadas para um ou outro cenário, muitas vezes restringindo o acesso ou sem considerar tolerância a interrupções. Tanto quanto podemos saber, não existe na literatura proposta para segurança em redes veiculares que viabilize a participação de todos os veículos com mecanismos de segurança implícitos, considerando as restrições de recursos dessas redes.

3. Modelo de rede e comunicação

Redes veiculares são heterogêneas em hardware, visto que devem estimular a participação de todos os motoristas através da utilização de seus *notebooks*, *palmtops*, *tablets* ou *smartphones*, com várias tecnologias de rede para comunicação entre esses nós. Alguns dos nós estão conectados, enquanto os demais podem não ter conectividade. Tais conexões podem cair a qualquer momento, devido a falhas, deslocamentos ou outros tipos de eventos. Nós podem participar ou deixar a rede dinamicamente.

Os nós da rede podem representar veículos, nós sensores para o monitoramento, pontos de acesso fixos nas vias para prover conexão externa aos veículos e servidores com pessoas qualificadas para controle do tráfego. Sensores e etiquetas RFID podem ser utilizados para monitoramento do trânsito e envio dessas informações. Os equipamentos podem estar limitados à comunicação *ad hoc* ou perderem conectividade com a rede metropolitana em alguns trechos das vias.

A característica de acesso eventual às redes metropolitanas pode ser usada para atualizar listas de usuários e chaves, mas não para garantir esses requisitos de segurança em tempo real. A comunicação *ad hoc* entre os veículos é motivada, pois não se pode considerar que há conexão constante com equipamentos de uma rede metropolitana sem fio (WMAN), disponibilizadas pelo poder público ou via redes celulares 3G/4G.

Dispositivos dos próprios usuários devem ser utilizados na rede veicular de forma que todos os motoristas possam participar sem a necessidade de adquirir novos equipamentos. A utilização do *smartphone* já adquirido causa maior incentivo que dispositivos novos embarcados no carro. Sempre que houver a tecnologia disponível, será utilizado o IEEE 802.11p que padroniza a comunicação em redes veiculares [4].

A comunicação entre os usuários pode ocorrer de forma direta, através de *Bluetooth* ou modo *ad hoc;* informações podem ser transmitidas para os pontos de acesso fixos e compartilhadas; além da sincronização de listas de usuários quando os equipamentos dos usuários estiverem diretamente ligados à Internet, na sua residência por exemplo.

Redes veiculares possibilitam uma variedade muito grande de configurações. A ampla diversidade de nós participantes, que vão desde nós sensores a robustos servidores de controle de tráfego, a mobilidade e as aplicações impedem que a caracterização do problema de provimento de segurança seja feita de forma única.

Objetiva-se neste trabalho propor soluções de segurança para redes veiculares sem hierarquia de nós e com formas de distinção entre as mensagens que devem ser consideradas para as decisões. Se uma informação afeta a rota utilizada pelo veículo no deslocamento, deve-se excluir a possibilidade do informante ser alguém diretamente interessado em rotas específicas, como um posto de combustível na via sugerida.

4. Mecanismos de Segurança para Redes Veiculares

Os dados enviados pelos motoristas devem ser tratados, de forma a garantir o anonimato e a privacidade dos usuários. Além disso, devem ser analisados para a detecção de informações maliciosas. Por exemplo, o dono de um posto anuncia que certa avenida está com um fluxo muito melhor que a realidade, pois quer aumentar a quantidade de carros que passam na região para aumentar o seu lucro, ou um usuário mal intencionado que quer provocar um engarrafamento em uma região da cidade.

Os requisitos de segurança em redes veiculares podem variar de acordo com as situações e cenários em que elas são utilizadas, um fator essencial considerado neste trabalho é a conectividade intermitente. A mobilidade presente nesse cenário se apresenta como um desafio fundamental para as redes veiculares, sendo necessário suportar recursos altamente variáveis dentro de curtos períodos de tempo, ou ainda atrasos de propagação extremamente longos.

Este trabalho propõe o roteamento com seleção de prioridade de pacotes para replicação, utilização de reputação e um mecanismo alternativo para substituição do gerenciamento de chaves; além de considerar a existência de mecanismos de detecção de intrusos, técnicas de criptografia e revogação de nós.

As soluções de segurança de maior interesse para este trabalho foram organizadas e classificadas a seguir em componentes de acordo com seus objetivos.

4.1. Roteamento prioritário

O roteamento é uma tarefa crítica porque um inimigo pode se inserir na rede para promover um ataque de negação de serviço. A maior parte dos protocolos de roteamento propostos para redes DTN não considera o aspecto da segurança como um dos objetivos principais. Portanto, é importante tratar o roteamento para garantir a segurança da rede.

Os protocolos de roteamento mais divulgados foram considerados: *Direct Delivery,* repasse de mensagens somente ao destinatário; *PRopHet,* repasse das mensagens para nós com maior probabilidade de entrega; *Epidemic,* distribuição de cópias das mensagens para toda a rede; *Spray and Wait,* distribuição de um número determinado de cópias para cada mensagem, que é dividido por dois a cada salto no modo binário [12].

Decidiu-se utilizar nesse trabalho o protocolo *Spray and Wait* no modo binário, pois a melhor defesa em redes DTN contra ataques de perda maliciosa de pacotes é o uso de caminhos múltiplos. O ataque mais simples consiste em fazer com que um nó descarte todos os pacotes que recebe. Para protocolos de repasse, cada descarte é um pacote perdido, pois não há cópia em outros nós.

Propõe-se o uso de *flags* para definir replicação de maior prioridade como indicador para caracterizar urgência e consequente tempo de vida (TTL) das mensagens, pois há mensagens que perdem o sentido se não forem entregues em certo espaço de tempo.

Tais *flags* auxiliam contra intrusos, que podem inundar continuamente a rede com envio de pacotes para qualquer nó e nunca repassarem qualquer pacote recebido de outros nós.

Permitir aos motoristas que expressem aspectos da importância das mensagens da rede pode ser um benefício considerável, tanto para os usuários quanto para a infraestrutura da rede que os suporta [1]. Dois aspectos que caracterizam a preferência dos usuários são uma noção de vida útil e uma relação de prioridade entre as mensagens, de forma que uma mensagem deve ser entregue antes de outra, se possível.

Esse trabalho considera que as aplicações que utilizam as redes veiculares devem obrigar a definição de prioridade das mensagens como: Baixa, Média, Alta ou Urgente. Mensagens definidas como urgentes possuem um tempo de vida determinado pela rede e inferior ao das demais, para descarte após término desse tempo.

4.2. Reputação

Quando um membro autenticado da rede atenta contra o funcionamento das aplicações, torna-se necessário o uso de mecanismos de segurança adicionais. Sistemas de reputação são utilizados para garantir o comportamento cooperativo em redes distribuídas como MANETs, VANETs ou redes *Peer-to-Peer* (P2P). [7]

Em VANETs, a reputação de um veículo pode ser considerada como um histórico sobre as informações repassadas aos outros veículos. Essa reputação é então utilizada como critério importante na tomada de decisões, tais como encaminhar ou descartar pacotes enviados por esse veículo, considerá-lo ou desconsiderá-lo como opção no roteamento de dados, considerar ou desconsiderar informações por ele repassadas, etc.

O princípio de reputação é uma recompensa para os usuários que se comportam bem na divulgação de informações do trânsito e repassam informações verdadeiras. Nesse trabalho, os veículos que divulgam informações úteis para outros veículos podem receber uma assinatura de bonificação em seu certificado por meio do mecanismo de reputação. Assim, é possível que outros usuários reconheçam-no como confiável segundo o mecanismo de reputação por meio de certificados proposto na seção 5.4.

4.3. Detecção de intrusos

Na presença de um intruso, a rede pode se comportar de diversas formas: pode continuar funcionando normalmente, sem permitir o acesso do intruso à rede, tampouco sofrer os efeitos da sua presença; pode reduzir a produção da rede, silenciando alguns nós, ou até mesmo interromper o funcionamento de toda a rede. Assim, faz parte dos objetivos deste trabalho permitir que as redes veiculares mantenham bom funcionamento e sejam confiáveis, mesmo sob a ação de um ataque.

Mesmo para ataques para os quais já existam mecanismos de prevenção, a necessidade da detecção de intrusos ou abordagens complementares se justifica porque mesmo os métodos mais eficazes de prevenção podem falhar. O mecanismo de reputação proposto neste trabalho poderá atuar em conjunto aos mecanismos preventivos. A vantagem de técnicas descentralizadas é a disponibilidade instantânea da informação, visto que os nós podem detectar intrusos no momento da comunicação.

4.4. Revogação de nós

A detecção de intrusos é normalmente seguida da revogação dos nós com comportamento indevido. A revogação é a exclusão do nó da rede, tornando impossível

para ele a comunicação com seus vizinhos. Esse processo deve ser autenticado para evitar a revogação de nós autênticos por intrusos. Como os nós não são protegidos contra violação física no modelo utilizado nesse trabalho, o mecanismo de reputação pode encaminhar implicitamente a revogação de nós. De outra forma, um nó intruso autenticado pela rede, provavelmente originado de uma violação física, poderia isolar nós autênticos, promovendo outros tipos de ataques de negação de serviço.

4.5. Técnicas criptográficas

O uso de criptografia com gerenciamento adequado de chaves é necessário para obter vários requisitos de segurança. A criptografia pode ser usada de forma a ocultar as informações do inimigo, garantir a autenticidade da informação ou ainda garantir a integridade e o frescor dos dados.

Para redes tolerantes a interrupções, uma técnica fim-a-fim para segurança não é muito atrativa. Existe a possibilidade de utilizar recursos escassos para mensagens indesejáveis quando se transporta tráfego por todo o caminho até seu destino sem realizar autenticação e checagem de controle de acesso. A alternativa é usá-los a cada passo do roteamento. Nesse caso, é necessário um compartilhamento de chaves entre os vários nós que precisam se comunicar diretamente para a execução do roteamento, conforme a proposta desse trabalho descrita na seção 5.2.

4.5.1. Encriptação

A criptografia pode ser utilizada em redes veiculares para garantir maior confiabilidade na comunicação, através de verificação de integridade ou autenticação de origem e destinatário das mensagens. A encriptação pode ser um processo salto-a-salto, feito cada vez que uma mensagem atinge um nó de repasse.

Um dos objetivos desse trabalho ocorre do fato que protocolos DTN devem prover um meio de encriptar elementos de forma que mensagens em trânsito não possam ser lidas na prática. O protocolo de agregação não provê nenhuma confidencialidade para a origem ou destino [2]. Similarmente, protocolos DTN devem possibilitar a aplicação de uma verificação de integridade de maneira que a identidade do nó origem seja provada e alterações em partes específicas da mensagem possam ser detectadas.

4.5.2. Assinatura digital

Assinatura é um processo criptográfico que adiciona um código autenticado com uma chave a uma mensagem e o recebedor tem que conhecer a chave para verificar a assinatura. Em redes DTN, as chaves devem ser compartilhadas pelos nós vizinhos para permitir a verificação salto-a-salto. Técnicas de assinatura tornam possível a autenticação e integridade na comunicação. O uso dessas técnicas de segurança pode evitar a inserção de pacotes falsos e a adulteração de mensagens.

Uma das diferenças das redes DTN é que uma mensagem autenticada usando uma assinatura digital, a princípio, pode ser verificada por qualquer elemento da rede no caminho. Se a mensagem contém informação suficiente, então qualquer nó pode pelo menos verificar a exatidão criptográfica da assinatura [14].

4.5.3. Gerenciamento de chaves

Um esquema de distribuição de chaves seguro e eficiente permite a autenticação dos nós da rede. O controle de acesso à rede pode impedir e eliminar diversos tipos de ataques, a menos que o inimigo comprometa nós legítimos da rede.

As características das redes DTN exigem novas abordagens para que seja possível atender os requisitos de segurança necessários a algumas aplicações. Por isso, nenhum esquema de gerenciamento de chaves ainda é reconhecido como adequado [6]. Este trabalho propõe um mecanismo alternativo para compartilhamento de chaves, por meio da utilização de certificados assinados pelo próprio usuário e quem confia nele.

Em redes DTN, ambos usuários e nós encaminhadores possuem pares de chaves e certificados, e os certificados dos usuários também indicam a classe de serviço [13]. Nós podem enviar seus pacotes com assinatura com sua chave privada, o que produz uma assinatura digital para o agregado específico. A assinatura permite aos recebedores confirmar a autenticidade do nó origem, a integridade da mensagem e os direitos relativos à classe de serviço, através do uso da chave pública do nó que enviou.

4.5.4. Criptografia baseada na identidade

Como uma área recente de pesquisa, os mecanismos que utilizam criptografia baseada na identidade [11] fornecem muitos dos benefícios da criptografia de chave pública e reduzem a sobrecarga envolvida na obtenção e verificação de chaves públicas. Embora esse mecanismo sofra alguns inconvenientes por requerer que os destinatários se comuniquem com um servidor, parece que simplesmente pré-estabelecer chaves para alguns nós com suas chaves privadas pode oferecer operações razoavelmente eficientes em redes com atrasos e interrupções, com um risco menor para segurança da rede.

Pode-se utilizar criptografia de chave pública como o ponto de partida para o estabelecimento de chaves. Roteadores e usuários finais recebem pares de chaves pública/privada, e um usuário tem de obter uma cópia assinada dessa chave pública de uma autoridade certificada da rede DTN. Todos os roteadores são considerados como pré-equipados com cópias de uma ou mais chaves públicas certificadas por autoridade DTN e o usuário então apresenta a chave assinada com a mensagem a ser encaminhada.

5. Modelo de Certificação Social

O modelo de certificação social, apresentado a seguir, objetiva prover comunicação segura para troca de informações entre os veículos e com as entidades responsáveis por organizar o trânsito. Os principais requisitos de segurança se interrelacionam, visto que dizem respeito à identidade de quem envia as informações, que não deve ser conhecida (anonimicidade), mas é preciso ser verificada e deve ter sua reputação computada. Ao receber informação de algum outro veículo, é importante saber se essa informação tem uma origem reconhecida, e caracterizar sua reputação como positiva ou negativa.

Em toda troca de mensagens na rede veicular, deve ser manifestado o interesse dos outros motoristas nas informações por meio de *flags* para indicar replicação de maior prioridade. Essa adequação, proposta na seção 4.1 deste trabalho, visa complementar o protocolo de roteamento *Spray and Wait* no modo binário, pois a replicação de mensagens no roteamento aumenta consideravelmente a tolerância a ataques.

5.1. Certificação Digital

A certificação digital pode garantir alguns requisitos de segurança essenciais na rede. Neste trabalho, no entanto, considera-se que a sua presença estará restrita a parte dos usuários, devido aos seus custos de implantação e manutenção. Pelo certificado, os usuários podem ter a garantia de acessar os sistemas corretos, de forma autêntica e privada.

O modelo deste trabalho visa aumentar a abrangência do acesso, especialmente em regiões sem cobertura da rede metropolitana, como em rodovias, não restringindo o acesso a nós autenticados por uma autoridade certificadora. Se a comunicação à rede metropolitana pudesse ser garantida em todo o tempo de operação da rede, os problemas de segurança seriam minimizados pelo uso dos certificados digitais dos servidores, e se restringiriam à manutenção das senhas e proteção dos sistemas contra invasões.

5.2. Certificados Assinados por Amigos

É necessário considerar alternativas para que os veículos consigam estabelecer os requisitos de segurança necessários, pois as redes veiculares não podem contar com acesso aos servidores de autenticação e os elementos da rede não possuirão certificados válidos, devido aos custos dos mesmos. A troca de material criptográfico deve ser feita de forma segura, por entrega direta sem uso da rede, ou de forma a garantir os requisitos básicos de segurança. Como os veículos não são certificados e não dispõem de nenhum material criptográfico inicial, não é possível garantir seus requisitos de segurança.

O problema deste trabalho pode ser abordado como uma rede complexa, adotando o modelo *"Small World"* para modelar redes veiculares como redes sociais. O modelo é apresentado como intermediário entre um grafo regular e um grafo aleatório [10].

A troca de material criptográfico de forma direta será usada para iniciar o estabelecimento dos requisitos de segurança. O funcionamento se baseia nos contatos diretos entre dois veículos, por exemplo, estacionados lado a lado ou em uma garagem. Os usuários num mesmo ambiente podem parear seus equipamentos por *Bluetooth*. Para garantir que não existe um intruso nessa comunicação, é possível estabelecer uma senha temporária que pode ser digitada por ambos os motoristas nesse contato inicial.

Esse contato direto entre amigos, definidos como duas pessoas que se conhecem e garantem sua identidade, possibilita que um assine o certificado do outro. Assinar o certificado do outro e compartilhar com ele sua chave pública possibilita uma rede de relacionamento na qual todos os veículos que possuam amigos em comum consigam validar os certificados usando as chaves do amigo. Como conhecem a chave pública do amigo, a utilizam para verificar a assinatura emitida pelo amigo para outro certificado.

Um certificado assinado por um terceiro, que não seja reconhecido como autoridade certificadora oficial, pode ser considerado válido se sua chave pública estiver disponível à outra parte. Assim, os veículos poderão validar todos os certificados assinados pelas chaves que eles conhecem. Como podem conhecer milhares de chaves, aumenta a probabilidade de encontrar uma chave em comum com os outros veículos.

5.3. Certificados Assinados por Amigos de Amigos

O princípio da assinatura por amigo de amigos é estender ao máximo as assinaturas sobre um certificado, incluindo assinaturas de amigos e certificados de reputação, para que os elementos de rede confiáveis possam ser diferenciados. As assinaturas de amigos podem ser estendidas também para os "amigos dos amigos", como em uma rede social. Isso ampliaria a possibilidade de compartilhamento de material criptográfico de dois veículos quaisquer que se encontrassem eventualmente no trânsito.

O funcionamento, baseado no PGP (*Pretty Good Privacy*)[1], ocorre seguinte forma:

1. O veículo gera um certificado auto-assinado;

[1] http://www.rnp.br/cais/keyserver/

2. O certificado pode ser assinado pelos amigos em contatos diretos, como em emparelhamento direto *Bluetooth*. Ao se aproximar do equipamento de um amigo, a geração das assinaturas dos certificados seria iniciada, por meio do compartilhamento de uma senha gerada no momento e trocada entre os amigos;

3. O certificado, assinado pelos amigos, pode ser compartilhado quando os dispositivos estiverem diretamente conectados à Internet, na sua residência por exemplo. Os amigos dos amigos reconheceriam o certificado, assinado por um amigo, e também o assinariam, compartilhando a convicção do amigo de que é um certificado confiável.

4. O veículo armazena as assinaturas de todos os amigos e amigos dos amigos, com as chaves públicas de todos. As assinaturas são usadas para validar a sua chave pública e as chaves dos amigos armazenadas para validar as chaves públicas dos demais.

5. Ao encontrar um veículo, a comunicação se inicia pela pesquisa de um amigo, ou amigo de amigo, conhecido. A identificação de um amigo em comum permite que os dois confirmem suas identidades pelas assinaturas e chaves públicas disponíveis.

6. Caso não esteja disponível a assinatura de alguém de sua rede social, é possível buscar uma assinatura conhecida originada por reputação. Essa assinatura poderá ser reconhecida se o emissor estiver na rede social de quem verifica o certificado. A reputação pode aumentar as chances de reconhecer o veículo informante, indicando que alguém da sua rede social foi beneficiado por esse veículo e reconheceu sua ajuda.

5.4. Reputação Certificada

Um mecanismo de reputação também é usado para validar o processo e confirmar que os usuários estão agindo em benefício da rede. A reputação terá efeito semelhante às assinaturas dos certificados, diferenciando apenas pelo indicativo do certificado de bonificação, emitidos pelos beneficiados por informações repassadas. Um certificado pode acumular várias assinaturas indicando sua reputação e será considerado se as recomendações positivas forem maior número que indicações negativas.

Por exemplo, um veículo que identifica um acidente e manifesta essa informação para rede, poderá receber uma bonificação de reputação positiva, se sua informação for útil a um terceiro. A bonificação será assinada com o certificado do veículo beneficiado com a informação. Todos os usuários que tiverem acesso à chave pública desse veículo serão capazes de confirmar a bonificação e reconhecer o veículo informante na rede.

A bonificação é mais uma possibilidade de reconhecimento do veículo pelos demais. Nesse caso, ele terá uma bonificação assinada que indica que alguém deu crédito à sua indicação e isso lhe foi útil. Se o veículo que deseja verificar as credenciais do outro é amigo, ou "amigo de amigo" do veículo que atribuiu a bonificação em questão, ele será capaz de identificar a validade da bonificação por meio do certificado.

5.5. Assinaturas Válidas

Este trabalho propõe três tipos de assinaturas emitidas para os certificados dos usuários que podem os credenciar para garantir a segurança em uma comunicação: assinaturas dos amigos, assinaturas dos amigos dos amigos e assinaturas de reputação. As assinaturas dos amigos são obtidas por contato direto e, por isso, implicam em uma relação próxima de conhecimento. Assinaturas de amigos dos amigos são obtidas por meio de interações em redes públicas como a Internet, sempre assinadas com os certificados dos respectivos amigos em comum.

De forma complementar, pode ser considerado o certificado de reputação que foi emitido por um desconhecido, caso tenha alguma intersecção em sua rede social e tenha sido beneficiado pela ação do veículo certificado com bonificação positiva.

Todo o processo deve ocorrer de forma rápida. Considerando uma centena de amigos, os "amigos dos amigos" poderiam contar com até 10.000 pessoas, porém como sempre existem amigos em comum, esse número tende a cair [10]. Usando mecanismos de chave pública curtas como logaritmo discreto ou curva elíptica, seria possível armazenar todas as chaves em alguns megabytes. Resumos das chaves de 20 bits poderiam ser usados para localizar chaves em comum. Esse processo deve ser rápido e encontrar as chaves em comum, então os certificados são validados com essas chaves.

6. Avaliação

Soluções de segurança para redes DTN devem ser avaliadas em relação a alguns aspectos críticos, como a latência e a probabilidade de entrega das mensagens. Além disso, a exatidão dos mecanismos de segurança utilizados nas redes veiculares deve ser verificada, de maneira que os nós apresentem comportamento adequado e não seja afetado o desempenho da rede.

Para validar o modelo de certificação aqui apresentado, um conjunto de simulações foi realizado para verificar o comportamento da rede, em função do número de elementos da rede e do número médio de elementos diretamente alcançáveis. O objetivo é mostrar o comportamento da rede veicular com utilização dos certificados auto-assinados, à medida que os motoristas passam a participar e serem autenticados pelos outros usuários, considerando os aspectos da tolerância a interrupções.

Como se espera uma grande variedade de usuários participando das redes veiculares, as simulações consideraram uma rede móvel e heterogênea, com o número total de nós variando entre 150 e 600 nós. Os nós são distribuídos pela rede de forma aleatória e deslocam-se de acordo com probabilidades definidas entre as regiões de um cenário urbano. Os pontos de interesse foram definidos como regiões centro, universidade e bairros. Os nós se deslocam entre as regiões pelas vias de uma cidade, como pode ser visualizado na Figura 2.

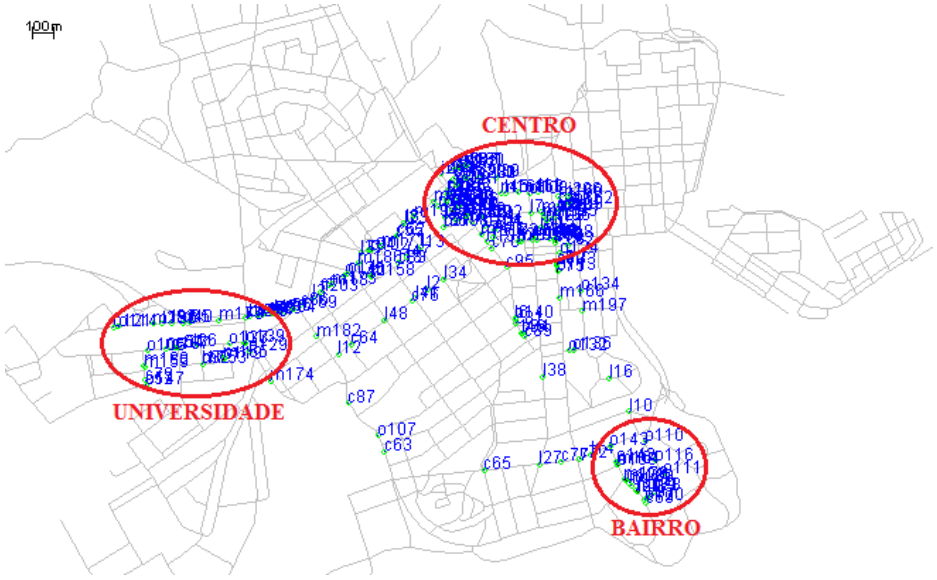


Figura 2 - Cenário das simulações

Por considerar o deslocamento em uma região urbana, o cenário utiliza um mapa e as rotas procuram obter o menor caminho possível para o destino dos nós. Foi utilizado o padrão de mobilidade "*Shortest Path Map Based Movement*" [15], que é uma derivação do "*Random Waypoint*", onde os nós usam o algoritmo de *Dijkstra* para o menor caminho para definir a rota do local atual até um destino selecionado de maneira randômica, através dos caminhos disponíveis.

Os nós se movimentam através da região abrangida pela rede, que possui tamanho de 4500 x 3400 metros, no entanto há pontos sem conectividade nessa área. Somente existe conexão *ad hoc* entre quaisquer dois nós quando ambos estão dentro do respectivo raio de transmissão. Considerando o padrão IEEE 802.11p [4] para redes veiculares, esse raio foi definido como 100m para cada nó, enquanto a velocidade de transmissão dos dados foi 250kbps e o buffer de mensagens 10MB para cada nó.

Para considerar as características DTN da rede veicular, utilizou-se o simulador The ONE (*Opportunistic Networking Evaluator*) [16], que mantém as mensagens em um *buffer* caso o nó não tenha conexão direta com o destino. Mensagens são geradas por algum nó da rede em intervalos curtos, a cada 30-45 segundos, e os períodos observados para avaliação foram simulações de 24 horas. Cada teste foi executado repetidamente, no mínimo oito vezes, sendo alterada a semente geradora do padrão de mobilidade.

Inicialmente, foi verificado o impacto da utilização do modelo de certificação social proposto na rede por meio do número médio de nós considerados confiáveis, ou seja, amigos ou amigos dos amigos. Logo após, foi analisado o comportamento da rede veicular à medida que os motoristas passam a participar e as mensagens serem autenticadas pelos outros usuários, considerando os aspectos da tolerância a interrupções. Os resultados são apresentados nos gráficos a seguir.

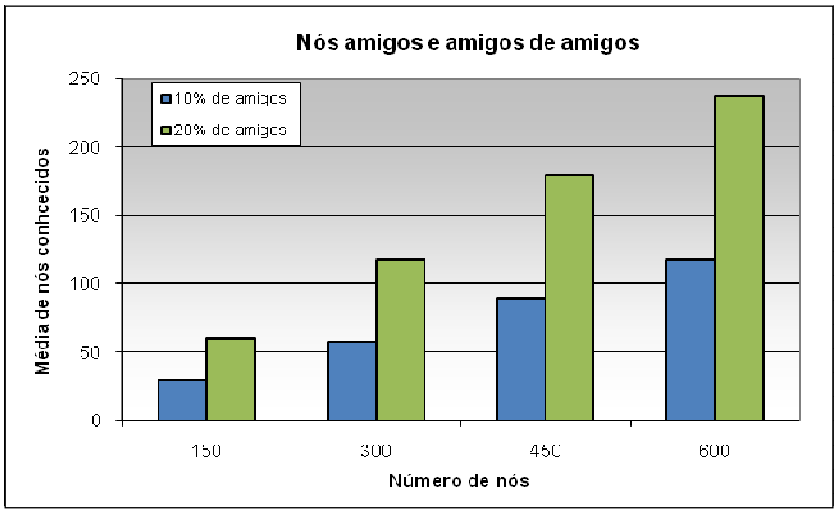


Figura 3 - Número médio de nós conhecidos

Na Figura 3, o percentual de amigos que cada nó da rede veicular possui foi estimado em 10 e 20% dos nós da rede. Pode-se observar que o percentual estipulado como número de amigos na simulação interfere diretamente na média de nós conhecidos, que considera também os amigos dos nós amigos. À medida que o número de nós da rede aumenta, há um incremento significativo na média de nós conhecidos pelos nós da rede, pois o maior número de amigos agrega outros usuários confiáveis.

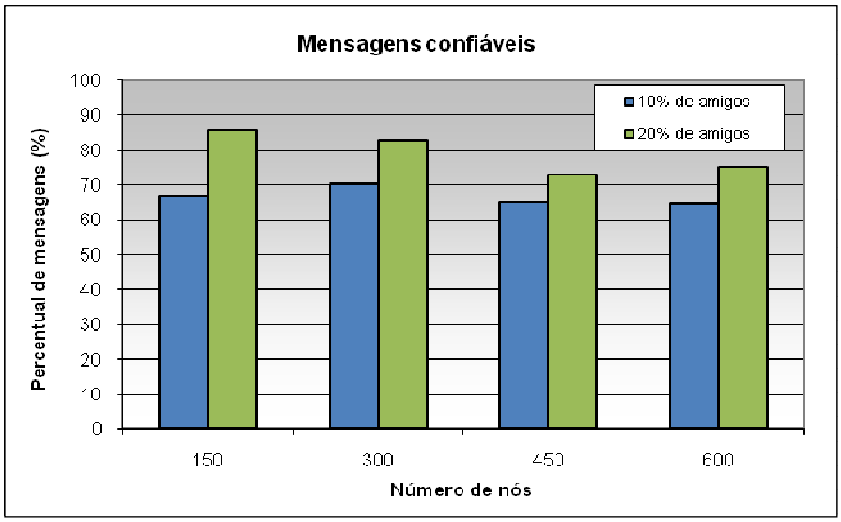


Figura 4 - Percentual de mensagens confiáveis das recebidas

A Figura 4 mostra o impacto da utilização dos certificados sociais como critério na seleção de mensagens confiáveis pelos usuários da rede veicular, considerando a conectividade imprevisível (DTN) e o número de mensagens recebidas pelos nós da rede. Com intervalo de confiança de 90%, o percentual de mensagens confiáveis em todos cenários analisados foi superior a 60%, atingindo mais de 85% em relação às mensagens entregues no cenário com 150 nós e 20% de amigos.

Apesar das características DTN da rede veicular, as mensagens recebidas pelos nós participantes foram analisadas para decidir sobre sua confiabilidade. A alteração do percentual de amigos de cada nó de 10 para 20% dos nós da rede representou até aproximadamente 20% de acréscimo no número de mensagens consideradas confiáveis, por serem enviadas por amigos do usuário ou amigos desses amigos.

Foi possível verificar nas simulações que a utilização do modelo de certificação social resulta em vantagens independente do número de nós da rede, pois representa maior segurança para os nós em relação à confiabilidade da rede. O número de mensagens entregues diminui, pois se restringe aos nós confiáveis. Como inicialmente não existiam chaves para autenticação das mensagens, o mecanismo proposto representa uma alternativa ao gerenciamento de chaves baseando-se numa rede social.

7. Conclusão e Trabalhos Futuros

Esse artigo apresenta um modelo de certificação social com foco na segurança de redes veiculares tolerantes a interrupções. Objetiva-se comunicação segura para troca de informações entre os veículos e com as entidades responsáveis por organizar o trânsito.

A proposta parte do relacionamento e conhecimento prévio dos usuários para estabelecer um grau de confiança em uma rede social. A utilização de certificados auto-assinados pelos motoristas possibilita o compartilhamento de chaves através de contato direto, entre duas pessoas que se conhecem e garantem sua identidade, que então assinam o certificado do outro mutuamente. Os certificados podem ser considerados válidos por outros usuários que tenham a chave pública da assinatura disponível.

Além disso, o mecanismo de reputação possibilita também que usuários beneficiados possam emitir uma assinatura com bonificação positiva para identificar o outro certificado como um elemento de rede confiável.

Como trabalhos futuros, propõe-se a avaliação do mecanismo de reputação proposto, bem como alterações do modelo de certificação proposto de acordo com a configuração inicial e possível integração com outras redes sociais como o Facebook.

Agradecimento: Este trabalho contou com recursos da RNP (http://www.rnp.br).

8. Referências

[1] Fall, K. (2004), "Messaging in difficult environments" – Intel Research Berkeley.

[2] Fall, K. (2003), "A Delay-Tolerant Network Architecture for Challenged Internets" – Intel Research Berkeley.

[3] Karagiannis, G., Altintas, O. et al. (2011). "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions" – IEEE Communications Surveys & Tutorials.

[4] Alves, R. S., Campbell, I. V. et al. (2009). "Redes Veiculares: Princípios, Aplicações e Desafios." Minicursos do Simpósio Brasileiro de Redes de Computadores, pp. 199-254.

[5] Raya, M. and Hubaux, J. P. (2005). "The Security of Vehicular Ad Hoc Networks." In Proceedings of ACM Workshop on Security of ad hoc and sensor networks.

[6] Symington, S. F., Farrell, S., Weiss, H. and Lovell, P. (2009), "Bundle Security Protocol Specification" – draft-irtf-dtnrg-bundle-security-08.txt.

[7] Paula, W. P.; Oliveira, S. and Nogueira, J. M. S. (2010). "Um Mecanismo de Reputação para Redes Veiculares Tolerantes a Atrasos e Desconexões." Simpósio Brasileiro de Redes de Computadores.

[8] Oliveira, T. R., Oliveira, S. and Nogueira, J. M. S. (2010), "Modelo de Gerenciamento de Segurança Adaptativo para Redes de Emergência" – In Simpósio Brasileiro de Redes de Computadores.

[9] Papadimitratos, P., Buttyan, L. et al. (2008). Secure vehicular communication systems: design and architecture. IEEE Wireless Communications Magazine, Vol. 46, No. 11, pp. 100–109.

[10] Gakenheimer, R. (1999). "Urban mobility in the developing world." Transportation Research Part A: Policy and Practice, vol. 33, no. 7-8, pp. 671-689.

[11] Seth, A. and Keshav, S. (2005), "Practical Security for Disconnected Nodes" – NPSEC.

[12] Mota, V. F. S., Silva, T. H. and Nogueira, J. M. S. (2009), "Introduzindo Tolerância a Interrupção em Redes Ad Hoc Móveis para Cenários de Emergência" – In Simpósio Brasileiro de Redes de Computadores.

[13] Durst, R. C. (2002), "An infrastructure security model for delay tolerant networks" – In http://www.dtnrg.org.

[14] Burgess, J., Bissias, G., Corner, M. D., Levine, B. N. (2007), "Surviving Attacks on Disruption-Tolerant Networks without Authentication" – ACM Mobihoc.

[15] Ekman, F., Keränen, A., Karvo, J. and Ott, J. (2008), "Working Day Movement Model" – In Proceeding of ACM SIGMOBILE workshop on Mobility models.

[16] Keranen, A. and Ott, J. (2007), "Increasing reality for DTN protocol simulations." Networking Laboratory, Helsinki University of Technology, Tech. Rep.

Mitigação de Rastreamentos em VANETs Através de Grupos Criptográficos e Ofuscação de Localizações

Eduardo Ferreira de Souza, Paulo André da S. Gonçalves

Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE)
50.740-560 – Recife – PE – Brasil

{efs, pasg}@cin.ufpe.br

Abstract. *Each vehicle in VANETs periodically broadcast messages with its current location. However, these messages allow attackers to improperly track any vehicle. The main mechanisms proposed to mitigate this problem are based either on cryptographic groups or obfuscations. The first approach allows attackers to easily track vehicles that are not in any group. The trace facility also occurs with the second approach, but when the vehicles are close together. This paper proposes a hybrid mechanism based on location obfuscation and cryptographic groups, which mitigates these tracking problems.*

Resumo. *Cada veículo nas VANETs transmite periodicamente mensagens contendo sua localização geográfica atual. Contudo, tais mensagens permitem que atacantes rastreiem indevidamente os veículos. Os principais mecanismos propostos para mitigar esse problema se baseiam ou no uso de grupos criptográficos ou no uso de ofuscações. A primeira abordagem permite que os veículos que não estejam em nenhum grupo possam ser facilmente rastreados. A facilidade de rastreamento também ocorre com a segunda abordagem, porém quando os veículos estiverem próximos entre si. Este artigo propõe um mecanismo híbrido, baseado em grupos criptográficos e ofuscação de localizações, que mitiga esses problemas de rastreamento.*

1. Introdução

As redes veiculares ad-hoc (VANETs) provêem um ambiente colaborativo para troca de informações entre os veículos. Em geral, as VANETs se baseiam na família de padrões IEEE 1609 para comunicação V2V (*vehicle to vehicle*) e V2I (*vehicle to infrastructure*). A família de padrões IEEE 1609 é motivada, principalmente, pelas aplicações de segurança no trânsito, isto é, pelas aplicações que informam os motoristas sobre situações de risco em potencial a fim de evitar colisões entre os veículos [IEEE P1609.2 Working Group 2006].

As principais aplicações voltadas para segurança no trânsito utilizam mensagens conhecidas como *CAMs* (*Cooperative Awareness Messages*), as quais são enviadas periodicamente por cada veículo e contêm a localização do emissor da mensagem. Através de tais mensagens, os veículos são capazes de monitorar a situação do trânsito, permitindo que sejam evitados acidentes. Apesar dos benefícios obtidos pela troca de informações de localização, essa comunicação em claro permite que qualquer veículo seja indevidamente rastreado através da captura de sucessivas *CAMs*. Dada essa vulnerabilidade, o

desafio é garantir a não rastreabilidade dos veículos, porém permitindo que sejam trocadas informações de localizações para viabilizar as aplicações de monitoração e segurança no trânsito.

Existem diversos esforços para padronizar a comunicação em redes veiculares [IEEE P1609.2 Working Group 2006, IEEE 802.11p Task Group 2010]. No entanto, o uso de múltiplos pseudônimos por cada veículo é a única diretriz estabelecida para que os ataques de rastreamentos não sejam sempre eficazes. Para isso, cada pseudônimo é utilizado apenas por um período de tempo limitado e, posteriormente, substituído. Grande parte dos mecanismos para mitigar os problemas de rastreamentos propõem que os veículos formem grupos criptográficos para a substituição de pseudônimos de forma coletiva [Freudiger et al. 2007, Stübing et al. 2011, Wasef and Shen 2010].

Um grupo criptográfico é formado por um conjunto de veículos e pode contar ou não com a presença de RSUs[1]. Esses grupos são utilizados para que os veículos troquem informações sigilosas de forma criptografada. A principal dessas informações é a localização de cada veículo. Nesse caso, um atacante não é capaz de obter tais informações com base nas mensagens trocadas internamente nos grupos. Contudo, os mecanismos baseados puramente em grupos apenas protegem as localizações enquanto os veículos pertencem a algum grupo.

Em [Chen and Wei 2012] é proposto um mecanismo baseado em técnicas de ofuscação para mitigar rastreamentos em VANETs. A ofuscação é a adulteração deliberada da precisão das localizações enviadas. Com isso, impede-se que os receptores identifiquem a localização exata do emissor da mensagem. Em VANETs, no entanto, cada veículo precisa conhecer a localização exata dos veículos em sua proximidade para que possam ser evitadas colisões entre eles. Para adequar-se a tal necessidade, em [Chen and Wei 2012] é proposto que os veículos próximos entre si informem suas localizações exatas em claro na rede. Assim sendo, os veículos tornam-se suscetíveis a rastreamentos nesses contextos.

Como citado, isoladamente as técnicas de grupos criptográficos e ofuscação utilizadas nos trabalhos relacionados apresentam vulnerabilidades de rastreamentos. Porém, através da união dos benefícios das duas técnicas, este artigo apresenta um mecanismo híbrido que mitiga os problemas de rastreamento. Desse modo, a solução proposta permite que todas as informações de localizações enviadas sejam protegidas ou por grupos criptográficos ou por ofuscações. O modelo de ataque utilizado considera um atacante global e passivo, isto é, o atacante é capaz de capturar simultaneamente todas as mensagens trocadas na rede, mas não envia mensagens.

O restante deste artigo está organizado da seguinte forma: A Seção 2 apresenta os trabalhos relacionados e como os mesmos se diferenciam da solução proposta neste artigo. Na Seção 3 é descrito o mecanismo proposto para mitigação de rastreamentos em VANETs. A Seção 4 avalia o desempenho da solução proposta em termos da entropia gerada, tempo máximo de rastreamento e colisões em potencial. Finalmente, a Seção 5 apresenta as conclusões.

[1] *Roadside Units* (RSUs): Dispositivos integrantes da infraestrutura e localizados às margens das rodovias.

2. Trabalhos Relacionados

As principais aplicações suportadas pela família de padrões IEEE 1609 são voltadas para segurança no trânsito e monitoração colaborativa [IEEE P1609.2 Working Group 2006]. A monitoração colaborativa permite que os usuários obtenham uma visão geral sobre as condições do tráfego nas rodovias. Por outro lado, as aplicações de segurança no trânsito atuam de forma mais específica, informando aos usuários sobre potenciais riscos de colisão. De acordo com os requisitos para o funcionamento de cada aplicação, descrito em [Hartenstein and Laberteaux 2008], é preciso que qualquer veículo da rede: (1) obtenha as localizações exatas dos veículos em sua proximidade (segurança no trânsito) e (2) possa estimar as localizações dos veículos em seu raio de alcance (monitoração colaborativa). Alguns trabalhos relacionados, no entanto, não se adequam a tais necessidades.

Atualmente, existem diversas propostas para mitigar os problemas de rastreamentos em VANETs [Freudiger et al. 2007, Stübing et al. 2011, Wasef and Shen 2010, Chen and Wei 2012]. Todas essas, exceto [Chen and Wei 2012], utilizam técnicas baseadas em grupos criptográficos. Em tais trabalhos, os grupos são utilizados apenas com foco em substituição de pseudônimos. Isto é, os grupos são finalizados após os pseudônimos serem substituídos. Como o atacante não sabe quais são os novos pseudônimos assumidos pelos veículos enquanto pertenciam aos grupos, dificulta-se que ele correlacione um dado veículo antes de ingressar no grupo como sendo o mesmo veículo após sair do grupo.

Em [Wiedersheim et al. 2010] e [Pan and Li 2012] são demonstradas as vulnerabilidades inerentes aos mecanismos que se propõem a impedir correlações entre pseudônimos. Através de simulações, em [Wiedersheim et al. 2010] é mostrada uma capacidade de rastreamento superior a 900 segundos, mesmo que os veículos substituam seus pseudônimos em curtos intervalos de 4 segundos. Isso ocorre porque as características de mobilidade de um dado veículo (localização, direção, sentido, velocidade, etc) permanecem após a substituição, sendo possível um atacante inferir que apenas o pseudônimo está modificado.

Na proposta apresentada em [Freudiger et al. 2007] são utilizados grupos situados em regiões fixas do mapa e gerenciados por RSUs. O mecanismo define que os grupos estejam localizados em cruzamentos entre vias para dificultar correlações de pseudônimos, visto que nessas regiões os veículos tendem a mudar suas características de mobilidade. Como os veículos protegem suas localizações apenas enquanto pertencentes aos grupos, em todos os outros contextos eles ficam suscetíveis a rastreamentos. Além disso, o trabalho restringe que veículos internos aos grupos não possam informar suas localizações aos veículos externos, mesmo que eles estejam próximos entre si. Desse modo, as aplicações de segurança no trânsito ficam inviáveis.

O esquema proposto em [Stübing et al. 2011] utiliza regiões pré-definidas no mapa, conhecidas por todos os veículos da rede, para que sejam formados grupos criptográficos. No mecanismo, é desnecessária a utilização de RSUs para gerenciar as chaves do grupo, visto que as chaves são definidas de forma colaborativa pelos veículos. Porém, assim como em [Freudiger et al. 2007], os veículos ficam vulneráveis a rastreamentos quando não pertencem aos grupos e, além disso, não é possível a comunicação entre veículos internos e externos aos grupos.

Diferentemente dos outros trabalhos citados até então, o mecanismo proposto em [Wasef and Shen 2010] não limita a formação de grupos a regiões fixas do mapa.

Nesse caso, quando um veículo precisa substituir seu pseudônimo, ele requisita a formação de um grupo. Para permitir que veículos externos obtenham a localização dos internos, o trabalho propõe que todos os veículos da rede conheçam as chaves secretas utilizadas em todos os grupos. Além disso, é assumido que os atacantes não conhecem tais chaves. Desse modo, pode-se evitar que as informações de localizações possam ser obtidas indevidamente enquanto os veículos pertencem aos grupos. Contudo, a suposição feita em relação às capacidades dos atacantes não é realística, pois as OBUs[2] não são restritas aos veículos. Portanto, um atacante também conheceria as chaves secretas dos grupos assim como qualquer veículo, caso possuísse uma OBU [IEEE P1609.2 Working Group 2006].

Sumarizando os trabalhos baseados em grupos analisados: os veículos protegem suas localizações apenas enquanto pertencem aos grupos. Nos outros momentos, porém, os veículos enviam publicamente suas localizações exatas, possibilitando rastreamentos. Além disso, em [Freudiger et al. 2007] e [Stübing et al. 2011] não é possível que veículos externos aos grupos detectem a proximidade em relação aos veículos internos, dado que não há troca de mensagens entre tais entidades.

As técnicas de ofuscação são frequentemente adotadas em redes de telefonia móvel [Ardagna et al. 2011], porém essa abordagem tem sido pouco explorada em VANETs. Em [Chen and Wei 2012] é proposto um esquema de ofuscação adaptável com foco em VANETs. Nele, o emissor das mensagens reduz a área das regiões de ofuscação ao detectar a proximidade com outro veículo. Nesse caso, se os veículos estiverem significativamente próximos, eles informam suas localizações exatas para permitir detecções de riscos de colisões. Porém, como há localizações exatas sendo enviadas em claro nesse contexto, os veículos ficam suscetíveis a rastreamentos. Ressalta-se que as situações de curtas distâncias entre os veículos são frequentes em vias de trânsito intenso, de modo que o mecanismo torna-se vulnerável a ataques.

Um problema comum a todos os trabalhos relacionados é a existência de contextos em que os atacantes podem obter as localizações exatas dos veículos. Para evitar rastreamentos de forma eficaz, é preciso assegurar que apenas veículos que realmente necessitam obter informações sobre localizações exatas o façam. A solução proposta neste trabalho impede o acesso inadequado às localizações exatas dos veículos através da união das técnicas de grupos criptográficos e ofuscação de localizações.

3. O Mecanismo Proposto

Este trabalho propõe simultaneamente novas técnicas de ofuscação e de grupos criptográficos. A ofuscação de localizações é realizada tanto para que as entidades obtenham estimativas das condições do trânsito quanto para que sejam detectadas as necessidades de formação de grupos. Para isso, os veículos sempre propagam suas localizações ofuscadas, independentemente de estarem presentes ou não em grupos. A comunicação através de grupos é estabelecida sempre que dois ou mais veículos encontram-se a uma distância que pode gerar riscos de colisão. Ao se comunicarem em grupos, os veículos passam a informar suas localizações exatas aos veículos internos ao grupo.

[2] *On-Board Units* (OBUs): Dispositivos que operam em movimento e suportam comunicação com outras OBUs e com as RSUs. Todos os veículos possuem OBUs embutidos, porém estes dispositivos não são restritos aos veículos, visto que OBUs podem ser utilizadas de forma portátil.

3.1. Ofuscação

A localização ofuscada é informada como uma região de circular onde o emissor está contido. Desse modo, um atacante não é capaz de obter a localização exata do veículo. Além disso, a sobreposição entre regiões de ofuscação de diferentes veículos eleva a dificuldade de rastreamentos. Essa dificuldade também pode ser definida como a entropia da rede.

A posição real do emissor pode ser qualquer ponto (x, y) contido na região de ofuscação, visto que tal região é calculada de forma pseudoaleatória. Para calcular o ponto central (x', y') da região de ofuscação, são gerados aleatoriamente dois valores: uma distância d em relação à posição real do veículo e um ângulo de inclinação α do segmento de reta entre (x, y) e (x', y'). Seja r o raio da região de ofuscação, então $0 \leq d \leq r$; e $0 \leq \alpha < 2\pi$. Assim sendo, x' e y' são definidos através da Equação (1):

$$\begin{aligned} x' &= x + d \times cos(\alpha), \\ y' &= y + d \times sen(\alpha). \end{aligned} \tag{1}$$

Ao receber uma mensagem contendo uma região ofuscada, o receptor não é capaz de obter a localização exata do emissor. Portanto, é possível apenas identificar que há algum veículo localizado dentro de tal região.

3.1.1. Situações de Risco

A necessidade de comunicação em grupo surge através da percepção de um risco de colisão. Considere dois veículos A e B não pertencentes a um mesmo grupo. Caso B receba a *CAM* enviada por A, o veículo B verifica se existe uma situação de risco. A verificação da situação de risco é calculada através da sobreposição entre a região ofuscada, contida na mensagem recebida, e a região de guarda do veículo receptor. A região de guarda é uma circunferência centrada na posição real do receptor da mensagem e com raio maior ou igual ao raio de ofuscação (r).

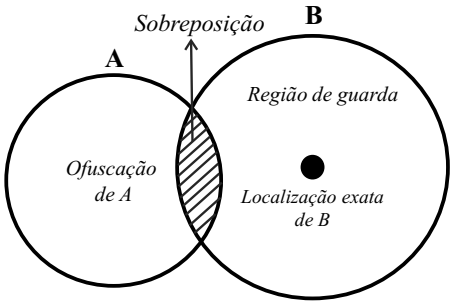


Figura 1. Situação de Risco detectada por *B*.

A Figura 1 ilustra a verificação de sobreposição realizada por B. No cenário ilustrado, caso B verifique que há sobreposição, uma mensagem de *Group Request* é enviada solicitando a formação de um grupo. No entanto, esta mensagem não contém a localização real de B, mas apenas sua localização ofuscada, pois os veículos ainda não pertencem a um mesmo grupo nesse momento. O grupo apenas será estabelecido caso

A também detecte a situação de risco através da localização ofuscada de B, contida no *Group Request*.

O raio da região de guarda (r_g) é definido por $r_g = r \times f_g$, onde f_g é o fator de guarda. Através do fator de guarda é possível aumentar, quando necessário, o raio da região de guarda em relação ao raio de ofuscação. O f_g é utilizado para minimizar a ocorrência de diferentes interpretações sobre a necessidade de formação de grupos entre A e B. Assim sendo, minimiza-se as circunstâncias onde B detecta a situação de risco ao receber a *CAM*, porém A não detecta ao receber o *Group Request*. Para isso, o raio da região de guarda (r_g) é aumentado ($f_g > 1$) especificamente no recebimento do *Group Request* durante a requisição inicial de comunicação em grupo.

3.2. Grupos Criptográficos

Os grupos criptográficos provêem um canal de comunicação seguro para a troca de localizações exatas, impedindo que um atacante global e passivo as obtenha. O gerenciamento de grupos proposto é realizado pelos próprios veículos e, portanto, independente de RSUs. Ao ingressarem em grupos, os veículos passam a enviar suas localizações exatas em um campo cifrado das *CAMs*. Porém, as mensagens podem conter simultaneamente informações de conhecimento público e informações restritas. Informações como número de identificação (ID) do grupo, pseudônimo e localização ofuscada do emissor são exemplos de informações enviadas em claro. Desta forma é possível que os veículos externos detectem situações de risco e ingressem em grupos pré-existentes. Destaca-se que, na prática, os pseudônimos são as chaves públicas utilizadas pelos veículos. Tais chaves são conhecidas e homologadas por uma Autoridade Certificadora[3] (AC) [IEEE P1609.2 Working Group 2006].

Visando minimizar a sobrecarga na rede decorrente do envio de *CAMs* distintas para os veículos internos e externos, apenas uma *CAM* é enviada contendo tanto a localização exata quanto a ofuscada. Porém, caso o veículo pertença a mais de um grupo, *CAMs* distintas são enviadas para cada um dos grupos. Nesse caso, a localização ofuscada do emissor é mantida constante ao enviá-las para grupos simultâneos, evitando o problema de ofuscações sobrepostas, descrito em [Ardagna et al. 2011].

3.2.1. Formação de Grupos

A Figura 2 ilustra a formação de grupo entre os veículos A e B. Os pontos nos centros das regiões de guarda representam a posição exata do receptor da mensagem. Imediatamente após A detectar a situação de risco, tal veículo envia para B um *Group Request*. A localização ofuscada de A, contida em tal mensagem, é utilizada para que B também verifique a existência de situação de risco entre as entidades. Caso também seja verificado, B poderá aceitar A em um grupo pré-existente ou criar um novo grupo para comunicação entre as entidades. Prioritariamente, a decisão tomada por *B* é aceitar A em um grupo pré-existente. Caso B pertença a mais de um grupo, ele aceitará A no grupo com maior número de veículos. Desse modo é minimizada a quantidade de grupos simultâneos que os veículos participam, minimizando também o *overhead* de troca de mensagens.

[3]Autoridade Certificadora (AC): Entidades confiáveis com permissões para autorizar e revogar a participação de dispositivos na rede.

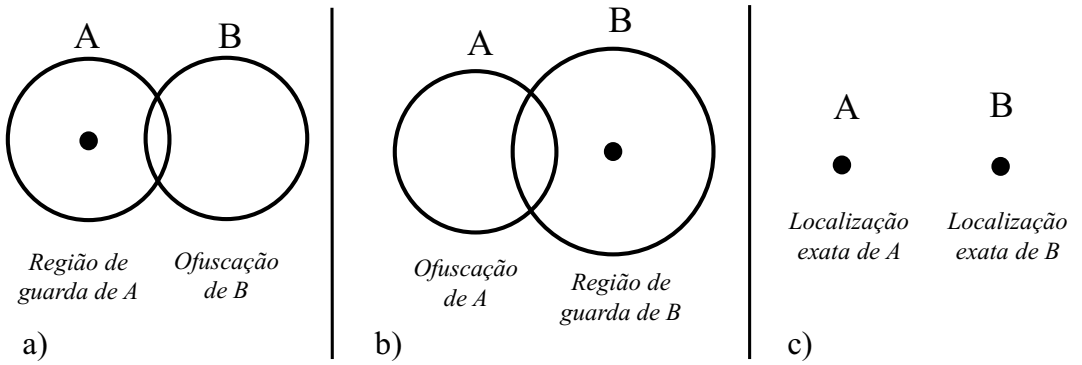


Figura 2. Exemplo de formação de Grupo. a) *B* envia um *CAM* e *A* detecta o risco; b) *A* envia a *Group Request* contendo sua localização ofuscada, e *B* detecta o risco; c) *B* envia a confirmação de formação de grupo, e os veículos passam a informar as localizações exatas.

A criação de um novo grupo é realizada apenas entre os dois veículos (A e B). Nesse processo, o veículo requisitado (B) torna-se o líder do grupo, isto é, o responsável por definir o ID do grupo e a chave simétrica a ser utilizada. Após verificar a situação de risco entre as entidades, B envia uma mensagem *Distribute Key* contendo os parâmetros do grupo. Caso seja utilizado um grupo pré-existente, tais parâmetros não são recalculados, mas apenas enviados para A. Destaca-se que, em grupos pré-existentes, qualquer veículo do grupo pode aceitar a entrada de novas entidades. Os parâmetros contidos na *Distribute Key* são cifrados através da chave pública do receptor.

3.2.2. Parâmetros do Grupo

Os identificadores do grupo, contidos nas mensagens internas, permitem que os receptores verifiquem se pertencem ao grupo ao qual uma mensagem está endereçada. A identificação do grupo é realizada através da *tupla* composta pelo ID do grupo ($group\_ID$) e a chave pública do líder ($lider\_pub\_key$). Tais informações são enviadas em claro e contidas em cada mensagem interna.

O ID do grupo é calculado pelo líder através de uma função de dispersão SHA–256, conforme a Equação (2) a seguir:

$$group\_ID = SHA\text{–}256(lider\_pub\_key \| contador\_de\_grupos). \quad (2)$$

Além da chave pública do líder, também é utilizado um contador de grupos, que é incrementado pelo líder a cada novo grupo criado. Desta forma, caso o líder crie mais de um grupo antes da substituição de sua chave pública, os IDs dos grupos serão diferentes. Assim como o ID do grupo, a chave simétrica (sim_k) é calculada da seguinte forma:

$$sim\_k = SHA\text{–}256(lider\_ID \| lider\_pub\_key \| group\_ID), \quad (3)$$

onde $lider\_ID$ é ID real do líder do grupo, $lider\_pub\_key$ é a chave pública do líder e $group\_ID$ é o identificador do grupo. Ressalta-se que o ID real de cada veículo é uma informação privada e conhecida apenas pelo próprio veículo e pela AC.

Naturalmente, é possível utilizar outras funções de dispersão, como SHA-3, para o cálculo da chave. É utilizada a função SHA-256 devido ao tamanho de saída de 256 bits, compatível com o tamanho máximo da chave do AES. O AES, por sua vez, é o algoritmo de criptografia simétrica utilizado na comunicação em grupos. Destaca-se que a SHA–256 é uma função de dispersão resistente à colisão e recomendada pelo NIST (*National Institute of Standards and Technology*).

Um dos requisitos de segurança em VANETs é o *não-repúdio*, isto é, garantir que o emissor de uma mensagem não possa negar sua autoria. Desse modo, as mensagens tornam-se passíveis de auditorias. Através da assinatura digital contida em mensagens cifradas, uma autoridade certificadora é capaz de identificar seus emissores, porém não consegue obter os textos-planos para eventuais auditorias. Diferentemente dos trabalhos relacionados, o esquema de cálculo das chaves definido neste trabalho garante à AC a capacidade de obter as chaves de todos os grupos e, consequentemente, os textos-planos das mensagens cifradas com as chaves simétricas dos grupos.

Para que a AC possa calcular a sim_k de um dado grupo, é necessário que seja obtida apenas uma mensagem interna de tal grupo. Dentre os três argumentos utilizados para o cálculo de sim_k, os campos $group\_ID$ e a $lider\_pub\_key$ trafegam em claro. Naturalmente, a chave não pode ser obtida apenas a partir de tais argumentos, visto que o $lider\_ID$ é necessário para calculá-la. No entanto o $lider\_ID$ é trivialmente obtido pela AC, pois a AC possui um mapeamento entre o ID dos veículos e todas as suas chaves públicas. Portanto, mesmo sem obter informações além das contidas nas mensagens, a AC é capaz de obter a chave utilizada para decifrá-las.

Em situações de auditoria, em geral, é necessário que sejam reportadas um conjunto de mensagens para a AC. Dependendo do protocolo definido para essas situações, as RSUs ou os próprios veículos podem reportar tais mensagens. Porém, vale ressaltar que quaisquer análises ou reportações realizadas no processo de auditoria estão fora do escopo deste trabalho.

3.2.3. Substituição e Término de Grupos

A mobilidade dos veículos gera um alto dinamismo em relação aos eventos de entradas e saídas em grupos. Como os veículos possuem características de mobilidades diferentes, é natural que alguns deles saiam do alcance do grupo. Como exemplo, haverá o estabelecimento de um grupo entre dois veículos caso eles se cruzem em sentidos opostos. Porém, esse grupo será desnecessário após o distanciamento dos veículos, dado que o grupo é composto apenas por eles. Portanto, é necessário que os veículos removam o grupo formado. Em grupos compostos por mais de dois elementos, a saída de veículos implicará substituição do grupo através do processo de RGP (*Replacement Group Procedure*).

Além da saída de veículos, as diferentes características mobilidade geram segregações do grupo ao longo do tempo. Para detectar tais eventos, todos os veículos do grupo realizam individualmente o monitoramento dos eventos de saída através do tempo de recebimento da última mensagem das outras entidades. Assim sendo, não há sincronização entre os veículos, centralização de responsabilidades ou pontos únicos de falhas. Portanto, cada elemento do grupo possui uma visão particular sobre a presença ou não dos outros elementos no grupo. O RGP permite que haja uma modificação no grupo

de forma que este possa refletir a realidade corrente dos veículos. Assim sendo, o RGP é importante para (1) remoção de veículos de saíram do grupo, (2) divisão do grupo em subgrupos, (3) eliminação de grupos desnecessários e (4) modificação da chave simétrica.

Um dado veículo A considera o grupo G_1 como redundante caso verifique que todos os veículos de G_1 estão contidos em outro grupo no qual A pertence. Apenas se G_1 não for redundante, A informa a necessidade de substituição do grupo aos outros veículos através de uma mensagem de requisição (*Group Replace Request*). Essa requisição é enviada após um período aleatório de espera (*time_to_request*). Desse modo, é possível minimizar as situações de requisições simultâneas enviadas por veículos distintos. Tal mensagem é responsável por indicar que houve um evento de saída e que o requisitante deseja substituir o grupo. Nesse caso, o emissor se tornará o líder do novo grupo caso sua requisição seja aceita pelos outros veículos. Como só recebem requisições as entidades que estão ao alcance do emissor, então os veículos que saíram do grupo, mesmo que ainda não identificados, não as receberão.

A Figura 3 ilustra a comunicação entre dois veículos (A e B) durante o RGP. Caso o grupo seja composto por múltiplas entidades, outros veículos, além de B, receberão a requisição feita por A. Assim sendo, a troca de mensagens ilustrada na Figura será realizada entre A e todos os veículos receptores da requisição.

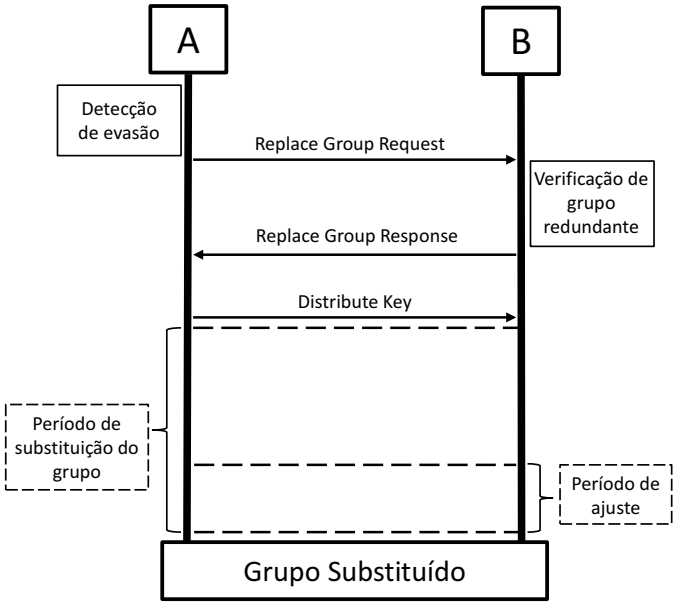


Figura 3. RGP - Troca de mensagens entre *A* e *B* para substituição de grupo.

Ao receber o *Group Replace Request*, B verifica se G_1 é um grupo redundante e, caso negativo, uma mensagem *Group Replace Response* é dada como resposta. O recebimento da primeira resposta à requisição feita indica que A deve calcular os parâmetros do novo grupo (G_2). Caso A não receba respostas ao *Group Replace Request*, uma nova requisição é enviada após um período aleatório (*time_to_request*). As assinaturas digitais do *Group Replace Request* e do *Group Replace Response* são cifradas através da chave simétrica do grupo. Portanto, apenas veículos pertencentes ao grupo podem enviá-las.

Após A calcular os parâmetros de G_2, a nova chave simétrica é cifrada através

da chave pública de B e é enviada como conteúdo da mensagem *Distribute Key*. Essa chave é cifrada com a chave pública de B, garantindo que apenas B seja capaz de decifrá-la. Especificamente no processo de RGP, a *Distribute Key* contém os IDs de G_1 e G_2, permitindo que o receptor valide a substituição do grupo, isto é, B verifica se o ID de G_1 corresponde ao grupo que A requisitou substituir.

Ao ser distribuída a sim_k de G_2 por A, inicia-se o *período de substituição do grupo* (Figura 3). Como não há sincronia sobre quais veículos estão presentes em cada grupo, este período é utilizado para que todos os veículos de G_1 tenham tempo hábil para substituí-lo. Após substituí-lo, cada veículo identifica G_1 como inativo. Assim sendo, não é aceito o ingresso de novos veículos em G_1. Portanto, caso haja situação de risco em relação a veículos fora de G_1, será utilizado outro grupo (como G_2) para comunicação cifrada entre as entidades. A remoção de G_1 ocorre após o período de substituição.

A movimentação dos veículos, a perda de mensagens e outros fatores podem colaborar para que um ou mais veículos de G_1 não ingressem em G_2 durante o período de substituição. Nesse caso, tais veículos podem passar a se comunicar de forma ofuscada, mesmo estando em situação de risco. Para mitigar esse problema, o *período de ajuste* (Figura 3) é utilizado. Nesse período, todos os veículos de G_1 verificam se o emissor de cada *CAM*: (1) está em situação de risco e (2) não está em outro grupo comum a ambos. Caso (1) e (2) sejam positivos, uma requisição direcionada de comunicação em grupo (*Group Request*) é enviada ao emissor da *CAM*. Nesse momento, como ambos os veículos ainda pertencem a G_1, o *Group Request* contém a localização exata de seu emissor.

Como citado, é possível que alguns veículos permaneçam simultaneamente em mais de um grupo ativo. Como exemplo, considere que o veículo V_1 esteja em situação de risco em relação a dois veículos, V_2 e V_3, porém que V_2 e V_3 não estejam em situação de risco entre si. Considerando que V_1 envia duas mensagens de *Group Request*, uma para V_2 e outra para V_3, então V_1 irá participar de dois grupos simultaneamente. Os grupos formados serão G_{1_2}, composto por V_1 e V_2, e G_{1_3}, composto por V_1 e V_3. Posteriormente, se V_2 e V_3 ficarem em situação de risco entre si e V_2 entrar em G_{1_3}, então G_{1_2} torna-se desnecessário. Para mitigar a participação em grupos desnecessários, cada veículo frequentemente verifica se há grupos totalmente contidos em outros grupos aos quais ele pertença e, se identificados, estes são removidos.

4. Avaliação de Desempenho

O desempenho do mecanismo proposto é analisado através de um simulador de troca de mensagens desenvolvido neste trabalho. O trabalho adota a estrutura de quadros do padrão IEEE 802.11p [IEEE 802.11p Task Group 2010]. Conforme as orientações da família de padrões IEEE 1609 para ambientes urbanos, o alcance máximo das mensagens utilizado na comunicação entre os veículos é de 300 metros. Além disso, os veículos enviam *CAMs* com frequência média de 200 milissegundos.

O mapa utilizado para simular a mobilidade dos veículos é uma área de 1 km^2 do centro da cidade de São Francisco – CA, nos Estados Unidos. O mapa de rodovias foi obtido através do U.S. Census Bureau [U.S. Census Bureau 2012]. Os registros das movimentações são gerados pelo simulador VanetMobiSim [Harri et al. 2007] e, posteriormente, importados para o simulador de troca de mensagens desenvolvido no trabalho.

Em [Santa et al. 2009] é apresentado o resultado de um experimento de

comunicação de veículos em um ambiente real. No experimento, a taxa de perda de mensagens foi afetada por fatores como a velocidade dos veículos e a densidade da rede. Contudo, a taxa máxima de perda obtida foi de 1,16%, isto é, 98,84% de taxa de entrega de mensagens. Com base em tal experimento, as simulações realizadas neste trabalho utilizam uma taxa de perda fixa de 1,5%.

Para a geração de mobilidade dos veículos com o simulador VanetMobiSim, são considerados uma velocidade máxima de 110km/h e aceleração máxima de 4,5 m/s^2. Naturalmente, ambos são limites superiores, contudo a velocidade e a aceleração desenvolvidas nas simulações dependem de diversos fatores, como a densidade de veículos na rodovia, a quantidade de sinais de trânsito e o número de faixas. Cada simulação realizada reflete em trinta minutos de movimentação real dos veículos ao longo do mapa. Além disso, são realizadas vinte simulações e os resultados são medidos com intervalo de confiança de 99%. São analisadas densidades de veículos variando entre 50 veículos/km^2 e 800 veículos/km^2, refletindo desde um trânsito pouco denso até um cenário de trânsito intenso e forte congestionamento. Os períodos estáticos utilizados para a simulação são: 9 segundos para *período de substituição do grupo*, 3 segundos para *período de ajuste* e 3 segundos para *período de remoção de redundância*. O fator de guarda utilizado é $f_g = 1 + 1/3$ para o caso de recepção de *Group Request*.

As métricas para avaliar o desempenho deste trabalho são as seguintes: a *entropia média dos veículos*, o *tempo máximo de rastreamento* e a quantidade de *colisões em potencial* enfrentadas pelos veículos. A *entropia* indica a dificuldade de rastreamento de um dado veículo para o atacante. O *tempo máximo de rastreamento* indica o maior período contínuo que o atacante consegue rastrear um veículo. Por fim, as *colisões em potencial* indicam os momentos em que os veículos ficam próximos entre si e sem se comunicarem através de grupos. Nas simulações não há modificações de pseudônimos. Caso houvesse, isso elevaria, potencialmente, a dificuldade de rastreamentos.

A dificuldade de rastreamento (*entropia*) de um veículo pode ser medida através do tamanho de seu conjunto de anonimato [Serjantov and Danezis 2003]. A *entropia* mede a quantidade de informação, em bits, que um atacante precisa para distinguir entre o veículo rastreado e os outros veículos da rede. Para isso, os veículos contidos em um mesmo conjunto de anonimato são simultaneamente indistinguíveis para um atacante global e passivo, dado que os elementos de um conjunto são considerados em distribuição uniforme. A *entropia* por si só apenas indica, de forma abstrata, quais cenários são mais difíceis para que um atacante realize rastreamentos. Porém, essa métrica é importante para que seja calculado o *tempo máximo de rastreamento* de um veículo.

Como o atacante não tem acesso às localizações exatas dos veículos, os tamanhos dos conjuntos de anonimato são computados apenas com base nas ofuscações. A análise realizada considera um conjunto de anonimato como um conjunto de veículos que enviam *CAMs* em uma amostra de tempo de 200 milissegundos, e que as áreas ofuscadas das mensagens possuem sobreposições simultâneas. A probabilidade do veículo n ser rastreado com sucesso é p_n, S_n é o conjunto de anonimato ao qual n pertence, $\|S_n\|$ é o tamanho de S_n e $H(n)$ é a *entropia* do veículo n, conforme a Equação (4) a seguir:

$$H(n) = -\sum_{n=1}^{\|S_n\|} p_n \log_2 p_n, \text{ onde } \sum_{n=1}^{\|S_n\|} p_n = 1. \tag{4}$$

Caso nenhum outro veículo além de n pertença ao conjunto de S_n, então $H(n)$ será 0 (zero). Nesse caso, n é trivialmente rastreável por um atacante. A Equação (5) apresenta a *entropia* média dos N veículos da rede ao longo de todo o tempo T de simulação.

$$H(T,N) = \frac{\sum_{t=1}^{T}\sum_{n=1}^{N} H(n)}{T \times N}. \quad (5)$$

A Figura 4 apresenta a *entropia* média para diferentes raios de ofuscação e densidades de veículos na rede. Percebe-se que ambas as variáveis impactam positivamente na *entropia* à medida que são incrementadas. Isso ocorre porque ambas geram um maior número de sobreposições entre as ofuscações, ocasionando em maior dificuldade de rastreamentos para um atacante.

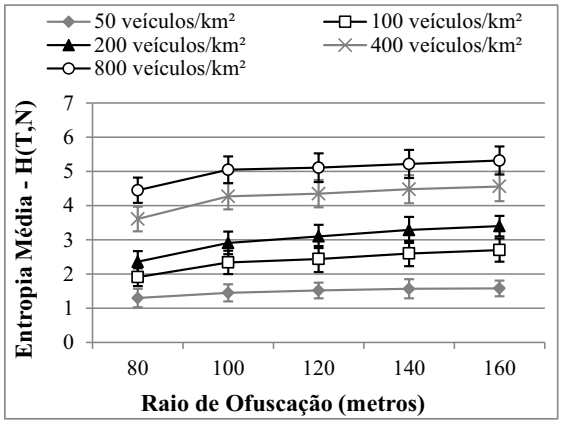


Figura 4. *Entropia* média da rede.

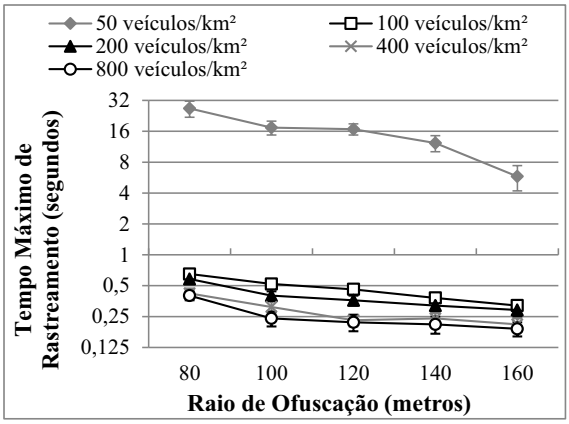


Figura 5. Média dos períodos máximos de rastreamento.

Como consequência da *entropia* $H(n)$, é avaliado o *tempo máximo de rastreamento* determinístico por um atacante global e passivo. Portanto, para que um atacante possa realizar o rastreamento com sucesso, é necessário que a *entropia* do veículo alvo seja zero. A Figura 5 apresenta a média do período máximo de rastreamento de todos os veículos da rede. Nota-se que no cenário com 50 veículos/km^2, onde a rede é pouco densa, o período de rastreamento varia entre 6 e 28 segundos, dependendo do raio de ofuscação utilizado. Naturalmente, como os veículos trafegam longas distâncias sem se aproximarem de outros veículos, não há sobreposição entre áreas de ofuscação, tornando-os vulneráveis a rastreamentos. Nos cenários com 100, 200, 400 e 800 veículos/km^2, os períodos de rastreamento máximos são inferiores a 1 segundo, demonstrando a maior eficiência do mecanismo em redes com densidades médias e altas.

Como citado, os veículos em conjuntos de anonimato de tamanho 1 são rastreáveis, visto que a *entropia* gerada para o atacante é 0. Em [Freudiger et al. 2007], [Stübing et al. 2011] e [Wasef and Shen 2010], os veículos participam de conjuntos de anonimato com tamanho maior que 1 apenas enquanto pertencem a grupos, pois este é o único momento que o atacante não obtém as localizações dos veículos. Em todos os outros momentos, porém, o envio de mensagens contendo localizações exatas permite que os veículos sejam rastreados. Portanto, para que um veículo não possa ser rastreado em tais mecanismos, é preciso que: (1) pelo menos dois veículos estejam próximos entre si e (2) um grupo seja formado entre as entidades. No entanto, a solução proposta neste trabalho

também garante que a *entropia* será maior que 0, caso (1) e (2) sejam verdadeiros. Assim sendo, o *tempo máximo de rastreamento* em [Freudiger et al. 2007], [Stübing et al. 2011] e [Wasef and Shen 2010] é, garantidamente, maior ou igual ao *tempo máximo de rastreamento* deste trabalho. De modo semelhante, em [Chen and Wei 2012] o tamanho de um conjunto de anonimato é maior que 1 apenas quando os veículos não estão próximos entre si. Porém, este trabalho garante a sobreposição de ofuscações de veículos próximos, de modo que o tamanho do conjunto de anonimato torna-se maior que 1 nessas situações. Desse modo, o *tempo máximo de rastreamento* em [Chen and Wei 2012] também é maior ou igual ao deste trabalho.

As métricas de *colisões em potencial* são utilizadas para medir a frequência e o tempo que os veículos ficam entre si em distâncias inferiores a 50% do raio de ofuscação quando eles não pertencem a um mesmo grupo. Na Figura 6 é mostrado que o percentual de *colisões em potencial* é inferior a 1% em todos os cenários. Além disso, o tempo médio em que essas situações perduram é inferior a 0,7 segundos, como apresentado na Figura 7. Portanto, os veículos permanecem sem o auxílio dos grupos para indicar potenciais riscos de colisão apenas em curtos intervalos de tempo. Considere o cenário de 800 veículos/km^2 e raio de ofuscação de 160 m: caso os veículos se aproximem a uma velocidade relativa de 80 km/h (22,2 m/s), então no intervalo de 0,7 segundos (Figura 7) eles se aproximam 15,6 m. Porém, como 50% do raio de ofuscação corresponde a 80 m, então os veículos detectam o risco a uma distância de 64,4 m, restando-lhes 2,9 segundos para realizarem manobras que evitem a colisão. Ressalta-se que essas situações ocorrem em menos de 1% dos momentos que os veículos precisam se comunicar em grupos, como apresentado no percentual de *colisões em potencial*.

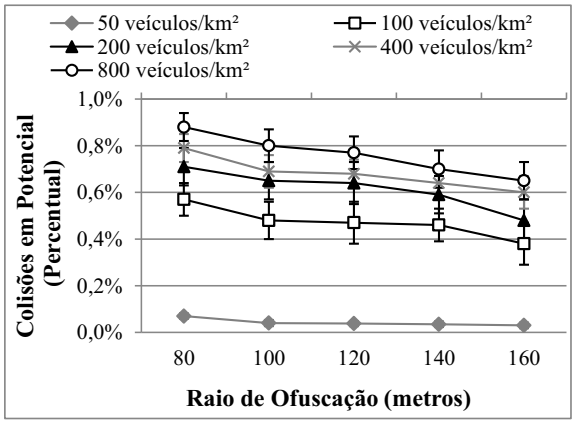


Figura 6. Percentual de *colisões em potencial*.

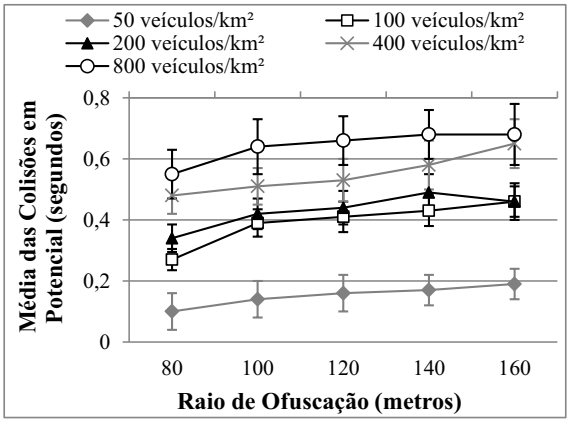


Figura 7. Duração média das *colisões em potencial*.

5. Conclusões

Neste trabalho foi proposto um mecanismo para mitigar o problema de rastreamento em redes veiculares. Foi utilizada uma abordagem híbrida, baseada em ofuscação de localizações e em grupos criptográficos. Tal abordagem se adequa às necessidades das aplicações de monitoração e segurança no trânsito. Diferentemente dos trabalhos relacionados, a localização exata dos veículos não trafega em claro na rede em nenhum momento. Foi demonstrado que a solução proposta garante um grau de *entropia* mais elevado que os trabalhos relacionados e, portanto, uma maior dificuldade de rastreamentos.

Através das simulações, foram analisadas as *colisões em potencial* sofridas pelos veículos. Em todos os cenários, o percentual de *colisões em potencial* foi inferior a 1%. Além disso, essas situações perduraram por menos de 0,7 segundos. Por outro lado, a principal métrica para avaliar o desempenho da solução foi o *tempo máximo de rastreamento* dos veículos. Em todos os cenários, essa métrica atingiu média inferior a 28 segundos, com destaque para os cenários com redes de médias e altas densidades, onde se obteve uma média inferior a 1 segundo. Portanto, o atacante não consegue rastrear os veículos durante períodos significativos de tempo.

Referências

Ardagna, C. A., Cremonini, M., di Vimercati, S. D. C., and Samarati, P. (2011). An Obfuscation-Based Approach for Protecting Location Privacy. *IEEE Transactions on Dependable and Secure Computing*, 8(1):13–27.

Chen, Y.-M. and Wei, Y.-C. (2012). SafeAnon: a Safe Location Privacy Scheme for Vehicular Networks. *Telecommunication Systems*, 50(4):339–354.

Freudiger, J., Raya, M., Félegyházi, M., Papadimitratos, P., and Hubaux, J.-P. (2007). Mix-Zones for Location Privacy in Vehicular Networks. In *Proc. of WSN4ITS*.

Harri, J., Fiore, M., Filali, F., and Bonnet, C. (2007). Vehicular Mobility Simulation for VANETs. In *Proc. of IEEE Annual Simulation Symposium*, pages 301–309.

Hartenstein, H. and Laberteaux, K. P. (2008). A Tutorial Survey on Vehicular Ad Hoc Networks. *IEEE Communications Magazine*, 46(8):164–171.

IEEE 802.11p Task Group (2010). IEEE 802.11p - IEEE 802.11 - Amendment 6: Wireless Access in Vehicular Environments.

IEEE P1609.2 Working Group (2006). Trial-Use Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages.

Pan, Y. and Li, J. (2012). An Analysis of Anonymity for Cooperative Pseudonym Change Scheme in One-dimensional VANETs. In *Proc. of IEEE CSCWD*, pages 251–257.

Santa, J., Tsukada, M., Ernst, T., Mehani, O., and Gómez-Skarmeta, A. F. (2009). Assessment of VANET multi-hop routing over an experimental platform. *Journal of Internet Protocol Technology*, 4(3):158–172.

Serjantov, A. and Danezis, G. (2003). Towards an information theoretic metricfor anonymity. *Lecture Notes in Computer Science*, 2482:41–53.

Stübing, H., Pfalzgraf, M., and Huss, S. A. (2011). A Decentralized Group Privacy Protocol for Vehicular Networks. In *Proc. of IEEE International Conference on Privacy, Security, Risk and Trust / IEEE SocialCom*, pages 1147–1154.

U.S. Census Bureau (2012). Topologically Integrated Geographic Encoding and Referencing. `http://www.census.gov/geo/maps-data/data/tiger.html`.

Wasef, A. and Shen, X. (2010). REP: Location Privacy for VANETs using Random Encryption Periods. *ACM Mobile Networks and Applications*, 15:172–185.

Wiedersheim, B., Ma, Z., Kargl, F., and Papadimitratos, P. (2010). Privacy in Inter-Vehicular Networks: Why simple pseudonym change is not enough. In *Proc. of Wireless On-demand Network Systems and Services*, pages 176–183.

Sistema para Monitoramento Descentralizado de Trânsito Baseado em Redes Veiculares Infraestruturadas

José Geraldo Ribeiro Júnior[1,2], **Igor M. Quintanilha**[1], **Miguel Elias M. Campista** [1], **Luís Henrique M. K. Costa**[1]

[1]Grupo de Teleinformática e Automação – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

[2]Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG
Leopoldina – MG – Brasil

{jgrjunior,quintanilha,miguel,luish}@gta.ufrj.br

Resumo. *As propostas atuais para o monitoramento automatizado de trânsito em vias públicas exigem um alto custo de instalação e manutenção, especialmente por serem dependentes de um elemento central responsável por calcular e divulgar as condições da via. Este artigo propõe um sistema para o monitoramento e divulgação das condições de trânsito de forma descentralizada, onde unidades de bordo e unidades de acostamento, que não precisam estar interligadas entre si, nem a um ponto central, trocam informações a fim de atualizar suas tabelas de condições sobre cada trecho da via. Para validar o sistema proposto, foram executados experimentos em um cenário montado na Universidade Federal do Rio de Janeiro utilizando uma rede IEEE 802.11b/g. A comparação dos resultados obtidos com os dados de um GPS mostrou alto grau de precisão tanto na detecção da posição dos veículos quanto na estimativa da condição da via.*

Abstract. *Current proposals of automated traffic monitoring systems in highways require high investments in installation and maintenance, especially because these systems depend on a central element, responsible to infer and disseminate the traffic conditions. This paper proposes a system to monitor and disseminate traffic conditions, using a decentralized structure. On board Units and Road Side Units do not need to be neither interconnected, nor connected to a central point. These elements exchange information to update their tables concerning traffic conditions of each route section. To validate the proposed system, a prototype was implemented using an IEEE 802.11b/g and experiments were performed on the Federal University of Rio de Janeiro. The comparison of the obtained results with data obtained from a GPS shows a high accuracy degree in detecting both the position of the vehicle and the estimative of the road condition.*

1. Introdução

Conhecer as condições de uma via em tempo real é fundamental para minimizar os problemas no trânsito. Apesar da maioria dos sistemas de monitoramento em todo o mundo utilizar câmeras de vídeos, propostas de sistemas automatizados começam

a se tornar uma realidade. Entre as propostas estão sistemas que usam: (1) sensores nas vias e nos carros [Edelmayer et al., 2010, Kassem et al., 2012], (2) sensores magnéticos [Cheung et al., 2004], (3) sensores acústicos [Fazenda et al., 2009], (4) técnicas de computação gráfica [Cucchiara et al., 2000], ou (5) a combinação de um GPS (*Global Positioning System*) [Yoon et al., 2007, Mohan et al., 2008], utilizado para definir a posição do veículo na via, com 3G [Google, 2011, Valerio et al., 2009] ou redes IEEE 802.11 [Thiagarajan et al., 2009], para enviar os dados sobre a movimentação do veículo. Outros sistemas dispensam o uso do GPS para localização do veículo [Ribeiro Júnior et al., 2012a], o que reduz o consumo de bateria, especialmente em equipamentos como *smartphones*. No entanto, esses sistemas dependem de um elemento centralizador, responsável por inferir e divulgar as condições na via. Essa central pode ser um elemento externo, conectado via Internet, ou fazer parte da rede local.

Em paralelo ao aumento constante do número de congestionamentos, está a popularização do uso de *smartphones*. Segundo o relatório da Cisco sobre previsão do crescimento do tráfego móvel global, até 2015 haverá um *smartphone* por habitante no planeta [Cisco, 2012]. Por outro lado, a alta demanda da tecnologia 3G/4G está deteriorando a qualidade e a garantia do serviço prestado aos usuários devido a congestionamentos na rede [Balasubramanian et al., 2010], fazendo as operadoras limitarem os planos a uma taxa máxima de uso.

Este trabalho propõe um sistema colaborativo para o monitoramento e divulgação das condições de trânsito de forma descentralizada. Nessa proposta, unidades de bordo e unidades de acostamento trocam informações a fim de atualizar suas TCTs (Tabelas de Condições dos Trechos), que contêm informações sobre cada trecho da via. Um diferencial desta proposta é o fato das unidades de acostamento não precisarem estar conectadas, nem entre si e nem tampouco a um ponto central. Utilizando kits de acostamento, já propostos na literatura [Ribeiro Júnior et al., 2011], o sistema pode ser utilizado mesmo em rodovias onde não existe infraestrutura, como energia ou cobertura celular.

No sistema proposto, as informações geradas pelos veículos são ponderadas para privilegiar as informações mais recentes e ainda para auxiliar na definição de um tempo de vida útil para a informação, uma vez que o objetivo é inferir a condição da via em tempo real. Por não haver um elemento centralizador, não há garantia de sincronismo entre os relógios, dessa forma foi necessário utilizar um mecanismo de controle semelhante ao controle de empréstimo (*lease*) do DHCP (*Dynamic Host Configuration Protocol*) [rfc, 1997], onde o tempo de vida útil é dado em segundos e cada dispositivo decrementa o tempo, baseado no relógio local, até chegar a zero.

Para validar o sistema proposto, um protótipo foi implementado no campus da Universidade Federal do Rio de Janeiro, na Ilha do Fundão, utilizando uma rede IEEE 802.11b/g. O cenário escolhido possui características semelhantes a vias públicas com grande fluxo de veículos, além de outras redes na mesma frequência. Foram coletados dados sobre cada etapa do processo utilizando um veículo. Entre as medições estão o tempo de associação e configuração de rede, o tempo para envio e atualização da tabela de condição do trecho e a detecção de passagem da unidade de bordo pela unidade de acostamento. A comparação dos resultados obtidos com os dados de um GPS de alta precisão (informa a posição 4 vezes por segundo) mostrou um alto grau de precisão tanto na detecção da posição dos veículos quanto na estimativa da condição da via.

O restante deste artigo possui a seguinte estrutura: a Seção 2 apresenta detalhes do sistema proposto, enquanto a Seção 3 apresenta as especificações de um protótipo implementado para testes utilizando redes IEEE 802.11b/g. A Seção 4 apresenta os experimentos e os resultados obtidos no campus da UFRJ. Finalmente, a Seção 5 apresenta as conclusões e indica os trabalhos futuros.

2. Sistema Proposto

A Figura 1 apresenta a arquitetura do sistema proposto, onde as unidades de acostamento estão instaladas em pontos de ônibus. Considera-se que um usuário do veículo possui um equipamento com a aplicação proposta em execução, a via possui trânsito nos dois sentidos e as distâncias entre as unidades de acostamento sejam previamente conhecidas. A conexão das unidades de acostamento a elementos externos à rede local permite a divulgação das informações da via para dispositivos externos.

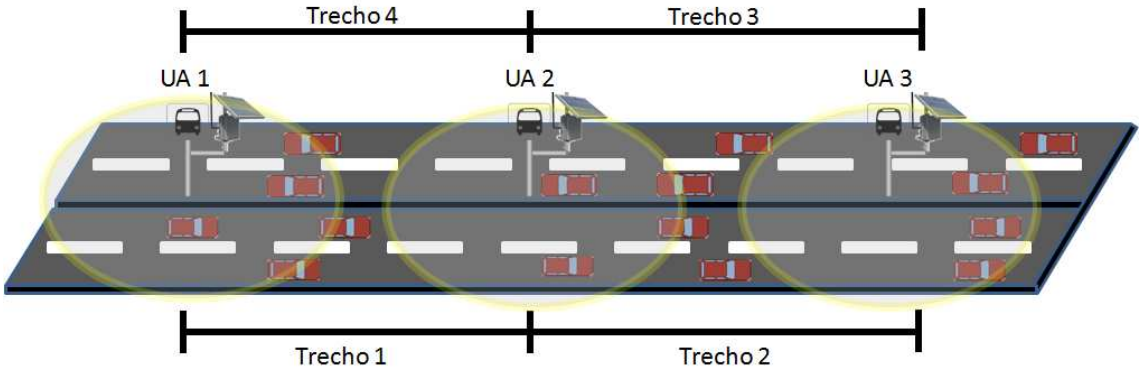


Figura 1. Arquitetura do sistema proposto.

Conhecida a distância entre as unidades de acostamento e o tempo utilizado para ir de um ponto ao outro, é possível calcular a velocidade do veículo no trecho. Considera-se um trecho o segmento da via que fica entre as unidades de acostamento (Figura 1). O número de trechos é proporcional à quantidade de unidades de acostamento. Para cada sentido da via é considerado um trecho diferente. O número de trechos (N_T) é dado pela equação $N_T = ((N_{UnA} - 1) * N_D)$, onde N_{UnA}, e N_D são, respectivamente, o número de unidades de acostamento e o número de direções, que nesta proposta será sempre múltiplo de 2.

A Tabela de Condição dos Trechos (TCT), vista na Figura 2, é usada para a troca de informação entre a unidade de bordo e a unidade de acostamento. Nessa tabela estão as seguintes informações: Trecho, Condição e TTL (*Time To Live*). O `Trecho` representa o identificador único do trecho em questão, que será utilizado na comparação entre as tabelas; a `Condição` representa a velocidade média atual no trecho; e o `TTL` representa o tempo de vida de cada entrada da tabela. O TTL tem basicamente duas funções: definir um tempo de vida útil para a informação e atribuir maior peso para informações mais recentes. Uma vez que a ideia é inferir a condição atual nos trechos, as informações defasadas recebem menor ou nenhum peso (no caso do TTL ser zero).

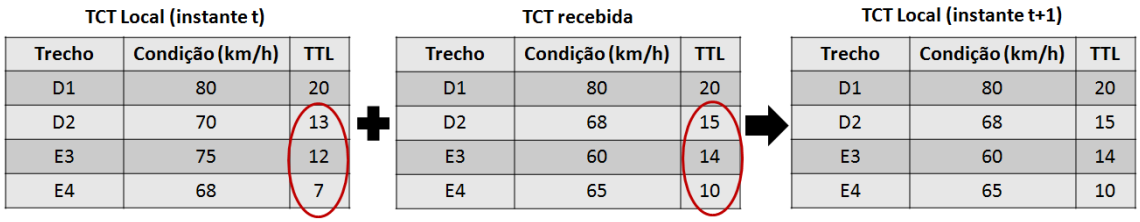


Figura 2. Atualização da TCT local.

2.1. Funções da Unidade de Bordo

A unidade de bordo é responsável por quatro funções: (1) calcular em tempo de execução a velocidade dentro do trecho, (2) atualizar a TCT local, (3) detectar o melhor momento pra envio da TCT à unidade de acostamento e (4) enviar a TCT a unidade de acostamento mais próxima. Ao enviar essas informações, o veículo atualiza a TCT da unidade de acostamento em, pelo menos, todas as linhas referentes aos trechos anteriores já que os valores do TTL são maiores. Analisando a Figura 1, por exemplo, quando um veículo acaba de passar pelo trecho 1, ele envia informações sobre as condições do trânsito para a unidade de acostamento 2, onde ao menos as informações sobre os trechos 1 e 4 serão mais atuais. Como não há conexão entre as unidades de acostamento, os veículos cumprem o papel dos enlaces de comunicação da arquitetura proposta, possibilitando que cada unidade de acostamento tenha informação sobre toda a via.

Como é apresentado no Algoritmo 1, ao receber a TCT, o veículo compara cada linha com a TCT local. Cada linha representa um trecho da via. As informações recebidas sobre trechos com TTL maior que as existentes na TCT atual são usadas para atualizar a TCT local. A Figura 2 ilustra a comparação, onde a tabela local, no instante $t + 1$, é atualizada após a comparação das tabelas no instante t. Ao detectar o momento em que o veículo passou pela unidade de acostamento, este calcula a velocidade no trecho. Em seguida a TCT local é atualizada e finalmente enviada para a unidade de acostamento.

2.2. Função da Unidade de Acostamento

Como mostra o Algoritmo 2, cada unidade de acostamento divulga sua TCT atual ao cliente após terminar o processo de conexão. Assim, cada unidade de acostamento divulga para os veículos próximos, em ambos os sentidos, uma visão das condições em trechos futuros. Quando a unidade de acostamento recebe uma TCT, ela também precisa comparar com a informação local e, se o TTL de cada entrada for maior do que a informação atual, atualizar seus dados. No entanto, como são vários veículos enviando informação simultaneamente, e os veículos podem apresentar velocidades distintas, de acordo com a condição de cada pista, a unidade de acostamento utiliza a média harmônica simples para calcular a condição atual do trecho. Utilizando a Equação 1, o sistema atribui um peso maior para as informações mais recentes.

$$MHS_{atual_{Trecho}} = \frac{2}{\frac{1}{v_i} + \frac{1}{MHS_{ant}}}, \quad (1)$$

onde v_i é a velocidade mais recente recebida do último nó móvel.

Algoritmo 1 Pseudo-código executado pela unidade de bordo.

...
$N_{Trechos} = ((N_{UnA} - 1) * N_D)$;
while true **do**
  if Entrou na área de cobertura da unidade de acostamento **then**
    Faz pedido de conexão;
  else
    Procura ESSID conhecido;
  end if
  Recebe TCT da unidade de acostamento;
  while $num_{Linha} < N_{Trechos}$ **do** //Compara TCT recebida com TCT local
    if $TTL_{atual_{LinhaX}} <= TTL_{recebida_{LinhaX}}$ **then**
      $TTL_{atual_{LinhaX}} \leftarrow TTL_{recebida_{LinhaX}}$;
      $Condicao_{Atual_{LinhaX}} \leftarrow Condicao_{Recebida_{LinhaX}}$;
    end if
  end while
  Detecta momento em que passou pela unidade de acostamento;
  Guarda o tempo em que passou pela unidade de acostamento;
  $Veloc_{Trecho} \leftarrow \frac{dist_{Trecho}}{Tempo_{UnA2} - Tempo_{UnA1}}$;
  Atualiza TCT local;
  Envia a TCT atualizada para a unidade de acostamento;
end while
...

Algoritmo 2 Pseudo-código executado pela unidade de acostamento.

...
$N_{Trechos} = ((N_{UnA} - 1) * N_D)$;
while true **do**
  Recebe pedido de conexão;
  Envia, junto com as configurações de rede, a TCT local atual;
  Aguarda por TCT atualizada;
  Recebe TCT atualizada;
  $num_{Linha} \leftarrow 1$;
  while $num_{Linha} < N_{Trechos}$ **do** //Compara TCT recebida com TCT local;
    if $TTL_{Atual_{LinhaX}} = 0$ **then**
      $TTL_{Atual_{linhaX}} \leftarrow TTL_{Recebida_{LinhaX}}$;
      $Condicao_{Atual_{linhaX}} \leftarrow Condicao_{Recebida_{LinhaX}}$;
    else
      if $TTL_{Atual_{LinhaX}} <= TTL_{Recebida_{LinhaX}}$ **then**
        $MHS_{Atual_{Trecho}} = \frac{2}{\frac{1}{Condicao_{Recebida}} + \frac{1}{MHS_{ant}}}$;
        $TTL_{Atual_{linhaX}} \leftarrow TTL_{Recebida_{LinhaX}}$;
        $Condicao_{Atual_{linhaX}} \leftarrow MHS_{Atual_{Trecho}}$;
      end if
    end if
  end while
  $num_{Linha} \leftarrow num_{Linha} + 1$;
end while
...

2.3. Cálculo do TTL

O sistema proposto não assume o sincronismo entre os relógios das unidades de bordo e de acostamento, de complexa realização uma vez que o cenário é parcialmente desconectado. Como solução mais simples, definiu-se que cada dispositivo fica responsável por decrementar o valor do TTL baseado no horário local. Dessa forma, o valor do TTL é decrementado a cada segundo. A falta de sincronismo é minimizada, ficando restrita apenas ao tempo de envio entre um elemento e outro.

O valor máximo do TTL depende de características da via, como número de pistas, extensão, velocidade máxima permitida, entre outras. Um tempo de TTL muito pequeno pode não ser suficiente para atualizar as condições da via em trechos distantes ou em vias onde o limite de velocidade é baixo. Um tempo de TTL muito grande pode fazer com que as unidades de acostamento tenham uma informação desatualizada sobre trechos distantes. O valor do TTL deve ser maior que tempo de defasagem, ou seja, maior que o tempo necessário para atualizar as informações sobre o trecho mais distante. O cálculo para encontrar o valor do tempo de defasagem (T_{Def}) é dado pela equação $T_{Def} = \sum_{i=1}^{n} T_{Trecho}$, onde n é o número de trechos e T_{Trecho} é o tempo médio para atravessar cada trecho.

3. Protótipo do Sistema Proposto Baseado em uma Rede IEEE 802.11

Um protótipo do sistema foi implementado utilizando uma rede IEEE 802.11b/g. As rotinas necessárias foram implementadas em *Python*. As seções a seguir apresentam detalhes da implementação da unidade de bordo e da unidade de acostamento.

3.1. Requisitos Básicos - Estrutura

O uso de rotinas específicas para a troca de TCT nas unidades de acostamento necessitou equipamentos que permitam a alteração do *firmware* padrão. Cada unidade de acostamento foi composta por um kit com autonomia de energia, ilustrado na Figura 3. Esse equipamento foi baseado no kit usado pelo projeto ReBUS [Ribeiro Júnior et al., 2011]. Cada kit usado no protótipo proposto é composto por um roteador D-Link modelo DIR-320, um pendrive USB 2.0 de 32 GB, um circuito regulador de tensão e uma bateria de 12V/7 Ah. A posição de cada unidade de acostamento era previamente conhecida. Foram utilizadas as antenas padrão que vem com os equipamentos.

Figura 3. Kit de acostamento.

Para determinar a distância entre dois pontos de acesso, foi utilizada a função de Haversine [Nordin et al., 2012]. Essa função é bastante usada em sistemas de navegação

e é capaz de fornecer a distância entre dois pontos de uma esfera utilizando coordenadas geográficas (latitude e longitude). Realizando uma aproximação da Terra como uma esfera perfeita, é possível apresentar um erro médio de 0.3% nos cálculos. A função de Haversine ($haversine(\theta)$) é definida como:

$$haversine(\theta) \equiv \sin^2\left(\frac{\theta}{2}\right) \tag{2}$$

Considerando dois pontos de uma esfera de raio R, com latitudes e longitudes (ϕ_1, λ_1), (ϕ_2, λ_2), respectivamente, a distância d é:

$$d = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right). \tag{3}$$

Ao utilizar um GPS com uma amostragem σ, a distância total D percorrida por um veículo no intervalo de tempo $[0, k * \sigma]$ é igual a $\sum_{n=0}^{k-1} d_{n,n+1}$, onde $d_{n,n+1}$ representa o cálculo da enésima distância utilizando a fórmula de Haversine para amostras espaçadas no tempo de σ.

3.2. Etapas Executadas no Nó Móvel

A execução do sistema na unidade de bordo do protótipo requer pelo menos seis passos: (1) varredura pelos ESSIDs conhecidos, associação e conexão com o ponto de acesso; (2) recebimento da TCT pelo veículo; (3) comparação e atualização da TCT recebida com a TCT local (no veículo); (4) detecção do momento em que o veículo passou pelo ponto de acesso, terminando o trecho; (5) cálculo da velocidade no trecho; e (6) envio da TCT do veículo para o ponto de acesso. É fundamental que o veículo envie a velocidade média calculada no trecho para o ponto de acesso antes que perca o sinal. A Figura 4 apresenta um diagrama com a execução dos passos para conexão e troca de informação entre veículo e ponto de acesso. Para que possa receber os quadros de sondas provenientes dos pontos de acesso, enquanto não estiver conectada, a interface de rede do cliente deve estar em modo monitor.

Como é possível observar na Figura 4, o cliente processa todos os quadros *beacon* recebidos a fim de determinar se o ESSID faz parte do sistema de monitoramento. Essa verificação é feita em arquivo XML local (Listagem 1). Para identificar em qual trecho o veículo se encontra ou definir a direção na via, usa-se a combinação dos dois últimos pontos de acesso, anterior e atual. Utilizando os momentos em que passou por cada ponto de acesso e a extensão do trecho, o veículo calcula a velocidade média. No código a seguir, é possível visualizar que trecho é composto por dois pontos de acesso e pela extensão. O arquivo armazena ainda as informações sobre cada ponto de acesso, que são: o ESSID, ou seja, o nome da rede; o endereço MAC e as coordenadas geográficas.

Listagem 1. Arquivo para definição do Trecho.

```
1 <root>
2    <path id='ID'>
3       <stretch>trecho</stretch>
```

```
4         <distance>extensao</distance>
5         <encodedPath>encodedPath</encodedPath>
6         <from>
7             <essid>essid</essid>
8             <macAddress>00:00:00:00:00:00</macAddress>
9             <location>
10                <latitude>latitude1</latitude>
11                <longitude>longitude1</longitude>
12            </location>
13        </from>
14        <to>
15            <essid>essid</essid>
16            <macAddress>ff:ff:ff:ff:ff:ff</macAddress>
17                <location>
18                    <latitude>latitude2</latitude>
19                    <longitude>longitude2</longitude>
20                </location>
21        </to>
22    </path>
23 </root>
```

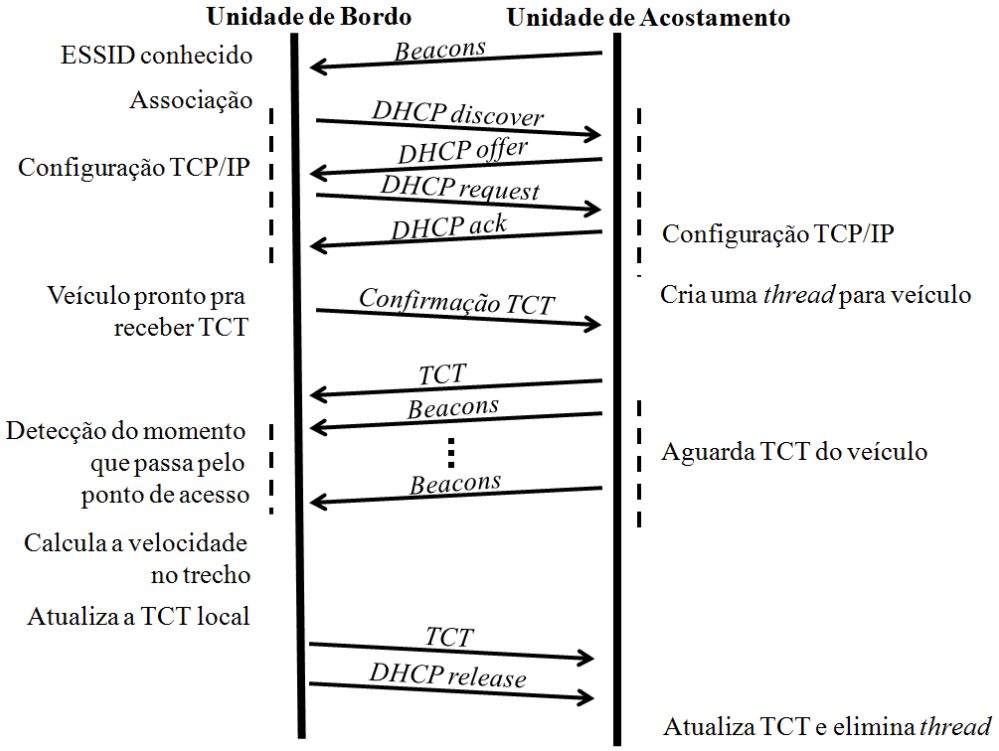


Figura 4. Diagrama das etapas de funcionamento do sistema proposto.

Uma vez que o ESSID destacado é conhecido, o veículo se associa ao ponto de acesso e aguarda o recebimento das informações de trânsito (via TCT) enviadas pelo ponto de acesso. Vale mencionar que a interface do cliente é retirada do modo monitor logo depois da associação com o ponto de acesso. Assim como demonstrado na Seção 2.1, o veículo atualiza a TCT local, levando em consideração o TTL em cada trecho. Após esse processo, o cliente volta a analisar os *beacons* a fim de determinar o

momento em que se encontra mais próximo do ponto de acesso. O algoritmo proposto em [Ribeiro Júnior et al., 2012b] considera tal momento como sendo aquele em que se recebe o *beacon* com maior potência de sinal. O veículo detecta que passou pelo ponto de acesso, finalizando o percurso do trecho, quando recebe um *beacon* com potência de sinal 10 dBm maior que a maior potência recebida do mesmo ponto de acesso. Após passar pelo ponto de acesso, o veículo calcula a velocidade no trecho e atualiza a TCT com a nova Condição e TTL. A tabela resultante é enviada para o ponto de acesso. Por fim, há a desassociação com o ponto de acesso e a procura pelo próximo ponto na trajetória.

Após o processo de desassociação, para evitar a reassociação, já que ele ainda pode receber *beacons* do ponto de acesso anterior, o endereço MAC passa a identificar o ponto de acesso anterior. Assim, o veículo sempre armazena o endereço MAC do ponto de acesso anterior e o atual (ou o próximo) para que possa definir o trecho.

3.3. Unidade de Acostamento

Na unidade de acostamento (ponto de acesso), há a necessidade de instalar um *firmware* de código aberto como o OpenWRT [Openwrt, 2011] para o funcionamento do algoritmo. Como se pode observar no diagrama da Figura 4, a unidade espera pelo momento em que um veículo se associa. Em seguida, o ponto de acesso espera uma confirmação de que o cliente está pronto para receber a TCT. Ao receber esta confirmação, o ponto de acesso envia a TCT. Para cada novo cliente conectado é instanciada uma nova *thread* para tratar a etapa de troca de informações separadamente. Uma vez enviada a TCT, a *thread* aguarda o envio da TCT da unidade de bordo. Caso exista mais de uma conexão simultânea, a unidade de acostamento deve sincronizar os processos, para que, ao atualizar a TCT local, considere todas as condições recebidas do último. Para isso, é realizado o cálculo da média harmônica simples. Nesta proposta não há divisão dos veículos em categorias uma vez que, como demonstrado em [Ribeiro Júnior et al., 2012a], utilizando a média harmônica para inferir a velocidade no trecho, a interferência de veículos preferenciais na condição do trecho só acontece em cenários muito específicos. Por outro lado, como demonstrado em [Treiber e Kesting, 2010], se houver veículos mais lentos que o fluxo normal, eles afetarão diretamente o tráfego nas outras pistas.

Para o protótipo utilizou-se uma rede aberta, no entanto o uso de chaves de criptografia não impede o funcionamento do sistema proposto.

4. Experimentos

Para realizar os experimentos, dois cenários foram montados no campus da Ilha do Fundão, na Universidade Federal do Rio de Janeiro. As unidades de acostamento foram compostas pelos kits apresentados na Figura 3. A aplicação cliente foi executada em um *laptop* Sony Vaio com processador I5-3210m, 6 GB de RAM e disco rígido de 640 GB e interface de rede sem fio. Para comparação dos resultados utilizou-se um GPS modelo "u-blox 5", que informa a posição quatro vezes por segundo.

4.1. Primeiro Cenário

No primeiro cenário, foram utilizados uma unidade de acostamento e um dispositivo móvel para verificar o tempo gasto na associação ao ponto de acesso e no envio das configurações de rede. Foi avaliada também a eficiência da proposta descrita na Seção 3.2

para detecção do momento em que o veículo passa pelo ponto de acesso podendo então desconectar. O trecho apresentado na Figura 5 possui 900 m de extensão. Uma limitação desse cenário é o limite de velocidade permitido nas vias, de 40 km/h.

Figura 5. Cenário - Experimento 1 - UFRJ.

A Tabela 1 apresenta os resultados do tempo de conexão, definido como o intervalo de tempo em que o veículo recebe o primeiro *beacon* do ESSID até o momento em que ele recebe as configurações de rede. As limitações do padrão IEEE 802.11b/g no que diz respeito a mobilidade e a atenuação por múltiplos caminhos aumentam a perda de quadros com o aumento da velocidade do veículo. Dessa forma, quanto mais rápido o veículo estiver, mais tempo ele leva para estabelecer a conexão. Estando mais lento, os efeitos da atenuação são menores, permitindo que a conexão seja concluída mais rapidamente. Durante os experimentos, foram detectadas cerca de 11 redes IEEE 802.11b/g utilizando os canais 9 e 11 (o canal 9 era utilizado pela unidade de acostamento).

O veículo sempre desconecta logo após passar pelo ponto de acesso, ao receber um sinal com potência 10 dBm menor que a máxima.

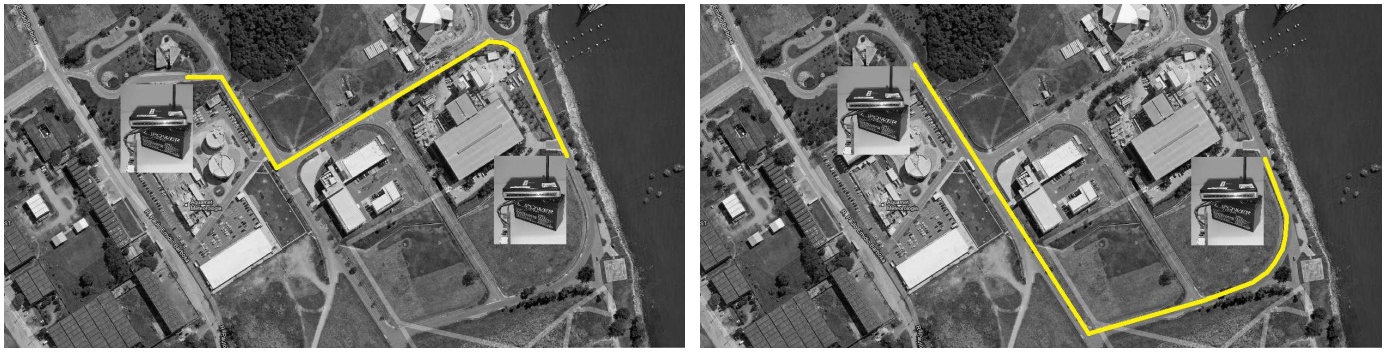
Tabela 1. Tempo de conexão entre unidade de bordo e unidade de acostamento.

Tempo de Conexão	
Velocidades	Tempo
20 km/h	< 1 seg
25 km/h	4 seg
35 km/h	7 seg
40 km/h	9 seg

4.2. Segundo Cenário

No segundo cenário, o experimento conta com duas unidades de acostamento e uma unidade de bordo. A Figura 6 apresenta os dois trechos utilizados. Para calcular a distância

entre os dois pontos de acesso, utilizando a fórmula de Haversine (Equação 2), a taxa de amostragem utilizada foi de 1 segundo. Na unidade de bordo mediu-se o tempo de associação, o tempo para receber as configurações TCP/IP, o tempo para executar as trocas de tabelas e o momento de desconexão.



(a) Trecho 1. (b) Trecho 2.

Figura 6. Experimento 2 - UFRJ.

A Figura 7 apresenta quatro momentos em que o veículo passa pelos pontos de acesso, executando todo o algoritmo. As linhas em destaque apresentam os momentos em que (1) o veículo encontrou o ponto de acesso, isto é, o momento em que o veículo recebeu o primeiro *beacon* daquele ESSID, (2) o momento em que o veículo terminou o processo de associação ao ponto de acesso e (3) o momento da desconexão. Os asteriscos representam os *beacons* recebidos. Como pode ser visto, a desconexão acontece logo após a unidade de bordo receber o sinal com maior potência, ou seja, quando a potência do sinal recebido é 10 dBm menor.

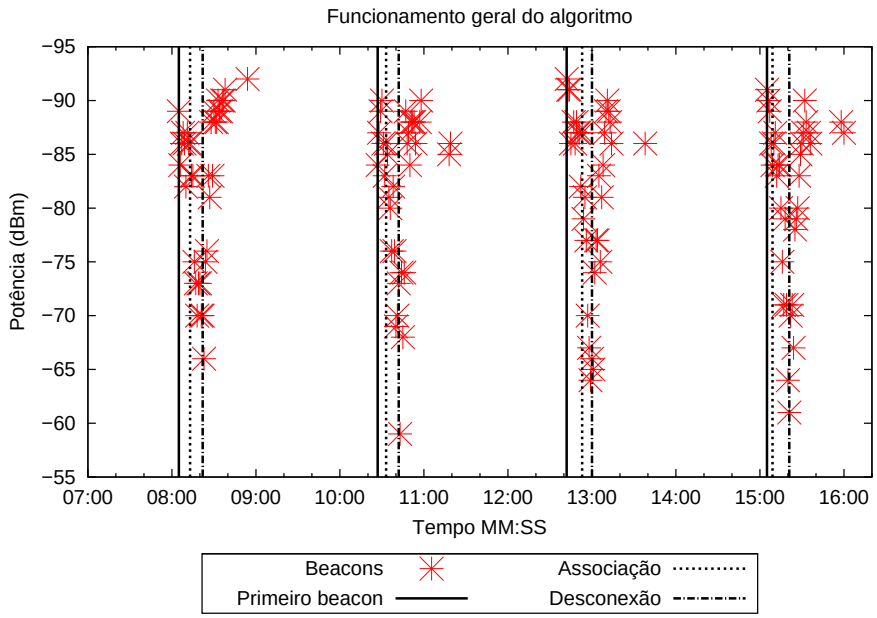


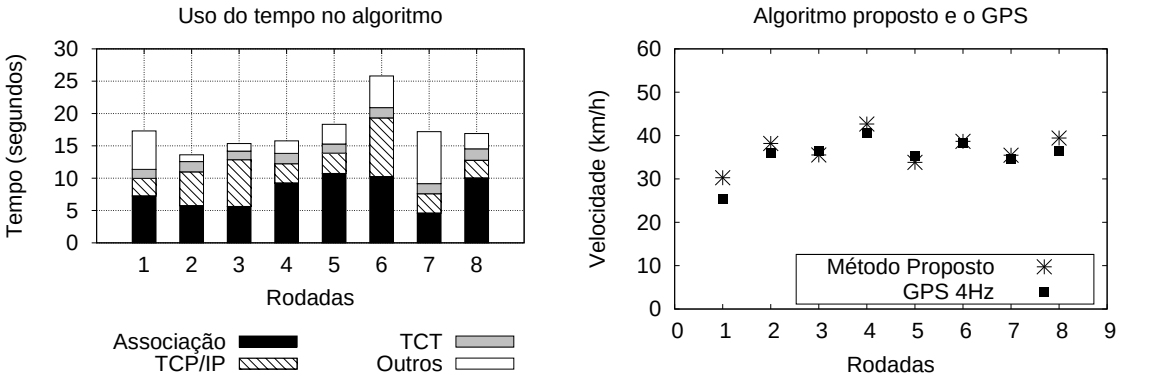
Figura 7. Funcionamento do algoritmo.

A Figura 8(a) detalha o processo para que seja possível visualizar o tempo gasto

em cada etapa. O intervalo de tempo entre o momento em que o veículo detectou um ESSID conhecido e se associou variou entre 6 e 11 segundos. Foram necessários de 3 a 9 segundos para receber as configurações de rede e enviar o aviso para receber a tabela. O tempo médio para recebimento da tabela com as condições dos trechos foi de 2 segundos. O veículo detectou que passou pelo ponto de acesso com precisão de 3 metros, no pior caso.

Como é possível observar, fazer a associação e receber as configurações de rede é a etapa que necessita de mais tempo. Ao menos duas opções podem reduzir o tempo de execução do algoritmo ao utilizar IEEE 802.11b/g. A primeira seria que o veículo utilizasse um endereço IP pré-definido ao detectar o ESSID conhecido. Outra opção seria tornar o primeiro ponto de acesso como o único responsável por distribuir os endereços IP, fazendo com que essa etapa aconteça somente no momento em que o veículo entra na via. Experimentos utilizando IP fixo na unidade de bordo reduziram o tempo necessário para receber as configurações de rede de 3 a 9 segundos para menos de 1 segundo.

A Figura 8(b) apresenta a condição inferida em cada trecho, tanto pelo método proposto quanto pelo GPS. Para gerar uma condição única utilizando o GPS, foi calculada a média harmônica de todos os valores obtidos no trecho. Como pode ser visto, os resultados são muito semelhantes. Considera-se que a condição inferida está correta quando os resultados estão dentro do mesmo intervalo que define as condições do trecho. Assim como Ribeiro Júnior et. al [Ribeiro Júnior et al., 2012b], foram definidos três intervalos. Caso a velocidade média esteja abaixo de 40 km/h, significa que o trecho apresenta condição lenta. Entre 40 e 80 km/h o trecho apresenta condição intermediária e acima de 80 km/h condição rápida.



(a) Histograma. (b) Velocidade experimental comparado com o GPS.

Figura 8. Tempo detalhado de cada etapa e Comparação de resultados.

A grande variação no tempo de associação com o ponto de acesso da Figura 8(a) é consequência do intervalo de tempo considerado, que inclui o primeiro *beacon* detectado e a associação completa com a unidade de bordo. Em regiões onde não há obstáculos, o alcance de um *beacon* é maior do que o necessário para que haja pareamento entre os dispositivos.

5. Conclusão e Trabalhos Futuros

Este artigo apresentou um sistema colaborativo para o monitoramento distribuído de veículos. Por não exigir que as unidades de acostamento estejam conectadas, o sistema pode ser utilizado mesmo em rodovias onde não existe infraestrutura, como energia ou cobertura celular.

Foram realizados experimentos em dois cenários com resultados muito similares aos obtidos por um GPS, com alta taxa de acerto ao inferir as condições dos trechos e a posição do veículo. Ao considerar a camada física no protótipo, uma parte expressiva do tempo de contato com o ponto de acesso era para executar rotinas de autenticação e endereçamento.

Como trabalho futuro pretende-se implementar um protótipo utilizando o padrão IEEE 802.11p e variar a estrutura proposta, interligando algumas unidades de acostamento para diminuir o tempo de defasagem entre pontos mais distantes. Pretende-se ainda fazer experimentos em maior escala, por meio de simulação, para analisar a quantidade de tráfego gerada pelo sistema e qual a relação com o aumento no número de carros.

Referências

(1997). Dynamic Host Configuration Protocol. *IETF RFC 2131*.

Balasubramanian, A., Mahajan, R. e Venkataramani, A. (2010). Augmenting Mobile 3G using WiFi. Em *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, p. 209–222, New York, NY, USA. ACM.

Cheung, S. Y., Coleri, S., Dundar, B., Ganesh, S., Tan, C.-W. e Varaiya, P. (2004). Traffic Measurement and Vehicle Classification with a Single Magnetic Sensor. Institute of transportation studies, research reports, working papers, proceedings, Institute of Transportation Studies, UC Berkeley.

Cisco (2012). Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016. Relatório técnico. Acessado em dezembro de 2012.

Cucchiara, R., Piccardi, M. e Mello, P. (2000). Image Analysis and Rule-based Reasoning for a Traffic Monitoring System. *Intelligent Transportation Systems, IEEE Transactions on*, 1(2):119 –130.

Edelmayer, A., Miranda, M. e Nebehaj, V. (2010). Cooperative Federated Filtering Approach for Enhanced Position Estimation and Sensor Fault Tolerance in Ad-hoc Vehicle Networks. *Intelligent Transport Systems, IET*, 4(1):82 –92.

Fazenda, B., Atmoko, H., Gu, F., Guan, L. e Ball, A. (2009). Acoustic Based Safety Emergency Vehicle Detection for Intelligent Transport Systems. Em *ICROS-SICE International Joint Conference 2009*. IEEE Xplore.

Google (2011). Google Maps. *Disponível em http://maps.google.com/support*. Acessado em dezembro de 2012.

Kassem, N., Kosba, A. e Youssef, M. (2012). RF-Based Vehicle Detection and Speed Estimation. Em *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, p. 1 –5.

Mohan, P., Padmanabhan, V. N. e Ramjee, R. (2008). Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones. Em *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, p. 323–336, New York, NY, USA. ACM.

Nordin, N., Zaharudin, Z., Maasar, M. e Nordin, N. (2012). Finding Shortest Path of the Ambulance Routing: Interface of A #x2217; Algorithm Using C# Programming. Em *Humanities, Science and Engineering Research (SHUSER), 2012 IEEE Symposium on*, p. 1569 –1573.

Openwrt (2011). OpenWRT - Wireless Freedom. Disponível em http://www.openwrt.org/. Acessado em dezembro de 2012.

Ribeiro Júnior, J. G., Campista, M. E. M. e Costa, L. H. M. K. (2012a). Opportunistic System for Collaborative Traffic Monitoring Using Existing IEEE 802.11 Networks. *Intelligent Vehicular Networking: V2V/V2I Communications and Applications - IEEE International Conference on Communications - ICC 2012*, p. 7294 –7298.

Ribeiro Júnior, J. G., Costa, L. H. M. K., Campista, M. E. M., Moraes, I. M., Alves, R. S., Couto, R. S., Silva, F. O. B., Valverde, L. G., Lanza, M. L. D., Camilo, B. C. V. e Amorim, M. D. (2011). GT-ReBUS: Redes de Acesso em Ônibus Universitários. *Disponível em http://www.gta.ufrj.br/gt-rebus*. Acessado em dezembro de 2012.

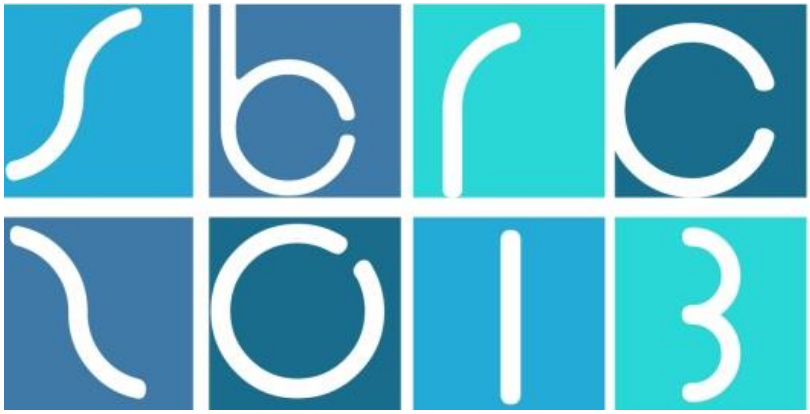
Ribeiro Júnior, J. G., Quintanilha, I. M., Campista, M. E. M. e Costa, L. H. M. K. (2012b). Evaluation of an Opportunistic Collaborative Traffic Monitoring System. *IFIP/IEEE Wireless Days Conference 2012*, p. 6.

Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S. e Eriksson, J. (2009). VTrack: Accurate, Energy-aware Road Traffic Delay Estimation Using Mobile Phones. Em *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, p. 85–98, New York, NY, USA. ACM.

Treiber, M. e Kesting, A. (2010). An Open-Source Microscopic Traffic Simulator. *Intelligent Transportation Systems Magazine, IEEE*, 2(3):6 –13.

Valerio, D., D'Alconzo, A., Ricciato, F. e Wiedermann, W. (2009). Exploiting Cellular Networks for Road Traffic Estimation: A Survey and a Research Roadmap. Em *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, p. 1 –5.

Yoon, J., Noble, B. e Liu, M. (2007). Surface Street Traffic Estimation. Em *Proceedings of the 5th international conference on Mobile systems, applications and services*, MobiSys '07, p. 220–232, New York, NY, USA. ACM.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 21

Redes Definidas por Software 2

Building upon RouteFlow: a SDN development experience

Allan Vidal[1,2], Fábio Verdi[2], Eder Leão Fernandes[1], Christian Esteve Rothenberg[1], Marcos Rogério Salvador[1],

[1] Fundação CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações
Campinas – SP – Brazil

[2]Universidade Federal de São Carlos (UFSCar)
Sorocaba – SP – Brazil

{allanv,ederlf,esteve,marcosrs}@cpqd.com.br, verdi@ufscar.br

Abstract. *RouteFlow is a platform for providing virtual IP routing services in OpenFlow networks. During the first year of development, we came across some use cases that might be interesting pursuing in addition to a number of lessons learned worth sharing. In this paper, we will discuss identified requirements and architectural and implementation changes made to shape RouteFlow into a more robust solution for Software Defined networking (SDN). This paper addresses topics of interest to the SDN community, such as development issues involving layered applications on top of network controllers, ease of configuration, and network visualization. In addition, we will present the first publicly known use case with multiple, heterogeneous OpenFlow controllers to implement a centralized routing control function, demonstrating how IP routing as a service can be provided for different network domains under a single central control. Finally, performance comparisons and a real testbed were used as means of validating the implementation.*

1. Introduction

Software Defined Networking (SDN) builds upon the concept of the separation of the data plane, responsible for forwarding packets, and the control plane, responsible for determining the forwarding behavior of the data plane. The OpenFlow protocol [McKeown et al. 2008], an enabling trigger of SDN, introduced the notion of programmable switches managed by a network controller / operating system: a piece of software that controls the behavior of the switches, forming a general view of the network and acting accordingly to application purposes.

The RouteFlow project [RouteFlow] aims to provide virtualized IP routing services on OpenFlow-enabled hardware following the SDN paradigm. Basically, RouteFlow links together an OpenFlow infrastructure to a virtual network environment running Linux-based IP routing engines (e.g. Quagga) to effectively run target IP routed networks on the physical infrastructure. As orchestrated by the RouteFlow control function, the switches are instructed via OpenFlow controllers working as proxies that translate protocol messages and events between the physical and the virtual environments.

The project counts with a growing user base worldwide (more than 1,000 downloads and more than 10,000 unique visitors since the project started in April, 2010). External contributions range from bug reporting to actual code submissions via the community-oriented GitHub repository. To cite a few examples, Google has contributed with an

SNMP plug-in and is currently working on MPLS support and new APIs of the Quagga routing engine. Indiana University has added an advanced GUI and run pilots with hardware switches in the US-wide NDDI testbed. UNIRIO has prototyped a single node abstraction with a domain-wide eBGP controller. UNICAMP has done a port to the Ryu OpenFlow 1.2 controller and is experimenting with new data center designs. While some users look at RouteFlow as "Quagga on steroids" to achieve a hardware-accelerated open-source routing solution, others are looking at cost-effective BGP-free edge designs in hybrid IP-SDN networking scenarios where RouteFlow offers a migration path to OpenFlow/SDN [Rothenberg et al. 2012]. These are ongoing examples of the power of innovation resulting from the blend of open interfaces to commercial hardware and open-source community-driven software development.

In this paper, we present re-architecting efforts on the RouteFlow platform to solve problems that were revealed during the first year of the public release including feedback from third party users and lessons learned from demonstrations using commercial OpenFlow switches.[1] The main issues we discuss include configurability, component flexibility, resilience, easy management interfaces, and collection of statistics. A description of our solutions to issues such as mapping a virtual network to a physical one, topology updates and network events will also be presented from the point of view of our routing application. The development experience made us review some original concepts leading to a new design that attempts to solve most of the issues raised in the first version [Nascimento et al. 2011].

One of the consequences of these improvements is that RouteFlow has been extended to support multiple controllers and virtual domains, becoming, as far as we know, the first distributed OpenFlow application that runs simultaneously over different controllers (e.g., NOX, POX, Floodlight, Ryu). Related work on dividing network control among several controllers has been proposed, for reasons of performance, manageability and scalability [Tavakoli et al. 2009, Heller et al. 2012]. We will present a solution that uses multiple heterogeneous controllers to implement a separation of routing domains from a centralized control point, giving the view of a global environment while keeping the individuality of each network and its controller.

Altogether, this paper contributes with insights on SDN application development topics that will certainly interest the vast majority of researchers and practitioners of the OpenFlow/SDN toolkit. We expect to further evolve discussions around traditional IP routing implemented upon SDN, and how it can be implemented as a service, opening new ways of doing hybrid networking between SDN and legacy IP/Eth/MPLS/Optical domains.

In Section 2 we present the core principles of RouteFlow discussing the previous design and implementation as well as the identified issues. In Section 3, we revisit the objectives and describe the project decisions and implementation tasks to refactor and introduce new features in the RouteFlow architecture. Section 4 presents results from the experimental evaluation on the performance of the middleware in isolation and the RouteFlow platform in action in two possible setups, one in a multi-lab hardware testbed and

[1]Open Networking Summit I (Oct/2011) and II (Apr/2012), Super Computing Research Sandbox (Nov/2011), OFELIA/CHANGE Summer School (Nov/2011), Internet2 NDDI (Jan/2012), 7^{th} API on SDN (Jun/2012). See details on: https://sites.google.com/site/routeflow/updates

another controlling multiple virtual network domains. Section 5 discusses related work on layered SDN application development, multiple controller scenarios, and novel routing schemes. Section 6 presents our work ahead on a research agenda towards broadening the feature set of RouteFlow. We conclude in Section 7 with a summary and final remarks.

2. Core Design Principles

RouteFlow was born as a Gedankenexperiment ("thought experiment") on whether the Linux control plane embedded in a 1U Ethernet switch prototype could be run out of the box in a commodity server with OpenFlow being the solely communication channel between the data and the control plane. Firstly baptized as QuagFlow [Nascimento et al. 2010] (Quagga + OpenFlow) the experiment turned out to be viable in terms of convergence and performance when compared to a traditional lab setup [Nascimento et al. 2011]. With increasing interest from the community, the RouteFlow project emerged and went public to serve the goal of connecting open-source routing stacks with OpenFlow infrastructures.

Fundamentally, RouteFlow is based on three main modules: the RouteFlow client (RFClient), the RouteFlow server (RFServer), and the RouteFlow proxy (RFProxy).[2] Figure 1 depicts a simplified view of a typical RouteFlow scenario: routing engines in a virtualized environment generate the forwarding information base according to the configured routing protocols (e.g., OSPF, BGP) and ARP processes. In turn, the routing and ARP tables are collected by the RFClient daemons and then translated into OpenFlow tuples that are sent to the RFServer, which adapts this FIB to the specified routing control logic and finally instructs the RFProxy, a controller application, to configure the switches using OpenFlow commands.

Matching packets on routing protocol and control traffic (e.g., ARP, BGP, RIP, OSPF) are directed by the RFProxy to the corresponding virtual interfaces via a software switch. The behavior of this virtual switch [3] is also controlled by the RFProxy and allows for a direct channel between the physical and virtual environments, eliminating the need to pass through the RFServer and RFClient, reducing the delay in routing protocol messages and allowing for distributed virtual switches and additional programmability.

2.1. Architectural issues

We identified the most pressing issues in the old architecture (see Figure 2(a)) as being:

Too much centralization. Most of the logic and network view was implemented and stored in the RFServer, without the help of third-party database implementations. The centralization of this design raised concerns about the reliability and performance of a network controlled by RouteFlow. It was important to relieve the server from this burden while providing a reliable storage implementation and facilitating the development of new services like GUI or custom routing logic (e.g. aggregation mode).

[2]As a historical note, the first QuagFlow prototype implemented RFServer and RFProxy as a single NOX application. After the separation (in the first RouteFlow versions) RFProxy was named RouteFlow controller. This caused some confusion, since it actually an application on top of an OpenFlow controller, so we renamed it. Its purpose and general design remain the same.

[3]We employ Open vSwitch for this task: http://openvswitch.org/

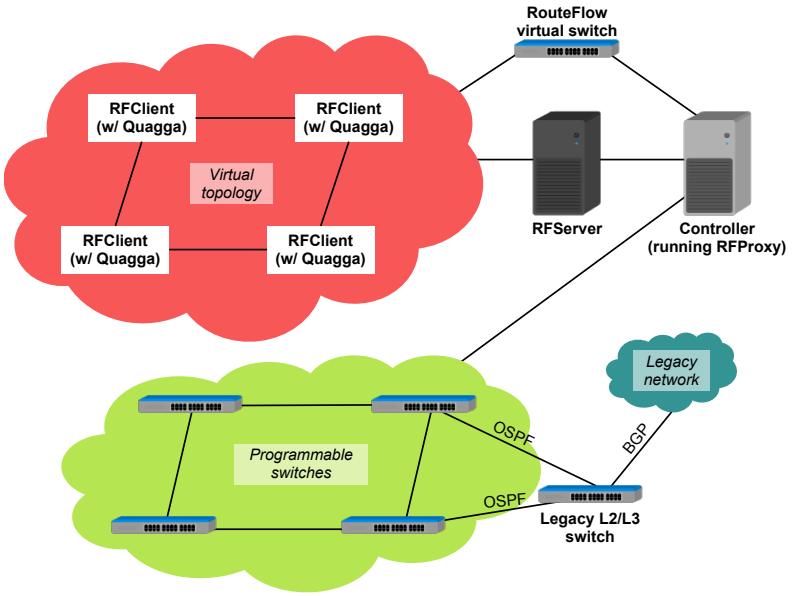


Figure 1. A typical, simplified RouteFlow scenario

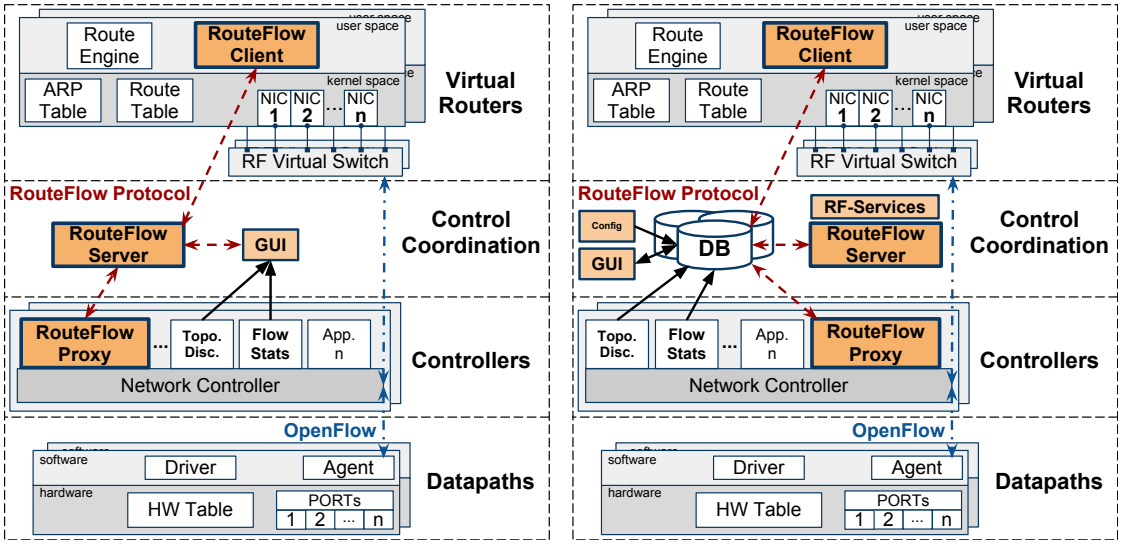
Deficits of inter-module communication. There was no clear and direct communication channel between the RFServer and the RFClients, and also the RFServer and the RFProxy application in the controller. An uniform method of communication was desired, that was extensible, programmer-friendly, and allowed to keep a convenient history of the messages exchanged by the modules to ease debugging and unit testing.

Lack of configurability. The most pressing issue was actually an implementation limitation: there was no way of telling the RouteFlow server to follow a defined configuration when associating the clients in the virtual environment with the switches in the physical environment. This forced the user to start the clients and connect the switches in a certain order without allowing for arbitrary component restart. A proper configuration scheme was needed to instruct the RFServer on how to behave whenever a switch or client joined the network under its control, rather than expect the user to make this match manually.

3. Project Decisions and Implementation

The new architecture, illustrated in Figure 2(b), retains the main modules and characteristics of the previous one. A central database that facilitates all module communication was introduced, as well as a configuration scheme and GUI tied to this database. While tackling the issues in the previous version, we also introduced new features, such as:

Make the platform more modular, extensible, configurable, and flexible. Anticipating the need for updating (or even replacing) RouteFlow components of the RouteFlow architecture, we have followed well-known principles from systems design that allow architectural evolvability [Ghodsi et al. 2011]: layers of indirection, system modularity, and interface extensibility. Meeting these goals involved also exposing configuration to the users in a clear way, reducing the amount of code, building modules with clearer purposes, facilitating the port of RFProxy to other controllers and enabling different services to be implemented on top of RFServer. The result is a better layered, distributed system, flexible enough to accommodate different virtualization use cases ($m : n$ map-



(a) First RouteFlow architecture (b) Redesigned RouteFlow architecture

Figure 2. Evolution of the RouteFlow architecture (as implemented)

ping of routing engine virtual interfaces to physical OpenFlow-enabled ports) and ease the development of advanced routing-oriented applications by the users themselves.

Keep network state history and statistics. One of the main advantages of centralizing network view is that it enables the inspection of its behavior and changes. When dealing with complex routing scenarios, this possibility is even more interesting, as it allows the network administrator to study the changes and events in the network, allowing to correlate and replay events or roll-back configurations.

Consider multi-controller scenarios. We have independently arrived at a controller-filtering architecture that is similar to the one proposed by Kandoo (as we will discuss later in Section 5). The hierarchical architecture allows for scenarios in which different networks (or portions of it) are controlled by different OpenFlow controllers. We can implement this new feature with slight changes to the configuration. Furthermore, the higher-level RouteFlow protocol layer abstracts most of the differences between OpenFlow versions 1.0/1.1/1.2/1.3, making it easier to support heterogeneous controllers.

Enable future works on replication of the network state and high availability. Originally, the RFServer was designed to be the module that took all the decisions regarding the network management, and we want to keep this role so that all routing policy and information can be centralized in a coherent control function. However, centralizing the server creates a single point of failure, and it is important that we consider possibilities to make it more reliable. By separating the network state from its responsibilities now, we can enable future solutions for achieving proper decentralization, benefiting from the latest results from the distributed systems and database research communities.

All the proposed changes are directly related to user and developer needs identified during the cycle of the initial release, some in experimental setups, others in real testbeds. In order to implement them, we went through code refactoring and architectural changes to introduce the centralized database and IPC and a new configuration scheme. The code refactoring itself involved many smaller tasks such as code standardization,

proper modularization, reduction of the number of external dependencies, easier testing, rewriting of the web-based graphical user interface and other minor changes that do not warrant detailed description in the scope of this paper. Therefore, we will focus on the newly introduced database and flexible configuration scheme.

3.1. Centralized database with embedded IPC

We first considered the issue of providing an unified scheme of inter-process communication (IPC) and evaluated several alternatives. Message queuing solutions like RabbitMQ or ZeroMQ,[4] were discarded for requiring a more complex setup and being too large and powerful for our purposes. Serializing solutions like ProtoBuffers and Thrift,[5] were potential candidates, but would require additional logic to store pending and already consumed messages, since they provide only the message exchange layer. When studying the use of NoSQL databases for persistent storage, we came across the idea to use the database itself as the central point for the IPC and natively keep a history of the RouteFlow workflow allowing for replay or catch-up operations. A publish/subscribe semantic was adopted for this multi-component, event-oriented solution.

After careful consideration of several popular NoSQL options (MongoDB, Redis, CouchDB)[6], we decided to implement the central database and the IPC mechanism upon MongoDB. The factors that lead to this choice were the programming-friendly and extensible JSON orientation plus the proven mechanisms for replication and distribution. Noteworthy, the IPC implementation (e.g., message factory) is completely agnostic to the DB of choice, should we change this decision.[7]

At the core of the RouteFlow state is the mapping between the physical environment being controlled and the virtual environment performing the routing tasks. The reliability of this network state in RFServer was questionable and it was difficult to improve this without delegating this function to another module. An external database fits this goal, allowing for more flexible configuration schemes. Statistics collection performed by the RFProxy could also be stored in this central database, based on which additional services could be implemented for data analysis or visualization.

The choice of delegating the core state responsibilities to an external database allows for better fault-tolerance, either by replicating the database or separating RFServer in several instances controlling it. The possibility of distributing RFServer takes us down another road: when associated with multiple controllers, it effectively allows for routing to be managed from several points, all tied by a unifying distributed database.

To wrap up, the new implementation is in line with the design rationale and best practices of cloud applications, and includes a scalable, fault-tolerant DB that serves as IPC, and centralizes RouteFlow's core state, the network view (logical, physical, and protocol-specific), and any information base used to develop routing applications (e.g., traffic histogram/forecasts, flow monitoring feedback, administrative policies). Hence,

[4]RabbitMQ: http://www.rabbitmq.com/; ZeroMQ: http://www.zeromq.org/.

[5]Thrift: http://thrift.apache.org; ProtoBuffers https://developers.google.com/protocol-buffers/.

[6]MongoDB: http://www.mongodb.org/; Redis: http://redis.io/; CouchDB: http://couchdb.apache.org/.

[7]While some may call Database-as-an-IPC an antipattern (cf. http://en.wikipedia.org/wiki/Database-as-IPC), we debate this belief when considering NoSQL solutions like MongoDB acting as a messaging and transport layer (e.g. http://shtylman.com/post/the-tail-of-mongodb/).

the DB embodies so-called Network Information Base (NIB) [Koponen et al. 2010] and Knowledge Information Base (KIB) [Saucez and et al. 2011].

3.2. Flexible configuration scheme

In the first implementation of RouteFlow, the association between VMs (running RFClients) and the OpenFlow switches was automatically managed by the RFServer with the chosen criteria being the order of registration: the nth client to register would be associated with the nth switch to join the network. The main characteristic of this approach is that it does not require any input from the network administrator other than taking care of the order in which switches join the network.

While this approach works for experimental and well-controlled scenarios, it posed problems whenever the switches were not under direct control. To solve this issue, we devised a configuration approach that would also serve as the basis for allowing multiple controllers to manage the network and ease arbitrary mappings beyond 1:1. In the proposed configuration approach, the network administrator is required to inform RouteFlow about the desired mapping. This configuration is loaded and stored in the centralized database. Table 1 details the possible states a mapping entry can assume. Figure 3 illustrates the default RFServer behavior upon network events.

Whenever a switch[8] joins the network, RFProxy informs the RouteFlow server about each of its physical ports. These ports are registered by the server in one of two ways explicited by Table 1: as (i) *idle datapath port* or (ii) *client-datapath association*. The former happens when there is either no configuration for the datapath port being registered or the configured client port to be associated with this datapath port has not been registered yet. The latter happens when the client port that is to be associated with this datapath (based on the configuration) is already registered as idle.

When a RFClient starts, it informs the RouteFlow server about each of its interfaces (ports). These ports are registered by the RFServer in one of two states shown in Table 1: as an idle client port or an client-datapath association. The association behavior is analogous to the one described above for the datapath ports.

After the association, the RFServer asks the RFClient to trigger a message that will go through the virtual switch to which it is connected and reach the RFProxy. When this happens, the RFProxy becomes aware of the connection between the RFClient and its virtual switch, informing the RFServer. The RFServer then decides what to do with this information. Tipically, the RFProxy will be instructed to redirect all traffic coming from the a virtual machines to the physical switch associated with it, and vice-versa. In the event of a switch leaving the network, all the associations involving the ports of the

Table 1. Possible association states

Format	Type
vm_id, vm_port, -, -, -, -, -	idle client port
-, -, -, -, dp_id, dp_port, ct_id	idle datapath port
vm_id, vm_port, dp_id, dp_port, -, -, ct_id	client-datapath association
vm_id, vm_port, dp_id, dp_port, vs_id, vs_port, ct_id	active client-datapath association

[8]Terms datapath and switch are used interchangeably.

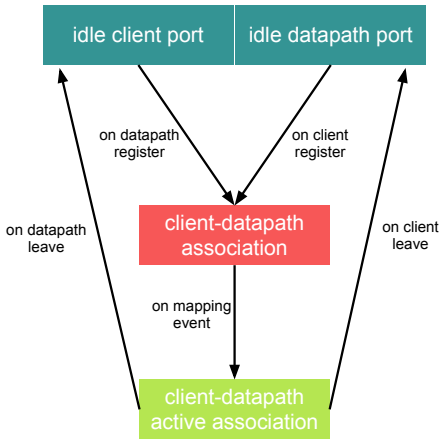


Figure 3. RFServer default association behavior

switch are removed, leaving idle client ports in case there was an association. In case the datapath comes back, RouteFlow will behave as if it were a new datapath, as described above, restoring the association configured by the operator.

An entry in the configuration file contains a subset of the fields identified in Table 1: `vm_id`, `vm_port`, `dp_id`, `dp_port`, `ct_id`. These fields are enough for the association to be made, since remaining fields related to the virtual switch attachment (`vs_*`) are defined at runtime. The `ct_id` field identifies the controller to which the switch is connected. This mechanism allows RouteFlow to deal with multiple controllers, either managing parts of the same network or different networks altogether.

Considering that the virtual environment can also be distributed, it becomes possible to run several routing domains on top of a single RFServer, facilitating the management of several routed networks under a single point. This segmentation presents a possible solution for provisioning of routing as a service [Lakshminarayanan et al. 2004]. In this sense, our solution is capable of controlling independent networks, being different ASes, subnetworks, ISPs or a combination of them, a pending goal of the original RouteFlow paper [Nascimento et al. 2011] to apply a PaaS model to networking.

4. Evaluation

To validate the new developments, we conducted a number of experiments and collected data to evaluate the new architecture and exemplify some new use cases for RouteFlow. The code and tools used to run these tests are openly available.[9] The benchmarks were made on a Dell Latitude e6520 with an Intel Core i7 2620M processor and 3 GB of RAM.

Simple performance measurements were made using the *cbench* tool [Tootoonchian et al. 2012], which simulates a number of OpenFlow switches generating requests and listening for flow installations. We adapted *cbench* to fake ARP requests (inside 60 bytes `packet-in` OpenFlow messages). These requests are handled by a modified version of the RFClient so that it ignores the routing engine. This way, we are effectively eliminating the influences of both the hardware and software which are not under our control, measuring more closely the specific delay introduced by RouteFlow.

[9]https://github.com/CPqD/RouteFlow/tree/benchmark

4.1. How much latency is introduced between the data and control planes?

In latency mode, *cbench* sends an ARP request and waits for the `flow-mod` message before sending the next request. The results for RouteFlow running in latency mode on POX and NOX are shown in Figure 4.

Each test is composed by several rounds of 1 second in duration, in which fake packets are sent to the controller and then handled by RFProxy that redirects them to the corresponding RFClient. For every test packet, the RFClient is configured to send a flow installation message. By doing this, we are testing a worst-case scenario in which every control packet results in a change in the network. These tests are intended to measure the performance and behavior of the new IPC mechanism.[10]

Figure 4 illustrates the cumulative distribution of latency values in three tests. Figure 4a shows the latency distribution for a network of only 1 switch. In this case, the IPC polling mechanism is not used to its full extent, since just one message will be queued every time. Therefore, the latency for the majority of the rounds is around the polling timeout. Figure 4b shows the accumulated latency, calculated considering all 4 switches as one. When compared to Figure 4c, which shows the average latency for all the 4 switches, the scales differ, but the behavior is similar. The accumulated latency shows that the IPC performs better in relation to the case in Figure 4a, mostly because the IPC will read all messages as they become available; when running with more than one switch, it is more likely that more than one message will be queued at any given time, keeping the IPC busy in a working cycle, not waiting for the next poll timeout.

Another comparison based on Figure 4 reveals that RouteFlow running on top of NOX (RFProxy implemented in C++) is more consistent in its performance, with most cycles lasting less than 60 ms. The results for POX (RFProxy implemented in Python) are less consistent, with more cycles lasting almost twice the worst case for NOX.

4.2. How many control plane events can be handled?

In throughput mode, *cbench* keeps sending as many ARP requests as possible in order to measure how many flow installations are made by the application. The throughput test stresses RouteFlow and the controller, showing how many flows can be installed in a single round lasting for 1 second. The results in Table 2 show how many flows can

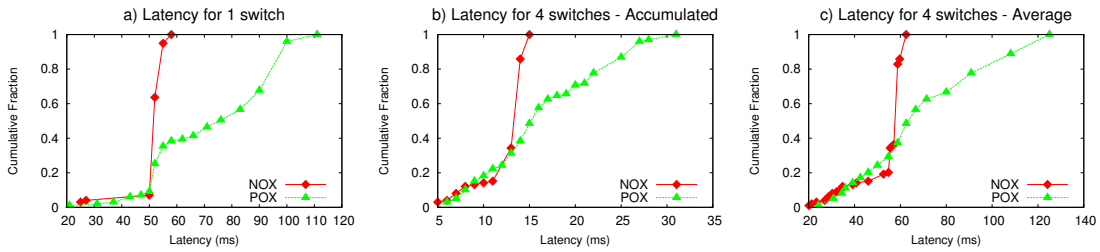


Figure 4. Latency CDF graphs for NOX and POX controlling a single network with 1 and 4 switches (taken from 100 rounds)

[10]The IPC mechanism uses a 50 ms polling time to check for unread messages. This value was chosen because it optimizes the ratio of DB access to message rate when running in latency mode. Whenever a polling timeout occurs, the IPC will read all available messages.

be installed in all of the switches in the network during a test with 100 rounds lasting 1 second each. The results show that the number of switches influence the number of flows installed per second, more than the choice of the controller.

Table 2. Total number of flows installed per second when testing in throughput mode (Average, standard deviation and 90% percentile taken from 100 rounds).

Controller	1 switch		4 switches	
	# Flows$_{avg}$	# Flows$_{90\%}$	# Flows$_{avg}$	# Flows$_{90\%}$
POX	915.05 $\pm$62.11	1013.0	573.87 $\pm$64.96	672.0
NOX	967.68 $\pm$54.85	1040.0	542.26 $\pm$44.96	597.0

4.3. What is the actual performance in a real network?

Test on a real network infrastructure were performed using the control framework of the FIBRE project,[11] with resources in two islands separated by 100 km (the distance between the CPqD lab in Campinas and the LARC lab at USP in São Paulo). To evaluate the delay introduced by the virtualized RouteFlow control plane, we measured the round-trip time from end-hosts when sending ICMP (*ping*) messages to the interfaces of the virtual routers (a LXC container in the RouteFlow host). This way, we are effectively measuring the compound delay introduced by the controller, the RouteFlow virtual switch, and the underlying network, but not the IPC mechanism. The results are illustrated in Table 3 for the case where the RouteFlow instance runs in the CPqD lab with one end-host connected in a LAN, and the second end-host located at USP. The CPqD-USP connectivity goes through the GIGA network and involves about ten L2 devices. The end-to-end delay observed between the hosts connected through this network for *ping*) exchanges exhibited line-rate performance, with a constant RTT around 2 ms. The results in Table 3 also highlight the performance gap between the controllers. The NOX version of RFProxy introduces little delay in the RTT, and is more suited for real applications

Table 3. RTT (milliseconds) from a host to the virtual routers in RouteFlow (average and standard deviation taken from 1000 rounds)

Controller	host@CPqD	host@USP
POX	22.31 $\pm$16.08	24.53 $\pm$16.18
NOX	1.37 $\pm$0.37	3.52 $\pm$0.59

4.4. How to split the control over multiple OpenFlow domains?

In order to validate the new configuration scheme, a simple proof-of-concept test was carried to show the feasibility of more than one network being controlled by RouteFlow. This network setup is illustrated in Figure 5, and makes use of the flexibility of the new configuration system. A central RFServer controls two networks: one contains four OpenFlow switches acting as routers being controlled by a POX instance, and the other contains a single OpenFlow switch acting as a learning switch being controlled by a NOX instance. In this test, RouteFlow was able to properly isolate the routing domains belonging to each network, while still having a centralized view of the networks.

[11] http://www.fibre-ict.eu/

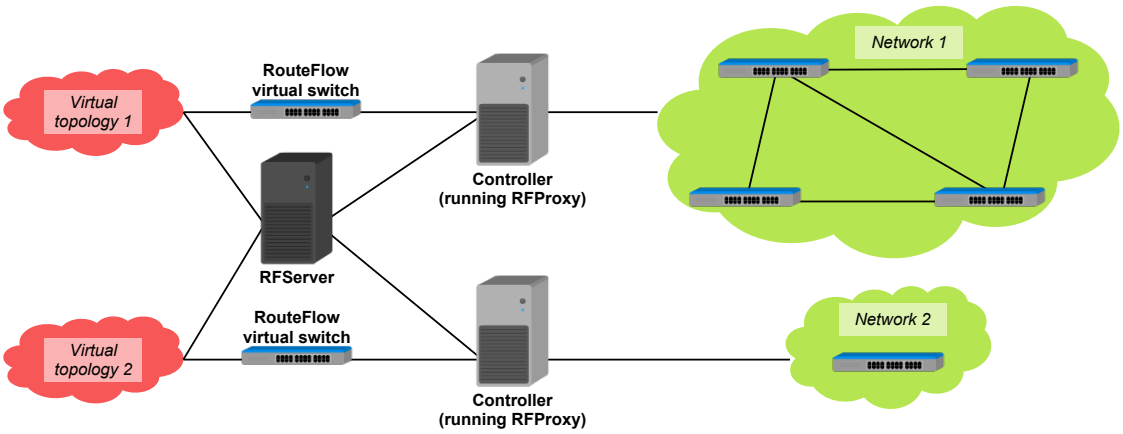


Figure 5. Test environment showing several controllers and networks

5. Related work

Layered controller application architectures. Our architectural work on RouteFlow is very similar to a recent proposal named Kandoo [Hassas Yeganeh and Ganjali 2012]. In Kandoo, one or more local controllers is directly connected to one or more switches. Messages and events that happen often and are better dealt with less latency when treated in these local (first hop) controllers. A root controller (that may be distributed), treats less frequent application-significant events, relieving the control paths at higher layers. Comparing RouteFlow and Kandoo, a notable similarity is adopting a division of roles when treating events. In RouteFlow, the RFProxy is responsible for dealing with frequent events (such as delivering `packet-in` events), only notifying the RFServer about some network-wide events, such as a switch joining or leaving. In this light, RFServer acts as a root controller in Kandoo. A key difference is the inclusion of a virtual environment on top of RFServer. This extra layer contains much of the application logic, and can be easily modified and distributed without meddling with the actual SDN application (RouteFlow). We also differ in message workflow because routing packets are sent from the RFProxy directly to the virtual environment, as determined by RFServer but without going through it. This creates a better performing path, partially offsetting the introduction of another logic layer in the architecture.

Trade-offs and controller placement. Though we do not directly explore performance and related trade-offs, some other works have explored the problem of controller placement [Heller et al. 2012] and realizing a logically centralized control functions [Levin et al. 2012]. Both lines of work may reveal useful insights when applying multiple controllers to different topologies using RouteFlow.

Network slicing. Flowvisor [Sherwood et al. 2010] bears some resemblance in that the roles of several controllers are centralized in a unique point with global view. In this case, the several instances of RFProxy behave as controllers, each with a view of their portion of a network, while RFServer centralizes all subviews and is a central point to implement virtualization policies. However, our work has much more defined scope around IP routing as a service, rather than serve as a general-purpose OpenFlow slicing tool.

Routing-as-a-Service. By enabling several controllers to be managed centrally by RouteFlow, we have shown a simple implementation towards routing as a ser-

vice [Lakshminarayanan et al. 2004] based on SDN. OpenFlow switches can be used to implement routers inside either Routing Service Provider (RSP) or directly at the ASes (though this would involve a considerably larger effort). These routers could be logically separated in different domains, while being controlled from a central interface. One of the key points of RouteFlow is its integration capabilities with existing routing in legacy networks. The global and richer view of the network facilitated by SDN may also help implement some issues related to routing as a service, such as QoS guarantees, conflict resolution and custom routing demands [Kotronis et al. 2012].

Software-defined router designs. Many efforts are going on into software routing designs that benefit from the advances of general-purpose CPU and the flexibility of open-source software routing stacks. Noteworthy examples include XenFlow [Mattos et al. 2011] that uses a hybrid virtualization system based on Xen and OpenFlow switches following the SDN control plane split but relying on software-based packet forwarding. An hybrid software-hardware router design called Fibium [Sarrar et al. 2012] relies on implementing a routing cache on the hardware flow tables while keeping the full FIB in software.

6. Future and ongoing work

There is a long list of ongoing activities around RouteFlow, including:

High-availability. Test new scenarios involving MongoDB replication, stand-by shadow VMs, and datapath OAM extensions (e.g. BFD triggers). While non-stop-forwarding is an actual feature of OpenFlow split architectures in case of controller disconnection, further work in the routing protocols is required to provide graceful restart. Multi-connection and stand-by controllers introduced in OpenFlow 1.x[12] will be pursued as well. The fast fail-over group tables in v1.1 and above allow to implement fast prefix-independent convergence to alternative next hops.

Routing services. A serious push towards a central routing services provider on top of RouteFlow can be made if we build the capabilities, improvements in the configurability and monitoring in the graphical user interface in order to provide more abstract user interfaces and routing policy languages to free users from low-lvel configuration tasks and experience a true XaaS model with the benefits of outsourcing [Kotronis et al. 2012]. In addition, router multiplexing and aggreagation will be furthered developed. New routing services will be investigated to assist multi-homing scenarios with policy-based path selection injecting higher priority (least R$ cost or lowest delay) routes.

Hybrid software/hardware forwarding. To overcome the flow table limits of current commercial OpenFlow hardware, we will investigate simple virtual aggregation techniques (IETF SimpleVA) and hybrid software/hardware forwarding approaches in spirit of smart flow caching [Sarrar et al. 2012].

Further testing. Using the infrastructure being built by the FIBRE project, we will extend the tests on larger-scale setups to study the impact of longer distances and larger networks. We intend to extend the federation with FIBRE islands from UFPA and UFSCar, and even internationally to include resources from the European partners like i2CAT. Further work

[12]More benefits from moving to the newest versions include IPv6 and MPLS matching plus QoS features. Currently, Google is extending RouteFlow to make use of Quagga LDP label info.

on unit tests and system tests will be pursued including recent advances in SDN testing and debugging tools [Handigol et al. 2012].

7. Conclusions

RouteFlow has been successful in its first objective: to deliver a software-defined IP routing solution for OpenFlow networks. Now that the first milestones have been achieved, our recent work helps to position RouteFlow for the future, enabling the introduction of newer and more powerful features that go much beyond its initial target. The lessons we learned developing RouteFlow suggest SDN practitioners to pay attention to issues such as the (i) amount of centralization and modularization, (ii) the importance of IPC/RPC/MQ, and (iii) flexible configuration capabilities for diverse practical setups.

While OpenFlow controllers often provide means for network view and configuration, their APIs and features often differ, making it important to speak a common language inside an application, making it much easier to extend and port to other controllers by defining so sought northbound APIs. It was also invaluable to have a central messaging mechanism, which provided a reliable and easy-to-debug solution for inter-module communication. As an added value to these developments, our recent work in providing graphical tools, clear log messages, and an easy configuration scheme are vital to allow an SDN application going "into the wild", since these tasks can become quite difficult when involving complex networks and real-world routing scenarios.

As for the future, we are excited to see the first pilots going live in operational trials and further advance on the missing pieces in a community-based approach. Building upon the current architecture and aiming for larger scalability and performance, we will seek to facilitate the materialization of Routing-as-a-Service solutions, coupled with high-availability, better configurability and support for more routing protocols such as IPv6 and MPLS. This will help make RouteFlow a more enticing solution to real networks, fitting the needs of highly virtualized environments such as data centers, and becoming a real alternative to closed-source or software-based edge boxes in use at IXP or ISP domains.

8. References

References

Ghodsi, A., Shenker, S., Koponen, T., Singla, A., Raghavan, B., and Wilcox, J. (2011). Intelligent design enables architectural evolution. In *HotNets '11*.

Handigol, N., Heller, B., Jeyakumar, V., Maziéres, D., and McKeown, N. (2012). Where is the debugger for my software-defined network? In *HotSDN '12*.

Hassas Yeganeh, S. and Ganjali, Y. (2012). Kandoo: a framework for efficient and scalable offloading of control applications. In *HotSDN '12*.

Heller, B., Sherwood, R., and McKeown, N. (2012). The controller placement problem. In *HotSDN '12*.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., et al. (2010). Onix: A distributed control platform for large-scale production networks. *OSDI'10*.

Kotronis, V., Dimitropoulos, X., and Ager, B. (2012). Outsourcing the routing control logic: better internet routing based on sdn principles. In *HotNets '12*.

Lakshminarayanan, K., Stoica, I., Shenker, S., and Rexford, J. (2004). Routing as a service. Technical Report UCB/EECS-2006-19.

Levin, D., Wundsam, A., Heller, B., Handigol, N., and Feldmann, A. (2012). Logically centralized?: state distribution trade-offs in software defined networks. In *HotSDN '12*.

Mattos, D., Fernandes, N., Duarte, O., and de Janeiro-RJ-Brasil, R. (2011). Xenflow: Um sistema de processamento de fluxos robusto e eficiente para migraç ao em redes virtuais. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC)*.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Nascimento, M., Rothenberg, C., Denicol, R., Salvador, M., and Magalhaes, M. (2011). Routeflow: Roteamento commodity sobre redes programáveis. *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC)*.

Nascimento, M. R., C. Esteve Rothenberg, Salvador, M. R., and Magalhães, M. F. (2010). QuagFlow: partnering Quagga with OpenFlow. *SIGCOMM CCR*, 40:441–442.

Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., Corrêa, C. N. A., Cunha de Lucena, S., and Raszuk, R. (2012). Revisiting routing control platforms with the eyes and muscles of software-defined networking. In *HotSDN '12*.

RouteFlow. Documentation. `http://go.cpqd.com.br/routeflow`. Acessado em 04/10/2012.

Sarrar, N., Uhlig, S., Feldmann, A., Sherwood, R., and Huang, X. (2012). Leveraging Zipf's law for traffic offloading. *SIGCOMM Comput. Commun. Rev.*, 42(1):16–22.

Saucez, D. and et al. (2011). Low-level design specification of the machine learning engine. EU FP7 ECODE Project. Deliverable D2.3.

Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. (2010). Can the production network be the testbed? In *OSDI'10*.

Tavakoli, A., Casado, M., Koponen, T., and Shenker, S. (2009). Applying nox to the datacenter. *Proc. HotNets (October 2009)*.

Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., and Sherwood, R. (2012). On controller performance in software-defined networks. In *Hot-ICE '12*.

Uma Arquitetura para o Aprovisionamento de QoS Interdomínios em Redes Virtuais baseadas no OpenFlow

Diego dos Passos Silva, Allan Borges Pontes, Edson Adriano Maravalho Avelar, Kelvin Lopes Dias.

Centro de Informática - Universidade Federal de Pernambuco (UFPE)
CEP: 50740-540 - Recife - PE - Brasil

{dps4,abp,eama,kld}@cin.ufpe.br

Abstract. *With the advent of software-defined networks (SDN) and, in particular, the OpenFlow platform, new solutions for QoS provisioning are needed to maintain the applications requirements, as they cross different administrative domains which will compose the new Future Internet ecosystem based on virtual networks. This article presents an architecture based on virtualization of networks with end-to-end QoS support considering two levels of mapping. The first one maps QoS specifications (QSPEC) between OpenFlow and IEEE 802.1p priority scheme. The second level provides mapping and interdomain interoperability through NSIS (Next Steps in Signaling) protocol. The performance results obtained from an OpenFlow testbed demonstrate the effectiveness of the proposal.*

Resumo. *Com o advento das redes definidas por software (SDN Software Defined Networks) e, em particular, da plataforma OpenFlow, novas soluções para o aprovisionamento de QoS são necessárias para manter os requisitos das aplicações, enquanto estas atravessam diversos domínios administrativos que constituirão o novo ecossistema da Internet do Futuro. Este artigo apresenta uma arquitetura, baseada em virtualização de redes, que fornece suporte à QoS fim-a-fim considerando dois níveis de mapeamento. O primeiro nível mapeia especificações de QoS (QSPEC) entre fluxos OpenFlow e o esquema de prioridades do IEEE 802.1p. O segundo fornece mapeamento e interoperabilidade interdomínios através do protocolo NSIS (Next Steps in Signaling). Os resultados de desempenho obtidos a partir de um testbed OpenFlow demonstram a eficácia da proposta.*

1. Introdução

Prover QoS (*Quality of Service*) fim-a-fim ainda é um dos maiores problemas para o sucesso de determinados serviços nos sistemas heterogêneos de telecomunicações usados ao redor do mundo. Abordagens para o aprovisionamento de QoS na Internet foram extensivamente discutidas no contexto das arquiteturas de Serviços Integrados (IntServ) e Serviços Diferenciados (DiffServ) [Gozdecki et al. 2003], com diversos mecanismos propostos e avaliados. Entretanto, algumas dessas soluções arquiteturais

* Este trabalho foi realizado com recursos do projeto ReVir: Redes Virtuais para a Internet do Futuro, financiado pelo CTIC/RNP

são complexas e/ou de difícil implementação ou aceitação pelos provedores de acesso e de *backbone*, já que ou apresentam custo muito elevado ou não possuem garantias de QoS fim-a-fim.

Com o advento da virtualização de redes e da plataforma OpenFlow [OpenFlow 2012], novas soluções para o aprovisionamento de QoS são necessárias para manter os requisitos das aplicações, enquanto estas atravessam diversos domínios administrativos que constituirão o novo ecossistema, baseada em redes virtuais, da Internet do Futuro [Schonwalder et al. 2009]. Dessa forma, surge a seguinte questão: "Como suportar aplicações com requisitos distintos e que atravessam redes que combinam equipamentos que operam na camada 3 (L3 – *Layer 3*) e, exclusivamente, na camada 2 (L2 – *Layer 2)*, garantindo QoS fim-a-fim no contexto de redes virtuais?".

Este artigo propõe e avalia uma arquitetura, denominada RME (*Resource Mediation Engine*), que provê suporte à QoS fim-a-fim em redes virtuais baseadas no OpenFlow. A proposta considera dois níveis de mapeamento. O primeiro nível mapeia especificações de QoS (QSPEC) entre fluxos OpenFlow e o esquema de prioridades do PCP (*Priority Code Point*) [IEEE 2011]. O segundo nível fornece mapeamento e interoperabilidade interdomínios através do protocolo NSIS (*Next Steps in Signaling*) [Fu et al. 2005].

O artigo está organizado da seguinte forma: Na Seção 2, os trabalhos relacionados são discutidos. A Seção 3 apresenta a arquitetura proposta. O *testbed* utilizado para validarmos a solução e os resultados obtidos é abordado na Seção 4. A Seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Esta seção discute os trabalhos relacionados ao OpenFlow e à QoS em redes virtuais. O OpenFlow é uma iniciativa recente e, por isso, há poucas propostas para provimento de QoS nesta tecnologia. Até a versão 1.0 da especificação do OpenFlow, o provimento de QoS era restrito à reserva de banda para fluxos ou *slices (*fatias de redes virtuais*)*. Por isso, este mecanismo é o mais utilizado ou estendido pelas propostas citadas nessa seção.

No trabalho [Civanlar et al. 2010] os autores propõem um controlador de rede virtual que configura as rotas dos fluxos com base em requisitos de QoS para tráfego de vídeo escalável (SVC - *Scalable Video Coding*) em termos de atraso fim-a-fim e de melhor esforço baseado na perda de pacotes. A lógica do controlador é implementada no NOX [Gude et al. 2008]. Os autores em [Egilmez et al. 2011] também propõem uma arquitetura de roteamento baseado em QoS para SVC através de redes OpenFlow. O artigo divide o tráfego em *SVC base layer* (QoS sem perda de pacotes) e *SVC enhancement layer* (QoS com certa tolerância a perda de pacotes e tráfego de melhor esforço). O algoritmo de roteamento utilizado é o "*Network Simplex*", o qual é implementado na ferramenta/biblioteca de otimização de rede LEMON [LEMON 2012].

Em [Min al. 2010] é proposto um mecanismo de roteamento em redes virtuais com base nos requisitos de usuário e estado da rede. A solução foi avaliada utilizando *switches* OpenFlow baseados em NetFPGA (*Hardware/Software* de prototipação). A proposta pode ser dividida em quatro componentes: ENVI, LAVI, componente de monitoramento de rede (*Network monitoring component*) e componente de roteamento

baseado em QoS (*QoS routing component*). Contudo, os autores não detalham os parâmetros e algoritmos utilizados nos dois últimos componentes da solução.

Em [Egilmez et al. 2012] os autores propõem uma solução chamada de OpenQoS. O OpenQoS seleciona as melhores rotas para fluxos OpenFlow baseado em requisitos de QoS. Contudo, apesar de os autores sugerirem um conjunto de parâmetros de QoS para a tomada de decisão, apenas a vazão disponível é utilizada. Em [Mattos e Duarte 2012] os autores propõem uma solução chamada QFlow. Esta proposta baseia-se no controle de recursos de roteadores virtuais OpenFlow usando Xen (XenFlow) como processamento, memória e vazão mínima e máxima de cada roteador. Em [Zhang et al. 2009] o encaminhamento de pacotes é melhorado e a QoS é fornecida através de OSPF, roteadores virtuais (VINI) e do DSCP (Differentiated Services Code Point). O artigo [Bozakov 2011] propõe o roteador virtual como um serviço e um algoritmo de custo mínimo de fluxo para lidar com a alocação dos fluxos.

Em [Puschita et al 2010], os autores propõem um modelo de QoS chamado I-NAME para escolher o melhor caminho fim-a-fim com base em perfis de QoS. No entanto, os autores não abordam soluções específicas para a virtualização de redes. Os autores em [Kassler et al 2012] propõem uma arquitetura baseada em elementos de controle inteligentes e no protocolo SIP (*Session Initiation Protocol*) para o provimento de QoS em redes OpenFlow. Contudo, a proposta não foi testada ou simulada. Com exceção do SIP nenhuma outra tecnologia foi sugerida para a implementação da solução.

Em resumo, encontrou-se apenas a proposta [Kim et al. 2010] que desenvolveu um mecanismo de priorização chamado de SSF (*Shortest Span First*) utilizado para, segundo os autores, maximizar a probabilidade de satisfazer os requisitos de desempenho de um novo fluxo e, ao mesmo tempo, minimizar o número de fluxos rejeitados. O SSF se baseia numa versão modificada do RCSP (*Rate-Controlled Static-Priority Queueing*) [Zhang e Ferrari 1993] para fornecer a prioridade entre os fluxos. Neste caso, ao contrário do RME, a proposta não fornece priorização entre pacotes de um mesmo fluxo. Os autores também propõem um controlador de QoS para redes OpenFlow. O controlador é responsável, de forma dinâmica, por dividir o tráfego entre diferentes *slices* de acordo com os requisitos de QoS (Vazão) de cada fluxo.

Em relação a propostas que utilizam NSIS e PCP podemos citar [Carmo et al. 2006] que sugere a substituição do uso de SBM (*Subnet Bandwidth Manager*) [Yavatkar et al. 2000] juntamente com RSVP por NSIS e PCP. Contudo a proposta não considera a utilização de redes virtuais. O provisionamento de qualidade de serviço fim-a-fim ainda é um desafio, e não existe uma solução consolidada, o que pode ser ratificado pela escassez de trabalhos correlatos apresentados nessa seção. Nenhum dos trabalhos citados propõe uma arquitetura para provimento de QoS fim-a-fim em redes virtuais, levando em consideração domínios administrativos distintos. Outro ponto que difere o RME dos outros trabalhos é a garantia de QoS baseado em fluxos de pacotes e não baseado apenas em portas de comutadores, como é feito na maioria dos trabalhos. O RME é avaliado em um testbed OpenFlow usando-se tanto métricas de QoS quanto métricas de QoE (Qualidade de Experiência).

3. Solução para provisão de QoS fim-a-fim em redes virtuais

Esta seção apresenta a arquitetura para provisão de QoS fim-a-fim e suas entidades. Neste caso: o RME *Switch*, o RME VLAN, o RME *Gateway*. Além disso, a interação desses módulos com o *framework* NSIS.

3.1. Visão Geral da Arquitetura

No cenário mostrado na Figura 1 existe um servidor em um domínio específico (Domínio 1), que transmite o tráfego para um cliente em outro domínio (Domínio 2). Em cada domínio existe um RME *Gateway* que faz a comunicação entre os domínios. Cada *gateway* se comunica com uma instância em execução do NSIS usando comunicação interprocessos através do D-Bus [D-Bus 2012]. Os dois *gateways* OpenFlow estão conectados diretamente a um *switch* Pronto 3290 (RME *Switch*) utilizado no testbed. Os *gateways* e o *switch* são gerenciados pelo controlador RME VLAN, o qual é baseado no controlador NOX [Gude 2008]. A seguir, são detalhados todos os passos necessários para a comunicação entre Clientes e Servidores dentro da arquitetura RME.

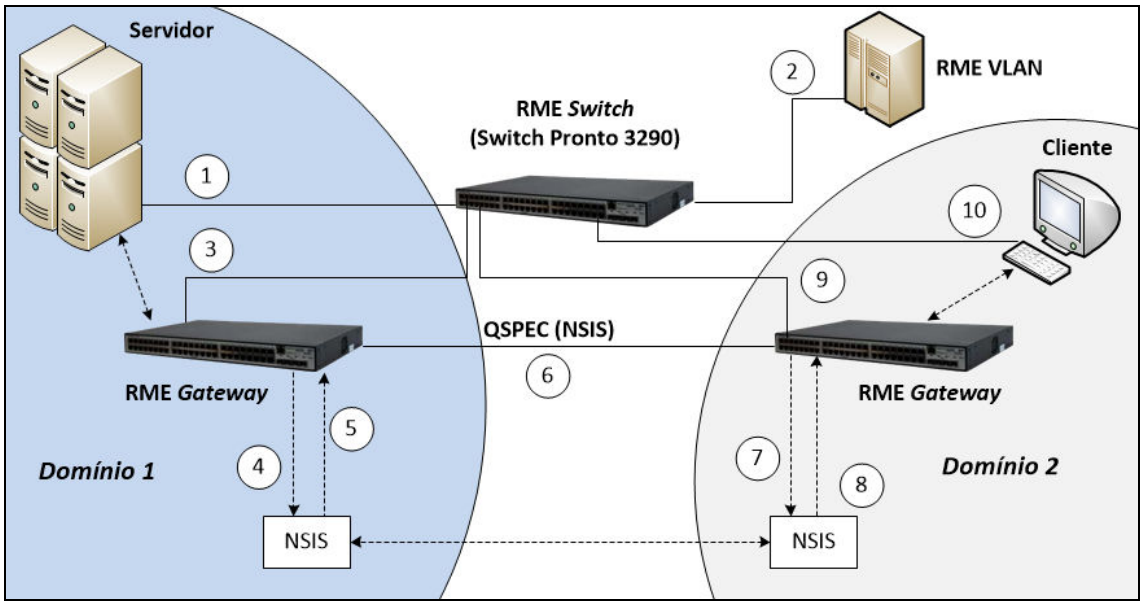


Figura 1. Visão conceitual do funcionamento do *gateway* Openflow e NSIS.

A comunicação entre servidor e cliente é realizada através dos seguintes passos:

1. O servidor envia o pacote para o cliente que é recebido pelo RME *Switch*.
2. Todo primeiro pacote de cada novo fluxo é enviado para o controlador (RME VLAN) que analisa o pacote e instala regras para esse fluxo nas tabelas de encaminhamento do RME *Switch.*
3. O pacote de entrada é então enviado para o RME *Gateway*.
4. Ao receber o pacote, o RME *Gateway* encaminha-o para o NSIS. Nesse instante o NSIS realiza o mapeamento L2/L3. Esse mapeamento consiste na conversão do

campo PCP (*Priority Code Point*) (Camada 2) para o campo DSCP (*Differentiated Services Code Point*) (camada 3) [Baker 2010].

5. Após o passo anterior, o NSIS envia o pacote mapeado para o cliente (outro domínio administrativo). Além disso, o fluxo é instalado na tabela de encaminhamento do RME *Gateway* para que todos os pacotes com a mesma característica não precisem mais ser enviado para o NSIS.
6. O Pacote passa de um domínio para o outro através dos RME *Gateways*.
7. O RME *Gateway* no domínio 2 recebe o pacote e envia para a instância NSIS de destino.
8. A instância NSIS de destino interpreta o pacote recebido e envia para o RME *Switch* de destino. Um novo fluxo OpenFlow é instalado na tabela de encaminhamento no RME *Gateway* e a reserva de recurso é feita para este fluxo.
9. O pacote é encaminhado ao cliente através do *switch* Pronto 3290 (RME-*Switch*).
10. Por fim, o usuário recebe o pacote enviado pelo servidor com QoS garantida em todo o trajeto. A partir deste momento, todo o pacote priorizado que sair do servidor para o cliente também terá garantias de QoS em ambos os domínios.

3.2. O RME *Switch*

O tráfego em uma rede é originado por uma variedade de aplicações com diferentes requisitos de desempenho, e uma forma de atender às necessidades destas aplicações consiste na utilização de tipos de tráfego pré-definidos ou Classes de Serviço (CoS – *Class of Service*) para permitir a implementação de esquemas de priorização.

O grupo de trabalho IEEE 802.1p dedicou-se ativamente entre 1995 e 1998 para desenvolver um padrão de CoS, e mesmo não existindo um padrão com esse nome publicado pelo IEEE, o termo IEEE 802.1p é utilizado como sinônimo de CoS. O esquema de priorização IEEE 802.1p utiliza um campo de 3 bits, do cabeçalho IEEE 802.1D/Q [IEEE 2004] [IEEE 2011], chamado PCP (*Priority Code Point*). O PCP especifica um valor de prioridade entre 0 e 7 (oito classes) usados para diferenciar o tráfego.

Umas das contribuições deste artigo é a implementação de uma solução de priorização de pacotes, baseada no PCP, no *firmware* do *switch* OpenFlow. Para tal, foi necessário utilizar uma distribuição de código aberto com suporte a OpenFlow e compatível com o "*hardware* de prateleira". Estes fatores motivaram a adoção da Indigo [OpenFlowhub.org 2012]. O Indigo foi criado em Stanford e o projeto disponibiliza uma plataforma de desenvolvimento chamada IODS (*Indigo Open Development System*), que permiti modificar e criar imagens do Indigo para os switches compatíveis com OpenFlow. O Indigo é baseado na implementação de referência do OpenFlow e atualmente dispõe de todas as características requeridas pela especificação OpenFlow 1.0. Entretanto, esta escolha também alguns limitações. Uma delas foi a impossibilidade da construção de um escalonador otimizado para o *hardware* adotado, pois os códigos fonte dos *drivers* não estão disponíveis, assim, optou-se por desenvolver este mecanismo na camada de *software*.

Tendo em vista a necessidade de uma solução robusta e de alta desempenho, também foi adotada uma fila de prioridade como escalonador entre os diversos fluxos. Essas filas são estruturas de dados que armazenam elementos associados às prioridades, sendo reordenadas a cada novo elemento inserido de forma que a sequência de remoção destes elementos ocorre de acordo com suas respectivas prioridades. Existem vários tipos de filas de prioridades, a mais simples são filas lineares encadeadas em que os elementos estão ordenados por prioridade decrescentes. Também são possíveis filas de prioridades fixas inserindo elementos nas filas que correspondem a uma específica prioridade. Porém, as mais robustas são baseadas em árvores. Por isso, optou-se por utilizar a PQLib [PQLib 2012], uma biblioteca que implementa uma fila de prioridades baseada em árvores escrita em C e adotada por projetos como o servidor Web Apache e o sistema de gerenciamento de bancos de dados PostgreSQL.

3.3. RME VLAN

O uso do PCP em uma rede de *testbed* exigiu a criação de VLANs (*Virtual Local Area Network*) utilizando-se o padrão IEEE 802.1Q entre os *hosts* utilizados. Este padrão define um sistema de *tagging* VLAN em *frames Ethernet*, as *tags* criam um domínio *broadcast* limitado à *tag* VLAN, ou seja, cada *tag* forma uma rede particular e separadas de outras redes.

Contudo, VLANs juntamente com alguns outros recursos da Ethernet como o *Spanning Tree* e o protocolo ARP (*Address Resolution Protocol*) ainda não são completamente suportados por todos os comutadores OpenFlow, como o Pronto 3290, e os diversos controladores, como o NOX. Por isso, tornou-se necessário o desenvolvimento de um componente (aplicação de rede) para o NOX capaz contornar algumas destas limitações da qual chamamos de RME-VLAN.

Um *switch* OpenFlow é dividido em parte de *hardware* e parte de *software*, as tabelas de encaminhamento ficam na parte de *hardware* chamada de Plano de Encaminhamento e o controlador, neste caso o NOX, se comunica com a parte de *software* do *switch* chamada de Plano de Dados através de um canal constituído pelo protocolo OpenFlow que pode ser ou não seguro (criptografado).

O NOX tem a função de manipular (controlar) a tabela de encaminhamento do *switch* OpenFlow. Todos os pacotes que chegam ao *switch* OpenFlow são analisados para determinar se pertencem a algum fluxo da tabela de encaminhamento do *switch*. Caso positivo, o pacote é enviado de acordo com a regra instalada, caso contrário, o pacote é enviado ao NOX para que este analise e instale regras específicas para aquele pacote.

3.4. RME-*Gateway*

Os dispositivos com suporte ao OpenFlow e ao NSIS atuam como *gateways* realizando o mapeamento entre domínios dos *bits* de prioridade usados na rede interna (PCP). Conforme dito anteriormente, isto é feito via NSIS QSPECs e DSCP. Indiretamente, também ocorre a reserva de recursos via RME *Gateway* através da comunicação entre o *switch* OpenFlow no espaço do usuário e o controlador OpenFlow como mostra a Figura 2.

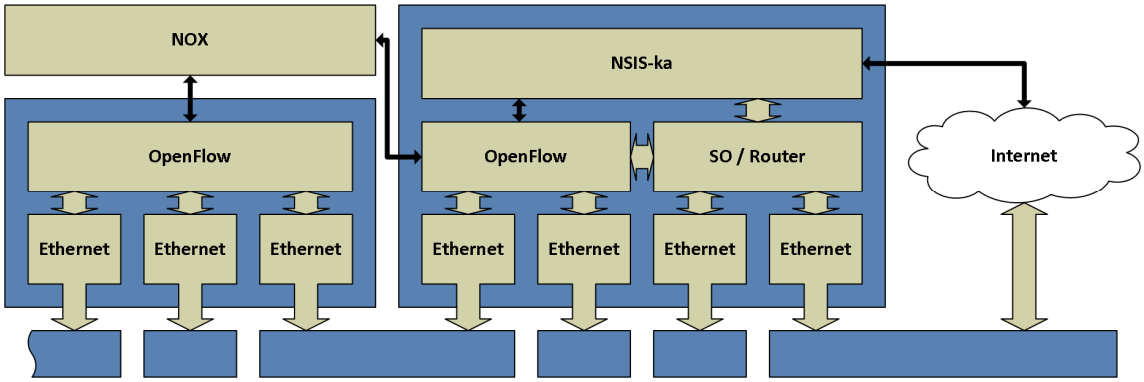


Figura 2. Reserva de recursos no NSIS-ka.

Em resumo, pode-se dizer que a principal atribuição deste módulo dentro da arquitetura proposta é mapear a QoS entre as camadas da pilha de protocolo, mantendo as respectivas atribuições dos seus componentes. O NSIS é responsável pela negociação e reserva de recursos interdomínios, e o conjunto OpenFlow/NOX pelas garantias de qualidade intradomínio. Para fazer a comunicação entre o OpenFlow e o NSIS-ka [NSIS-ka 2012] (implementação aberta do NSIS) utilizou-se o D-Bus, conforme dito na Seção 3.1. Vale ressaltar que esta proposta não se destina a modificar os protocolos de sinalização, mas apenas torná-los complementares. O *testbed* assim como os resultados obtidos é abordado na Seção 4.

4. *Testbed* e Resultados Obtidos

Esta seção aborda o *testbed* desenvolvido para validação da proposta bem como os resultados obtidos. Os *hosts* da rede foram implementados via máquinas virtuais em um *bare-metal hypervisor* para que o cenário pudesse ser heterogêneo em termos de sistemas operacionais, quantidade de interfaces de rede de um *host*, memória RAM etc. Neste caso, foi utilizado o VMware vSphere Hypervisor (ESXi) 5.0 [VMware 2012] em três servidores. Também foram utilizados, um *switch* OpenFlow (Pronto 3290) e quatro PCs, um para acesso central aos sistemas de gerenciamento da rede, e os demais como servidores de streaming e *upload* de dados.

Foram criados dois *testbeds*. Para verificamos e validarmos a implementação do PCP na plataforma Indigo, o primeiro *testbed* foi realizado com um servidor de *upload* e dezesseis clientes, todos diretamente conectados através do *RME-Switch* (Pronto 3290). O segundo testbed foi realizado com o RME-*Switch* e o RME-*Gateway* conforme mostrado anteriormente na Seção 3.4. O RME-*Switch* foi usado no segundo *testbed* como *switch* central para que houvesse um tráfego ainda maior que o do primeiro *testbed* percorrendo este. Com isso, pudemos verificar a escalabilidade das nossas soluções.

Para termos uma visão da qualidade do ponto de vista da administração da rede e do usuário, esta seção é dividida em duas partes: Uma com os resultados obtidos a partir da avaliação de QoS (Seção 4.1) e a segunda com os resultados da avaliação de QoE (Seção 4.2) conforme a seguir.

4.1 Resultados da Avaliação de QoS

Para testarmos a priorização do tráfego e NSIS provendo QoS interdomínios, foram criados clientes com acesso pelo *switch* Pronto a um servidor de *upload*. Cada cliente envia para este servidor, simultaneamente, um tráfego UDP priorizado e um não priorizado através da ferramenta Iperf [Iperf 2012]. Neste caso, foram realizados três testes:

- **Cenário 1:** Tráfego priorizado total de 32 Mbps e não priorizado de 992 Mbps.
- **Cenário 2:** Tráfego priorizado total de 64 Mbps e não priorizado de 960 Mbps.
- **Cenário 3:** Tráfego priorizado total de 128 Mbps e não priorizado de 896 Mbps.

As relações entre o tráfego priorizado e não priorizado são de 1:31 (Cenário 1), 1:15 (Cenário 2) e 1:7 (Cenário 3), respectivamente. O Iperf também foi utilizado para a medição da perda de pacotes. Em todos os casos o tráfego priorizado não sofreu perda ao contrário do tráfego não priorizado que sofreu uma perda média de pacotes de 4,77% no Cenário 1, de 6,44% no Cenário 2 e de 16,47% no Cenário 3 conforme mostram as Figuras 3 e 4. O comportamento é esperado considerando que a vazão total não atinge o limiar teórico de 1 Gbps por enlace e quanto maior o tráfego priorizado maior tende a ser o descarte de pacotes não priorizados.

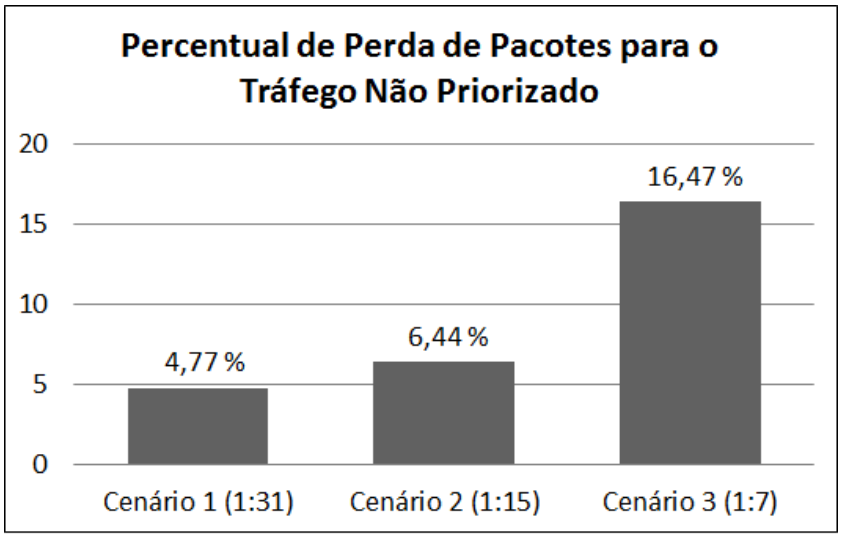


Figura 3. Percentual de Perda de Pacotes versus proporção entre o tráfego priorizado e o não priorizado.

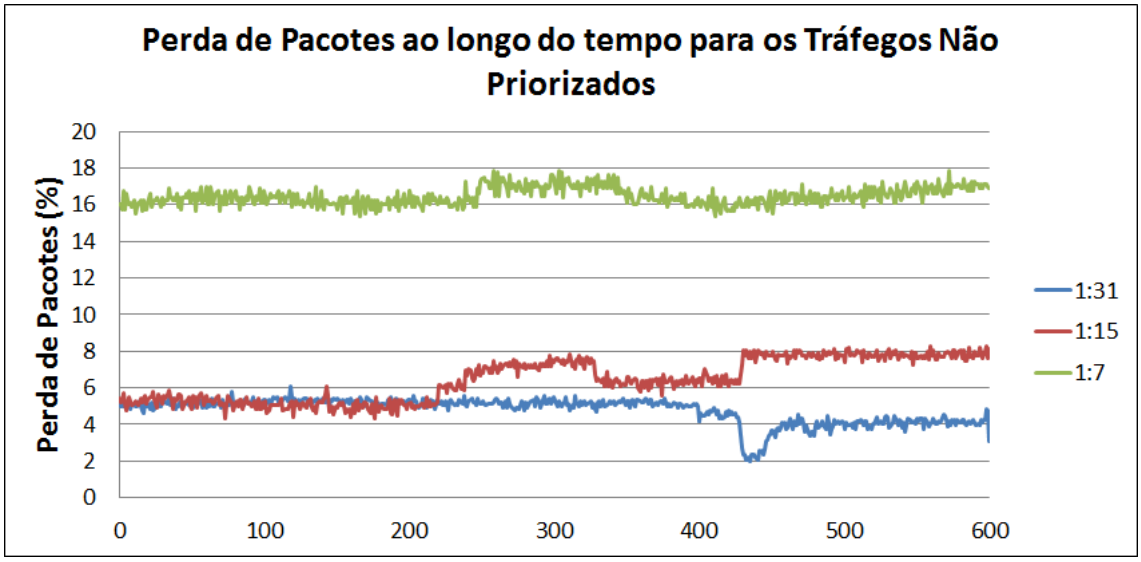


Figura 4. Percentuais de Perda de Pacotes ao longo do tempo proporção para tráfego não priorizado.

Os testes mostraram que com a implementação do *switch* no espaço do usuário permite-se que a vazão máxima fica em torno dos 30 Mbps. As médias foram de 0,804% para o tráfego priorizado e de 4,453 % para o trafego não priorizado. A avaliação da QoE do RME-*Gateway* e abordado na Seção 4.2.

4.2 Resultados da Avaliação de QoE

Os aspectos e métricas de QoS (*e.g.*:vazão, atraso e *jitter*) são importantes para análises de desempenho de protocolos e arquiteturas do ponto de vista da rede, mas não em termos de percepção humana. Desta forma, as novas arquiteturas não estão sendo mais avaliadas apenas em termos de aspectos de QoS, mas também quanto ao suporte à Qualidade de Experiência (*Quality of Experience*). O PSNR (*Peak Signal to Noise Ratio*) [WINKLER 2005] é uma métrica tradicional de QoE que estima a qualidade do vídeo em decibéis, comparando o vídeo original com o vídeo recebido pelo usuário. Para cada faixa de valores de PSNR, há uma qualificação para o vídeo que foi recebido conforme mostra a Tabela 1.

Tabela 1. Valores de classificação do PSNR.

PSNR(dB)	> 37	31 – 37	25 – 31	20 – 25	< 20
Qualidade	Excelente	Bom	Aceitável	Pobre	Péssimo

Nos servidores foi instalado o Darwin Streaming Server [DSS 2012] para fornecer serviços de *streaming* em uma rede através dos protocolos RTP e RTSP sobre UDP. Para avaliação do vídeo no cenário foi utilizado trailers do filme “Os Vingadores” [Os Vingadores (Trailer) 2012]. Todos possuem a mesma duração, o mesmo codec e o mesmo número de frames, porém com resoluções diferentes conforme mostra a Tabela 2.

Tabela 2. Vídeos utilizados nos testes.

Nome do vídeo	Resolução	FPS	Codec	*Bitrate*	Duração	*Frames*
Vídeo 1	640x360	24	H.264/AVC	488 kbps	2m e 3s	2970
Vídeo 2	1280x720	24	H.264/AVC	1732 kbps	2m e 3s	2970
Vídeo 3	1920x1080	24	H.264/AVC	3371 kbps	2m e 3s	2970

Nas medições, exatamente o mesmo vídeo era transmitido pelas redes Priorizada (PCP=5) e Não Priorizada (PCP=0). Cada medição foi repetida 10 vezes. Os vídeos além de transferidos também foram armazenados nos clientes no formato mpeg-4. Como vimos na avaliação da priorização do tráfego, a rede sem prioridade perde uma grande quantidade de pacotes. No uso de aplicações de vídeo isso implica na perda de *frames* e, consequentemente, na degradação da QoE. A Tabela 3 mostra que o vídeo recebido na rede sem prioridade perde uma grande parte das informações, isso se torna mais evidente quando a rede é mais exigida, no caso do vídeo em HD (1920x1080 *pixels*), enquanto que o vídeo na rede com prioridade tem pouco ou nenhuma perda.

Tabela 3. Vídeos utilizados nos testes.

	Original	Recebido Sem Priorização	Recebido Com Priorização
Vídeo 1 (640x360)	9,5 MB	9,5 MB	9,5MB
Vídeo 2 (1280x720)	30,2 MB	25,8 MB±0,2MB	30 MB±0,2MB
Vídeo 3 (1920x1080)	56,7 MB	30,9 MB±0,32MB	55,2MB±1,2MB

O calculo do PSNR foi feito com a ferramenta MSU VQMT [MSU 2012]. A Figura 5 mostra o PSNR dos vídeos recebidos nas redes Priorizada e Não Priorizada. Percebe-se que os valores neste caso são semelhantes, isso se deve ao fato do Vídeo 1 com resolução de 640x360 não exigir muitos recursos da rede.

A Figura 6 mostra o PSNR do Vídeo 2 (1280x720), neste caso, torna-se mais evidente o efeito da priorização, pois o vídeo com esta resolução precisa de mais recursos para trafegar pela rede. Porém, o PSNR médio ainda está em torno de 25, o que torna o vídeo aceitável segundo os valores de PSNR mostrados anteriormente (Tabela 1).

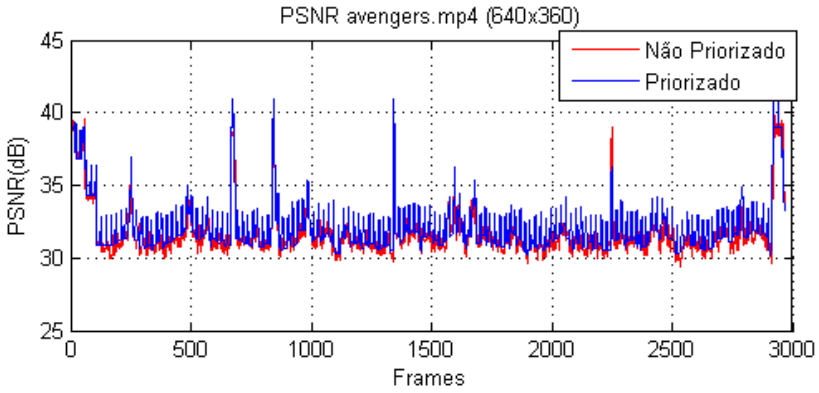


Figura 5. PSNR dos vídeos 640x360 recebidos na rede priorizada e não priorizada.

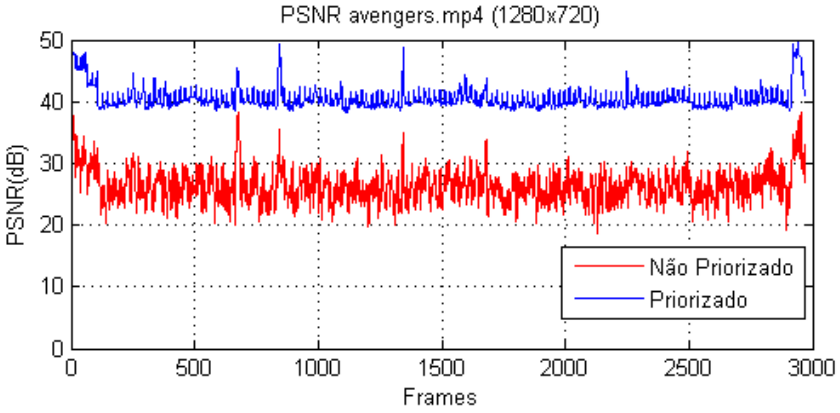


Figura 6. PSNR dos vídeos 1280x7620 recebidos na rede priorizada e não priorizada.

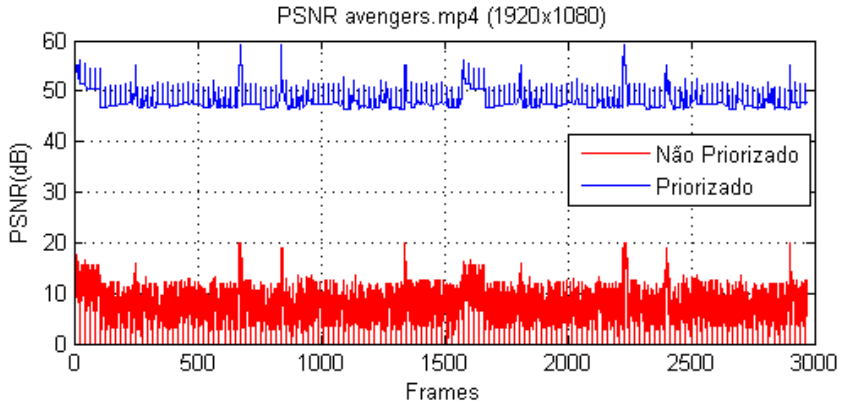


Figura 7. PSNR dos vídeos 1920x1080 recebidos na rede priorizada e não priorizada.

Com o vídeo em alta definição os efeitos da priorização são muito visíveis (Figura 7). Este vídeo exige muito recursos da rede como uma alta vazão (em média 3371 Mbps), por exemplo, e a o esquema de priorização reserva a maior parte desta para a rede priorizada tornando o vídeo não priorizado muito degradado. Mesmo assim, há perda de QoE na Rede Priorizada.

A Tabela 4 mostra os PSNR médios obtidos. Se considerarmos os PSNR médios entre as redes Priorizada e Não Priorizada para cada um dos vídeos transmitidos. Os ganhos entre estes são de 1,4%, 54% e 520% para os Vídeos 1, 2 e 3 respectivamente.

Tabela 4. PSNR médios dos vídeos recebidos pelas Redes Priorizada e Não Priorizada.

Vídeos	PSNR (Média)	
	Rede Priorizada	Rede Não Priorizada
Vídeo 1 (640x360)	32.1412±1.90	31.6751±2.71
Vídeo 2 (1280x720)	40.5352±2.70	26.3133±5.0234
Vídeo 3 (1920x1080)	47.9077±1.78	7.7237±3.65

A perda ou duplicação dos pacotes na rede ocasionam queda da QoE oferecida ao usuário. A figura abaixo mostra um exemplo de *frame* do filme em alta definição. Percebe-se que este *frame* na Rede Priorizada não difere do *frame* original, porém a perda de pacotes faz com que o vídeo recebido na Rede Não Priorizada seja um *frame* degradado.

Figura 11. Comparação do 1412° frame entre o vídeo original e os recebido nas redes priorizada e não priorizada.

5. Conclusão

O surgimento da virtualização de redes e do conceito de redes definidas por software implementado pela plataforma OpenFlow trouxe grande inovação para as redes tradicionais. Porém, por serem um conceito em desenvolvimento, novas soluções precisam ser propostas visando esses novos conceitos. Neste contexto, garantir qualidade de serviço fim-a-fim é um dos desafios. Este trabalho apresentou uma proposta de solução para a provisão de QoS fim-a-fim na plataforma OpenFlow através do PCP e do *framework* NSIS. A solução proposta tem o objetivo de garantir QoS em redes com tecnologia OpenFlow, além de permitir a extensão dessa garantia em domínios administrativos distintos. As decisões de projeto foram tomadas tendo em mente a adoção de conceitos previamente estabelecidos como o modelo de QoS do IEEE 802.1p e a sinalização do NSIS.

A proposta foi avaliada em um testbed OpenFlow utilizando diversas ferramentas e dispositivos usados nas empresas e universidade, o que coloca a proposta é um cenário bem realista. Na avaliação, mostrou-se que a proposta possui um ganho significativo tanto considerando avaliação de métricas de QoS, que refletem o comportamento da rede, quando avaliação de métricas subjetivas de QoE que estão mais próximas aos critérios dos usuários. Como trabalhos futuros, pretendemos utilizar o IPv6 e o mapeamento e negociação de QoS não apenas nas redes Ethernet, mas também em cenários heterogêneos com tecnologias como por exemplo, o IEEE 802.11 e IEEE 802.16.

6. Referências

Gozdecki, J. et al. (2003) "Quality of service terminology in IP networks," Communications Magazine, IEEE , vol.41, no.3, pp.153,159, March.

OpenFlow Switch Specification - Version 1.0.0. (2012) http://www.OpenFlow.org/documents/OpenFlow-spec-v1.0.0.pdf, Março.

Schonwalder, J. et al. (2009) "Future Internet = content + services + management," Communications Magazine, IEEE , vol.47, no.7, pp.27,33, July.

IEEE (2011) "IEEE Standard for Local and metropolitan area networks--Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks". IEEE Standard 802.1Q-2011.

Fu, X. et al. (2005) "NSIS: a new extensible ip signaling protocol suite" Communications Magazine, IEEE, v. 43, n. 10, p. 133–141.

Civanlar, S. et al. (2010) "A QoS-Enabled OpenFlow Environment for Scalable Video Streaming", IEEE Globecom 2010 Workshop on Network of the Future (FutureNet-III), Miami, USA.

Gude, N. et al. (2008). "NOX: towards an operating system for networks" In: SIGCOMM Comput. Commun. Rev. 38, 3, July, p. 105-110.

Egilmez, H.E. et al. (2011) "Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing," Image Processing (ICIP), 2011 18th IEEE International Conference on , vol., no., pp.2241,2244, 11-14 September.

LEMON (2012) "Library for Efficient Modeling and Optimization in Networks." http://lemon.cs.elte.hu, March.

Min, S.H. et al. (2010) "Implementation of a Programmable Service Composition Network using NetFPGA-based OpenFlow Switches", 1st ASIA NetFPGA Developer's workshop, Korea, June.

Egilmez, H.E. et al. (2012) "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks", *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific* , vol., no., pp.1,8, 3-6 December.

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, pp. 536-549, Ouro Preto, MG, Brazil, May.

Zhang, Y et al. (2009) "A QoS-Oriented Network Architecture Based on Virtualization," Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on , vol.2, no., pp.959-963, 7-8 March.

Bozakov, Z. (2011) "Architecture and algorithms for virtual routers as a service," Quality of Service (IWQoS), 2011 IEEE 19th International Workshop on , vol., no., pp.1-3, 6-7 June. Civanlar, S. et al. (2010) "A QoS-Enabled OpenFlow Environment for Scalable Video Streaming", IEEE Globecom 2010 Workshop on Network of the Future (FutureNet-III), Miami, USA.

Puschita, E. et al. (2010) "An Innovative QoS Paradigm Based on Cognitive In-network Management of Resources for a Future Unified Network Architecture: I-NAME QoS Model," Advances in Future Internet (AFIN), 2010 Second International Conference on , vol., no., pp.37-43, 18-25 July.

Kassler, A. et al. (2012) "Towards QoE-driven Multimedia Service Negotiation and Path Optimization with Software Defined Networking" International Conference on Software, Telecommunications and Computer Networks (SoftCOM), September.

Kim, W. et al. (2010) "Automated and Scalable QoS Control for Network Convergence," Proceedings of the INM/WREN'10.

Zhang, H. and Ferrari, D. (1993) "Rate-controlled static-priority queueing" In Proceedings of the IEEE INFOCOM.

Carmo, M. et al. (2006) NSIS-based quality of service and resource allocation in Ethernet networks. In: Braun T et al. (ed) WWIC 2006, LNCS 3970. Springer, Berlin, pp 132–142.

Yavatkar, R. et al (2000) "SBM (subnet bandwidth manager): A protocol for RSVP-based admission control over IEEE 802-style networks," May, internet RFC 2814.

D-Bus (2012) http://www.freedesktop.org/wiki/Software/dbus, Março.

Baker, F.; Polk, J.; Dolly, M. (2010) A Differentiated Services Code Point (DSCP) Capacity-Admitted Traffic. IETF RFC 5865.

IEEE (2004) "IEEE Standards for Local and Metropolitan Area Networks: Media access control (MAC) bridges" IEEE Standard 802.1D-2004.

OpenFlowhub.org (2012) Indigo, http://www.OpenFlowhub.org/display/Indigo/Indigo+-+Open+Source+OpenFlow+Switches, Março.

PQLib (2012) "Priority Queue Library", http://www.ohloh.net/p/pqlib, Março.

NSIS-ka (2012) http://nsis-ka.org/, Março.

VMware (2012) "VMware vSphere Hypervisor (ESXi) 5.0, http://www.vmware.com/br/products/datacenter-virtualization/vsphere/mid-size-and-enterprise-business/overview.html, Março.

Iperf (2012) http://sourceforge.net/projects/iperf/, Março.

Winkler, S. (2005) Perceptual video quality metrics – a review, in Digital Video Image Quality and Perceptual Coding, cap. 5, CRC Press.

DSS (2012) "Darwin Streaming Server" http://dss.macosforge.org/, Março.

Os Vingadores (Trailer) (2012) Direção: Joss Whedon. [S.l.]: Walt Disney Home Entertainment, 2012. 1 DVD (2m e 30 s) NTSC, color. Título original: The Avengers.

MSU (2012) "MSU Video Quality Measurement Tool" http://compression.ru/video/quality_measure/video_measurement_tool_en.html, Março.

ProViNet: Uma Plataforma para Gerenciamento de Redes Virtuais Programáveis

Wanderson Paim de Jesus[1], **Ricardo Luis dos Santos**[1], **Oscar Maurício Caicedo Rendón**[1]
Lisandro Zambenedetti Granville[1]

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

{wpjesus, rlsantos, omrendom, granville}@inf.ufrgs.br

Abstract. *With the evolvement of virtualization and network programming techniques, high-level applications can be used to define the behavior of network traffic while keeping isolation. However, to ensure a harmonious relationship between users, network applications and virtual networks, considerable management efforts are needed. In this paper we propose the ProViNet platform, a solution for managing the deployment of network applications in Programmable Virtual Networks (PVN). In addition to the management facilities, ProViNet contributes with an architecture that allows sharing the control plane of PVN in a scalable way. During the development of this work we identified the need for a standard representation of programmable virtual infrastructures, so it is also proposed an extension of the programmable virtual networks description language VXDL. In order to verify the feasibility of the proposed platform, we implemented a prototype, which is analyzed and evaluated in this paper.*

Resumo. *Ao passo que evoluem as técnicas de virtualização e programação de redes, aplicativos de alto nível podem ser utilizados para definir o comportamento de tráfegos de rede isoladamente. Entretanto, garantir um relacionamento harmônico entre usuários, aplicativos de rede e redes virtuais exige grandes esforços de gerenciamento. Neste trabalho propomos a plataforma ProViNet, uma solução para o gerenciamento da implantação de aplicativos de rede em Redes Virtuais Programáveis (RVP). Além de agregar facilidades no gerenciamento, ProViNet contribui com uma arquitetura que permite o compartilhamento do plano de controle de RVP de forma escalável. Durante o desenvolvimento do trabalho foi identificada a necessidade de uma representação padrão da infraestrutura virtual programável, para tanto propõe-se também, uma extensão da linguagem de definição de redes virtuais VXDL. A fim de verificar a viabilidade da plataforma proposta, foi implementado um protótipo, o qual é analisado e avaliado neste trabalho.*

1. Introdução

Historicamente, o núcleo das redes de computadores, quando comparado com os servidores, *desktops* e dispositivos móveis da borda das redes, é um ambiente hostil à inovação. No contexto específico da Internet, esse fato é geralmente referenciado como ossificação [Hausheer *et al.* 2011]. Para exemplificar, soluções propostas a mais de dez anos, como IPv6 e IPSec, ainda não estão amplamente em uso. São apontadas como possíveis causas: *(i)* a necessidade de modificações globais, ocasionalmente exigindo a substituição de

equipamentos; *(ii)* a lentidão no processo de padronização que trata da interoperabilidade com serviços legados; *(iii)* e a abordagem adotada pelas fabricantes de equipamentos, de implementar e implantar soluções baseadas em seu retorno financeiro.

Na intersecção entre o conceito de Virtualização de Redes [Chowdhury e Boutaba 2010] e o de Programabilidade de Redes [Kanaumi *et al.* 2010], emergem as Redes Virtuais Programáveis (RVP), as quais promissoramente prometem reverter o cenário de lentidão testemunhado nas redes de computadores. Uma das abordagens adotadas pelas RVP segue o formato das Redes Definidas por *Software* (SDN - *Software-Defined Networks*) [Lantz *et al.* 2010], que define o desacoplamento dos planos de controle e de dados. Algumas implementações do plano de controle se baseiam na Arquitetura Orientada a Serviços (SOA - *Service Oriented Architecture*) para prover a comunicação com aplicativos de rede. Dessa forma, utilizando uma interface padronizada de definição de serviços, os aplicativos de rede podem ser programados em linguagens distintas, se tornam menos dependentes da tecnologia utilizada no plano de controle.

Por um lado, o desacoplamento entre aplicativos de rede, plano de controle e infraestrutura programável torna a arquitetura SDN flexível e escalável [Gutz *et al.* 2012]. Por outro lado, induz uma grande complexidade no gerenciamento. Modelos de gerenciamento utilizados nas redes comuns não são adequados às redes programáveis, uma vez que não tratam da implantação dinâmica de novos serviços. Além disso, dependendo das políticas de acesso às infraestruturas programáveis, um grande número de usuários poderão propor e implantar seus próprios aplicativos de rede. Nesse cenário, harmonizar os aplicativos, usuários e redes virtuais, mantendo a confiabilidade e escalabilidade dos serviços implantados é um problema em aberto.

As propostas para o gerenciamento de RVP variam de acordo com o ambiente de implantação e com os requisitos dos usuários. Nos ambientes de *testbeds*, as propostas focam em prover aos experimentadores e cientistas soluções para o controle do provimento de *Slices*. Entretanto, o gerenciamento da programabilidade que os usuários possuem sobre os *Slices* ainda é incipiente. São exemplos desse tipo de proposta o ProtoGENI [ProtoGENI 2012] e o OFELIA *Control Framework* [Kopsel 2011]. Nos ambientes de *Cloud*, mais direcionados ao mercado, implementam-se soluções para gerenciar os serviços que os provedores de *Cloud* oferecerão aos seus clientes. Ou seja, o foco maior está em prover controle aos gerentes e administradores da *Cloud*, deixando o usuário final sem chance de propor novos aplicativos de rede. As propostas da CITRIX, XenServer *Distributed vSwitch Controller* e da CISCO, OnePK, são exemplos.

O problema de pesquisa deste trabalho está em como prover acesso de múltiplos usuários finais a uma infraestrutura de RVP, agregando facilidades no gerenciamento e implantação de aplicativos de forma a encorajar o desenvolvimento de novas soluções de rede. Propõe-se para isso a plataforma de gerenciamento ProViNet (***Pro****grammable* ***Vi****rtual* ***Net****work Managemet Platform*). ProViNet contribui para o estado-da-arte em quatro pontos: *(i)* na elaboração de uma arquitetura para gerenciamento de RVP que provê escalabilidade e alta disponibilidade de serviços; *(ii)* em uma abordagem para implantação dinâmica de novos serviços no plano de controle; *(iii)* na extensão da linguagem de descrição de rede virtual programável, *Virtual Resources and Interconnection Networks Description Language* (VXDL) [Koslovski *et al.* 2008]; *(iv)* e por fim, no de-

senvolvimento, como parte da plataforma, de um sistema com interface de acesso Web que facilita a compreensão e interação dos usuários finais com ambientes de RVP.

O restante do artigo está organizado conforme segue. Na Seção 2, são descritos os principais trabalhos que envolvem o gerenciamento de RVP. Na Seção 3 é apresentada a plataforma ProViNet, discutindo a arquitetura conceitual e conceitos empregados. Em seguida, na Seção 4 é detalhado o protótipo utilizado como base para a avaliação e análise apresentada na Seção 5. Por fim, a Seção 6 conclui o artigo com as considerações finais e perspectivas para trabalhos futuros.

2. Trabalhos Relacionados e Contextualização

Os conceitos de Virtualização de Redes [Chowdhury e Boutaba 2009] [Chowdhury e Boutaba 2010] e a Programabilidade de Redes [Campbell *et al.* 1999] [Lin *et al.* 2011] são bem discutidos na literatura. A junção desses conceitos formam as Redes Virtuais Programáveis (RVP), as quais são mais comumente aplicadas em dois ambientes, nos projetos de plataformas de testes, também conhecidos como *testbeds* e em Nuvens privadas (*Private Clouds*). Os trabalhos relacionados apresentados neste capítulo são organizados conforme esses dois ambientes, buscando evidenciar em cada um deles o nível de abstração da infraestrutura virtual, a abordagem utilizada e a natureza da licença.

Dentre as propostas no contexto dos *testbeds*, as quais focam em auxiliar os pesquisadores em seus experimentos, destaca-se o *framework* de controle OFELIA (OCF) [Kopsel 2011], o qual é uma derivação da plataforma Expedient proposta pela Universidade de Stanford. Esse *framework* auxilia os pesquisadores na criação de *Slices* utilizando recursos de várias federações. Além disso, lhes oferece também a capacidade de associar esses *Slices* a controladores previamente configurados. Apesar dessa funcionalidade de associação através da interface gráfica, o OCF não provê o gerenciamento da implantação de aplicativos de rede, considerada uma das maiores preocupações da virtualização de redes [Chowdhury e Boutaba 2010].

Outra proposta, ainda no contexto dos *testbeds*, foi criada no projeto GENI [GENI 2011] e é chamada ProtoGENI [ProtoGENI 2012]. Tal proposta também implementa uma interface de acesso via Web que assiste aos pesquisadores na criação de *Slices* com recursos oriundos de diversas federações. Ao interagir com a interface do ProtoGENI, pesquisadores podem instanciar nós virtuais e conecta-los dinamicamente. Assim como a maioria das propostas desenvolvidas nesse mesmo contexto, o foco está no provimento do *Slice* e não na programabilidade do mesmo. Em geral, essas soluções são livres de licença, entretanto existe um conjunto burocrático de regras para utilização e acesso aos recursos geridos pelas ferramentas citadas.

As soluções no contexto de *Cloud Computing* se diferem das propostas de *testbeds*, principalmente pelo foco comercial. Os ambientes de *Cloud* em geral demandam uma grade quantidade de recursos de rede, pois comercializam serviços com alta taxa de disponibilidade e qualidade de serviço. Portanto, a criação de serviços de rede customizados para atender a demandas especiais é visto como um grande atrativo para os provedores de *Cloud*. Para atender esses provedores, surgiram soluções que aumentam o poder de customização dos serviços providos nas redes virtuais de seus *datacenters*. Algumas dessas soluções são comercializadas, tais como Nexus 1000v e OnePK da CISCO e DVSC (*Distributed Virtual Switch Controller*) da CITRIX. Outras são de código aberto,

como os *plugins open-vSwitch*, Ryu Plugin e o *restproxy*, para a plataforma OpenStack [OpenStack 2011].

Tanto em *Cloud* quanto em *testbeds*, é crescente o número de propostas que empregam os conceitos da arquitetura SDN em suas soluções de programabilidade em redes virtuais. Rubio *et al.* [Rubio-Loyola *et al.* 2011], por exemplo, propôs um plano de orquestração, o qual é complementar aos planos originalmente definidos na arquitetura SDN (controle e dados). O propósito desse novo plano é controlar dinamicamente o comportamento das redes virtuais em resposta às constantes variações, gerando assim um sistema de gerenciamento autonômico. Embora esse sistema tenha vantagens, como resposta rápida às falhas geradas por variações no comportamento da rede, requer soluções padronizadas e bem testadas. Em consequência, o usuário final continua distante da criação e implantação de aplicativos na rede virtual programável. Tal fato, contribui para baixa taxa de inovação nas redes, pois mantem a natureza restritiva e a pequena quantidade de pessoas aptas a desenvolver e implantar novas soluções.

Em resumo, apesar de existirem propostas recentes envolvendo redes virtuais programáveis, nenhuma das pesquisadas promove o gerenciamento da programabilidade em uma infraestrutura de RVP, considerando o acesso de múltiplos usuários finais. Além disso, maior parte das propostas analisadas se limitam ao provisionamento da rede virtual (controle de máquinas virtuais e configuração da rede virtual), não oferecendo funcionalidades para o gerenciamento dos aplicativos de rede que podem ser instalados dinamicamente nas redes virtuais programáveis.

3. ProViNet

Uma das abordagens para as Redes Virtuais Programáveis é a utilização da arquitetura de Redes Definidas por *Software*. Tal arquitetura define que os planos de controle e de dados sejam desacoplados. Sendo assim, eles necessitam de um protocolo para comunicação. Recentemente tem se empregado o termo *Southbound API* (SBAPI) para se referir aos protocolos que provejam essa comunicação entre o plano de controle e de dados. Conforme ilustrado na Figura 1 a SBAPI **(2)** é utilizada para comunicação entre o *Pool* de Controle e os *Slices*, que serão descritos mais diante.

Ao passo que surgiram diversas soluções para a camada de controle, e cada implementação adota um padrão de linguagem, os aplicativos de rede criados seguiam tal heterogeneidade, ficando assim dependentes das tecnologias dos controladores e isoladas entre si. Atualmente a tendência é que as diferentes implementações do plano de controle ofereçam uma interface padrão de comunicação com aplicativos de rede externos. É comum se referir a esse tipo de interface como *Northbound API* (NBAPI) **(1)**. Em geral, elas independem de linguagem, baseando-se na arquitetura *Representational State Transfer* (REST), por exemplo. Desse modo, novos aplicativos que venham a interagir com a camada de controle necessitam apenas das especificações de serviços providas por cada implementação de controlador.

Conforme ilustrado na Figura 1, a plataforma ProViNet auxilia os usuários finais no gerenciamento e implantação de aplicativos de rede em *Slices* de Redes Virtuais Programáveis. Para isso, se apoia no conceito de separação de planos, sejam eles: o plano de dados, representado pelos elementos que formam a rede virtual nos *Slices* e utilizam a SBAPI para se comunicarem com o plano de controle; o plano de controle, formado

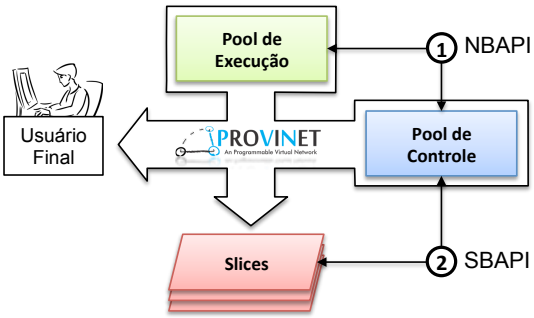


Figura 1. Módulos e relacionamentos

pelos controladores, que são agrupados no *Pool* de Controle para prover alta disponibilidade, conforme será apresentado na Subseção 3.1; e pelo plano de aplicativos, presente no *Pool* de Execução, o qual organiza, armazena e executa os aplicativos de rede que se aproveitam do conceito de NBAPI para comunicação com o plano de controle.

A plataforma ProViNet se aplica a qualquer ambiente que utilize infraestrutura compatível com o conceito de RVP. Em geral, a aplicação se dá em dois níveis, um no nível de rede virtual, no qual o plano de dados seriam representado pelos *vSwitches*, e outro no nível físico, utilizando os *switches* físicos compatíveis como plano de dados. Ou até mesmo em um modelo híbrido, com controle em ambos os níveis.

3.1. Arquitetura Conceitual

A arquitetura apresentada na Figura 2 ilustra os componentes da plataforma ProViNet assim como suas relações em alto nível. O usuário interage com a plataforma através de uma interface Web que expõe graficamente as funcionalidades providas por seus módulos. Conforme ilustrado, essas funcionalidades se subdividem em quatro interações. Na interação **(a)**, o módulo Controle de Usuários atende a requisições de autenticação e cadastro. O processo de gerenciamento de implantação e execução de aplicativos, executado na interação **(b)**, é tratado pelo módulo Controle de Aplicativos. As solicitações de infraestrutura virtual são enviadas na interação **(c)** e tratadas pelo módulo de Controle de *Slices* e Referências. Por fim, as configurações inerentes aos três módulos citados são apresentadas em uma interface de administração **(d)** para o Administrador do ambiente de implantação do ProViNet.

ProViNet fornece escalabilidade utilizando uma arquitetura baseada no conceito chamado de *Resource Pool*. Cada *Pool* representa um conjunto de servidores de virtualização (*hypervisors*) interligados e controlados por uma plataforma de gerenciamento única. O *Pool* de Controle é utilizado para execução de máquinas virtuais com sistemas idênticos, que executam uma implementação de controlador pré-definida e compatível com o conceito de NBAPI. Os aplicativos escritos pelos usuários são executados em máquinas virtuais individuais alocadas no *Pool* de Execução. A comunicação entre os controladores do *Pool* de Controle e as VMs do *Pool* de Execução é provida pela NBAPI. Já a comunicação entre os controladores e os elementos do plano de dados nos *Slices* é provida pela SBAPI. As requisições realizadas pelos módulos do ProViNet aos *Pools* utilizam as interfaces de comunicação providas pela plataforma de virtualização adotada. Finalmente, a requisição ao Provedor de Infraestrutura Virtual ocorre por meio de uma requisição HTTP, enviando um documento de descrição de infraestrutura virtual, a ser

comentado mais adiante.

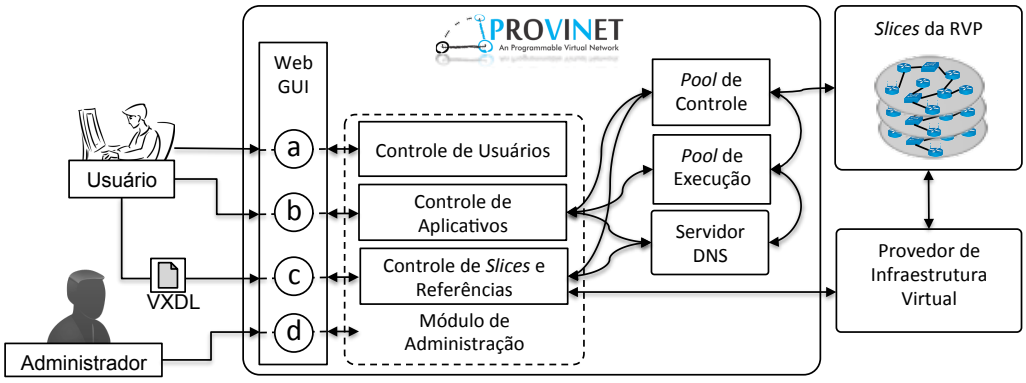


Figura 2. Arquitetura conceitual

Analisando novamente a Figura 2, percebe-se a esquerda o Usuário Final e a direita a Rede Virtual Programável. Entre esses elementos está a plataforma ProViNet, provendo a ligação entre eles. Nessa posição, o ProViNet oferece o gerenciamento da implantação de aplicativos, o controle de acesso para ambientes com múltiplos usuários e os requisitos não funcionais de disponibilidade, confiabilidade e escalabilidade. Com exceção do módulo de Controle de Usuários e de Administração, devido a simplicidade, os outros são detalhados nas subseções seguintes.

3.2. Controle de *Slices* e Referências

Uma Rede Virtual Programável (RVP) dispõe de recursos computacionais e de rede necessários para a criação de redes isoladas e programáveis. Para isso, se apoiam em tecnologias de virtualização (tais como XenServer, VMWare e Virtualbox) e de programabilidade (como o OpenFlow, empregado neste trabalho). Seja qual for a tecnologia e a abordagem de provimento da RVP, o resultado é que o usuário terá uma rede virtual programável que interliga exclusivamente suas máquinas virtuais. Por compartilharem os recursos do provedor (ou *datacenter*), é conveniente e usual se referir ao subconjunto de recursos de rede, computacionais e de armazenamento pertencentes a um usuário, de *Slice*.

Não é função da plataforma ProViNet o provimento do *Slice*, ou seja, inicializar máquinas virtuais e configurar a rede virtual que interliga as mesmas. Portanto, o módulo Controle de *Slices* e Referências tem, entre outras, a função de receber e encaminhar um documento de descrição de infraestrutura virtual especificando a topologia e os recursos a serem alocados pelo Provedor de Infraestrutura Virtual (PIV). Existem diversas propostas para esse tipo de documento, tais como Rspec (GENI), NDL-OWL(RENCI), NMC (OGF) e *Virtual Resources and Interconnection Networks Description Language* (VXDL) do projeto INRIA [Koslovski *et al.* 2008]. Neste trabalho propomos uma extensão para a linguagem VXDL, tornando-a compatível com arquiteturas de programabilidade de redes em que haja separação de planos (de controle e de dados).

Originalmente, a linguagem VXDL, apresentada no trabalho de Koslovski *et.al.* [Koslovski *et al.* 2008], define que os recursos possuem um nome e podem ter uma lista de funções, parâmetros, softwares e uma localização conforme se nota no trecho destacado de sua definição e apresentado abaixo:

```
<resource> ::= "resource" "(" <name> ")" "{"
               ["function" <elementary-functions>]
               ["parameters" <resource-parameters>]
               ["software" <software-list>]
               ["anchor" <location>] "}"
```

O atributo *function* pode assumir diversos valores, tais como *endpoints*, *aquisition* e *router*. Propõe-se a adição de um novo atributo inerente ao valor *router*, o qual nomeia-se <*controller-list*>. Esse valor representa uma lista de controladores. Foi adotado uma lista pois alguns roteadores ou *switches* compatíveis com SDN aceitam redundância de controladores. O *switch* virtual Open vSwitch por exemplo aceita a configuração de um *master* e diversos *slaves*. Cada elemento <*controller*> é composto por atributos que definem o tipo de conexão (ftp, ptcp e ssl são exemplos), o endereço IP e a porta em que o controlador remoto está configurado.

```
<elementary-functions> ::= <function> ("," <function> )*
<function> ::= "endpoint" | "aquisition" | "storage"
    | "computing" | "visualization" | "network_sensor"
    | "router" "(" "ports" <ports> [<controller-list>] ")"
<ports> ::= <number>
<controller-list> ::= <controller> ("," <controller> )*
<controller> ::= "(" <connection-type> <ip-address> <port> ")"
```

O usuário inicialmente deve elaborar este arquivo sem as informações sobre o controlador, pois estas serão adicionadas automaticamente pelo ProViNet durante o processo de requisição de infraestrutura virtual. Isso se justifica pelo fato do usuário não ter informações sobre os endereços de IP dos controladores, até mesmo porque eles são dinâmicos, conforme será apresentado adiante.

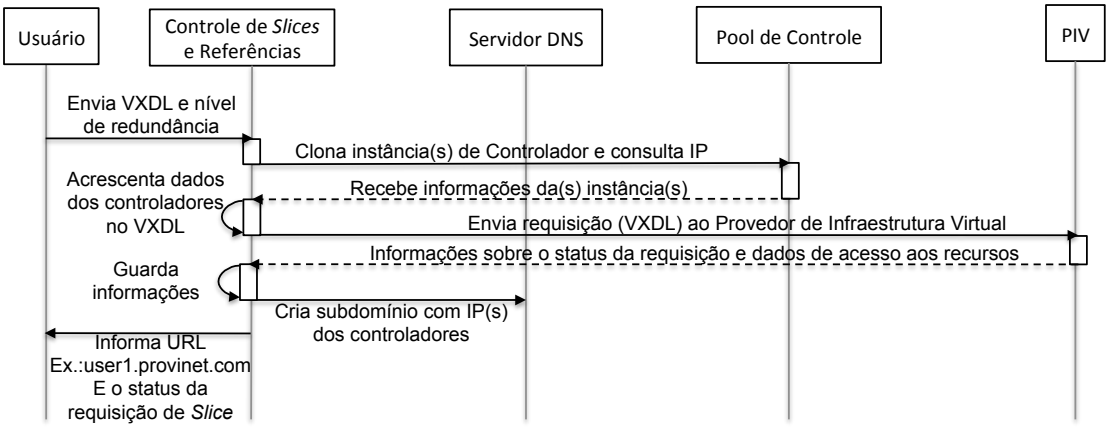


Figura 3. Diagrama de sequência da abordagem de criação de *Slices*

De acordo com o diagrama da Figura 3, quando o usuário já elaborou o documento VXDL, ele o envia ao módulo Controle de *Slices* e Referências via formulário de *upload*. No mesmo formulário, o usuário deve definir um nível de redundância de controladores no *Pool* de Controle que pretende ser associado ao *Slice* requerido. Após a instanciação dos controladores, o *Pool* de Controle reponde a requisição informando o IP dos mesmos. Esses IPs são adicionados ao VXDL juntamente com o tipo de conexão e porta. Em se-

guida, o documento é enviado ao PIV por uma requisição HTTP, gerada automaticamente pela plataforma.

Ao receber a resposta sobre o status da requisição e com informações sobre o acesso aos recursos do *Slice*, a plataforma registra no Servidor DNS os IPs dos controladores (os mesmos adicionados no documento VXDL). Esse registro é a adição do nome do usuário como subdomínio do domínio *www.provinet.local*. Ou seja, cada usuário tem um subdomínio no qual são associados os IPs dos controladores por ele requeridos. Caso o usuário tenha definido grau e redundância maior que um, o Servidor DNS aplicará o algoritmo Round Robin entre as referências. Abaixo segue um exemplo de configuração de uma domínio em que o "user1" possui dois controladores e o "user2" tem um.

```
...
                NS      servidor.provinet.local.
servidor        A       xxx.xxx.xxx.xxx
user1           A       <IP-controlador-1>
user1           A       <IP-controlador-2>
user2           A       <IP-controlador-3>
...
```

Por fim, o módulo apresenta ao usuário a URL a ser utilizada em seus aplicativos para fazer chamadas aos serviços (Ex.: http://user1.provinet.local). São fornecidos, também, os dados de acesso aos recursos virtuais recebidos do PIV. Dessa forma, o aplicativo do usuário poderá enviar requisições aos serviços providos pelos controladores instanciados no *Pool* de Controle.

3.3. Controle de Aplicativos

Uma vez que a estrutura de programabilidade já está estabelecida, ou seja, o usuário possui um *Slice* e um plano de controle devidamente configurado, resta ao usuário desenvolver e executar os aplicativos de rede. Para que o desenvolvimento seja possível, o usuário precisa saber quais são os serviços a sua disposição. Por isso, é disponibilizado através da interface Web do ProViNet, uma documentação completa sobre tais serviços, os quais dependem da tecnologia utilizada no *Pool* de Controle.

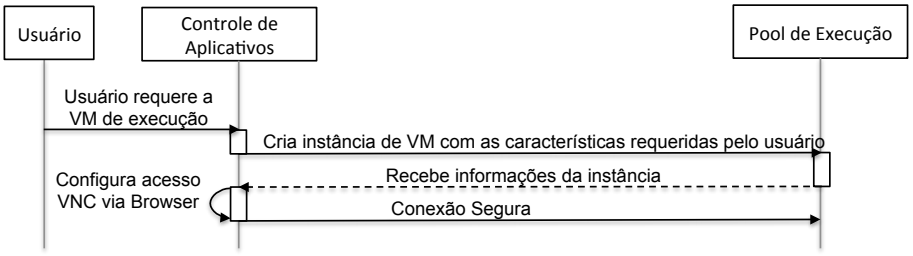


Figura 4. Abordagem de referências para provimento de escalabilidade e disponibilidade

De acordo com o diagrama apesentado na Figura 4, no primeiro passo o usuário acessa a plataforma e requer a VM que rodará seus aplicativos. Nesta requisição são apresentados perfis de VM, variando o Sistema Operacional e quantidade de recursos, tais como memória, processamento e armazenamento. Após a escolha de um perfil, a plataforma, por meio de uma interface de comunicação com o *Pool* de Execução requer uma VM com tais características. Para evitar qualquer tipo de bloqueio por *firewall*, ao

receber os dados da VM criada, o módulo Controle de Aplicativos configura uma interface de acesso remoto via *Browser*. Ou seja, o usuário é capaz de visualizar e interagir com a VM criada por meio de um terminal apresentado em seu *Browser*.

Como ilustrado na Figura 5, o aplicativo do "user1" rodando no *Pool* de Execução poderia fazer a chamada *http://user1.provinet.local/getTopology* para consultar a topologia da rede virtual disponibilizada em seu *Slice*. Se o usuário tem mais de um controlador, as requisições dos aplicativos serão direcionadas para os controladores conforme o algoritmo Round Robin, implementado pelo Servidor DNS. Assim, existirá um balanceamento de carga entre os controladores. Além disso, em caso de falha de um controlador, terá outro para atender as requisições.

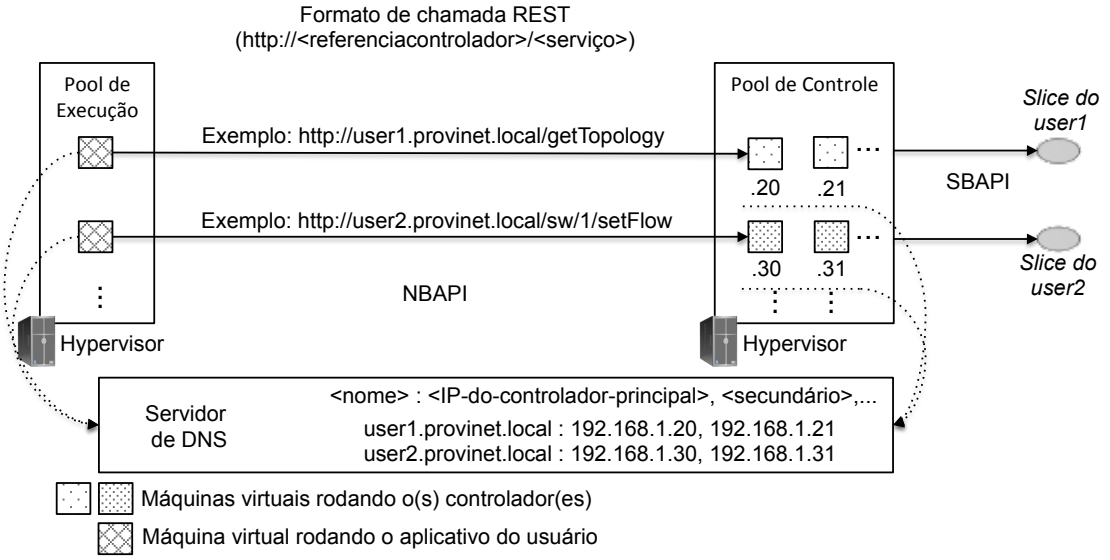


Figura 5. Abordagem de referências para provimento de escalabilidade e confiabilidade

O conjunto de serviços disponíveis deve variar de acordo com a tecnologia utilizada no plano de controle. Todavia, diante da necessidade de um novo serviço, o mesmo pode ser desenvolvido e instalado. Em geral, os controladores seguem uma arquitetura que permite o acoplamento de módulos com a implementação de novos serviços. Baseado nas informações de quais controladores pertencem a quais usuários, a plataforma ProViNet auxilia o usuário na instalação desses módulos.

Para implementar o módulo a ser instalado, o usuário deve seguir os tutoriais[1] disponibilizados pela fabricante do controlador adotado no plano de controle. O processo de instalação de um novo módulo se inicia pelo envio do módulo compactado para a plataforma. O módulo responsável então o encaminha a todos os controladores do usuário que estão associados a um determinado *Slice*. Em cada controlador, um *deamon* configurado para fazer a implantação de módulos executa uma rotina de instalação. Uma vez que os módulos estão instalados, os aplicativos dos usuários podem enviar requisições a ele (Ex.: *http://user1.provinet.local/modules/newModule/service*).

[1]Tomando como exemplo o controlador Floodlight, estão disponíveis no endereço *http://www.openflowhub.org/display/floodlightcontroller/How+to+Write+a+Module*, um conjunto de instruções para o desenvolvimento de novos módulos.

4. Protótipo

Com o objetivo de demonstrar a viabilidade da arquitetura conceitual detalhada na Figura 2 do Capítulo 3, foi desenvolvido um protótipo. Sua implementação baseia-se no desenvolvimento dos cinco principais módulos, a Interface Web, o Controle de Usuários, o Controle de Aplicativos, o Controle de *Slices* e Referências e a Administração do ProViNet. Tais módulos foram implementados utilizando o *framework* Django 1.4.3, a linguagem Python 2.7.3 e o sistema gerenciador de banco de dados PostgreSQL 9.1.6. A fim de fornecer maior compatibilidade do sistema, foi utilizado o servidor Web Apache 2.2.23.

O módulo Controle de Usuários fornece na interface do ProViNet, formulários de registro e login. O Controle de Aplicativos apresenta na interface uma área especial para apresentação dos serviços disponibilizados no plano de controle e uma área para requisição, controle e interação com as VMs no Pool de Execução. É disponibilizado pelo módulo de Controle de *Slices* e Referências um formulário para *upload* do documento VXDL e definição do nível de redundância. Finalmente, a área de Administração apresenta um conjunto de funcionalidades de configuração, que incluem o endereço de IP e dados de acesso dos *Pools*, do Servidor DNS, do Provedor de Infraestrutura Virtual e ainda a definição dos perfis de máquinas virtuais que serão disponibilizadas aos usuários.

Os *Pools* de Execução e de Controle são, na prática, servidores com alguma plataforma de virtualização instalada. No protótipo desenvolvido foi utilizado o *hypervisor* XenServer da CITRIX. Sendo assim, a comunicação entre os módulos da plataforma ProViNet e os Pools se dão pela utilização do XenServer SDK. Com o SDK é possível controlar e monitorar o *hypervisor* através de chamadas XML-RPC. O último módulo da plataforma é o Servidor DNS, o qual foi instalado em uma máquina com os sistemas Bind9 e Apache2. Para o controle dinâmico de configuração do DNS, um serviço web foi implementado para que, a partir de chamadas HTTP a uma URL específica, seja feita a adição e remoção de entradas no arquivo de configuração do bind9.

A implementação de controlador utilizada foi o Floodlight, por prover uma API RESTfull que possibilita o consumo dos serviços disponibilizados no controlador por meio de chamadas HTTP. Outras implementações poderiam ser utilizadas, desde que seja possível a instalação de módulos para provimento de serviços customizados e a disponibilização de serviços utilizando a arquitetura REST.

5. Caso de Estudo e Análise de Resultados

Para avaliar a solução apresentada foi elaborado um caso de estudo que aborda cada um dos diagramas de sequência apresentados no Capítulo 3. O cenário de execução é composto por servidores Intel Xeon CPU E3-1220 3.1GHz, 4GB RAM, com o *hypervisor* XenServer 6.1 representando o *Pool* de Controle e de Execução. O controlador OpenFlow utilizado foi o Floodlight v0.90, e é iniciado, por *script* durante a inicialização da VM (Ubuntu 12.04 com 1 vCPU e 384MB RAM) no *hypervisor*. Para executar o *framework* Django com o ProViNet foi utilizado um *laptop* Intel Core i7 2.8GHz e 4GB RAM. Por fim, o Servidor DNS foi instalado e configurado em um terceiro PC (Intel Core 2 Duo 2.33GHz e 4GB RAM) na mesma rede local que os outros PCs. Vale ressaltar que os tempos apresentados neste trabalho foram obtidos após 30 execuções, e representam a média uma vez que o coeficiente de variação foi muito próximo de zero.

Instanciar dois controladores	Adicionar informações no VXDL	Provimento da infraestrutura virtual (PIV)	Configuração de subdomínios	Total
12,824s	0,003s	57,81s	0,04s	75,677s

Tabela 1. Tempos para requisição de *Slice* e configuração de subdomínios

Atraídas pelo baixo custo de manutenção, segurança e armazenamento, é cada vez mais comum que empresas migrem parte de sua infraestrutura de computação para ambientes de *Cloud*, seja ela pública, privada ou híbrida. Para representar esse caso de estudo, consideramos uma rede composta por 7 *switches* e 4 *hosts*, organizados em uma topologia de árvore. Consideramos também que o usuário responsável pela migração já está registrado na plataforma ProViNet. Inicialmente, o usuário faz a requisição da infraestrutura virtual enviando um arquivo VXDL com a descrição da topologia (*switches*, *hosts* e *links*). A avaliação desse processo é apresentada na Tabela 1 e pode ser acompanhada pelo diagrama da Figura 3. Os tempos apresentados consideram nível de redundância igual a dois, ou seja, o *Slice* terá dois controladores associados.

Vale ressaltar que o tempo gasto para instanciar a infraestrutura virtual deve variar de acordo com a abordagem de mapeamento utilizada pelo PIV. A utilização de máquinas virtuais ao invés de *bridges* para representar os *switches* virtuais, por exemplo, certamente acarreta em maior custo de tempo. Os tempos apresentados consideram um sistema desenvolvido em um trabalho anterior chamado HyFS [Wickboldt *et al.* 2012] como provedora da infraestrutura virtual programável. O HyFS é atualmente a única plataforma de *private Cloud* capaz de receber requisições de rede virtual programável com topologias variadas. A topologia virtual é criada utilizando o modelo *overlay*, no qual *switches* e *hosts* são implantados em máquinas virtuais. Uma vez instanciados, os *software switches* são configurados para receberem informações de controle de controladores externos, estabelecidos pela plataforma ProViNet.

Após o provimento da infraestrutura virtual, o usuário responsável pela migração deve avaliar os serviços presentes no cenário anterior a migração, tais como *Firewall*, Balanceadores de carga e outros serviços necessários. Uma vez que a rede virtual a ser criada é programável, esses serviços podem ser implementados em forma de aplicativos de rede. Utilizando uma notação simplificada, um exemplo de aplicativo para tratamento de ataques de negação de serviço (DDoS) é apresentado no Algoritmo 1. Esse aplicativo roda no plano de execução, em uma VM que o usuário acessa através do *Browser*.

O algoritmo apresentado analisa o tráfego recebido por um *switch* a cada 10 segundos, se esse tráfego for superior a um limite pré definido o sistema toma alguma ação. Essa ação depende do número de ocorrências em que se detectou tráfego superior ao limite definido. Na segunda ocorrência o sistema bloqueia o tráfego ICMP, na terceira, o tráfego Web (porta 80) é bloqueado, e por fim, na quarta ocorrência, todo tráfego é bloqueado e um gerente é contactado. Certamente essa não é a melhor solução, mas é um exemplo de um possível aplicativo de rede que pode ser desenvolvido pelo usuário e implantado no *Pool* de Controle para proteger seu *Slice* desse tipo de ataque.

Para medir desempenho das requisições do aplicativo do usuário, foram realizados experimentos com varições no nível de redundância no plano de controle. Espera-se uma variação no desempenho das chamadas, pois ao utilizar uma maior quantidade de controladores, o balanceamento de carga realizado pelo Servidor DNS torna o processo

Algorithm 1 Exemplo de Aplicativo: Solução ProViNet para DDoS

Require: limite - Limite de tráfego considerado normal
Require: swdpid - DPID do switch de entrada de tráfego rede

```
1: limite ← 2000MB
2:
3: anterior ← httpRequest(user1.provinet.local/readstatus/swdpid)
4: ocorrencias ← 1
5: while True do
6:     atual ← httpRequest(user1.provinet.local/readstatus/swdpid)
7:     if (atual − anterior) ≥ limite then
8:         if ocorrencias == 2 then
9:             httpRequest(user1.provinet.local/addFlow/swdpid/blockICMP)
10:        else if ocorrencias == 3 then
11:            httpRequest(user1.provinet.local/addFlow/swdpid/blockWeb)
12:        else if ocorrencias ≥ 3 then
13:            httpRequest(user1.provinet.local/addFlow/swdpid/blockAll)
14:            CallManager()
15:        end if
16:        ocorrencias + +
17:    end if
18:    anterior ← atual
19:    sleep(10)
20: end while
```

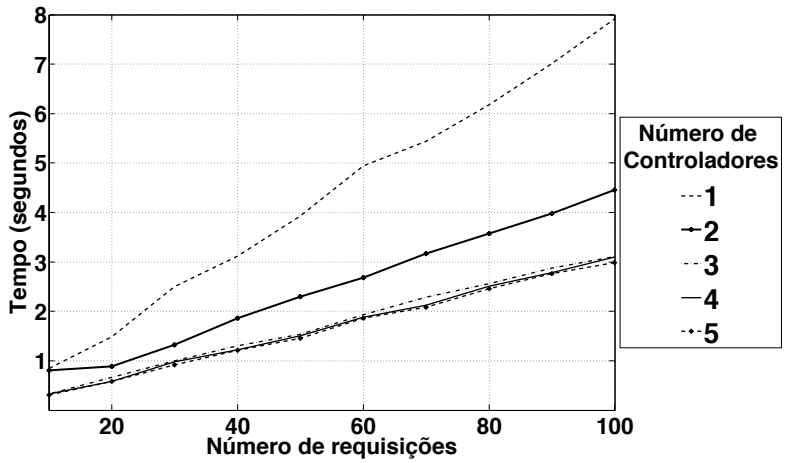


Figura 6. Performance do plano de controle com balanceamento de carga

mais rápido. Tal fato pode ser acompanhado no gráfico da Figura 6, que mostra o tempo gasto para concluir X requisições, sendo X, valores entre 10 e 100 com intervalos de 10 unidades. A requisição executada para obtenção dos valores apresentados foi *provinet.local/wm/core/controller/switches/json*, a qual retorna a lista de *switches* presentes no *Slice*.

Analisando os valores apresentados no gráfico 6, percebe-se que o balanceamento de carga provido pela abordagem proposta e gerenciado pelo ProViNet é efetivo e implica

em uma redução do tempo gasto para execução das requisições. Entretanto, o ganho se torna menos significativo para um número de controladores maior que 3. Ou seja, utilizando apenas 1 controlador, foram gastos em média 7,91 segundos para concluir 100 requisições, ao passo que com 2 controladores esse valor caiu para 4,45. O ganho ao aumentar o número de controladores de 4 para 5, não é tão expressivo quanto de 1 para 2, saindo de 3,09 para 2,98 segundos nesse caso. Discussões mais aprofundadas nesse contexto foram apresentadas por Heller *et.al.* [Heller *et al.* 2012].

Para avaliar a funcionalidade de implantação de módulos sob demanda, desinstalou-se do Floodlight um módulo que originalmente já vem instalado, o módulo de *firewall*. Compactou-se tal módulo em um arquivo e, através da interface do ProViNet, foi feita a requisição de instalação. Uma vez que o módulo Controle de Aplicativos possui cadastrado o endereço IP de todos os controladores e seus respectivos usuários, após receber o arquivo por *upload*, um *script* de envio é disparado. O papel desse *script* é acessar via ssh a VM de cada controlador, fazer a cópia do novo módulo para uma pasta específica na VM e ativar um segundo *script* na VM que faz a instalação do mesmo. Esse segundo *script* segue as informações disponibilizadas no site do Floodlight para instalação de módulos. O tempo médio gasto nesse processo foi de 23,435 segundos.

6. Conclusões e Trabalhos Futuros

A diversidade de ambientes computacionais requerem distintos serviços de comunicação em redes. As soluções desenvolvidas no passado, e implementadas de acordo com as vontades das fabricantes de equipamentos de redes, podem não ser mais suficientes. Entretanto, com o surgimento de propostas abertas de virtualização e programabilidade, como as Redes Definidas por *Software*, a criação de novas soluções se torna, de certa forma, mais democrática. Ou seja, depende menos dos anseios financeiros das grandes fabricantes.

Todavia, a complexidade inerente ao gerenciamento de ambientes de Rede Virtual Programável (RVP) ainda representa um grande desafio. Neste trabalho, propomos a plataforma ProViNet para o gerenciamento da implantação de aplicativos de rede em ambiente de RVP. A plataforma ProViNet contribui com uma arquitetura escalável, utilizando o conceito de *Resource Pool*, com uma abordagem para a instalação dinâmica de novos módulos no plano de controle, com a extensão da linguagem de definição de infraestrutura virtual de rede virtual programável, chamada VXDL, e por fim, com o desenvolvimento de um sistema com interface de acesso Web, facilitando a compreensão e interação com usuários finais.

Como trabalhos futuros pretende-se investigar mais precisamente, como os ambientes de *Cloud* poderiam prover pratilheiras de serviços de rede dinâmicas. As quais seriam ocupadas por soluções desenvolvidas e consumidas pelos próprios usuários de *Cloud*.

Referências

Campbell, A. T., Meer, H. G. D., Kounavis, M. E., Miki, K., Vicente, J. B., e Villela, D. (1999). A survey of programmable networks. *Computer Communication Review*, 29:7–23.

Chowdhury, N. e Boutaba, R. (2009). Network virtualization: state of the art and research challenges. *Communications Magazine, IEEE*, 47(7):20 –26.

Chowdhury, N. M. K. e Boutaba, R. (2010). A survey of network virtualization. *Computer Network*, 54(5):862–876.

GENI (2011). Global Environment for Network Innovations. Disponível em: http://www.geni.net/. Acessado em: Julho 2012.

Gutz, S., Story, A., Schlesinger, C., e Foster, N. (2012). Splendid isolation: a slice abstraction for software-defined networks. Em *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, páginas 79–84, New York, NY, USA. ACM.

Hausheer, D., Parekh, A., Walrand, J., e Schwartz, G. (2011). Towards a compelling new internet platform. Em *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, páginas 1224 –1227.

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, páginas 7–12, New York, NY, USA. ACM.

Kanaumi, Y., Saito, S., e Kawai, E. (2010). Deployment of a programmable network for a nation wide randd network. Em *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, páginas 233 –238.

Kopsel, W. (2011). Ofelia - pan-european test facility for openflow experimentation.

Koslovski, G. P., Primet, P. V.-B., e Charão, A. S. (2008). Vxdl: Virtual resources and interconnection networks description language. volume 2 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, páginas 138–154. Springer.

Lantz, B., Heller, B., e McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. Em *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, páginas 19:1–19:6, New York, NY, USA. ACM.

Lin, P., Bi, J., Hu, H., Feng, T., e Jiang, X. (2011). A quick survey on selected approaches for preparing programmable networks. Em *Proceedings of the 7th Asian Internet Engineering Conference*, AINTEC '11, páginas 160–163, New York, NY, USA. ACM.

OpenStack (2011). Open source software for building private and public clouds. Disponível em: http://www.openstack.org/. Acessado em: Julho 2012.

ProtoGENI (2012). Control Framework for GENI Cluster C. Disponível em: http://www.protogeni.net/trac/protogeni. Acessado em: Dezembro 2012.

Rubio-Loyola, J., Galis, A., Astorga, A., Serrat, J., Lefevre, L., Fischer, A., Paler, A., e Meer, H. (2011). Scalable service deployment on software-defined networks. *Communications Magazine, IEEE*, 49(12):84 –93.

Wickboldt, J. A., Granville, L. Z., Schneider, F., Dudkowski, D., e Brunner, M. (2012). A new approach to the design of flexible cloud management platforms. Em *8th International Conference on Network and Service Management (CNSM)*, páginas 155–158, Las Vegas, USA.

KeyFlow: Comutação por Chaves Locais de Fluxos Roteados na Borda via Identificadores Globais

Rafael Emerick Zape de Oliveira[1], **Rômulo Vitoi**[2],
Magnos Martinello[2], **Moisés Renato Nunes Ribeiro**[1]

[1]Laboratório de Telecomunicações (LabTel) – Depto. de Engenharia Elétrica
[2]Laboratório de Pesquisa em Redes e Multimídia (LPRM) – Depto. de Informática

Universidade Federal do Espírito Santo (UFES)
CEP 29075-910 - Vitória - ES - Brasil

rafael.emerick@ufes.br,{ravitoi,magnos}@inf.ufes.br, moises@ele.ufes.br

Abstract. *The large bulk of packets/flows in OpenFlow-based networks will require a level of efficiency in the processing of switching elements that do not compare to the classical methods of query tables flows. Simplifying procedures of lookup in core network is a necessary and ongoing effort to enable the transport of data at high rates and low latency. This paper presents and studies the performance in dataplane an approach for modification of OpenFlow switches named KeyFlow. Its goal is to enable simplified identifiers for global operations flows through the rest of the division between the global label and the local key of the switch.*

Resumo. *O grande volume de pacotes/fluxos nas redes baseadas em OpenFlow vai exigir um nível de eficiência no processamento dos elementos de comutação que não dispomos ainda nos métodos clássicos de consulta a tabelas de fluxos. Simplificar os procedimentos de lookup do núcleo da rede é um esforço necessário e contínuo para viabilizar o transporte de dados em altas taxas e baixa latência. Este artigo apresenta e estuda o desempenho no plano de dados de uma abordagem para modificação dos comutadores OpenFlow nomeada KeyFlow. Seu objetivo é viabilizar identificadores simplificados para fluxos globais via operações de resto da divisão entre o identificador e a chave local do comutador.*

1. Introdução

As recentes aplicações que rodam em nuvens privadas ou públicas são significativamente influenciadas pelo projeto da infraestrutura de rede. Muitas aplicações precisam trocar informações com nodos remotos para efetuar sua computação local. Por exemplo, aplicações MapReduce e sistemas de arquivos frequentemente requerem acesso a nodos remotos antes de prosseguir para as operações de entrada e saída (E/S). Neste caso, é preciso que a rede possa prover comunicação com largura máxima de banda e atrasos mínimos sem comprometer a qualidade dos serviços transportados.

Um dos elementos críticos nesta infraestrutura é o tamanho das tabelas de encaminhamento nos comutadores. Tradicionalmente, a abordagem para escalar o projeto do

tamanho das tabelas tem sido adicionar mais recursos de memória no silício do comutador ou permitir o uso de recursos de memória externa. Entretanto, com o aumento da densidade dos comutadores em redes de *data center* combinado com a necessidade de eficiência energética e custo, há uma demanda importante por novas formas de encaminhamento.

Uma tabela de encaminhamento ou *Forwarding Information Base* (FIB) implementada em um comutador é usada para roteamento, encaminhamento e funções similares para determinar a interface apropriada para a qual um comutador deve transmitir um pacote. Quando estas tabelas atingem suas capacidades máximas problemas de desempenho ocorrem. Um exemplo é o aprendizado de endereços MAC: se o conjunto de endereços MAC ativos na rede (função do número de máquinas virtuais na rede) for maior que o tamanho da tabela de encaminhamento, então haverá *flooding* para descobrir os endereços que não estiverem nas tabelas, afetando o desempenho da rede.

Recentemente, o *OpenFlow* [McKeown et al. 2008] tem sido amplamente adotado para identificação e tratamento de fluxos em redes experimentais e acadêmicas. A arquitetura baseia-se em um controlador externo aos *switches Openflow* que centraliza a gerência de encaminhamento de pacotes através de uma visão global e manutenção de estados dos fluxos ativos na rede. Assim, o *OpenFlow* pode encontrar problemas de escalabilidade no futuro em função da necessidade de manutenção completa de estados (*fullstate*) dos fluxos ativos e da necessidade, a cada novo fluxo, de comunicação com o controlador e seu processamento. Soma-se a isto o fato do contínuo crescimento da capacidade de transmissão por enlaces pesar sobre a capacidade de processamento eletrônico dos pacotes para roteamento ou encaminhamento. Desta forma, iniciativas para o desenvolvimento de técnicas sem manutenção de estados (*stateless*), atingindo um bom compromisso entre eficiência e complexidade, são necessárias.

Este trabalho apresenta uma abordagem de modificação da forma de comutar pacotes, analizando a alteração dos comutadores *OpenFlow* para encaminharem pacotes com base em operações mais simples, em contraste ao encaminhamento tradicional baseado em consultas a tabelas de fluxos. O objetivo é apresentar um protótipo de implementação que viabilize um processo simplificado de *lookup* para os identificadores globais de fluxos via operações de resto da divisão entre o rótulo e a chave local, e com isto obter um indicativo de quanto a solução proposta pode beneficiar a criação de circuitos para redes de experimentação e de produção. Sendo assim, busca-se alternativas para criação de conectividade entre nodos de maneira a atender, no plano de dados, demandas de conexões dinâmicas e flexíveis com o mínimo de ocupação de recursos possível. A proposta é implementada e validada num ambiente de prototipação baseado em *Mininet*, tendo o comutador *OpenFlow* padrão 1.0 como referência de desempenho. A avaliação compara os atrasos *Round-Trip Time* (RTT) (média, desvio padrão, melhor e pior caso) variando-se o número de elementos de comutação no núcleo da rede, assim como o tamanho e a ocupação nas tabelas de fluxo dos comutadores.

A seção que segue apresenta alguns trabalhos relacionados. Na seção 3 é apresentada a abordagem de encaminhamento do protótipo do *KeyFlow*. A seção 4 descreve a metodologia adotada, o ambiente de testes como prova de conceito e apresenta os resultados obtidos da avaliação de desempenho comparativa. Por fim, a seção 5 resume o trabalho com as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Para melhorar a eficiência na comutação de pacotes em redes *OpenFlow*, uma estratégia é minimizar as interações dos comutadores com o controlador [Mogul et al. 2010]. Embora as limitações não sejam salientes quando se utilizam regras fixas, quando empregado em redes com muitos elementos e sob demandas muito dinâmicas, uma implementação conhecida pode iniciar apenas poucas centenas de fluxos por segundo. Uma estimativa deste valor no equipamento desenvolvido pela fabricante HP, o ProCurve 5406z, é de 275 fluxos por segundo. Isto indica que a manutenção de estados completa (*statefull*) dos fluxos ativos é um problema para a escalabilidade das redes definidas por *software* (SDN).

O trabalho de [Bianco et al. 2010] apresenta uma análise de desempenho considerando a ocupação das tabelas de manutenção de estado, em diferentes tipos de comutadores implementados em Linux. O trabalho indica que as implementações mais eficientes deverão dispor de memórias mais rápidas, e consequentemente mais caras e com maior consumo de energia.

No que tange ao plano de controle das redes, o atraso de inserção de regras em muitos comutadores *OpenFlow* disponíveis no mercado gira em torno de 10ms para um único fluxo e chega a 1s para 1000 fluxos [Rotsos et al. 2012]. Isto mostra o quanto as redes definidas por *software* (SDN) podem sofrer com a complexidade de informações no plano de controle. Assim, busca-se no presente trabalho viabilizar o encaminhamento por meio de rótulos mais simples de maneira a facilitar o processo de inserção e atualização das regras de identificação dos fluxos.

Um esquema de roteamento na origem com rótulo único para encaminhamento pensado para redes ópticas de comutação de pacotes [Wessing et al. 2002], serve de base para a presente proposta. O artigo original teve por objetivo a não alteração do cabeçalho nos nodos de núcleo em função da dificuldade de tal operação no domínio óptico. O nosso interesse, entretanto, é explorar especificamente sua forma simples de encaminhamento em cada salto com propósito de redução de latência. Ela nos permite uma mudança conceitual importante de mecanismos convencionais de busca em tabelas para operações elementares sobre o identificador de fluxo. Tais operações podem ser realizadas com poucos ciclos de relógio combinando potencialmente redução na latência e processamento demandado por pacote em trânsito pela porta do comutador *OpenFlow*. A nossa proposta, por ser um mecanismo determinístico, também diferencia-se de técnicas baseadas em Bloom Filter, como por exemplo em [Rothenberg et al. 2010] que tem desempenho probabilístico em função da estrutura de dados utilizada.

Em [Casado et al. 2012] é apresenta a necessidade das SDN suportarem um núcleo "fabric", com a preocupação principal de entrega/encaminhamento de pacotes de uma origem a um destino sem utilização do cabeçalho do pacote original para encaminhamento no núcleo. Sugere-se, assim, a utilização de comutadores OpenFlow diferenciados em borda e núcleo, onde o núcleo utilizaria de funções comuns ao MPLS. A proposta do KeyFlow se alinha fortemente com as propostas de [Casado et al. 2012], porém, como detalhado nas seção 3, no que tange a solução para rede "fabric" entende-se que o KeyFlow pode propiciar uma maior simplificação em relação ao MPLS, o que favorece a sintetização de hardware mais simples e especializado sem a ocupação de recursos de estado em cada nodo do caminho. Além disso, pela utilização do KeyFlow, é possível a definição dos caminhos unicamente pela informação adicionada na borda, o que pro-

piciará o estabelecimento de conectividade para as redes definidas por software de uma maneira ainda mais flexível.

Outra contribuição da presente proposta consiste em estender o conceito de isolamento do plano de dados em relação ao plano de controle no contexto de *OpenFlow 1.0*, pela assimetria na relação comutador(es)-controlador. Os nodos de núcleo não executam operações de consulta no controlador, uma vez que os nodos de borda se responsabilizam por esta tarefa de modo a prover o roteamento na origem. Como consequência, há uma redução no número de consultas, além de agilizar substancialmente o processo de reconfiguração da rede por eliminação do tempo de instalação de regras no núcleo.

3. Abordagem KeyFlow: Rótulos Globais e Chaves Locais

O conceito de encaminhamento adotado pelo *KeyFlow* consiste na utilização do Esquema de Informação por Chave (KIS), proposto em [Wessing et al. 2002] para redes ópticas, na criação de topologias *overlay* em redes *OpenFlow*. Os rótulos globais estão relacionados com as chaves locais de tal modo que o resultado da operação de resto da divisão (*mod*) entre o rótulo transportado no pacote e a chave local, em cada nodo da rede, resulta na porta de saída no comutador, sem utilizar operações de consulta em tabelas e sem a necessidade de sinalização no núcleo da rede após as definições das chaves locais. Deste modo, o processamento nos nodos de núcleo é realizado de maneira eficiente, tendendo a ser uma solução escalável com o crescimento da rede e de baixo custo operacional.

Para geração dos rótulos globais, utiliza-se o Teorema Chinês do Resto (TCR). Para cada caminho (circuito virtual), a computação é efetuada a partir de dois vetores. O primeiro vetor é formado pelas chaves locais de todos os comutadores que compõem o caminho de interesse. O segundo vetor é formado pelo número da porta de saída de cada um destes comutadores. Como condição necessária para a criação de chaves válidas, restringe-se que todas as chaves definidas, em cada caminho, sejam primas entre si, conforme descrito a seguir na seção 3.1.

O estabelecimento dos caminhos prévios da rede, viabiliza uma importante vantagem do KeyFlow em relação ao MPLS. O estabelecimento de caminhos nestas redes é feitos por protocolos de sinalização como o RSVP e o LDP, em ambos os casos, é necessário a manutenção de estados em cada nodo. Mesmo considerando que, em alguns casos, esta manutenção de estados tem uma ocupação de recurso insignificante para circuitos lógicos estáveis, é importante ressaltar que no estado transitório de estabelecimento do caminho, esta sinalização pode impactar significativa na flexibilidade do atendimento à demandas dinâmicas. Como no KeyFlow é a borda que define o caminho que cada pacote seguirá, os fluxos poderão seguir diferentes caminhos, com ou sem redundância, independentemente do núcleo tomar conhecimento desta necessidade. Sendo assim, a possibilidade de melhor ocupação dos enlaces disponíveis e a qualidade de serviço a ser garantida para as aplicação tenderá a ser mais flexivel e mais simples, com possibilidade de suporte de tráfego unicast e multicast entre as bordas da rede.

A Fig. 1 ilustra uma visão geral da integração do KeyFlow a SDN. Nela há o controlador principal atendo as demandas dinâmicas da borda, composta por comutares *OpenFlow* extendidos para o suporte aos rótulos que definirão os caminhos utilizados pelos fluxos na rede "fabric". Nesta, verifica-se o controlador KeyFlow especializado em computar rótulos, conforme apresentado na Seção 3.1.

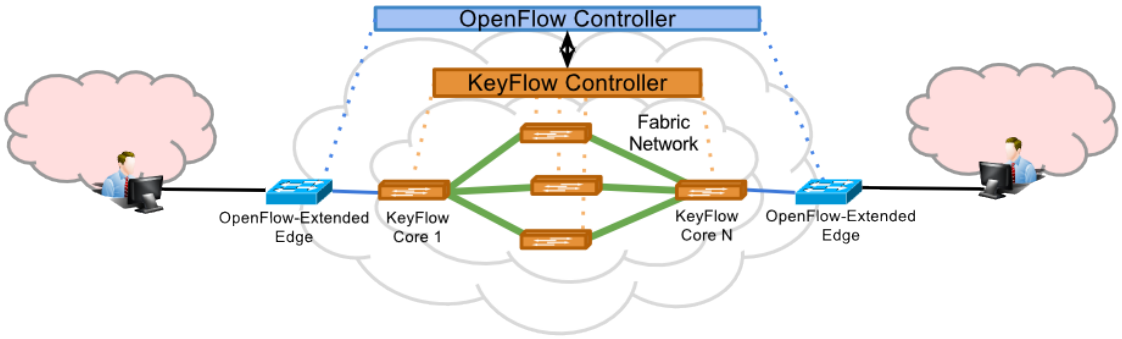


Figura 1. Arquitetura possível para uma rede definida por sofware integrada à solução KeyFlow.

3.1. Teorema Chinês do Resto

Conforme descrito em [Wessing et al. 2002], considere a existência de dois vetores, $\bar{k}$ e $\bar{s}$, respectivamente o conjunto de chaves e o conjunto das portas de saída dos comutadores que compõe um determinado caminho. Se o caminho em questão possui n nodos, tem-se:

(1) $\bar{k} : (k_1, k_2, \ldots, k_n)$ e $\bar{s} : (s_1, s_2, \ldots, s_n)$

O rótulo global r é tal que seja possível restaurar o vetor $\bar{s}$ usando o vetor $\bar{k}$. Para isto, utiliza-se o escalar k,

(2) $k = k_1 \cdot k_2 \cdot k_3 \cdots k_n$

Utilizando-se k e $\bar{k}$, cria-se o vetor $\bar{m}$, de modo que para cada elemento deste vetor, a condição $m_i \ mod \ k_j = 0$ seja atendida para todos $j \neq i$, o que é satisfeito pela seguinte equação:

(3) $m_i = \frac{k}{k_i}$, para $i \leqslant n$

Cria-se, então, outro vetor $\bar{c}$, baseado nos vetores $\bar{m}$ e $\bar{k}$, de maneira que seus elementos sejam da forma:

(4) $c_i = m_i(m_i^{-1} \ mod \ k_i)$, para $i \leqslant n$

O termo m_i^{-1} representa o inverso multiplicativo de m_i definido por $m_i m_i^{-1} \equiv 1 (\text{mod } k_i)$, e que pode ser obtido de maneira computacional. Sendo assim, tanto m_i e k_i devem ser primos entre si. Como m_i é múltiplo de todos os n elementos de $\bar{k}$, exceto k_i, verificamos que todas as chaves, de cada comutador de um dado caminho, devem ser primas entre si. O escalar r é finalmente calculado utilizando-se os vetores $\bar{c}$ e $\bar{s}$, assim:

(5) $r = (s_1 c_1 + s_2 c_2 + s_3 c_3 + \cdots + s_k c_k) \ mod \ k$

Dado o escalar r e o vetor $\bar{k}$, é possível recuperar o vetor $\bar{s}$, por meio da seguinte operação:

(6) $s_i = r \ mod \ k_i$ para $i \leqslant n$

Portanto, pelo TCR é possível gerar um escalar r a partir de dois vetores $\bar{k}$ e $\bar{s}$, onde todos os elementos do primeiro devem ser primos par-a-par. Uma análise mais detalhada a respeito do tamanho do rótulo em relação ao tamanho da rede pode ser obtida em [Wessing et al. 2002].

3.2. Implementação do Protótico de Comutador KeyFlow

Pela utilização do protocolo *OpenFlow* modificado no plano de controle da rede KeyFlow pode-se implementar o algoritmo de definição dos rótulos diretamente em uma aplicação no controlador e então distribuir as chaves para o núcleo da rede e os rótulos para as bordas. Com base na visão centralizada oferecida pela arquitetura, pode-se calcular todos os rótulos previamente na fase de inicialização da rede, por exemplo. Em segundo momento, define-se os melhores caminhos a serem repassados aos elementos de borda. Este repasse pode ser de maneira pró-ativa, pela instalação imediata de todas as regras, ou reativa, pela interação do elemento de borda com controlador.

O protótipo foi implementado a partir da implementação de *Stanford* do comutador em software. A diferença entre o protótipo de comutador *KeyFlow* e o padrão de referência da especificação *OpenFlow 1.0* pode ser verificada no fluxograma da Fig. 2.

Os pacotes ao entrarem no comutador *OpenFlow 1.0* têm os cabeçalhos analisados e então inicia-se a busca para verificar se eles pertencem a um determinado fluxo. Isto é feito pela busca sequencial nas tabelas existentes, do menor ao maior índice. Com isto, constata-se que o comutador de referência prioriza as regras mais específicas pela busca inicial na tabela *hash* (*table*=0, máx. 131.070 entradas). Caso não haja uma identificação do fluxo nesta tabela, a consulta é feita na tabela linear (*table*=1, máx. 100 entradas), que possui os registros curingas (com itens caracterizados pelo *, de "não importa"). Se nenhuma regra for encontrada, o pacote é encapsulado e enviado ao controlador da rede, caso contrário, as ações definidas na regra são aplicadas e o pacote é encaminhado para a respectiva porta de saída.

No fluxograma, verifica-se que antes mesmo de qualquer interação com o controlador da rede, de onde já se espera um atraso natural para caracterização do fluxo, há um forte componente aleatório de atraso na pesquisa pela regra nas tabelas de fluxo. Quanto maior for o número destas tabelas, e o respectivo tamanho ocupado, maior será a variação de tempo necessário para se aplicar as ações e enviar o pacote para seu destino. Sendo assim, para um grande número de fluxos ativos no comutador *OpenFlow* é esperada uma maior variação nos tempos de encaminhamento de cada pacote. Vale lembrar que é possível reduzir o tempo de acesso por meio da utilização de memórias mais rápidas, contudo estas memórias têm restrições ligadas ao seu tamanho e ao seu custo, o que implica numa limitação natural para a escalabilidade do encaminhamento.

Ainda na Fig. 2, verifica-se que o comutador *KeyFlow* é um comutador *OpenFlow* instrumentado para comutar pacotes com base em operações de resto da divisão no tratamento de fluxos. Neste comutador, há uma nova variável que controla a chave local do equipamento. Caso ela não esteja instalada (valor padrão zero), o fluxo de tratamento de pacotes segue de maneira idêntica ao padrão *OpenFlow* tradicional. Contudo, se houver a devida definição dessa chave, o fluxo do processamento é deslocado diretamente para a execução da função módulo sobre o campo identificador de vlan em relação à chave local. Uma vez com a chave instalada, o comutador passará a atuar exclusivamente como KeyFlow, não havendo um comportamento híbrido. Com isto, espera-se um tratamento dos pacotes/quadros de uma forma menos aleatória, uma vez que os processos de pesquisa em tabela são eliminados.

O campo identificador de vlan, do quadro Ethernet, foi utilizado no protótipo para

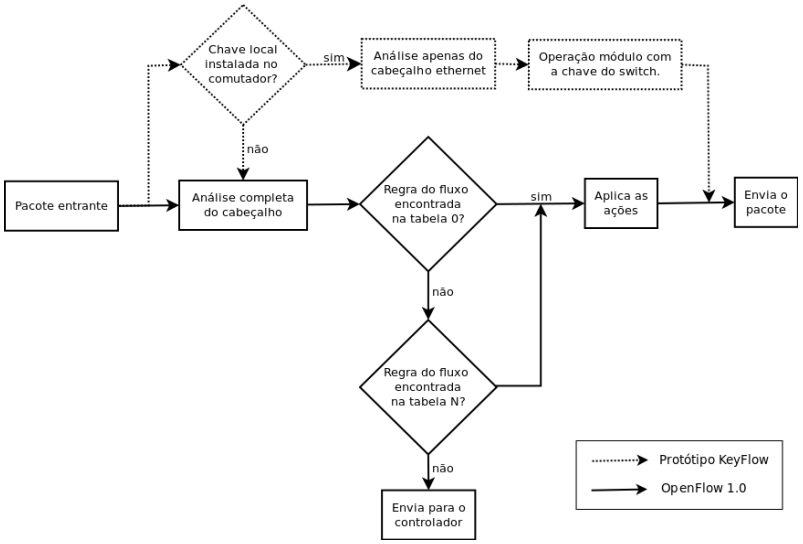


Figura 2. Fluxograma do tratamento de pacotes da especificação *OpenFlow 1.0* [Heller et al. 2009] e do protótipo de comutador KeyFlow.

validação da proposta. Devido ao seu pequeno tamanho (2^{12}, 4096 rótulos) entende-se que este rótulo não seja o ideal para marcação de grande escala. O campo MPLS, conforme apresentado em [Wessing et al. 2002], apresenta boa escalabilidade e possui uma quantidade de bits suficiente para criação de circuitos lógicos com algumas restrições de roteamento.

Alternativamente, pode-se criar um rótulo específico para o encapsulamento dentro do núcleo KeyFlow por meio de alteração mais profunda nos nodos de encaminhamento, dependendo da solução de estabelecimento de caminhos considerada no controlador. Caso mantida a utilização de cabeçalho já existentes, torna-se necessária uma reserva de faixa de rótulos para que circuitos ativos não interfiram em eventuais experimentos de usuários. No tocante a utilização de um novo rótulo, este deve ser totalmente transparente para os pontos finais, sendo utilizado apenas para estabelecimento de caminhos no núcleo da rede. Assim, o núcleo passa a um comportamento similar a um grande comutador "fabric" que interconecta os usuários da borda.

4. Metodologia de Validação

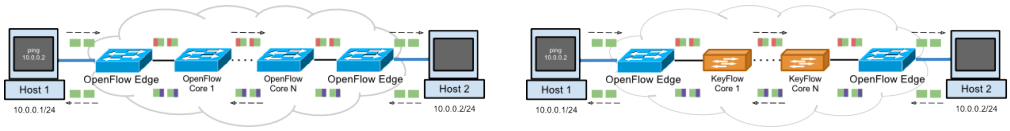
4.1. Cenários Avaliados

Para avaliar a eficiência da comutação baseada em chaves locais, desenvolveu-se um protótipo de comutador em *software*. Para medir a eficiência da comutação, utilizou-se o ambiente de prototipação do *Mininet* [1]. Neste ambiente, foi criada uma topologia linear com múltiplos comutadores virtuais conectando duas máquinas reais externas ao ambiente de emulação. Verificou-se a qualidade da experiência de um usuário conectado a esta rede por meio da medição do RTT dos pacotes até um alvo em outra extremidade do caminho. Foi realizada uma análise comparativa por meio da realização de testes utilizando os nodos de núcleo com comutadores *OpenFlow*, Fig. 3(a), onde se realizava pesquisa na tabela de fluxos salto a salto, comparativamente com o núcleo composto por

[1] http://mininet.github.com

comutadores *KeyFlow*, Fig. 3(b), que realizam a operação matemática módulo (resto da divisão) para determinar a porta de saída do pacote. Neste caso, a pesquisa por tabela acontecia apenas nos elementos de borda do caminho.

Nas extremidades foi configurada uma rede IP, e dela foram disparadas rajadas uniformes de pacotes ICMP para testes de continuidade e avaliação dos tempos de ida e volta dos pacotes. A rede da nuvem entre os nodos fica responsável pelo estabelecimento de conectividade fim-a-fim. Para se verificar a escalabilidade da solução, variou-se o número de comutadores de núcleo. Para avaliação da sensibilidade da transmissão frente a ocupação das tabelas de fluxos, variou-se o volume de regras instaladas nos elementos. Para identificação dos fluxos, utilizou-se diferentes 'id vlan' para se determinar o sentido do encaminhamento dos pacotes. Isto teve por objetivo criar um ambiente de encaminhamento similar às redes baseadas em comutação por rótulos.



(a) Núcleo com *lookup* em tabela de fluxo *OpenFlow*

(b) Núcleo com comutação KeyFlow

Figura 3. Topologia utilizada para avaliação do RTT do caminho lógico.

Na topologia da Fig. 3(a), utilizou-se apenas comutadores *OpenFlow* em todo o caminho. A cada salto, fez-se necessária uma consulta na tabela de fluxo em busca da regra válida para o encaminhamento no sentido correto. Os elementos da borda eram os responsáveis por introduzir e retirar devidamente o campo da vlan, fazendo a definição do circuito virtual, por meio de uma regra devidamente pré-instalada. Na Fig. 3(b), utilizou-se uma topologia idêntica, mas com comutadores *KeyFlow*. O resultado desta operação definia a porta de saída do pacote, sem a necessidade de ocupação de uma tabela de fluxos.

Toda a instalação das regras e a definição de chaves foram realizadas antes do inicio dos testes ICMP. Isto foi realizado por meio de *scripts* em *shell* do Linux hospedeiro do *Mininet*. Para estas sinalizações do plano e controle foi utilizado o aplicativo 'dpctl' com conexão direta pela *loopback* do sistema em cada comutador virtual, via conexão em diferentes portas TCP, uma para cada elemento. Todos os comutadores virtuais foram executados com privilégio de usuário no sistema. Para instalação da chave foi utilizada uma versão modificada do dpctl capaz de manipular a chave no comutador *KeyFlow*.

4.2. Ambiente de Testes

Para realização dos experimentos descritos foram utilizadas três estações. Duas máquinas físicas idênticas realizavam as funções de usuários em rede IP, representados por 'Host 1' e 'Host 2', nas Figs. 3(a) e 3(b). Neste ambiente, cada algoritmo que representa um comutador é executado em um processo isolado que trata cada pacote recebido de maneira idêntica a um comutador real. A diferença do ambiente emulado para o real está na inexistência de transmissão dos dados, uma vez que cada processo envia e recebe os pacotes, ou mais especificamente quadros Ethernet, de interfaces virtuais do ambiente Linux, sem a necessidade de realização de E/S na comunicação inter-processos. Apenas os processos dos comutadores de borda foram conectados às interfaces físicas da máquina hospedeira,

sendo possível assim, a entrega dos quadros para as máquinas alvos, tornando o serviço de entrega quadros mais próximo de um ambiente real. A Fig. 4 apresenta esquematicamente o ambiente experimental, e a Tabela 1, as configurações de cada máquina utilizada e as suas respectivas versões dos softwares.

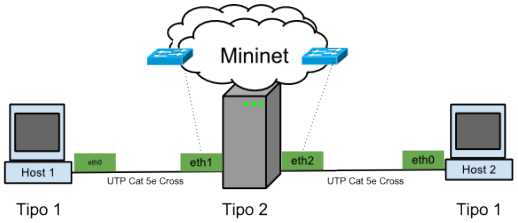


Figura 4. Esquemático do ambiente real de testes.

Tipo	Hardware	Software
1	PC i686 genérico. Interface GigaEthernet.	CentOS 5.0 GNU/Linux 2.6.18-274.el5
2	CPU Intel Xeon 3075 (2Núcleos), 2.66GHz, Cache: L1 32KB, L2 4MB Memória DDR2 800MHz 4GB 2 PCI GigaEthernet 1 FastEthernet (eth0) (acesso remoto não ilustrado) Placa Mae: ProLiant ML110 G5	*Mininet* Ver. 1.0.0 Ubuntu 11.10 GNU/Linux 2.6.38-12-generic i686

Tabela 1. Detalhamento da configuração utiliza no ambiente de testes.

4.3. Avaliação de Desempenho

Os testes foram realizados pela medição do RTT dos pacotes ICMP no caminho entre o 'Host 1' e o 'Host 2', de acordo com os cenários da Fig. 3(a) e Fig. 3(b). Para análise da escalabilidade, variou-se o número de elementos de núcleo de 1 até 15, somando-se os dois elementos da borda, o número total de comutadores no caminho variou de 3 a 17. O tamanho dos pacotes gerados pelo usuário foi de 54 bytes.

Para avaliação da sensibilidade do circuito com a ocupação das tabelas de fluxo, variou-se a ocupação das tabelas *hash* de todos os elementos *OpenFlow* pertencentes ao caminho. Testes preliminares mostraram que a ocupação completa da tabela linear, Fig. 5, para o ambiente emulado utilizado, não apresentou comprometimento significativo na comutação dos pacotes, possivelmente pelo baixo número de registros frente à capacidade computacional da máquina hospedeira do *Mininet*.

Sendo assim, para análise da ocupação das tabelas de fluxos, utilizou-se a tabela linear sempre cheia, com 98 regras aleatórias inválidas para a comutação, e 2 regras válidas responsáveis pelo encaminhamento e/ou pela retirada da marcação do pacote, no caso de entrada e saída da rede. Estas regras eram inseridas no final da tabela linear. Além disso, utilizou-se quatro ocupações distintas da tabela *hash*: 0%, 25%, 50%, 75%. Estas porcentagens eram relativas à capacidade máxima da referida tabela.

Os pacotes marcados com o número 5 foram encaminhado para a direita (porta 2, Fig. 5), e com o número 4 eram encaminhados para o sentido contrário. Os elementos de borda eram responsáveis por adicionar e retirar estes dados do pacote na interface com o usuário IP da rede. No caso dos testes com o *KeyFlow*, os elementos de borda atuavam da mesma maneira e com a mesma ocupação da tabela *hash*. Apenas os elementos centrais, que receberam a chave 3, determinavam a porta de saída pela função módulo, conforme mostra a Fig. 5.

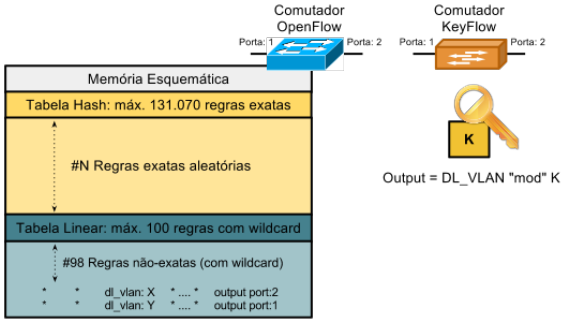


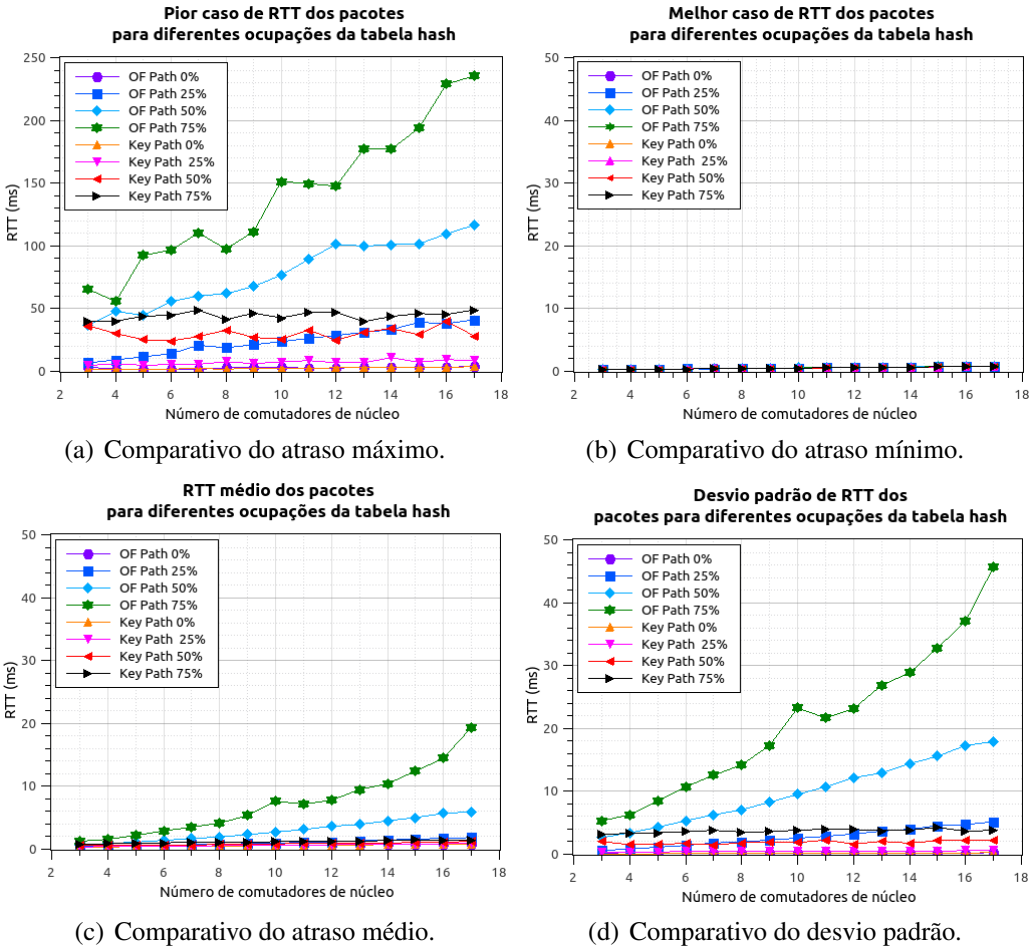
Figura 5. Informações de controle de encaminhamento em cada comutador.

4.3.1. Análise dos atrasos de envio dos pacotes

Para cada teste, foram gerados 5 mil pacotes, em intervalo constante, a partir do 'Host 1' em direção ao 'Host 2', e extraídas as estatísticas do atraso RTT de cada pacote. Foram utilizados os valores de média, de máximo (pior caso), de mínimo (melhor caso), e o desvio padrão. Para todos os testes não foram detectadas perdas de pacotes.

Verifica-se, pelo gráfico da Fig. 6(a), que há um crescimento aproximadamente linear para o RTT máximo no caminho com comutadores de núcleo com consulta em tabela. Para os caminhos com comutadores KeyFlow, não há variação do RTT máximo com o crescimento do número de elementos. O atraso de pior caso é um bom indicador da eficiência da implementação em ambiente emulado, pois ele representa o atraso sofrido pelos pacotes que sofreram a pior oferta de poder computacional do sistema de emulação. Ou seja, o pior caso, ocorreu para os pacotes que foram processados sem a utilização de cache da CPU. Nota-se que para carga de ocupação da tabela igual, ou superior, a 50%, devido à sobrecarga na memória cache L2 interna à CPU, os resultados mostram-se mais expressivos, com um crescimento acentuado no atraso máximo para os caminhos totalmente baseado em consulta em tabelas. Em contrapartida, o atraso máximo para os caminhos com o *KeyFlow* se mantém constante a partir do valor imposto pela busca nos elementos de borda. Para valores de carga inferiores a 50%, é visível um crescimento mais suave do RTT máximo para o caminho com lookup de tabela. Para tabela vazia, os cenários utilizados não apresentam diferença significativa. Isto se deve à baixa ocupação da memória cache da CPU, o que foi verificado em testes preliminares. Mesmo assim, para uma ocupação de 25%, para o caminho mais longo, o RTT máximo chega a ser 80% menor para o circuito que utiliza o núcleo com o comutador *KeyFlow*.

No gráfico da Fig. 6(b), verifica-se que o melhor caso ocorreu para todos os pacotes que se beneficiaram da memória rápida da CPU, a cache L2 de 4MB. Isto indica que



(a) Comparativo do atraso máximo.

(b) Comparativo do atraso mínimo.

(c) Comparativo do atraso médio.

(d) Comparativo do desvio padrão.

Figura 6. Comparativo entre atrasos.

quanto mais houver memórias rápidas próximas ao sistema responsável pela comutação, mais eficiente será o encaminhamento. Contudo, isto implicará significativamente nos custos e no consumo de energia dos equipamentos.

Para o RTT médio, Fig 6(c), nota-se um crescimento mais discreto para cargas iguais ou superiores a 50% e uma diferença pouco expressiva na média para cargas inferiores a 25%. Novamente, isso ocorreu devido à capacidade de armazenamento intrínseco da CPU do sistema de emulação. Até que esta memória interna fosse saturada, a maioria dos pacotes sofriam atraso mínimo devido ao cache do processador. A partir do momento da saturação deste cache, a maior parte dos pacotes teve maior atraso devido a necessidade de busca na tabela de fluxo localizada na memória RAM. Fica evidente que a proposta baseada em chaves é eficiente e escalável, pois manteve a média em valores inferiores a 3ms independentemente do tamanho do caminho e da ocupação das tabelas das bordas.

O desvio padrão, apresentado no gráfico da Fig. 6(d), representa a variabilidade do RTT de cada circuito, ou seja, temos o comparativo do desvio padrão percebido em cada experimento. Quanto menor a variabilidade da rede de pacotes, mais estas redes se assimilarão às redes de circuitos, e mais uniforme será a comutação dos quadros. Sendo assim, verifica-se que os caminhos baseado em comutadores *KeyFlow* apresentaram uma significativa redução de desvio padrão no tempo de ida e volta dos pacotes enviados. No

pior caso, com ocupação de 75% da tabela *hash*, obteve-se o desvio padrão constante inferior a 5 ms, independentemente do número de saltos do caminho. Já para o caminho baseado em busca em tabela, o pior caso apresenta uma variabilidade média superior em 40ms.

O gráfico da Fig. 7 pode ser utilizado para analisar a eficiência na redução dos atrasos dos caminhos baseados em comutadores *KeyFlow* em relação aos tempos obtidos em caminhos com os comutadores *OpenFlow* no núcleo da rede. Para o atraso mínimo, devido a capacidade computacional do emulador, a melhora notada é inferior a 5%. Pela mesma razão, percebe-se que a eficiência na redução da média do RTT cresce com o aumento da ocupação sobre a tabela *hash*, chegando próximo a 60% e a 75% no dois casos de maior ocupação da tabela. O desvio padrão apresentou melhora de 70% para os três patamares de ocupação significativa da tabela de fluxos. O RTT máximo também foi reduzido para estes casos em aproximadamente 60%.

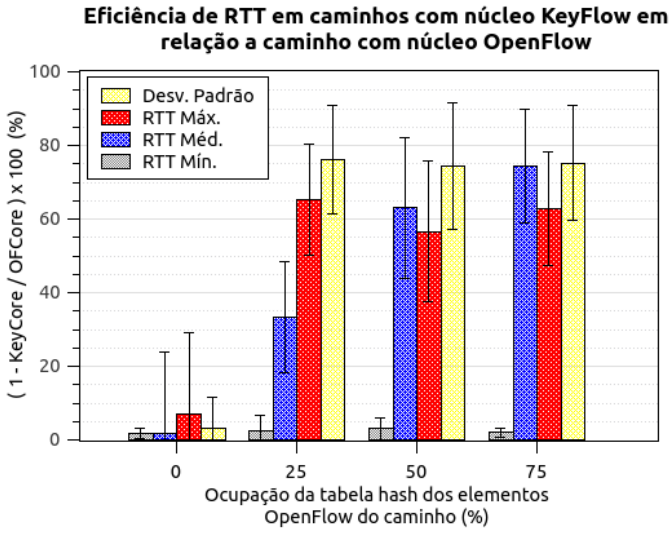


Figura 7. Eficiência na redução do RTT fim-a-fim no caminho com comutadores de núcleo baseados em chaves locais (via *KeyFlow*) em relação a comutadores com busca de tabela de estado na memória (via *OpenFlow*).

4.3.2. Análise do volume de dados nas tabelas de fluxo

Para cada experimento realizado foi criado um arquivo modelo com as regras aleatórias de ocupação da tabela *hash*. Ao se analisar o tamanho destes arquivos, chegou-se aos dados presentes na Tabela 2. A partir dela, verifica-se que uma regra exata, para a especificação *OpenFlow* utilizada possui cerca de 270 bytes (volume de bytes / número de fluxos). A chave implementada utiliza um tipo de dado inteiro de 4 bytes. Sendo assim, verifica-se que o volume total de dados necessários para definição de caminhos baseados no núcleo KeyFlow é praticamente insignificante quando comparado ao tamanho de uma regra exata.

Também pela Tabela 2, constata-se que volume de dados, a serem transferidos no plano de dados, necessário para atualização de tabelas com grande quantidade de regras

é bastante significativo. No caso de reconfigurações da rede, este volume de dados será multiplicado pelo número de comutadores relacionados aos fluxos ativos na rede. Com isso, haverá uma maior exigência sobre os recursos do controlador da rede e representará uma grande sobrecarga no plano de controle. Com a utilização do roteamento nas bordas, apenas estas deverão ser atualizadas em caso de recuperação de falhas, reduzindo assim a demanda de trabalho sobre o plano de controle.

Tipo (máx. de entradas)	Ocupação (%)	Fluxos (un)	Volume (MB)	Volume (Mb)
Linear (100)	100	100	0,004	0,032
Hash (131.070)	0	0	0	0
	25	32.767	8,7	69,6
	50	65.535	18	144
	75	98.302	26	208

Tabela 2. Relação entre ocupação da tabela *hash* e o volume de dados.

5. Conclusões e trabalhos futuros

Este trabalho apresentou uma implementação de uma proposta de comutação baseada em chaves com o objetivo de melhorar a eficiência e a escalabilidade em redes com alto volume de fluxos. Uma prova de conceito é executada, por meio de um protótipo baseado na implementação *OpenFlow 1.0*. Este protótipo utilizou o campo 'vlan id', do quadro Ethernet, para a identificação global dos fluxos e de uma chave local, e instalada em cada comutador de núcleo. Os testes de validação foram criados sobre o ambiente de emulação, com a criação um caminho virtual entre duas estações. Buscou-se, assim, comparar o tempo de encaminhamento para elementos baseados no *KeyFlow*, em relação ao comutador *OpenFlow* convencional.

Os resultados mostraram que o *KeyFlow* apresenta uma significativa redução na variabilidade da entrega dos quadros em aproximadamente 70%. Além disso, a percepção de pior caso de RTT, na visão fim a fim, apresentou uma curva de tendência bem comportada em relação ao aumento do número de saltos e da carga de fluxos ativos na tabela de encaminhamento do comutador.

O *KeyFlow* também mostrou-se como uma alternativa em relação aos comutadores convencionais com memórias rápidas associadas à interface de processamento. É importante ressaltar que estas memórias possuem um alto custo e um elevado consumo de energia. Sendo assim, o *KeyFlow* pode viabilizar a implementação em comutadores com alto desempenho e baixo custo.

Por fim, verificou-se que a identificação dos fluxos pela combinação rótulo global/chave local reduz drasticamente o volume de dados necessários para ativação de fluxos nos comutadores de uma rede *OpenFlow*. Isso facilita também a reconfiguração da rede em caso de falhas, além de reduzir significativamente o tráfego no canal de controle.

Como trabalho futuro, espera-se a implementação do *KeyFlow* em outros comutadores de alto desempenho para realizar os testes de vazão, como em NetFPGA [Naous et al. 2008], e também em soluções embarcadas em dispositivos com menor capacidade de processamento, para validação da solução em ambiente real.

Referências

Bianco, A., Birke, R., Giraudo, L., and Palacin, M. (2010). Openflow switching: Data plane performance. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5.

Casado, M., Koponen, T., Shenker, S., and Tootoonchian, A. (2012). Fabric: A retrospective on envolving sdn. In *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, Helsink. ACM SIGCOMM.

Heller, B., Pfaff, B., Heller, B., Talayco, D., Erickson, D., Gibb, G., Appenzeller, G., Tourrilhes, J., Pettit, J., Yap, K., Casado, M., Kobayashi, M., McKeown, N., Balland, P., Price, R., Sherwood, R., and Yiakoumis, Y. (2009). In *OpenFlow Switch Specification, Version 1.0.0*.

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.

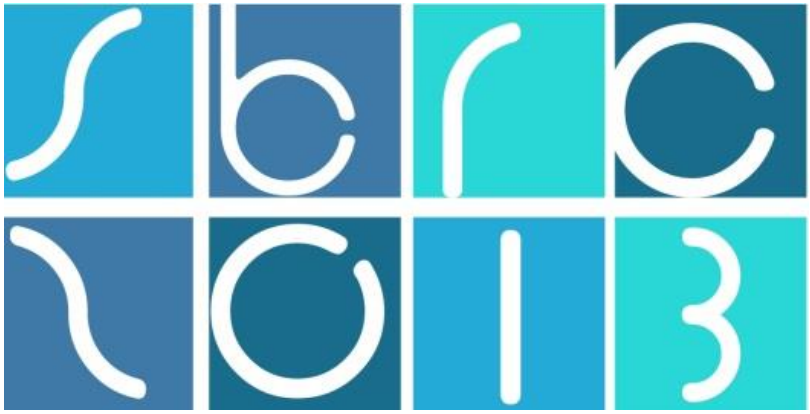
Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., Curtis, A. R., and Banerjee, S. (2010). Devoflow: cost-effective flow management for high performance enterprise networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 1:1–1:6, New York, NY, USA. ACM.

Naous, J., Erickson, D., Covington, G., A., and Appenzeller, G., M. N. (2008). Implementing an openflow switch on the netfpga platform. In *ANCS '08 Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 1–9.

Rothenberg, C. E., Macapuna, C. A. B., Verdi, F. L., Magalhães, M. F., and Zahemszky, A. (2010). Data center networking with in-packet bloom filters. In *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Rotsos, C., Sarrar, N., Uhlig, S., Sherwood, R., and Moore, A. W. (2012). Oflops: an open framework for openflow switch evaluation. In *Proceedings of the 13th international conference on Passive and Active Measurement*, PAM'12, pages 85–95, Berlin, Heidelberg. Springer-Verlag.

Wessing, H., Christiansen, H., Fjelde, T., and Dittmann, L. (2002). Novel scheme for packet forwarding without header modifications in optical networks. *Lightwave Technology, Journal of*, 20(8):1277 – 1283.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 22

Computação nas Nuvens

Um Middleware para Encenação Automatizada de Coreografias de Serviços Web em Ambientes de Computação em Nuvem

Leonardo Leite, Nelson Lago, Marco A. Gerosa, Fabio Kon [1]

[1]Departamento de Ciência da Computação
IME - Universidade de São Paulo (USP)

{leofl,lago,gerosa,fabio.kon}@ime.usp.br

Abstract. *Automated deployment is an important issue in large-scale distributed systems, such as web services choreographies that implement business processes encompassing multiple organizations. In this paper, we present the CHOReOS Enactment Engine, a novel extensible open source middleware system that provides a platform for automation of the distributed deployment of web service choreographies in cloud computing environments. We evaluated the choreographies enactment time experimentally, varying the amount of services to be deployed and the amount of virtual machines available to the deployment process. The CHOReOS Enactment Engine enabled the automated deployment of service compositions on the cloud in an automated and reproducible way.*

Resumo. *A implantação automatizada é uma importante necessidade em sistemas distribuídos de grande escala, como é o caso de coreografias de serviços web, que implementam processos de negócios envolvendo várias organizações. Neste artigo, apresentamos o CHOReOS Enactment Engine, um novo sistema de middleware extensível que fornece uma plataforma para a implantação distribuída e automatizada de coreografias de serviços web em ambientes de computação em nuvem. Para avaliar a eficácia e escalabilidade do sistema, medimos experimentalmente o tempo de encenação de coreografias, tendo como carga a quantidade de serviços a serem implantados, e como recursos a quantidade de máquinas virtuais disponíveis para o processo de implantação. Nosso sistema de middleware está disponível como software livre e possibilita a implantação de complexas composições de serviços na nuvem de forma automatizada e reprodutível.*

1. Introdução

Profissionais da indústria já reconhecem a necessidade de processos totalmente automatizados para a implantação de sistemas [Humble and Farley 2011]. No entanto, muitas organizações ainda realizam a implantação de seus sistemas de forma manual, tornando esse processo moroso, propenso a erros e não-reprodutível [Dolstra et al. 2005]. O problema se agrava na implantação de sistemas distribuídos, pois o esforço de implantação cresce com a quantidade de nós do sistema.

Sistemas distribuídos estão migrando para ambientes de nuvem, nos quais são compostos e mantidos de forma descentralizada por várias organiza-

ções [Steen et al. 2012]. A computação em nuvem possibilita o acesso a um conjunto compartilhado de recursos computacionais que podem ser providos rapidamente [Mell and Grance 2011], favorecendo a criação de um processo de implantação totalmente automatizado. O modelo de Infraestrutura como Serviço (IaaS) fornece acesso a recursos virtualizados, como máquinas virtuais, de forma programática. Em contrapartida, no modelo de Plataforma como Serviço (PaaS), como o adotado pelo Google App Engine[1], os desenvolvedores da aplicação não se preocupam diretamente com a gerência dos recursos virtualizados ou com a configuração dos ambientes nos quais a aplicação será implantada. A nuvem também pode ser púlica, com clientes externos, como no caso da nuvem da Amazon[2], ou privada, com clientes internos, situação na qual a organização utilizam ambientes baseados em middleware como o OpenStack[3] [Zhang et al. 2010].

Serviços web são componentes que se comunicam pela rede por meio da utilização de protocolos baseados em padrões da Web [W3C 2004], e que podem ser compostos de forma a se obter um sistema mais complexo, mas com baixo acoplamento entre seus componentes. Em particular, coreografias são composições que implementam processos de negócios distribuídos, normalmente entre organizações, de maneira a diminuir o número de mensagens trocadas e distribuir a lógica do negócio entre as organizações envolvidas, dispensando o uso de coordenadores centralizados, uma vez que cada serviço "sabe" quando executar suas operações e com quais outros serviços interagir [Barker et al. 2009]. Um exemplo de notação para a especificação de coreografias é o BPMN2 [OMG 2011]. Para coreografias de serviços web, o processo de implantação deve ser distribuído. A *encenação* de uma coreografia deve ser coordenada, pois além de implantados no ambiente distribuído, os serviços da coreografia precisam conhecer a localização dos outros serviços, informação disponível apenas em tempo de implantação.

Neste artigo, apresentamos o CHOReOS Enactment Engine, um sistema de middleware que fornece uma plataforma como serviço para a implantação distribuída e automatizada de composições de serviços web, com ênfase em coreografias, em ambientes de computação em nuvem. O CHOReOS Enactment Engine é software livre[4] e desenvolvido no contexto do projeto CHOReOS, financiado pela Comissão Europeia com o objetivo de possibilitar a utilização de coreografias de serviços web em cenários de escala ultra grande. Nosso trabalho se diferencia dos trabalhos relacionados ao explorar como o ambiente de computação em nuvem traz benefícios ao processo de implantação, bem como ao considerar as restrições que esses ambientes impõem, como a falta de previsibilidade dos endereços das máquinas em tempo de configuração do serviço.

O CHOReOS Enactment Engine recebe uma especificação declarativa da coreografia, realiza a encenação da coreografia e devolve ao cliente informações sobre a localização de cada serviço implantado. A especificação da coreografia é uma descrição arquitetural, que deve ser criada pelo cliente da operação de encenação. Isso pode ser feito manualmente, mas também por processos automatizados por meio da conversão de modelos gráficos de mais alto nível, como o BPMN2[5]. Durante a implantação, o middleware

[1] `developers.google.com/appengine`

[2] `aws.amazon.com`

[3] `www.openstack.org`

[4] Código-fonte disponível em `github.com/choreos/choreos_middleware`.

[5] Essa é a abordagem de utilização do CHOReOS Enactment Engine implementada no contexto do projeto CHOReOS [Ben Hamida et al. 2012].

também realiza a troca de localizações dos nós da coreografia e configura dinamicamente os serviços para que a coreografia possa ser encenada.

Soluções comerciais de Plataforma como Serviço normalmente impõem restrições severas quanto às possíveis tecnologias a serem usadas por seus usuários no desenvolvimento de aplicações. Para amenizar esse problema, nosso sistema possui pontos de extensão que possibilitam a adequação do middleware a novas tecnologias. Esses pontos de extensão não dizem respeito apenas às tecnologias de desenvolvimento de serviços web, mas também à interação com serviços de infraestrutura, possibilitando que cada organização escolha sua plataforma de computação em nuvem.

Um aumento significativo na carga de um sistema normalmente tem grande impacto no seu tempo de execução. Consideramos um sistema escalável se for possível amenizar esse efeito por meio de um aumento proporcional nos recursos utilizados pelo sistema, ou seja, se um aumento nos recursos do sistema implicar em um aumento diretamente proporcional na sua capacidade de processamento [Law 1998]. Assim, para avaliar a eficácia e escalabilidade do CHOReOS Enactment Engine, medimos experimentalmente o tempo de implantação de coreografias variando a carga – quantidade de serviços – e os recursos disponíveis – quantidade de máquinas virtuais disponíveis para o processo de implantação.

Na próxima seção descrevemos alguns dos principais trabalhos relacionados à nossa abordagem. A Seção 3 apresenta a solução proposta e sua implementação. A Seção 4 apresenta resultados experimentais e, na Seção 5, descrevemos nossas conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Um dos aspectos fundamentais sobre processos de implantação automatizados é a linguagem para sua configuração. A linguagem pode ser procedimental [Dolstra et al. 2005] ou declarativa [Magee and Kramer 1996, Balter et al. 1998]. Outro aspecto relevante é o escalonamento de recursos computacionais para os serviços a serem implantados, o que pode ser feito, por exemplo, com o auxílio de sistemas de computação em grade [Watson et al. 2006].

Antes de instalar um serviço é preciso configurar adequadamente o sistema operacional e a plataforma na qual o serviço será implantado. No uso de ferramentas como Chef[6], Capistrano[7] e Nix [Dolstra et al. 2005], os usuários escrevem *scripts* que implementam o processo de configuração do ambiente e a implantação do serviço. No caso do Chef, um script configura a máquina na qual o serviço é instalado, enquanto que o Capistrano possibilita a coordenação da implantação de serviços em diferentes nós. Com as expressões Nix, é possível também unificar a especificação da implantação com o *build* da aplicação em um único script, possibilitando a edição parametrizada de arquivos de configuração da aplicação em função do local de implantação.

A abordagem procedimental possibilita ao usuário especificar a implantação de vários tipos de sistemas, mas normalmente requer especialização de seus usuários, pois vários detalhes do processo devem ser especificados. Esses *scripts* de implantação tam-

[6] `www.opscode.com/chef`
[7] `github.com/capistrano`

bém precisam ser desenvolvidos com o mesmo rigor do código da aplicação, inclusive com o uso de testes automatizados [Humble and Farley 2011]. Caso contrário, o processo de implantação torna-se pouco robusto e não confiável. Uma alternativa que evita essa sobrecarga no processo de desenvolvimento é o uso de sistemas especializados na implantação de determinados tipos de aplicações e que recebam, como entrada, uma simples especificação declarativa do sistema.

Um exemplo de abordagem declarativa é encontrado no uso de Linguagens de Descrição Arquitetural (ADLs), como a Darwin [Magee and Kramer 1996]. A linguagem Darwin se foca nos aspectos estruturais de sistemas distribuídos, descrevendo a conexão entre os módulos do sistema, mas sem descrever implementações ou sequências de interações entre os módulos. Em nosso trabalho, também descrevemos o sistema a ser implantado por meio de sua descrição estrutural, uma vez que é esse o aspecto necessário para que se possa automatizar o processo de implantação.

Regis, o ambiente de execução da linguagem Darwin [Magee et al. 1994], possui duas políticas de distribuição de programas por estações de trabalho. A primeira é o mapeamento definido pelo usuário de forma estática, abordagem não apropriada para ambientes de computação em nuvem. A segunda opção de política é a alocação automática em função da carga na CPU das estações de trabalho, não havendo flexibilidade para a consideração de outros recursos, como espaço em disco ou memória, por exemplo. Uma similaridade entre Regis e o CHOReOS Enactment Engine é o uso do middleware para o envio de mensagens contendo referências remotas dos componentes implantados para que eles possam estabelecer ligações dinâmicas entre si.

Olan [Balter et al. 1998] é um ambiente para a descrição, configuração e implantação de aplicações distribuídas em ambientes heterogêneos, e que também utiliza uma ADL própria. Baseando-se na entrada descrita na ADL, Olan gera scripts de Configuração de Máquina, que definem a execução do processo de implantação dos componentes no ambiente distribuído e do ajuste dos canais de comunicação entre esses componentes. A abordagem de gerar um script de configuração a partir de uma especificação declarativa é também implementada pelo CHOReOS Enactment Engine. A ADL de Olan também possibilita a especificação de restrições sobre a localização da implantação do componente, porém sem flexibilidade para a adoção de estratégias dinâmicas de alocação de nós.

O estudo de Quéma et al. [Quéma et al. 2004] realiza avaliações empíricas sobre desempenho e escalabilidade do processo de implantação proposto, que é executado de forma distribuída por agentes que se comunicam de forma assíncrona e hierárquica de acordo com a estrutura da composição sendo implantada, descrita por uma ADL. Essa estrutura hierárquica, no entanto, é apenas um caso particular das possibilidades na topologia de uma coreografia de serviços, o que impossibilita que essa solução seja diretamente adotada em nosso contexto. A avaliação de escalabilidade resulta em um aumento aproximadamente proporcional no tempo de implantação com o aumento proporcional da quantidade de componentes e da quantidade de máquinas disponíveis.

Os trabalhos descritos até aqui apresentam abordagens simples para o problema da distribuição dos componentes implantados pelas máquinas disponíveis. Já o trabalho de Watson et al., apresenta uma abordagem mais completa para esse problema com o uso de grades computacionais [Watson et al. 2006]. O foco da solução apresentada está

em escolher dinamicamente o provedor de infraestrutura e a máquina em que um serviço web deve ser implantado considerando os requisitos não-funcionais do serviço web. Isso é realizado não somente para a primeira implantação do serviço web, mas também para as replicações que ocorrem quando as instâncias existentes não conseguem mais atender aos requisitos não-funcionais. Uma desvantagem dessa abordagem é a carga adicional gerada pela análise dos requisitos não-funcionais a cada troca de mensagens efetuada pelos serviços implantados. Embora o trabalho de Watson et al. avance na problemática da distribuição dos serviços, nenhum dos trabalhos analisados considera as potencialidades e desafios dos ambientes de computação em nuvem, que oferecem serviços de infraestrutura para a gerência de recursos virtualizados.

3. CHOReOS Enactment Engine

Nesta seção descreveremos o funcionamento do CHOReOS Enactment Engine, abordando sua interface, arquitetura e pontos de extensão.

3.1. Interface

As funcionalidades do CHOReOS Enactment Engine são expostas como serviços REST, sendo a encenação de coreografias a principal operação disponível. Para executar essa operação, o middleware recebe uma descrição arquitetural da coreografia em formato XML, estruturado de acordo com o modelo de classes apresentado na Figura 1. A descrição de uma coreografia consiste numa lista de descrições de seus serviços, sendo cada serviço modelado de acordo com a classe ChorServiceSpec. Com base nessa descrição, o CHOReOS Enactment Engine implanta os artefatos em diversos nós nos ambientes de nuvem disponíveis. O cliente da operação de encenação recebe como resposta dados que modelam os serviços implantados de acordo com a classe Service, estrutura que informa em que nó cada serviço da coreografia foi implantado, assim como a URI de acesso ao serviço.

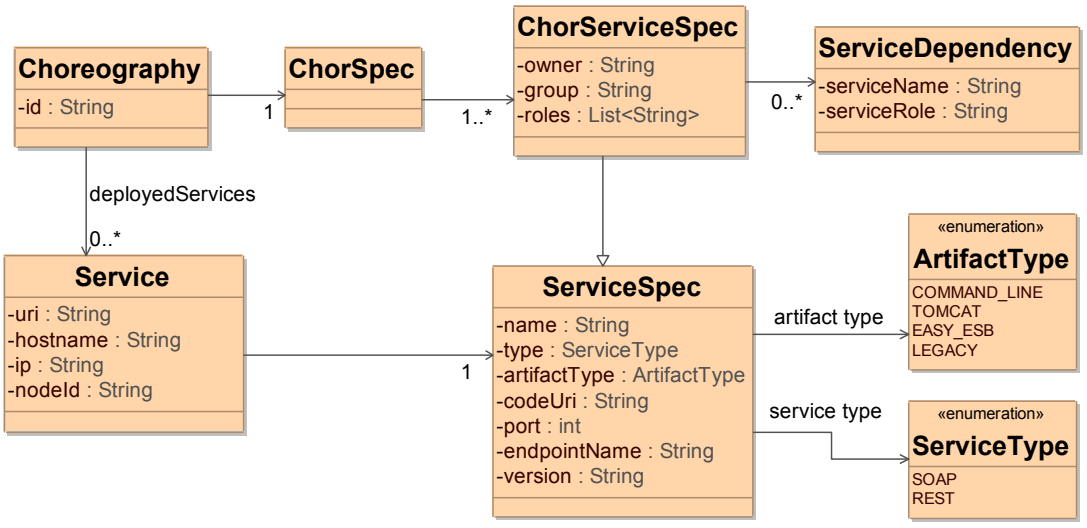


Figura 1. Estrutura da descrição arquitetural de uma coreografia.

O middleware implanta cada serviço acessando seu artefato implantável, identificado pelo atributo `codeUri`, e o instalando de acordo com procedimentos relacionados

ao tipo do artefato, identificado pelo atributo `artifactType`. Caso o tipo do artefato seja `COMMAND_LINE`, por exemplo, o artefato deve ser um arquivo jar que possa ser executado com o comando `java -jar`. Para que o middleware identifique a URI de acesso ao serviço, ainda é preciso informar na especificação do serviço os atributos `endpointName` e `port`, que são partes constituintes dessa URI.

Cada serviço na coreografia pode consumir operações de outros serviços da coreografia. Para isso, é preciso que cada serviço consumidor conheça a URI de acesso a cada serviço provedor do qual depende. Em uma implantação em um ambiente de nuvem, em que máquinas virtuais podem ser criadas no momento da encenação, esse endereço será conhecido apenas após a implantação de cada serviço. Assim, para possibilitar a ligação dinâmica entre os serviços da coreografia, cada serviço consumidor deve implementar uma operação denominada `setInvocationAddress`, que recebe a URI de um serviço provedor que pode desempenhar o papel de uma das dependências do serviço consumidor. A relação de quais serviços satisfazem as necessidades de outros serviços é definida na especificação da coreografia, utilizando a lista de elementos do tipo ServiceDepency, pertencentes a cada especificação de serviço.

Uma descrição completa da estrutura da especificação da coreografia, incluindo o *schema* XML correspondente e exemplos de descrições de coreografia em XML, está disponível em `http://ccsl.ime.usp.br/baile/files/ee_rest_api.pdf`. Esse documento contém também a descrição da interface REST do middleware.

3.2. Arquitetura

O CHOReOS Enactment Engine é composto por componentes que são instanciados nas infraestruturas das organizações participantes da coreografia ou em nuvens públicas. A relação de uso entre esses componentes é exibida na Figura 2. Os componentes Choreography Deployer e Deployment Manager são fornecidos pelo próprio middleware. Os componentes Chef Client e Chef Server são partes do Chef, uma ferramenta de código aberto para a gerência de configuração de software. Por fim, o componente Cloud Gateway é algum serviço de infraestrutura para a gerência de máquinas virtuais, como, por exemplo, o serviço Amazon EC2. A encenação de uma coreografia requer uma instância do componente Choreography Deployer e que cada organização possua, em sua infraestrutura, uma ou mais instâncias dos componentes Cloud Gateway, Chef Server e Deployment Manager. A seguir, descreveremos brevemente cada um desses componentes.

Cloud Gateway. Cria e destrói máquinas virtuais em uma infraestrutura de computação em nuvem. Neste contexto, essas máquinas virtuais são também chamadas *nós*. Esse componente é utilizado pelo Deployment Manager, que decide quando criar ou destruir os nós. Uma instância do Deployment Manager deve ser configurada de acordo com o Cloud Gateway a ser utilizado. Atualmente, podem ser utilizados como Cloud Gateway o serviço EC2 dos Web Services da Amazon, ou uma instalação do OpenStack. Outras tecnologias de nuvem ou virtualização podem ser acrescentadas mediante extensão do Deployment Manager.

Chef Server. Em um sistema gerenciado pelo Chef, as configurações de software de um nó são especificadas através de scripts denominados "receitas". As receitas podem especificar configurações do sistema operacional (p.ex., instalar o Java), instalação e configuração de middleware (p.ex., instalar o Tomcat), assim como a instalação de um serviço final

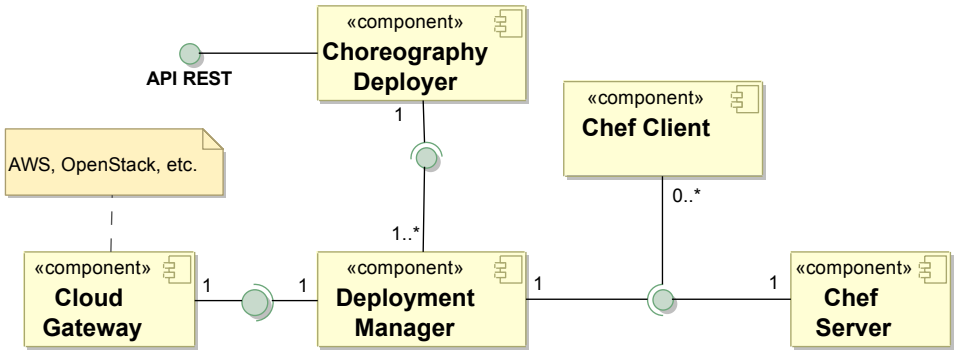


Figura 2. Arquitetura do CHOReOS Enactment Engine.

implantado sobre o middleware já configurado (p.ex., um serviço web). Essas receitas e as associações entre receitas e nós são armazenadas no Chef Server.

Chef Client. Em cada nó gerenciado pelo Chef, é instalado um programa denominado Chef Client, que utiliza a API REST do Chef Server para obter as receitas relacionadas ao nó e assim atualizar suas configurações de software.

Deployment Manager. Implanta os serviços em um ambiente de nuvem associado a uma organização. Através de uma operação REST, o Deployment Manager recebe a especificação de um serviço e seleciona um nó adequado para sua implantação, possivelmente considerando os requisitos não-funcionais do serviço. A especificação do serviço é descrita em formato XML e modelada de acordo com a classe ServiceSpec (Figura 1). O atributo `artifactType` determinará o procedimento para converter a especificação declarativa do serviço em uma receita Chef que implemente seu processo de instalação. Após gerar a receita, o Deployment Manager a armazena no Chef Server e, também utilizando o Chef, cria uma associação entre essa receita e o nó selecionado. Utilizando outra operação REST do Deployment Manager pode-se requisitar que um determinado nó seja atualizado de acordo com as receitas a ele relacionadas. Essa divisão do processo em duas chamadas ao Deployment Manager é adotada para evitar a sobrecarga de execuções desnecessárias do Chef Client.

Choreography Deployer. Expõe uma API REST de operações que automatizam a implantação de coreografias de serviços web. O cliente do Choreography Deployer fornece a especificação declarativa da coreografia (Figura 1). Baseado nessa especificação, o Choreography Deployer coordena invocações aos Deployment Managers implantados nas diferentes organizações que participam da coreografia. Após a implantação dos serviços, para cada par de serviços que se comunicam na coreografia, o Choreography Deployer informa ao serviço consumidor a localização do serviço provedor.

O diagrama de sequência apresentado pela Figura 3 ilustra o processo de encenação de uma pequena coreografia de dois serviços, na qual o serviço A é consumidor do serviço B. O diagrama de sequência ajuda a evidenciar detalhes da interação entre os componentes do sistema.

O processo de implantação se inicia com duas requisições ao Choreography De-

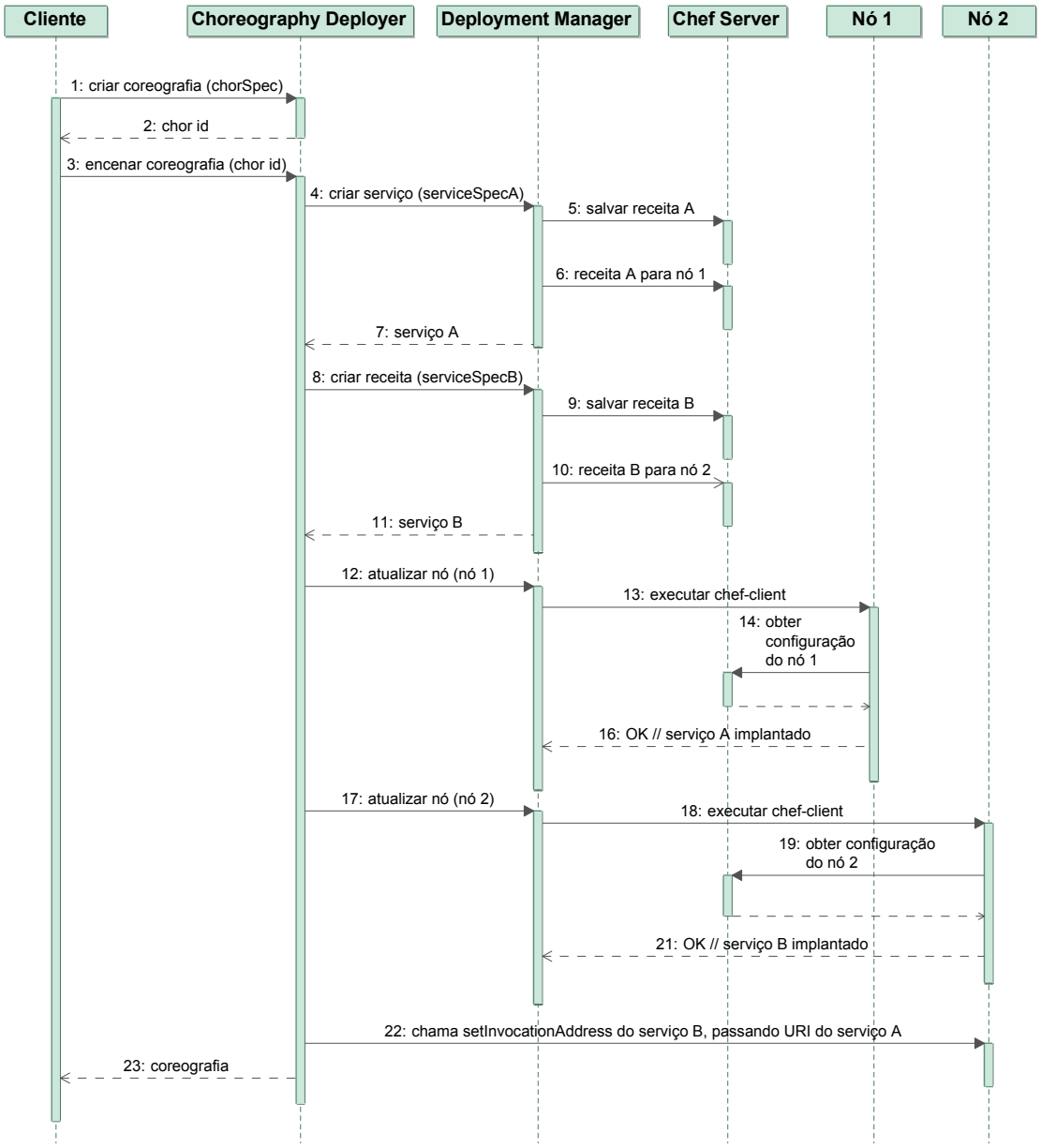


Figura 3. Exemplo de encenação de uma coreografia de dois serviços.

ployer: uma para a criação do recurso[8] *coreografia* associado à especificação fornecida pelo cliente e outra para a encenação da coreografia. Para cada especificação de serviço contido na especificação da coreografia, o Choreography Deployer requisita a criação de um recurso *serviço* ao Deployment Manager. Para isso, o Deployment Manager cria uma receita Chef que automatiza a instalação do serviço, armazena-a no Chef Server e associa a receita a algum nó por ele selecionado. O Deployment Manager retorna ao Choreography Deployer o recurso *serviço*, contendo a URI de acesso a esse serviço e uma referência ao nó selecionado. Em seguida, o Choreography Deployer requisita ao Deployment Manager a atualização de todos os nós selecionados. Quando o Deployment Manager re-

[8]Usamos aqui o termo "recurso" em seu sentido usual em arquiteturas REST.

cebe um pedido de atualização de nó, ele se conecta por SSH a esse nó e executa o Chef Client, que se comunica com o Chef Server para obter as receitas associadas àquele nó e instalar os serviços com base nas receitas obtidas. O Choreography Deployer ainda invoca a operação `setInvocationAddress` do serviço B, informando ao serviço B a URI de acesso do serviço A. Finalmente, o Choreography Deployer devolve ao cliente a representação da coreografia, contendo informações da implantação, especialmente as URIs dos serviços em execução.

Uma observação sobre o diagrama da Figura 3 é que todas as operação foram dispostas de forma sequencial para facilitar a leitura do diagrama, mas o Choreography Deployer realiza as chamadas de criação dos serviços em paralelo, assim como as chamadas de atualização dos nós e as invocações às operações `setInvocationAddress`.

3.3. Pontos de Extensão

Para possibilitar uma maior flexibilidade aos usuários do CHOReOS Enactment Engine em relação às possíveis escolhas de tecnologias adotadas, nosso middleware oferece os pontos de extensão seguintes.

CloudProvider. A implementação dessa interface permite a utilização de outros serviços de infraestrutura para a gerência de máquinas virtuais. Um exemplo seria codificar uma implementação VirtualBoxCloudProvider que utilizasse os recursos do VirtualBox[9] para criar máquinas virtuais no próprio ambiente do desenvolvedor.

NodeSelector. A implementação dessa interface define uma nova política de alocação de serviços em nós da nuvem, que pode levar em conta os requisitos não funcionais do serviço e propriedades dos nós à disposição. Um exemplo simples é a implementação AlwaysCreateNodeSelector, que cria um novo nó para cada requisição realizada.

RecipeBuilder. A implementação dessa interface possibilita o suporte a novos tipos de artefatos implantáveis. Um exemplo seria a codificação de uma implementação PythonRecipeBuilder, que gerasse receitas de serviços web codificados em Python e acessíveis pelo servidor web Apache.

ContextSender. A implementação dessa interface possibilita que o Enactment Engine ofereça suporte a novos tipos de serviços ao se definir um procedimento para a invocação da operação `setInvocationAddress`. Um exemplo de nova implementação seria o JMSContextSender, que definiria uma convenção para a construção da operação `setInvocationAddress` de serviços JMS participantes de uma coreografia.

4. Avaliação Experimental

Quando implantadas continuamente nos ambientes de teste e homologação, aplicações oferecem retorno rápido para os desenvolvedores sobre os efeitos de suas modificações e sobre eventuais defeitos no sistema [Humble and Farley 2011]. Quanto mais demorado for o processo de implantação, mais demorado será o *feedback* aos desenvolvedores e, em alguns casos, a demora nesse processo desmotiva sua execução frequente. Por esses motivos, um dos objetivos do CHOReOS Enactment Engine é oferecer um processo de implantação escalável, no qual o aumento na disponibilidade de recursos oferecidos (tais

[9] www.virtualbox.org

como máquinas virtuais) deve possibilitar que um aumento na carga do processo de implantação (por exemplo, a quantidade de serviços a serem implantados) não provoque um grande aumento no tempo de execução.

Nesta seção, apresentamos a avaliação de escalabilidade do CHOReOS Enactment Engine. A unidade de carga do sistema é a quantidade de coreografias a serem implantadas, enquanto que a unidade de recursos fornecidos ao sistema é a quantidade de máquinas virtuais alocados no ambiente de computação em nuvem. Cada coreografia possui dois serviços: o serviço `TravelAgency`, que é invocado pelo cliente da coreografia, e o serviço `Airline`, que é invocado pelo serviço `TravelAgency`.

Definimos uma *execução* do experimento como o processo de encenações simultâneas de N coreografias, sendo os valores utilizados para N: 1, 5, 10, 15, 20, 25. Cada execução é composta por três *fases*: 1) a criação de $2 * N$ nós no ambiente de computação em nuvem (um nó por serviço), 2) a implantação das N coreografias nos nós criados na fase anterior e 3) a invocação das coreografias para verificação do funcionamento adequado do processo de negócio. Para cada fase em cada execução, registramos três valores: o *tempo total* (t) de duração da fase, o *tempo médio* (m) gasto por cada coreografia naquela fase e o *desvio padrão* (s) dos valores usados no cálculo do tempo médio. Tomando como exemplo a fase de implantação de uma execução com $N = 5$, o tempo médio é a média aritmética dos tempos de implantação das 5 coreografias, o desvio padrão é o desvio padrão dos tempos de implantação e o tempo total é o tempo gasto para que as coreografias sejam implantadas. Para cada valor de N, coletamos uma amostra de 10 execuções sem falhas[10], desconsiderando assim execuções com falhas provenientes do provedor de IaaS.

O Cloud Gateway utilizado foi o serviço EC2 da Amazon e a política de alocação de nós foi a de alocar um nó por serviço. As execuções foram realizadas com os componentes Choreography Deployer e Deployment Manager instanciados em um Mac Book Pro 8 (8GB de RAM, processador Intel Core i7 com 2,7GHz) utilizando o sistema operacional GNU/Linux (kernel 3.6.7) e a JVM OpenJDK 6. O Chef Server utilizado estava instalado em um servidor localizado em nossa universidade, de onde as requisições para as execuções foram originadas. A versão do código-fonte utilizada nas execuções foi marcada com a tag `sbrc2013` em nosso repositório público[11].

Os resultados experimentais da fase de criação de nós são apresentados na Tabela 1, os resultados da fase de implantação na Tabela 2, os resultados da fase de invocações na Tabela 3 e os tempos totais das execuções na Tabela 4. As tabelas com dados de fases (Tabelas 1, 2 e 3) mostram os seguintes valores para cada valor de N: média aritmética dos tempos médios ($\overline{m}$); mediana dos tempos médios ($m_{1/2}$); média aritmética dos desvios padrões ($\overline{s}$); média aritmética dos tempos totais ($\overline{t}$); mediana dos tempos totais ($t_{1/2}$); e desvio padrão dos tempos totais (s_t).

O principal resultado observado é que um aumento de 25 vezes no número de coreografias implantadas apenas dobrou o tempo total das execuções, subindo o tempo

[10]Os dados brutos coletados durante as execuções se encontram em
`http://ccsl.ime.usp.br/baile/files/ee_scalability_sbrc2013.zip`

[11]O código dessa versão pode ser baixado diretamente de
`http://github.com/choreos/choreos_middleware/archive/sbrc2013.zip`

Tabela 1. Tempos da fase de criação dos nós (s)

	m : tempo médio da criação de um nó s : desvio padrão do tempo de criação de nó			t : tempo total da fase de criação de nós		
N	$\overline{m}$	$m_{1/2}$	$\overline{s}$	$\bar{t}$	$t_{1/2}$	s_t
1	103.5	114.0	6.5	120.1	116.5	11.9
5	108.1	107.9	8.0	127.9	125.0	11.4
10	128.0	126.1	18.3	185.3	154.9	95.0
15	141.6	140.5	16.2	180.2	175.7	11.7
20	146.4	142.6	24.3	239.8	208.4	69.7
25	170.0	173.1	26.3	238.0	234.7	16.2

Tabela 2. Tempos da fase de implantação (s)

	m : tempo médio da implantação de uma coreografia s : desvio padrão do tempo de implantação			t : tempo total da fase de implantações		
N	$\overline{m}$	$m_{1/2}$	$\overline{s}$	$\bar{t}$	$t_{1/2}$	s_t
1	155.6	154.2	0.0	155.6	154.3	11.4
5	157.2	159.5	9.2	170.0	169.0	13.7
10	189.1	187.4	12.1	211.1	208.1	10.4
15	217.3	217.8	11.9	242.9	239.3	12.5
20	243.0	243.1	15.6	275.7	274.9	11.7
25	271.3	270.6	18.6	313.7	308.6	21.8

Tabela 3. Tempos da fase de invocação (s)

	m : tempo médio de uma invocação s : desvio padrão do tempo de invocação			t : tempo total da fase de invocações		
N	$\overline{m}$	$m_{1/2}$	$\overline{s}$	$\bar{t}$	$t_{1/2}$	s_t
1	0.80	0.80	0.00	0.80	0.80	0.05
5	0.82	0.81	0.04	0.88	0.88	0.05
10	0.88	0.88	0.06	1.01	0.98	0.06
15	0.87	0.87	0.06	1.02	0.98	0.13
20	0.89	0.88	0.07	1.08	1.01	0.13
25	0.92	0.92	0.14	1.49	1.42	0.48

Tabela 4. Tempos totais das execuções (s)

	t : tempo total de execução		
N	$\bar{t}$	$t_{1/2}$	s_t
1	276.5	281.3	17.2
5	298.7	296.4	20.1
10	397.4	365.9	92.9
15	424.1	419.0	22.5
20	516.6	486.7	75.0
25	553.1	540.1	34.8

de encenação de 4,6 minutos em média para apenas 9,2 minutos em média, o que consideramos um resultado muito satisfatório. Podemos também observar o crescimento dos tempos de execuções e de cada uma de suas fases nas Figuras 4 e 5. É interessante notar que a fase de invocação consome aproximadamente apenas 1 segundo, com uma variância muito pequena. Esse comportamento para a fase de invocação era esperado, considerando que os serviços utilizados são bem leves, que as máquinas virtuais criadas são equivalentes em suas capacidades computacionais e que cada nó recebeu apenas um serviço.

Quando verificamos a evolução da encenação de 1 coreografia até 25 coreografias, notamos que o aumento na média do tempo de criação de nós foi de 64%, enquanto que o aumento na média do tempo total da fase de criação de nós foi de 98%. A média do tempo de implantação das coreografias subiu 75% e o aumento na média do tempo da fase de implantação de coreografias foi de 102%. O aumento maior nos tempos totais das fases se explica pela natureza dos dados coletados: para a fase de criação dos nós, verificamos que a maior parte dos nós criados segue um bom comportamento mas, a cada execução, é comum que pelo menos um dos nós da Amazon EC2 demore bem mais para ficar pronto. Às vezes também ocorre de algum nó não ficar pronto para uso, situação na qual nosso middleware requisita a alocação de um novo nó. Como o tempo total da fase de criação de nós é próximo ao tempo mais demorado de criação de um nó, fica claro que eventos

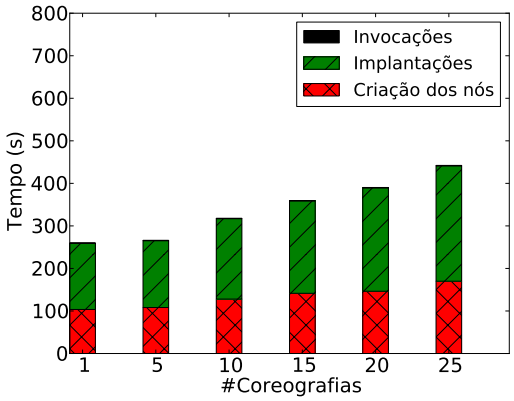


Figura 4. Média dos tempos de encenação das coreografias

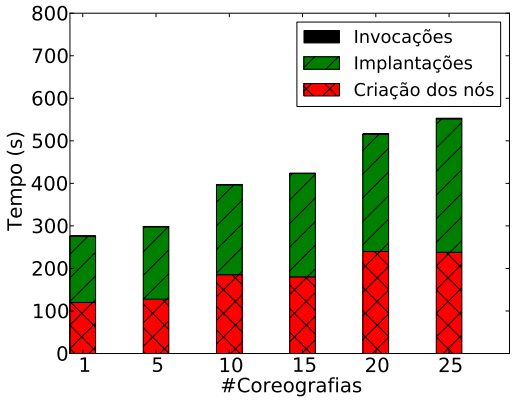


Figura 5. Média dos tempos totais de encenação das coreografias

esporádicos de nós muito demorados afetam mais contundentemente o tempo total da fase do que o tempo médio de criação de nós. Fenômeno similar ocorre, em menor escala, na fase de implantação da coreografia.

A presença de eventos esporádicos que são mais demorados explica também o fato de que as medianas para os tempos totais de fases sejam menores do que as médias correspondentes. Portanto, a mediana representa melhor do que a média o valor esperado de um evento como a criação de um nó. Por outro lado, o fato de a mediana dos tempos médios ser bem próxima à média correspondente sugere que os eventos que "distorcem" a média dos tempos totais são de fato bastante raros. Outra conclusão que podemos tirar sobre a presença desses eventos esporádicos é a de que não podemos assumir uma distribuição normal para eventos aleatórios de criação de nós e de implantações de coreografias. Esses eventos esporádicos também fazem com que os valores representando os desvios dos tempos totais das fases sejam maiores do que os desvios dos tempos médios.

Outra observação relevante é que os desvios padrões da criação de nós são maiores do que os desvios padrões da implantação de coreografias. Pode-se chegar à mesma conclusão com o coeficiente de variação (razão entre desvio padrão e a média) no lugar do desvio padrão, já que os desvios na criação de nós são maiores do que os desvios das implantações, mesmo sendo os tempos de criação de nós menores do que os tempos de implantação. Essa análise evidencia que a variabilidade do tempo de alocação de um nó pela Amazon é o principal fator na variabilidade do tempo de execução. Em especial, observamos na Tabela 1 que os desvios padrões das fases de criação de nós foi muito maior para $N = 10$ e $N = 20$, o que se deve principalmente aos já mencionados eventos esporádicos. Na Figura 5, pode-se notar como a presença mais forte dos eventos esporádicos na criação de nós faz com que não haja um crescimento contínuo no tempo total da fase de criação de nós. Outra evidência de como esse comportamento se deve aos eventos esporádicos é a pouca influência no padrão de crescimento dos tempos médios visto na Figura 4.

Avaliamos o resultado obtido como sendo bastante satisfatório, dado que demonstra que o sistema possui escalabilidade adequada, considerando as restrições impostas por sistemas externos. O crescimento e a variação observados nos tempos referentes à fase

de implantação podem ser explicados pelo crescimento na carga (e, portanto, no tempo de resposta) do Chef Server. O servidor em questão manteve-se com 100% de uso de CPU durante longos períodos, sugerindo que o desempenho pode ser significativamente melhorado através da distribuição e replicação do Chef Server. No entanto, como mencionado, o componente responsável pelo maior impacto negativo na escalabilidade é o serviço da Amazon. É possível que diferentes provedores ofereçam resultados melhores, dependendo de suas políticas de alocação de máquinas virtuais, que também podem levar em conta aspectos como economia de energia, distribuição de carga etc.

5. Conclusão

Neste artigo, apresentamos o CHOReOS Enactment Engine, um sistema de middleware disponibilizado como software livre que oferece serviços de plataforma para a implantação distribuída e automatizada de coreografias de serviços web em ambientes de nuvem. Esse sistema disponibiliza pontos extensíveis que ajudam a amenizar a falta de flexibilidade comum a soluções de Plataforma como Serviço (PaaS). Avaliamos a escalabilidade do tempo de implantação, que se mostrou satisfatória dentro da faixa de operação avaliada, uma vez que o tempo de implantação apenas dobrou para um aumento de 25 vezes no número de serviços implantados ao se utilizar sempre uma máquina virtual por serviço. Esse resultado é particularmente positivo quando comparado aos resultados do único artigo de nosso conhecimento que apresenta avaliações empíricas sobre a escalabilidade do processo de implantação [Quéma et al. 2004], no qual o tempo de implantação é aproximadamente proporcional ao aumento conjunto de carga e recursos.

Em trabalhos futuros utilizaremos diferentes ambientes de computação em nuvem em uma mesma encenação de coreografia. Analisar também o desempenho do middleware variando a quantidade de serviços em uma coreografia e realizando experimentos de maior porte, com centenas ou milhares de serviços. Finalmente, pretendemos adequar ainda mais o CHOReOS Enactment Engine aos requisitos de sistemas de grande escala, considerando especialmente a falha ou mau comportamento de sistemas externos, bastante comuns, como pudemos observar na interação com os serviços da Amazon durante nosso experimento. Um dos planos para isso é o uso de *caches* e *pools* de máquinas virtuais para amenizar o impacto do tempo de criação das máquinas virtuais no processo de encenação.

Agradecimentos

Esta pesquisa foi financiada pela Comissão Europeia (FP7/2007-2013) através do projeto CHOReOS (processo 257178) e pela HP através do projeto Baile. Agradecemos aos colegas envolvidos com a implementação de nosso middleware: Daniel Cukier, Carlos Eduardo do Santos, Felipe Pontes e Alfonso Diaz, e também à Nathália Demétrio pelo suporte ao tratamento estatístico dos dados.

Referências

Balter, R., Bellissard, L., Boyer, F., Riveill, M., and Vion-Dury, J.-Y. (1998). Architecturing and configuring distributed application with Olan. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, Middleware '98, pages 241–256. Springer-Verlag.

Barker, A., Walton, C. D., and Robertson, D. (2009). Choreographing Web Services. *IEEE Transactions on Services Computing*, 2(2):152–166.

Ben Hamida, A., Kon, F., Ansaldi Oliva, G., Santos, C., Lorré, J.-P., Autili, M., Angelis, G., Zarras, A., Georgantas, N., Issarny, V., and Bertolino, A. (2012). An integrated development and runtime environment for the future internet. In *The Future Internet*, volume 7281 of *Lecture Notes in Computer Science*, pages 81–92. Springer Berlin Heidelberg.

Dolstra, E., Bravenboer, M., and Visser, E. (2005). Service configuration management. In *Proceedings of the 12th international workshop on Software configuration management*, SCM '05, pages 83–98. ACM.

Humble, J. and Farley, D. (2011). *Continuous Delivery*. Addison-Wesly.

Law, D. (1998). Scalable means more than more: a unifying definition of simulation scalability. In *Proceedings of the I998 Winter Simulation Conference*, pages 781–788. IEEE.

Magee, J., Dulay, N., and Kramer, J. (1994). A constructive development environment for parallel and distributed programs. In *Proceedings of 2nd International Workshop on Configurable Distributed Systems, 1994*, pages 4–14.

Magee, J. and Kramer, J. (1996). Dynamic structure in software architectures. In *Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering*, SIGSOFT '96, pages 3–14. ACM.

Mell, P. and Grance, T. (2011). The NIST definition of cloud computing (draft). Disponível em http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf. Acessado em 2 de dezembro de 2012.

OMG (2011). Business process model and notation (BPMN), version 2.0. http://www.omg.org/spec/BPMN/2.0.

Quéma, V., Balter, R., Bellissard, L., Féliot, D., Freyssinet, A., and Lacourte, S. (2004). Asynchronous, hierarchical, and scalable deployment of component-based applications. In *Component Deployment*, volume 3083 of *Lecture Notes in Computer Science*, pages 50–64. Springer Berlin Heidelberg.

Steen, M., Pierre, G., and Voulgaris, S. (2012). Challenges in very large distributed systems. *Journal of Internet Services and Applications*, 3(1):59–66.

W3C (2004). Web services architecture. http://www.w3.org/TR/ws-arch/.

Watson, P., Fowler, C., Kubicek, C., Mukherjee, A., Colquhoun, J., Hewitt, M., and Parastatidis, S. (2006). Dynamically deploying web services on a grid using Dynasoar. In *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC 2006*, page 8.

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.

Sobre o Uso de Dispositivos de Alta Granularidade, Alta Volatilidade e Alta Dispersão em *Just in Time Clouds*

Rostand Costa[1 2], Diénert Vieira[2], Francisco Brasileiro[1], Dênio Mariz[2], Guido Lemos[2]

[1] Universidade Federal de Campina Grande - Departamento de Sistemas e Computação
Av. Aprígio Veloso, s/n, Bodocongó 58.429-900 - Campina Grande, PB - Brasil
{*rostand, fubica*}*@lsd.ufcg.edu.br*

[2]Universidade Federal da Paraíba - Departamento de Informática
Laboratório de Aplicações de Vídeo Digital 58.050-900, João Pessoa, PB
{*rostand, dienert, guido, denio*}*@lavid.ufpb.br*

Abstract. *The Just in Time Clouds (JiT Clouds) approach considers the provision of cloud computing using outsourced computing resources. Depending on their characteristics and location in the spectrum of scale, outsourced resources can provide different levels of quality of service, scalability and elasticity. The level of quality of service offered by a JiT Cloud is entirely dependent on the level of quality of service supported by the resources used, which is strongly related to the pattern of granularity, volatility and dispersion that they exhibit. In this paper, we study the use of computing resources for the composition of JiT Clouds through the use of specific mechanisms for its discovery, allocation and coordination, in the worst case scenario of high granularity, high volatility and high dispersion. Our simulation results show that, even in scenarios of high turnover of autonomous and geographically distributed nodes, it is possible to build JiT Clouds with the appropriate collective availability to achieve controlled levels of computational throughput.*

Resumo. *A abordagem Just in Time Clouds (JiT Clouds) considera o provimento de computação na nuvem usando recursos computacionais terceirizados. Dependendo de suas características e da sua localização no espectro de escala, os recursos terceirizados podem fornecer diferentes níveis de qualidade de serviço, elasticidade e escalabilidade. O nível de qualidade de serviço oferecido por uma JiT Cloud é inteiramente dependente do nível de qualidade de serviço suportado pelos recursos usados, o qual está fortemente relacionado ao padrão de granularidade, volatilidade e dispersão dos recursos. Neste trabalho, nós estudamos o uso de recursos para a composição de JiT Clouds através do uso de mecanismos específicos para a sua descoberta, alocação e coordenação, no cenário mais desafiador, caracterizado por alta granularidade, alta volatilidade e alta dispersão. Nossos resultados de simulação mostram que, mesmo em cenários de altíssima rotatividade de nós autônomos e distribuídos geograficamente, é possível construir JiT Clouds com a disponibilidade coletiva adequada para atingir níveis controlados de vazão computacional.*

1. Introdução

É cada vez maior a quantidade de dados que estão sendo gerados atualmente nas mais diversas áreas de conhecimento, por sensores, experimentos científicos e modelos de simulação, dentre outras fontes. Alguns conjuntos de dados são tão grandes que a única maneira viável para processá-los em um tempo razoável é quebrar o processamento em tarefas menores e executá-las em paralelo no maior número disponível de processadores.

Obviamente, paralelismo em larga escala só pode ser alcançado se houver unidades de processamento disponíveis e um nível relativamente elevado de independência entre as tarefas que compõem a aplicação paralela. Felizmente, muitas das cargas de trabalho das aplicações paralelas podem ser mapeadas em tarefas que podem ser processadas de forma completamente independente uma das outras, compondo uma classe de aplicações conhecida como "bag-of-tasks" (BoT) [Cirne et al. 2003]. O

fato de que as tarefas de uma aplicação BoT são totalmente independentes, não só faz o agendamento trivial, no tocante ao escalonamento e execução das tarefas, mas também faz com que a tolerância a falhas seja mais simples de tratar, já que um mecanismo de repetição pode ser usado para recuperar tarefas que eventualmente falhem durante a execução.

O paradigma da computação na nuvem permite o fornecimento de infraestrutura de Tecnologia da Informação sob a forma de um serviço que os clientes adquirem sob demanda e pagam apenas pela quantidade de serviços que realmente consomem. Muitas aplicações que processam grandes cargas de trabalho em paralelo poderiam potencialmente se beneficiar da elasticidade oferecida pelos provedores de computação na nuvem pois permite ao cliente aumentar ou diminuir a capacidade de sua infraestrutura de TI sem qualquer custo adicional. Essa característica faz com que o ônus dos custos e riscos associados ao planejamento da capacidade da infraestrutura de TI passem do cliente para o provedor do serviço. Entretanto, o estado-da-prática em provimento de computação na nuvem impõe um limite a essa elasticidade para que se possa garantir uma disponibilidade suficientemente elevada para os serviços e, ao mesmo tempo, manter os custos operacionais em um nível aceitável. Isso restringe o escopo das aplicações que poderiam se beneficiar do paradigma de computação em nuvem. Em particular, esse é o caso para muitas aplicações científicas do tipo BoT [Costa et al. 2011b].

Em trabalhos anteriores, os autores propuseram as *Just in Time Clouds* ou *JiT Clouds* [Costa et al. 2011a], uma abordagem alternativa para a construção de provedores de computação na nuvem baseada na utilização de recursos terceirizados, na qual os provedores de serviço apenas incorrem em custos de provisionamento quando os recursos terceirizados que eles usam para fornecer os seus serviços são demandados pelos seus clientes e apenas durante o período que eles são necessários, permitindo uma ampliação de algumas ordens de magnitude no limite que precisa ser imposto aos clientes. Dessa forma, as *JiT Clouds* se apresentam como uma infraestrutura adequada para a execução de aplicações BoT de larga escala.

As *JiT Clouds* podem ser montadas sobre recursos que estejam distribuídos por todo o espectro de recursos terceirizados de baixa escala[1]. Uma das missões do operador de uma *JiT Cloud*, o *JiT Provider*, é descobrir e explorar o potencial dos recursos tercerizados disponíveis alinhando-os com as necessidades das aplicações dos clientes. Dependendo de suas características, os recursos terceirizados podem fornecer diferentes níveis de qualidade de serviço, elasticidade e escalabilidade. O nível de qualidade de serviço oferecido por uma *JiT Cloud* é inteiramente dependente do nível de qualidade de serviço suportado pelos recursos usados para montá-la, o qual está relacionado ao padrão de granularidade, volatilidade e dispersão dos recursos.

Por *granularidade*, entende-se o nível de capacidade computacional presente em cada recurso tercerizado. Nesta classificação, servidores de alta capacidade e *clusters*, representam *recursos terceirizados de baixa granularidade* (*coarse-grained*), que são mais densos e poderosos, enquanto computadores pessoais representam *recursos terceirizados de alta granularidade* (*fine-grained*), mais leves e de menor capacidade.

A *volatidade*, por sua vez, representa o nível de disponibilidade e confiabilidade que o recurso tercerizado oferece quando alocado para uma determinada tarefa. Dedicação exclusiva, mecanismos de contingenciamento e tolerância a falhas caracterizam os *recursos terceirizados de baixa volatilidade*, enquanto que o uso oportunista e a falta de garantias de funcionamento são algumas das principais características dos *recursos terceirizados de alta volatilidade*.

A última propriedade considerada, a *dispersão*, está relacionada com o nível de distribuição dos recursos terceirizados. Os recursos concentrados em centros de dados representam *recursos terceirizados de baixa dispersão* enquanto que recursos individuais, distribuídos geograficamente, são *recursos terceirizados de alta dispersão*.

[1]Nesta categoria situam-se todos os detentores de recursos ociosos que não possuem, sozinhos, capacidade excedente suficiente para uma atuação solo como provedores públicos de computação na nuvem, como fez a *Amazon Bookstore*, por exemplo.

Quando os recursos estão concentrados em centros de dados e sua capacidade está localizada mais próxima do topo da magnitude que limita a baixa escala de recursos tercerizados, ou seja, são recursos computacionais semelhantes aos usados pelos provedores de tradicionais de computação na nuvem, os níveis de serviço oferecidos são consistentes com os praticados no mercado. Dessa forma, *JiT Clouds* baseadas em recursos de baixa granularidade, baixa volatilidade e baixa dispersão podem ser usadas para hospedar aplicações tipicamente suportadas por computação na nuvem.

No outro extremo do espectro da escala, quando os recursos terceirizados são de grão pequeno e distribuídos, eles precisam ser agrupados e coordenados pelo *JiT Provider* para a sua exploração. Estes recursos de alta granularidade, alta volatilidade e alta dispersão podem ser **convencionais**, representados por equipamentos padrão de processamento, e **não convencionais**, como receptores de TV Digital, por exemplo. Boa parte desses dispositivos não convencionais, notadamente os mais recentes, são equipados com processadores poderosos e quantidade razoável de memória, permitindo-lhes a execução de aplicações [Costa et al. 2012]. No entanto, como estes dispositivos são tipicamente recursos não dedicados e voláteis, uma *JiT Cloud* baseada neles é menos confiável do que aquela que é construído sobre recursos privados e dedicados. No entanto, há evidências suficientes de que existem clientes dispostos a utilizar serviços com qualidade de serviço do tipo *best-effort*: por um lado, a mera existência das *spot instances*[2] da AWS é uma boa indicação disso; por outro lado, a abundância de aplicações HTC[3] científicas e industriais adaptáveis para execução em ambientes de nuvem com qualidade de serviço equivalente ao proporcionado pelas *spot instances*, fornecem evidências adicionais de que um serviço altamente elástico e escalável de computação na nuvem é de muita utilidade.

Neste trabalho, nós analisamos o potencial de uso de recursos de alta granularidade, alta volatilidade e alta dispersão para a composição de *JiT Clouds* através do uso de mecanismos específicos para a sua descoberta, alocação e coordenação. Nossos resultados de simulação mostram que, mesmo em cenários de altíssima rotatividade de nós autônomos e distribuídos geograficamente, é possível construir *JiT Clouds* com a disponibilidade coletiva [Andrzejak et al. 2008] adequada para atingir níveis controlados de vazão computacional.

O restante do documento está organizado da seguinte forma. Na Seção 2, para permitir o entendimento do restante do trabalho, apresentamos a arquitetura geral do mecanismo considerado para a construção de *JiT Clouds* dinâmicas baseadas em recursos voláteis e distribuídos. Nessa seção nós descrevemos formalmente o modelo de operação do sistema, e apresentamos os desafios trazidos pelas características particulares das *JiT Clouds* estudadas neste artigo. A Seção 3 traz uma descrição de como foi realizada a nossa avaliação, incluindo uma discussão do modelo de simulação utilizado. Na Seção 4, analisamos os resultados obtidos nos nossos experimentos, enquanto que na Seção 5 discutimos alguns trabalhos relacionados. Finalmente, a Seção 6 traz as nossas considerações finais.

2. *JiT Clouds* Baseadas em Dispositivos de Alta Granularidade, Alta Volatilidade e Alta Dispersão

A fim de construir *JiT Clouds* dinâmicas e de alta vazão baseadas em recursos terceirizados dispersos, de pequena capacidade e não dedicados é necessário fornecer uma maneira de acessar, individualmente, uma grande quantidade de processadores, enviar programas e, possivelmente, dados, para todos e, remotamente, desencadear a execução do código transmitido. Em seguida, é necessário reunir os resultados produzidos, e, finalmente, liberar os recursos alocados de forma que outras aplicações possam usá-los.

A ideia de alocar uma enorme quantidade de recursos através da abstração de um *JiT Cloud*,

[2]Nessa modalidade, o usuário faz uma oferta para instâncias do Amazon EC2 disponíveis e as executa sempre que sua oferta exceder o preço *spot* atual, o qual varia em tempo real com base na demanda e cujo aumento pode ocasionar a retomada da instância a qualquer tempo.

[3]Em uma classificação bastante ampla, a computação paralela é normalmente dividida em Computação de Alto Desempenho (HPC, do inglês *High Performance Computing*) e Computação de Alta Vazão (HTC, do inglês *High Throughput Computing*) [Litzkow et al. 1988].

habilitá-los para o processamento distribuído de aplicações paralelas (centenas de milhares de computadores conectados via Internet, por exemplo) e fazê-lo a um custo muito menor do que alternativas tradicionais, apesar de atrativa, representa um desafio não trivial. A questão principal é: **onde** encontrar milhões de processadores terceirizados disponíveis e **como** configurá-los em conformidade e instantaneamente para o uso em *JiT Clouds* dinâmicas voltadas para os requisitos de alta vazão de aplicações BoT? Além disso, como executar esta tarefa com um atraso mínimo?

Neste sentido, uma categoria singular destes dispositivos tercerizados desperta um interesse especial para este trabalho: aqueles que podem ser organizados em uma rede de *broadcast*[4]. Uma rede de *broadcast* possui o potencial de permitir a comunicação simultânea com todos os dispositivos conectados, os quais podem ser coordenados para realizar uma determinada ação. Nesta abordagem, programas transmitidos através do canal de *broadcast* podem ser carregados e executados concomitantemente por todos os recursos computacionais conectados à rede de *broadcast* em um dado momento. Este mecanismo torna possível construir, de uma forma realmente rápida[5] e controlada, *JiT Clouds* distribuídas e dinâmicas.

2.1. *Infraestrutura Computacional Distribuída Sob Demanda*

Usando o conceito de redes de *broadcast*, os autores propuseram em trabalhos anteriores [Costa et al. 2009] uma nova arquitetura para construir *Distributed Computing Infrastructures* (DCIs) dinâmicas baseadas em recursos computacionais de alta granularidade, alta volatilidade e alta dispersão que é, ao mesmo tempo flexível e altamente escalável, sendo aplicável para a execução eficiente de aplicações BoT de larga escala e curta duração. Com esta abordagem, um cliente poderá alocar, sob demanda, um conjunto com um grande número de unidades de processamento, chamada de instância DCI, que executará sua aplicação BoT de forma tão eficiente quanto possível. Após completar a execução, o cliente liberará a instância DCI que foi criada. Por causa desta singularidade, a arquitetura é chamada de *Infraestrutura Computacional Distribuída Sob Demanda* (ou OddCI, do inglês *On-Demand Distributed Computing Infrastructure*).

A arquitetura OddCI é formada por um *Provider*, um *Backend*, uma ou mais redes de *broadcast*, cada uma contendo um canal de *broadcast* e um *Controller*, e *Processing Node Agents* (PNA). Estes últimos são programas a serem enviados e executados em cada um dos recursos computacionais acessíveis pelo *Controller* através da sua rede de *broadcast* correspondente. Além disso, é assumido que os recursos computacionais também possuem um canal bidirecional, chamado de *canal direto*, o qual os conecta tanto com o *Backend* quanto com o seu respectivo *Controller* (1).

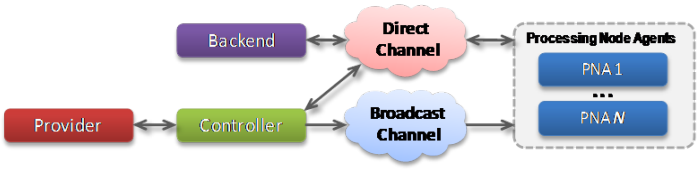


Figura 1. Visão Geral da Arquitetura OddCI

O *Provider* (provedor) é responsável por criar, gerenciar e destruir as instâncias OddCI de acordo com as solicitações dos clientes e também pela autenticação do cliente e pela verificação das

[4]O termo *broadcasting* está, originalmente, relacionado a transmissões de rádio ou televisão e significa a distribuição, de forma simultânea e através de um meio físico específico e unidirecional (o canal de *broadcast*), de sinais de áudio e/ou vídeo contendo programação para uma determinada audiência. Considerando o mesmo princípio de transmissão de um-para-muitos, será usado o termo **rede de *broadcast*** para representar uma rede composta por um transmissor digital de dados, um **canal de *broadcast***, um conjunto de equipamentos receptores com capacidade de processamento de aplicações paralelas e possibilidade de acesso a um canal de interação *full-duplex*, comumente uma conexão com a Internet.

[5]Na verdade, o quão rápido o *software* será carregado dependerá do tamanho dos dados a serem transmitidos e da velocidade do canal de *broadcast*.

suas credenciais para usar os recursos que estão sendo requisitados. O *Controller* (controlador) é encarregado de configurar a infraestrutura, conforme instruído pelo *Provider*, e formatar e enviar, via canal de *broadcast*, mensagens de controle e imagens (executáveis) para os PNAs, necessárias para construir e manter as instâncias OddCI. A responsabilidade do *Backend* (retaguarda) é o gerenciamento das atividades específicas de cada aplicação sendo executada: distribuição de tarefas, provisionamento de dados de entrada e recepção e pós-processamento dos resultados gerados pela aplicação paralela. Cabe aos *Processing Node Agents - PNA* (agentes processadores) o gerenciamento da execução da aplicação do cliente no dispositivo computacional e o envio de sondas periódicas (*heartbeat messages*) para sinalizar o seu estado.

O *Direct Channel* (canal direto), por sua vez, é uma rede de comunicação bidirecional que permite a comunicação entre todos os componentes da arquitetura, tal como a Internet, enquanto que o *Broadcast Channel* (canal de *broadcast*) é um canal unidirecional para envio de dados do *Controller* para os dispositivos. Pode ser, por exemplo, um canal de TV Digital ou uma ERB de uma rede celular.

2.2. Modelo de Operação OddCI

Nesta subseção é feita uma descrição mais formal do modelo de operação de sistemas OddCI que foi utilizado na nossa avaliação.

Consideramos uma rede de *broadcast* que pode acessar um conjunto $\mathbb{D}$ de dispositivos. Seja $A(d,t)$ uma função boleana no tempo que indica se um dispositivo $d \in \mathbb{D}$ está ativo no momento t. O conjunto de dispositivos *ativos* no momento t, $\mathbb{D}^a(t)$, é dado por $\mathbb{D}^a(t) = \{ d \mid d \in \mathbb{D} \bigwedge A(d,t) = true\}$ e o conjunto de dispositivos *inativos* no momento t, $\mathbb{D}^i(t)$, é dado por $\mathbb{D}^i(t) = \mathbb{D} \backslash \mathbb{D}^a(t)$. É assumido que os dispositivos são voláteis, ou seja, os dispositivos podem alternar entre os estados *ativo* e *inativo* em qualquer momento e, portanto, um mesmo dispositivo $d \in \mathbb{D}^a(t')$ pode pertencer a $\mathbb{D}^i(t''), t' \neq t''$.

Seja o serviço demandado pelos clientes de um provedor de um sistema OddCI definido por uma sequência de tuplas r_1, r_2, ..., r_n com $r_j =< t_j, q_j, l_j >$, onde t_j é o momento no qual r_j é submetida, q_j é a quantidade desejada de dispositivos simultâneos que devem ser alocados e l_j é a duração do intervalo de tempo no qual os q_j recursos serão necessários ($t_j, q_j, l_j \in \mathbb{N}$). A instância OddCI $\mathbf{I}_j$, $1 \leq j \leq n$, representa o atendimento da requisição r_j pelo sistema.

Seja $L(d,t)$ a função boleana que indica se o dispositivo d está alocado a alguma instância no tempo t, o conjunto $\mathbb{D}^a(t)$ pode ser decomposto em $\mathbb{D}^a(t) = \mathbb{D}^l(t) \cup \mathbb{D}^d(t)$, onde $\mathbb{D}^l(t)$ é o subconjunto dos dispositivos ativos e alocados a instâncias no momento t ($\mathbb{D}^l(t) = \{d \mid d \in \mathbb{D}^a(t) \bigwedge L(d,t) = true\}$) e $\mathbb{D}^d(t)$ é o subconjunto dos dispositivos ativos que estão disponíveis no momento t ($\mathbb{D}^d = \mathbb{D}^a(t) \backslash \mathbb{D}^l(t)$).

Um controlador ao ser designado, através de uma estratégia de escalonamento, pelo provedor para o atendimento de uma demanda r_j, tentará fazer a alocação dos q_j dispositivos solicitados através do envio de mensagens de controle para a rede de *broadcast* que controla. Seja m uma mensagem de controle enviada através do canal unidirecional no momento t, então todos os dispositivos pertencentes a $\mathbb{D}^d(t + T(m))$ receberão e processarão m, onde $T(m)$ é a duração da transmissão da mensagem de controle m. $T(m)$ é uma função da taxa de transmissão e do retardo médio do canal unidirecional e do tamanho da mensagem m.

Seja $\mathbb{D}^r(m) \subseteq \mathbb{D}^d(t + T(m))$ o subconjunto dos dispositivos ativos *disponíveis* em $t+T(m)$ que responderem, através dos seus respectivos canais bidirecionais, à convocação do controlador feita pela mensagem m. O subconjunto $\mathbb{D}^v(m)$ com os primeiros q_j dispositivos de $\mathbb{D}^r(m)$ que atendam a um critério de elegibilidade definido pelo *Controller* em função dos requisitos do cliente serão alocados para a instância $\mathbf{I}_j$. Os demais dispositivos, $\mathbb{D}^r(m) \backslash \mathbb{D}^v(m)$, serão descartados[6].

Para lidar com a volatilidade do sistema, assumimos que o sistema de tarifação adotado pelo

[6]Por simplicidade foi assumido que todos os dispositivos na rede de *broadcast* simulada atendiam ao critério de elegibilidade desejado.

provedor para o uso de seus recursos é baseado na apuração de cada intervalo de tempo com duração σ, chamado *slot* de processamento, durante o qual um dispositivo permanece ativo e alocado a uma instância. Sempre que um dispositivo d é alocado para a instância $\mathbf{I}_j$ em um momento t, o *slot* de processamento $s_{j,d,t}$ é iniciado. O *slot* $s_{j,d,t}$ é dito completado se d permanece alocado para a instância $\mathbf{I}_j$ até o momento $t + \sigma$. Apenas *slots* completados são tarifados.

Seja $\mathbb{S}_j^i$ o conjunto de todos os *slots* iniciados na instância $\mathbf{I}_j$ e seja $O(j,d,t)$ uma função boleana que indica se o *slot* $s_{j,d,t}$ foi completado, então o conjunto de *slots* completados na instância $\mathbf{I}_j$ é dado por $\mathbb{S}_j^c = \left\{ s_{j,d,t} \mid s_{j,d,t} \in \mathbb{S}_j^i \bigwedge O(j,d,t) = true \right\}$. Uma instância $\mathbf{I}_j$ é completada quando um mínimo de $\left\lceil \frac{l_j}{\sigma} \right\rceil \times q_j$ *slots* de processamento *completados* é atingido, ou seja, $\left|\mathbb{S}_j^c\right| \geq \left\lceil \frac{l_j}{\sigma} \right\rceil \times q_j$. Caso $\mathbf{I}_j$ ainda não tenha sido completada quando o *slot* $s_{j,d,t}$ for completado, o dispositivo d será realocado à instância $\mathbf{I}_j$, iniciando o *slot* $s_{j,d,t+\sigma}$.

Seja $I(d,t)$ a função que indica a qual instância o dispositivo $d \in \mathbb{D}^a(t)$ está alocado com exclusividade no tempo t:

$$I(d,t) = \begin{cases} j, \text{ se } \textit{d} \text{ está alocado à instância } \textit{I}_\textit{j} \text{ no momento } \textit{t} \\ 0, \text{ se } \textit{d} \text{ não está alocado em nenhuma instância no momento } \textit{t} \end{cases}, d \in \mathbb{D}^a(t),$$

então o conjunto de dispositivos alocados à instância $\mathbf{I}_j$ no momento t, $\mathbb{D}_j^l(t)$, é dado por $\mathbb{D}_j^l(t) = \{d \mid d \in \mathbb{D}^a(t) \bigwedge I(d,t) = j\}$.

2.3. O Desafio da Alta Volatilidade

Como os dispositivos acessíveis pela rede de *broadcast* são voláteis, os dispositivos ativos alocados à instância $\mathbf{I}_j$ podem, eventualmente, se tornar inativos em qualquer momento e tais perdas de dispositivos precisam ser identificadas e repostas. Para lidar com a eventual perda de dispositivos ativos alocados para uma instância $\mathbf{I}_j$ e recompor o seu tamanho para o valor desejado, q_j, o controlador pode enviar novas mensagens de controle para alocar os dispositivos faltantes sempre que necessário. A frequência de envio de mensagens de controle para manter o tamanho solicitado de uma instância é definida pela estratégia de provisionamento adotada.

A reposição de dispositivos para a instância $\mathbf{I}_j$ no momento t através do envio de uma mensagem de controle m levará o tempo $T(m)$ para atingir os dispositivos ativos disponíveis no momento $t + T(m)$, $\mathbb{D}^d(t + T(m))$. Neste sentido, a estratégia de provisionamento adotada pelo controlador precisa considerar a reposição tanto dos dispositivos já perdidos por $\mathbf{I}_j$ no momento t, quanto dos que poderão ser perdidos adicionalmente até o momento $t + T(m)$.

Para evitar o uso ineficiente de recursos do controlador e da rede de *broadcast* causado pelo descarte de um número grande de dispositivos, a quantidade de dispositivos que responderem à mensagem de controle m, $|\mathbb{D}^r(m)|$, deve ser o mais próximo possível da quantidade de dispositivos que serão alocados a $\mathbf{I}_j$ em decorrência do envio de m, $|\mathbb{D}^v(m)|$. Para tal, é atribuído para cada mensagem de controle m enviada, um valor $P(m)$ que representa a probabilidade de cada dispositivo em $\mathbb{D}^d(t + T(m))$ responder ou não à mensagem m enviada no momento t. O cálculo de $P(m)$ leva em consideração a quantidade de dispositivos que se necessita e a quantidade total de dispositivos que estarão disponíveis: $P(m) = |\mathbb{D}^v(m)| / |\mathbb{D}^d(t + T(m))|$. Neste sentido, como o estado dos dispositivos da rede de *broadcast* pode mudar constantemente, é necessário dispor de algum mecanismo para fazer, em t, uma estimativa do número de dispositivos disponíveis em um momento futuro, $t + T(m)$.

Por outro lado, para minimizar a perda de dispositivos em $\mathbf{I}_j$, o controlador precisa adotar algum critério de elegibilidade para indicar, dentre os dispositivos existentes em $\mathbb{D}^d(t + T(m))$ que irão responder a m, aqueles dispositivos que possuam uma expectativa de maior permanência no estado ativo.

Do ponto de vista do cliente, a existência da volatilidade do sistema implica na necessidade de adequar o tamanho máximo das tarefas da sua aplicação como um divisor do tamanho do *slot* de processamento adotado pelo provedor, ou seja, deve ser possível a conclusão total ou parcial (via

checkpoints) de uma ou mais tarefas durante a duração de um *slot* de processamento.

3. Descrição dos Experimentos

O objetivo principal da nossa avaliação, baseada em simulação, foi investigar o potencial de uso de recursos tercerizados em *JiT Clouds* no cenário mais desafiador, caracterizado por alta granularidade, alta volatilidade e alta dispersão através do uso de mecanismos específicos para a sua descoberta, alocação e coordenação.

No contexto considerado, os recursos alocados para a construção de *JiT Clouds* possuem um grau extremo de distribuição e granularidade, sendo necessário acessar cada dispositivo diretamente e escalonar individualmente cada *slot* de processamento. Além disso, os dispositivos são eminentemente voláteis e alocar um determinado conjunto de recursos para processar tarefas em paralelo implica que, ao longo do tempo, o conjunto sofrerá redução no seu tamanho. Assim, é necessário reparar a perda esperada de nós através de alguma estratégia de antecipação ou recuperação, que chamamos de algoritmos compensatórios.

A utilização de métodos de predição para suportar mecanismos que assegurem a disponibilidade coletiva (*collective availability* [Andrzejak et al. 2008]) de uma coleção volátil de recursos foi estudada por *Andrzejak et al.* O estudo mostra que através de previsão, métodos acurados podem ser usados para garantir que um subconjunto ω qualquer de nós em um conjunto volátil Ω esteja disponível durante um período ΔT ao custo de uma sobrecarga de controle razoável. As técnicas de predição avaliadas foram bem sucedidas também para avaliar o custo de disponibilidade do subconjunto em termos de nível de redundância necessário e da sobrecarga de migração entre recursos não dedicados e dedicados para compensar a perda de nós.

A taxa de sucesso (*success rate*) obtida quando se tenta manter um subconjunto ω de dispositivos disponíveis em um dado período é dependente do tempo médio de disponibilidade dos dispositivos do conjunto volátil (*historical turnover rate*) e do tamanho de ω, mas pode ser equilibrada através de um nível adequado de redundância R através da alocação de $|\omega| + R$ recursos computacionais. Os resultados de *Andrzejak et al.* indicam que a solução mais prática para controle da disponibilidade coletiva é uma combinação de uma abordagem de previsão simplificada com o ranqueamento dos dispositivos de acordo com sua disponibilidade histórica. Com base nisso, uma sequência de *bits* 0 ou 1 pode representar a disponibilidade histórica de cada dispositivo em instantes de tempo específicos e um modelo de predição processa as sequências de *bits* dos dispositivos gerando um *ranking* de regularidade que pode permitir ou não a sua escolha para o atendimento de requisitos de disponibilidade específicos. Em nossa abordagem, uma variação escalável desse método é obtida através do cálculo e manutenção das informações históricas e do ranqueamento de disponibilidade corrente nos próprios dispositivos (PNA) e enviadas ao *Controller* em cada *heartbeat message* transmitida. Um alvo de ranqueamento é informado pelo *Controller* juntamente com os outros requisitos para o dispositivo nas mensagens de controle enviadas e, considerando o histórico de disponibilidade do dispositivo, o PNA decide se irá juntar-se ou não à instância. Com o uso de ranqueamento, o critério de elegibilidade do PNA primeiro verifica o *ranking* do dispositivo e depois aplica o fator probabilístico indicado em $P(m)$, o qual deve ter sido calculado considerando uma estimativa da quantidade de dispositivos disponíveis que atendem ao alvo de ranqueamento desejado. Eventualmente, o controlador pode precisar diminuir o *ranking*-alvo para ajustá-lo à condição atual de ranqueamento dos dispositivos disponíveis e conseguir obter a quantidade necessária de dispositivos para repor o tamanho da instância.

Para analisar como a volatilidade e a contenção de recursos presentes na rede de *broadcast* podem afetar a disponibilidade coletiva necessária, foram considerados dois cenários de uso:

- *Atendendo a Aplicações Sensíveis ao Tempo*: No primeiro cenário, chamado *Vazão Mínima*, o controlador tenta garantir que a duração esperada para a instância $\mathbf{I}_j$ seja observada, ou seja, que os $\left\lceil \frac{l_j}{\sigma} \right\rceil \times\ q_j$ *slots* solicitados sejam completados no tempo l_j. Em tal contexto, o número de *slots* completados na instância $\mathbf{I}_j$ precisa permanecer em uma média maior ou igual a q_j durante todo o ciclo de vida de $\mathbf{I}_j$. Para lidar com a eventual perda de dispositivos e

mesmo assim garantir uma vazão mínima q_j, o controlador deve aplicar um determinado nível de redundância sobre o tamanho mínimo desejado para a instância. Para isso, são enviadas, proativamente, mensagens de controle para regenerar o tamanho da instância para um valor alvo $q_j + X$, onde X é a quantidade adicional necessária para compensar as eventuais perdas de dispositivos que ocorrerão até o envio do próximo comando de regeneração. Baseado na última consolidação de *heartbeat messages*, o controlador calcula X, o momento t para envio de cada mensagem de controle m para a instância $\mathbf{I}_j$ e também $|\mathbb{D}^d\left(t + T\left(m\right)\right)|$ em função da taxa histórica de perda de dispositivos observada na rede de *broadcast* em um dado período de referência, cujo momento inicial padrão é o momento de submissão da demanda r_j, ou seja, t_j. O valor $P\left(m\right)$ é definido pelo controlador para cada mensagem de controle m considerando q_j, X, $|\mathbb{D}_j^l(t)|$ e $|\mathbb{D}^d\left(t + T\left(m\right)\right)|$. Neste cenário, é aceitável que o tamanho solicitado para a instância (q_j) seja excedido para compensar regimes de maior volatilidade.

- *Lidando com Capacidade Limitada no Backend*: No segundo cenário, chamado *Paralelismo Máximo*, o controlador tenta cumprir, tanto quanto possível, o limite do tamanho q_j solicitado para a instância sem excedê-lo. Assim, o número de dispositivos alocados para a instância $\mathbf{I}_j$, tende a permanecer em uma quantidade sempre igual ou menor do que q_j durante todo o seu ciclo de vida para respeitar a condição de que o *Backend* do cliente só consegue tratar, no máximo, q_j dispositivos simultaneamente. Sempre que a perda de dispositivos causada pela volatilidade da rede de *broadcast* atingir um determinado limite Y, ou seja, $\left|\mathbb{D}_j^l\left(t\right)\right| \leq q_j - Y$, serão enviadas, reativamente, mensagens de controle para regenerar o tamanho da instância para o valor alvo q_j. O valor adequado de Y, que representa o tempo de reação para regeneração da instância, é definido pelo controlador a partir do tempo $T\left(m\right)$ necessário para transmissão da mensagem de controle m, bem como em função da taxa histórica de perda de dispositivos observada na rede de *broadcast*. O valor $P\left(m\right)$ é definido pelo controlador para cada mensagem de controle m considerando q_j, Y, $|\mathbb{D}_j^l(t)|$ e $|\mathbb{D}^d\left(t + T\left(m\right)\right)|$. Neste cenário, é aceitável que a duração solicitada (l_j) não seja cumprida em regimes de maior volatilidade.

3.1. Implementação do Modelo de Simulação

O simulador usado nos experimentos foi baseado no ambiente OMNeT++ [Varga and Hornig 2008]. O OMNeT++ é composto por uma biblioteca de componentes e de um *framework* de simulação modular e flexível, que pode ser estendido usando a linguagem C++ para a lógica dos componentes, enquanto que a linguagem *NEtwork Description* (NED) é usada para descrição da topologia da rede, portas de comunicação, canais, conexões, dentre outros parâmetros. Para essa avaliação, algumas extensões nos componentes originais foram realizadas. Em particular, foram acrescentados os aspectos de transmissão em *broadcast* e o comportamento dos componentes da arquitetura, de acordo com o modelo de operação descrito na Seção 2.2 e os mecanismos descritos na Seção 3[7].

Parte da configuração do simulador foi baseada em uma etapa anterior da pesquisa na qual foram obtidas medições de campo em um *testbed* real: um protótipo de sistema OddCI para redes de TV Digital [Costa et al. 2012], cujos resultados permitiram confirmar o comportamento linear na transmissão de mensagens de controle por radiodifusão, a adequação dos recursos de comunicação direta dos receptores para troca de tarefas/resultados e o potencial de processamento de STBs de baixo custo (*low-end*).

O comportamento estocástico do sistema OddCI simulado foi modelado usando algumas variáveis independentes (aleatórias). A população total de dispositivos computacionais (ou nós) potencialmente acessíveis através da rede de *broadcast*, denominada $\mathbb{D}$, é determinada, *a priori*, como um parâmetro de simulação. Entretanto, a quantidade de nós ativos (i.e, que podem ser efetivamente atingidos por uma mensagem de controle) no início da simulação é modelada como uma variável aleatória com distribuição uniforme: $|\mathbb{D}^a(0)| = U(\mu, |\mathbb{D}|)$, onde μ é o número mínimo de dispositivos

[7]O modelo completo do simulador usado neste trabalho pode ser encontrado no sítio http://www.lsd.ufcg.edu.br/∼rostand/JiTDC_OddCISim.zip.

acessíveis através da rede de *broadcast*. Uma vez que o número inicial de dispositivos ativos $|\mathbb{D}^a(0)|$ é determinado no início da simulação, os dispositivos ativos iniciais são selecionados entre a população de dispositivos, $\mathbb{D}$, com igual probabilidade. Sempre que um nó individual é selecionado para ser ativado, ele permanece ativo por um tempo de sessão τ_{ON} e então é desativado por um período de espera (*standby*) τ_{OFF}. Dessa forma, os dispositivos ativos em um determinado momento na rede de *broadcast* configuram um processo estocástico que depende das seguintes variáveis: tamanho da população $|\mathbb{D}|$, o número inicial de dispositivos ativos, $|\mathbb{D}^a(0)|$, o tempo em sessão, τ_{ON}, e o tempo em *standby*, τ_{OFF}. Foi assumido um mesmo *ranking* de disponibilidade para os dispositivos em $\mathbb{D}$.

A volatilidade ($\mathcal{V}$) inserida no sistema simulado foi normalizada, através das probabilidades utilizadas em τ_{ON} e τ_{OFF} (que foram modeladas como variáveis aleatórias com distribuição Bernoulli), de forma a obter uma variação percentual controlada da quantidade de dispositivos que alternam entre o estado ativo e inativo na rede de *broadcast* dentro de cada período de tempo de tamanho σ, o intervalo de referência considerado, mas mantendo o total de ativos em qualquer tempo próximo da disponibilidade inicial configurada. Em resumo, o parâmetro $\mathcal{V}$ regula o percentual de dispositivos ativos ganhos e perdidos em um dado intervalo de tempo de tamanho σ, o mesmo adotado como duração de um *slot* de processamento. Uma vantagem desta associação da volatilidade à duração do *slot* de processamento é tornar os resultados obtidos em uma dada configuração aplicáveis em outros cenários de tarifação e granularidade de tarefas.

Para analisar o comportamento do sistema sob alta volatilidade em regimes de contenção de recursos, a carga de trabalho utilizada teve como objetivo estressar dois gargalos potenciais: a disponibilidade de dispositivos para atendimento da demanda e a concorrência pelo uso do canal de transmissão em *broadcast*. Para tal, foi fixado um pico de demanda ($\mathcal{P}$), representando o máximo da soma de dispositivos alocados para instâncias em um dado momento de um período de observação. A partir de $\mathcal{P}$, as cargas de trabalho de cada experimento foram construídas de forma relativa usando dois parâmetros do simulador: quantidade de instâncias simultâneas ($\mathcal{S}$) e a duração das instâncias em *slots* ($\mathcal{D}$). Assim, o *workload* de cada experimento é baseado na sua configuração e formado por $\mathcal{S}$ instâncias simultâneas iguais, todas iniciando no mesmo momento ($t_j = 0$), solicitando a mesma quantidade de dispositivos ($q_j = \frac{\mathcal{P}}{\mathcal{S}}$) pelo mesmo intervalo de tempo ($l_j = \mathcal{D} \times \sigma$). O tamanho de $\mathbb{D}$ é regulado pela aplicação de um *fator de contenção*, ζ, sobre $\mathcal{P}$: $|\mathbb{D}| = \zeta \times \mathcal{P}$.

3.2. Parâmetros do Sistema

Para atribuição dos parâmetros do sistema foram usadas duas estratégias: projeto de experimento (DoE, do inglês *Design of Experiment*) e varredura de parâmetros. Inicialmente, os parâmetros foram tratados em cada cenário considerado através de um DoE do tipo 2^k fatorial [Jain 1991].

Os fatores considerados no DoE foram: *Volatilidade* ($\mathcal{V}$), *Tamanho da População* ($|\mathbb{D}|$), *Tamanho da Imagem* ($\mathcal{T}$), *Instâncias Simultâneas* ($\mathcal{S}$) e *Duração da Instância* ($\mathcal{D}$).

Para o tamanho da imagem da aplicação, o qual está associado ao tempo de uso do canal de transmissão em *broadcast* para envio de cada mensagem de controle, foram considerados dois valores diferentes: *pequeno* (representativo do tamanho de módulos clientes de aplicações como o SETI@home [Anderson et al. 2002] e *grande* (representando "*workers*" de implementações padrão de *desktop grids* como o OurGrid [Cirne et al. 2006]). As imagens do tipo *pequeno* têm 512 Kbytes de tamanho e correspondem ao nível **baixo** do fator, enquanto que as imagens do tipo *grande* possuem tamanho de 5 Mbytes e representam o nível **alto** do fator. Os níveis atribuídos para os demais fatores em cada DoE estão apresentados nas Tabelas 1 e 2.

A variável de resposta considerada para o cenário do *Vazão Mínima* foi o *coeficiente médio de vazão* ($\overline{\Phi}$) das instâncias, o qual representa a relação entre a quantidade média de *slots* completados por ciclo e a quantidade necessária para que a duração esperada para a instância seja cumprida. Essa métrica é dada por $\overline{\Phi} = (\sum_{j=1}^{\mathcal{S}} (|\mathbb{S}_j^c|/\mathcal{D}/q_j))/\mathcal{S}$ e seu valor de referência é 1.

Para o cenário do *Paralelismo Máximo* foi escolhida a variável de resposta *coeficiente médio*

Tabela 1. DoE 2^k: Fatores, níveis e efeitos para o cenário *Vazão Mínima*

Fator	Baixo	Alto	Efeito Estimado	Soma dos Quadrados	Contribuição
A: Volatilidade ($\mathcal{V}$)	5%	75%	$-0,3335$	$0,8896$	$28,41\%$
B: População ($\|\mathbb{D}\|$)	$(1+\mathcal{V})\times\mathcal{P}$	$10\times\mathcal{P}$	$0,2672$	$0,5710$	$18,24\%$
C: Tamanho da Imagem ($\mathcal{T}$)	$512Kb$	$5Mb$	$-0,1667$	$0,2223$	$7,10\%$
D: Instâncias Simultâneas ($\mathcal{S}$)	10	100	$-0,1729$	$0,2392$	$7,64\%$
E: Duração da Instância ($\mathcal{D}$)	$10\ horas$	$100\ horas$	$-0,0184$	$0,0027$	$0,09\%$

de paralelismo ($\overline{\Pi}$) das instâncias, o qual representa a relação entre a quantidade efetiva de dispositivos fornecida e a quantidade de dispositivos solicitada. Esta métrica é dada por $\overline{\Pi} = (\sum_{j=1}^{\mathcal{S}}(|\overline{\mathbb{D}_j^l}|/q_j))/\mathcal{S}$ e seu valor de referência também é 1.

Tabela 2. DoE 2^k: Fatores, níveis e efeitos para o cenário *Paralelismo Máximo*

Fator	Baixo	Alto	Efeito Estimado	Soma dos Quadrados	Contribuição
A: Volatilidade ($\mathcal{V}$)	5%	75%	$-0,2216$	$0,3927$	$16,17\%$
B: População ($\|\mathbb{D}\|$)	$(1+\mathcal{V})\times\mathcal{P}$	$10\times\mathcal{P}$	$0,0449$	$0,0161$	$0,66\%$
C: Tamanho da Imagem ($\mathcal{T}$)	$512Kb$	$5Mb$	$-0,2327$	$0,4331$	$17,83\%$
D: Instâncias Simultâneas ($\mathcal{S}$)	10	100	$-0,2412$	$0,4653$	$19,16\%$
E: Duração da Instância ($\mathcal{D}$)	$10\ horas$	$100\ horas$	$0,0080$	$0,0005$	$0,02\%$

Foram conduzidas várias repetições dos 32 experimentos previstos no DoE realizado para cada um dos cenários considerados para obter médias com 95% de confiança com erro máximo de 5% para mais ou para menos. A contribuição de cada fator em cada cenário é mostrada nas Tabelas 1 (*Vazão Mínima*) e 2 (*Paralelismo Máximo*).

No cenário de *Vazão Mínima*, os fatores da *Volatilidade* e do *Tamanho da População* foram preponderantes com participação de $28,41\%$ e $18,24\%$, respectivamente (Tabela 1). Enquanto que no cenário de *Paralelismo Máximo*, além da *Volatilidade*, que responde por $16,17\%$, os fatores *Tamanho da Imagem* com $17,83\%$ e *Instâncias Simultâneas* com $19,16\%$ foram determinantes na variação da métrica observada (Tabela 2).

Como resultado da análise dos efeitos através de ANOVA [Jain 1991], o F-Value de $164,4793$ (*Vazão Mínima*) e $252,9781$ (*Paralelismo Máximo*) implicam que os modelos são significativos. O R^2 ajustado indica que os modelos explicam $98,75\%$ e $98,27\%$ da variação observada e o R^2 de predição está dentro de $0,20$ do R^2 ajustado, representando uma boa capacidade de predição dos modelos. Maiores detalhes sobre este estudo, incluindo os gráficos de diagnóstico, cubo e interação, podem ser encontrados *online* no mesmo endereço citado na Seção 3.1.

Para a realização das simulações, os valores dos parâmetros que não afetaram o comportamento da variável de resposta foram ajustados para os valores médios entre os níveis "Alto" e "Baixo" usados em cada DoE[8]. Para os fatores mais relevantes: *Volatilidade* e *Tamanho da População* (*Vazão Máxima*) e *Volatilidade*, *Tamanho da Imagem* e *Instâncias Simultâneas* (*Paralelismo Máximo*), foi aplicada uma varredura de parâmetros. Para a varredura não foi necessário ampliar os níveis usados no DoE, posto que já ocorreram restrições relevantes nos respectivos intervalos.

A Tabela 3 mostra como o sistema foi configurado para os experimentos dos dois cenários usando o resultado do DoE, os valores obtidos no *testbed* real e alguns padrões de mercado, como no caso da duração do *slot* de processamento baseada na forma de tarifação usada nas *spot instances*.

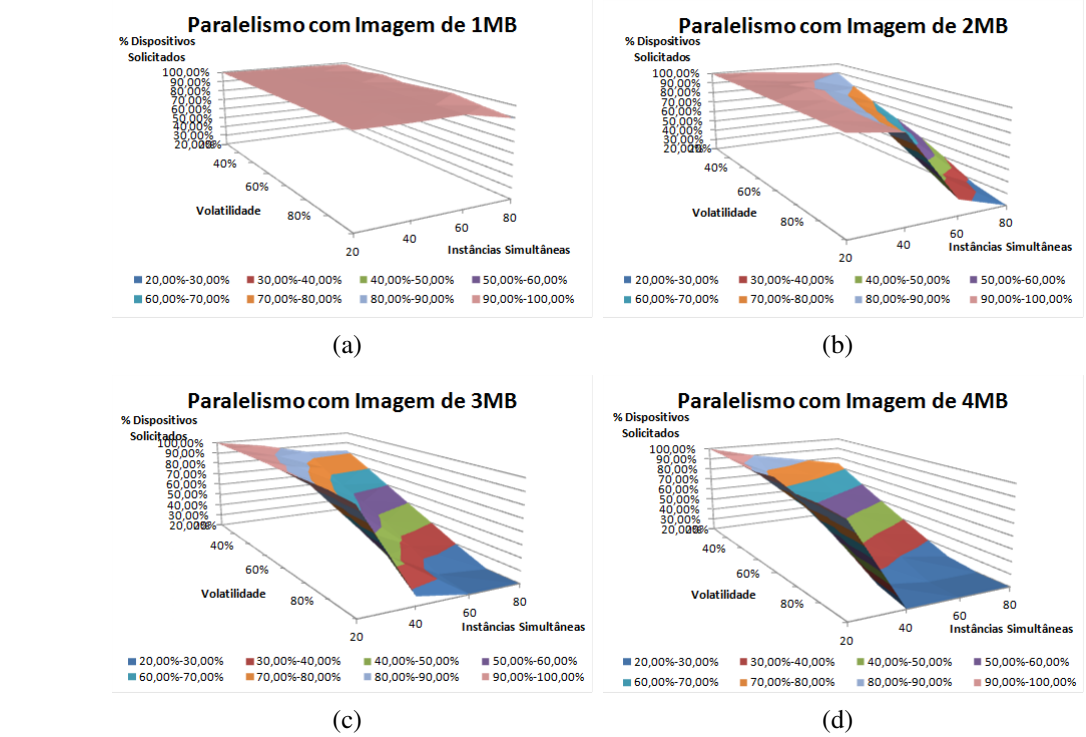
[8]Exceto no caso da *Duração da Instância*, com contribuição irrelevante, onde foi usado o nível "Baixo" com o objetivo de diminuir o tempo de execução de cada experimento.

Tabela 3. Parâmetros Usados nas Simulações

Parâmetro	**Cenário *Vazão Mínima***	**Cenário *Paralelismo Máximo***
Pico de Demanda ($\mathcal{P}$)	10.000 dispositivos	10.000 dispositivos
Taxa Canal Direto	1 Mbps	1 Mbps
Taxa Canal de *Broadcast*	1 Mbps	1 Mbps
Duração *slot* de processamento (σ)	1 hora	1 hora
Retardo Máximo	5 segundos	5 segundos
Disponibilidade Inicial ($\lvert\mathbb{D}^a(0)\rvert$)	100% da população	100% da população
Duração da Instância ($\mathcal{D}$)	10 *slots*	10 *slots*
Instâncias Simultâneas ($\mathcal{S}$)	50 instâncias	{20,40,60,80} instâncias
Tamanho da Imagem ($\mathcal{T}$)	{ 2, 5MB	{1MB,2MB,3MB,4MB}
População ($\lvert\mathbb{D}\rvert$)	{2.$\mathcal{P}$,3.$\mathcal{P}$,4.$\mathcal{P}$,5.$\mathcal{P}$, 6.$\mathcal{P}$,7.$\mathcal{P}$, 8.$\mathcal{P}$,9.$\mathcal{P}$}	10.$\mathcal{P}$
Volatilidade ($\mathcal{V}$)	{20%,30%,40%,50%, 60%,70%,80%,90%}	{20%,30%,40%,50%, 60%,70%,80%,90%}

4. Resultados e Análise

No primeiro experimento, realizado para o cenário de *Paralelismo Máximo*, o objetivo foi observar como a variação da volatilidade ($\mathcal{V}$), da quantidade de instâncias simultâneas ($\mathcal{S}$) e do tamanho da imagem da aplicação ($\mathcal{T}$) impacta na manutenção da quantidade desejada de dispositivos ativos para cada instância. Para eliminar a variável de contenção de dispositivos, a população foi configurada para 10 vezes o total da demanda concomitante esperada ($\lvert\mathbb{D}\rvert = 10 \times \mathcal{P}$). Para cobrir todas as combinações dos parâmetros de entrada foram realizados 128 experimentos - repetidos até que as médias obtidas tivessem 95% de confiança e erro máximo de 5% para mais ou para menos. A métrica de interesse observada foi o coeficiente médio de paralelismo das instâncias, $\overline{\Pi}$. Os resultados obtidos estão exibidos graficamente na Fig. 2.



(a) (b)

(c) (d)

Figura 2. *Paralelismo Máximo*: Métrica $\overline{\Pi}$ para diferentes tamanhos de imagem ($\mathcal{T}$)

Como pode ser observado na Fig. 2(a), quando lida com imagens de aplicação pequenas, o

controlador consegue compensar a perda de dispositivos em praticamente todos os regimes de volatilidade simulados, mesmo quando coordenando muitas instâncias simultâneas. Entretanto, à medida que o tamanho da imagem aumenta, aumenta o tamanho da mensagem de controle correspondente e diminui a capacidade do controlador de restabelecer o nível de paralelismo máximo das instâncias devido ao aumento proporcional do tempo de transmissão de cada mensagem de controle (Fig. 2(b)). Isso fica ainda mais evidenciado com o incremento no número de instâncias simultâneas, o que implica, na prática, no enfileiramento de mensagens de controle para serem enviadas pelo transmissor de *broadcast*. Esse efeito, que pode ser visualizado também nas figuras 2(c) e 2(d), é ampliado pelas restrições ao paralelismo máximo impostas neste cenário de uso, que ao limitar o tamanho que pode ser praticado para cada instância, não permite uma compensação antecipada das perdas através de redundância, o que diminuiria a quantidade de mensagens de controle reparatórias a serem enviadas e, consequentemente, a concorrência das instâncias pelo canal de *broadcast*. Associadamente, a inclusão de mecanismos adequados no controle de admissão pode otimizar o uso dos recursos do sistema através de um melhor escalonamento das instâncias ao longo do tempo.

No segundo experimento, realizado para o cenário de *Vazão Mínima*, o objetivo foi observar como a variação da volatilidade ($\mathcal{V}$) e do tamanho da população de dispositivos ($|\mathbb{D}|$) impactam na manutenção da quantidade desejada de *slots* de processamento completados, ou vazão, obtida em cada instância. Para controlar o nível de contenção de recursos, o tamanho da população foi iniciada em um patamar operacional mínimo, correspondente ao pico da demanda esperada acrescido da volatilidade inserida ($|\mathbb{D}| = \mathcal{P} \times (1+V)$), e foi sendo aumentada pela aplicação de um fator de contenção (um fator 2 equivale a uma população com o dobro da quantidade operacional mínima, um fator 3, ao triplo, e assim por diante). Para cobrir todas as combinações dos parâmetros de entrada foram realizados 64 experimentos - repetidos até que as médias obtidas tivessem 95% de confiança e erro máximo de 5% para mais ou para menos. A métrica de interesse principal foi a mesma usada no DoE, o coeficiente médio de vazão das instâncias, $\overline{\Phi}$. Os resultados obtidos estão exibidos na Fig. 3.

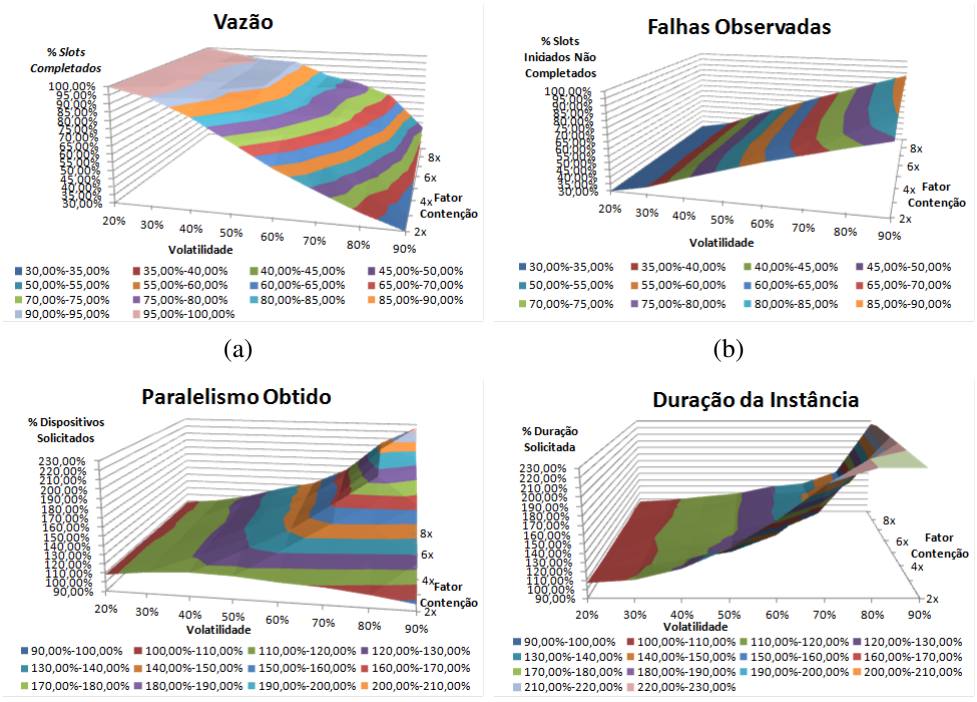


Figura 3. *Vazão Mínima*: Vazão, Duração, Paralelismo e Falhas observadas

Como ilustrado na Fig. 3(a), a quantidade média de *slots* de processamento completados por ciclo é fortemente afetada à medida que é inserida mais volatilidade no sistema. Nas configurações

com até 40% de volatilidade, ou seja, onde até 40% dos dispositivos alocados às instâncias falham em cada ciclo, foi possível manter níveis de vazão apenas 10% abaixo do solicitado, dependendo do fator de contenção do tamanho da população aplicado. Em tais níveis de volatilidade, o esforço de coordenação do provedor também é mantido controlado, como pode ser visto na Fig. 3(b), a qual traz o percentual de *slots* iniciados que não foram completados. Entretanto, à medida que a volatilidade é incrementada, a vazão entregue diminuiu consideravelmente apesar do aumento do custo operacional do provedor, com perdas de até 90% para a obtenção de vazão de apenas 30%. Cada *slot* não finalizado implica em custos operacionais, diretos e indiretos, para o provedor, principalmente no consumo de recursos de comunicação via canal de *broadcast* e canal direto dos dispositivos.

A métrica coeficiente médio de paralelismo das instâncias, $\overline{\Pi}$, também foi apurada para esse experimento. Pode ser visualizado na Fig. 3(c) que, por não haver restrição de tamanho para as instâncias, a quantidade de dispositivos ativos nas instâncias foi sendo aumentada à medida que a volatilidade percebida no sistema aumentava e ainda havia disponibilidade de recursos. O resultado do aumento do paralelismo repercute em uma atenuação dos efeitos da volatilidade sobre a vazão, como pode ser visualizado na Fig. 3(d), na qual a duração das instâncias torna a diminuir nos cenários com menor contenção de recursos mesmo em regimes de maior volatilidade. Obviamente, em contextos cuja disponibilidade de recursos não apresentem restrições ao nível de redundância praticados, como é o caso de redes de TV Digital com milhões de dispositivos, é possível aplicar níveis de paralelismo ainda maiores nas instâncias e ampliar a faixa de volatilidade onde alta vazão pode ser praticada. Entretanto, é necessário conciliar o nível de paralelismo com a capacidade do *Backend* e com o custo operacional do provedor.

5. Trabalhos Relacionados

Dentro do nosso conhecimento, nós somos o primeiro grupo a investigar o potencial do uso de redes de *broadcast* para a construção de infraestruturas computacionais distribuídas instantâneas e sob demanda [Batista et al. 2007, Costa et al. 2009]. Existem, entretanto, alguns outros trabalhos que apresentam convergência com a nossa pesquisa.

Fedak *at al.* [Fedak et al. 2010] construíram uma plataforma experimental para computação distribuída usando dispositivos de baixa capacidade conectados através de banda larga, chamada DSL-Lab, que oferece a possibilidade para pesquisadores realizarem experimentos em condições próximas àquelas que normalmente estão disponíveis com conexões domésticas com a Internet. Os resultados confirmam que é possível construir uma pilha completa de *software* em uma plataforma de design leve e de baixo custo sobre os dispositivos conectados em banda larga implementando gestão de recursos, eficiência energética, segurança e conectividade.

Neill *at al.* [Neill et al. 2011] investigam o uso de uma arquitetura de sistema heterogêneo que combina um *cluster* de computadores tradicionais, com um conjunto integrado de *set-top-boxes* para executar aplicações paralelas. Os resultados experimentais também confirmam que a rede de banda larga de processadores embarcados é uma nova e promissora plataforma para uma variedade de aplicações paralelas com uso intensivo de processamento e armazenamento (*computationally intensive and data-intensive grid applications*) e já é capaz de proporcionar ganhos significativos de desempenho para algumas classes de aplicações Open MPI.

6. Conclusão

Com o objetivo de investigar o uso de recursos terceirizados de alta granularidade, alta volatilidade e alta dispersão para a construção de *JiT Clouds* de alta vazão e usando uma nova arquitetura, chamada de *On-demand Distributed Computing Infrastructure* (OddCI), nós estudamos o comportamento do sistema e o impacto que os seus parâmetros têm sobre a sua eficiência através de simulação.

Nossos resultados mostram que, mesmo em cenários de altíssima rotatividade de nós autônomos e distribuídos geograficamente, é possível construir *JiT Clouds* com a disponibilidade coletiva adequada para atingir níveis controlados de vazão computacional usando os mecanismos de coordenação adequados.

No caso particular da aplicabilidade de sistemas OddCI para a descoberta, alocação e operação de *JiT DCs* dinâmicos, ficou evidenciado que a concorrência pelo uso do canal de *broadcast*, notadamente em contextos que envolvam a coordenação de muitas DCIs simultaneamente, requer a inclusão de mecanismos específicos em nível de controle de admissão e também na otimização da utilização dos recursos de comunicação de forma a permitir conciliar a qualidade do serviço prestado pelo provedor com os custos operacionais envolvidos. A investigação de mecanismos aplicáveis em tais aspectos será tratada em etapas futuras da nossa pesquisa, bem como a avaliação de aspectos de QoS e de consumo de energia (e recursos) nos dispositivos usados nas instâncias.

Referências

Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M., and Werthimer, D. (2002). Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45:56–61.

Andrzejak, A., Kondo, D., and Anderson, D. P. (2008). Ensuring collective availability in volatile resource pools via forecasting. In *Proceedings of the 19th IFIP/IEEE International Workshop on Distributed Systems*, DSOM '08, pages 149–161, Berlin, Heidelberg. Springer-Verlag.

Batista, C. E. C. F., de Araujo, T. M. U., dos Anjos, D. O., de Castro, M., Brasileiro, F., and Filho., G. (2007). Tvgrid: A grid architecture to use the idle resources on a digital tv network. In *Proc. 7th IEEE Intl Symposium on Cluster Computing and the Grid*, Rio de Janeiro, Brazil.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., and Mowbray, M. (2006). Labs of the World, Unite!!! *Journal of Grid Computing*, 4(3):225–246.

Cirne, W., Paranhos, D., Costa, L., Santos-Neto, E., and Brasileiro, F. e. a. (2003). *Running Bag-of-Tasks applications on computational grids: the MyGrid approach*. IEEE.

Costa, R., Bezerra, D. H. D., Vieira, D. A., Brasileiro, F., Sousa, D. M., and Filho, G. S. (2012). Oddci-ginga: A platform for high throughput computing using digital tv receivers. *IEEE/ACM International Conference on Grid Computing - GRID'12*, 0:155–163.

Costa, R., Brasileiro, F., de Souza Filho, G. L., and Sousa, D. M. (2009). Oddci: on-demand distributed computing infrastructure. In *2nd Workshop on Many-Task Computing on Grids and Supercomputers*, volume 16, pages 1–10, Portland, Oregon. ACM.

Costa, R., Brasileiro, F., Lemos, G., and Mariz, D. (2011a). Just in Time Clouds: Enabling Highly-Elastic Public Clouds over Low Scale Amortized Resources. In *3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011)*, Athens - Greece.

Costa, R., Brasileiro, F., Lemos, G., and Mariz, D. (2011b). Sobre a Amplitude da Elasticidade dos Atuais Provedores de Computação na Nuvem. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2011)*, Campo Grande - MS.

Fedak, G., Gelas, J.-P., Herault, T., Iniesta, V., Kondo, D., and Lefèvre, L. (2010). DSL-Lab: a platform to experiment on domestic broadband internet. In *9th Intl Symposium on Parallel and Distributed Computing (ISPDC'2010)*, Istanbul, Turkey.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. John Wiley and Sons.

Litzkow, M., Livny, M., and Mutka, M. (1988). Condor - a hunter of idle workstations. In *Proc. of the 8th International Conference of Distributed Computing Systems*, pages 104–111. IEEE.

Neill, R., Carloni, L. P., Shabarshin, A., Sigaev, V., and Tcherepanov, S. (2011). Embedded processor virtualization for broadband grid computing. In *Proc. of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID '11, pages 145–156, Washington, DC, USA. IEEE.

Varga, A. and Hornig, R. (2008). An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 60:1–60:10, Brussels, Belgium. ICST.

Replicação de Máquina de Estados Tolerante a Faltas Bizantinas usando Máquinas Virtuais Gêmeas

Fernando Dettoni[1], Lau Cheuk Lung[1,4], Miguel Correia[2], Aldelir Fernando Luiz[3,4]

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

[2]Instituto Superior Técnico - Universidade Técnica de Lisboa / INESC-ID - Portugal

[3]Câmpus Avançado de Blumenau - Instituto Federal Catarinense - Brasil

[4]Departamento de Automação e Sistemas - Universidade Federal de Santa Catarina - Brasil

{fdettoni, lau.lung}@inf.ufsc.br, miguel.p.correia@ist.utl.pt,

aldelir@das.ufsc.br

Abstract. *We present an architecture and an algorithm for Byzantine fault-tolerant state machine replication. Our algorithm explores the advantages of virtualization to reliably detect and tolerate faulty replicas, allowing the transformation of Byzantine faults into omission faults. Our approach reduces the total number of physical replicas from* $3f + 1$ *to* $2f + 1$*. Our approach is based on the concept of twin virtual machines, where there are two virtual machines in each physical host, each one acting as failure detector of its twin.*

Resumo. *Neste artigo é apresentada uma arquitetura e um algoritmo para replicação de máquina de estados tolerante a faltas bizantinas. São exploradas as vantagens fornecidas pela virtualização para detectar e tolerar réplicas faltosas, transformando faltas bizantinas em faltas de omissão. Com esta transformação, nossa abordagem reduz o número total de réplicas físicas necessárias de* $3f + 1$ *para* $2f + 1$*. Nossa abordagem é baseada no conceito de máquinas virtuais gêmeas, ou seja, na execução de duas máquinas virtuais em cada máquina física, cada uma funcionando como um detector de falhas de sua gêmea.*

1. Introdução

Cada vez mais, sistemas computacionais são usados em aplicações críticas e por isso necessitam operar corretamente apesar da presença de faltas. Estas faltas causam a parada total, como faltas de *crash*, ou arbitrárias, também chamadas de faltas bizantinas [Lamport et al. 1982]. Para garantir que o sistema funcione corretamente na presença de faltas é necessário o desenvolvimento de técnicas e algoritmos tolerantes a faltas bizantinas. Uma das técnicas mais utilizadas é a *replicação de máquina de estados* (RME) [Schneider 1990], que utiliza máquinas de estados determinísticas para oferecer um serviço replicado. Muitas abordagens tolerantes a faltas bizantinas (BFT) foram desenvolvidas utilizando RME (e.g. [Castro and Liskov 1999, Yin et al. 2003, Kotla et al. 2008]). Dentre estas, o algoritimo PBFT [Castro and Liskov 1999] é frequentemente considerado um pilar, pois foi o primeiro algoritmo com desempenho suficiente para muitas aplicações práticas.

Outra técnica de tolerância a faltas é o uso de *detectores de falhas não-confiáveis* [Chandra and Toueg 1996]. Apesar de originalmente criada para tolerar faltas de *crash*, algumas propostas posteriores foram capazes de estender a ideia para suportar faltas bizantinas [Doudou et al. 1999, Kihlstrom et al. 2003]. Estes detectores de falhas oferecem indicações sobre réplicas que aparentam estar faltosas.

A *virtualização* pode também ser considerada uma técnica de tolerância a faltas bizantinas pois introduz um nível de isolamento entre máquinas virtuais. Diversos trabalhos usam virtualização com o objetivo de proteger alguns componentes de falhas (ou ataques) de outros [Jiang and Wang 2007, Garfinkel and Rosenblum 2003, Laureano et al. 2004]. Tecnologias de virtualização são bastante aceitas pela indústria, e são largamente utilizadas atualmente, por exemplo por serviços de computação em nuvens como Amazon Web Services e Windows Azure.

O PBFT e vários outros algoritmos de replicação de máquinas de estados BFT têm um alto custo de implementação pois possuem uma resiliência de $n \geq 3f + 1$, ou seja, precisam de $n > 3f$ réplicas para tolerar f réplicas faltosas. Para diminuir esse custo, surgiram algumas abordagens que usam um *componente confiável* para limitar o comportamento das réplicas faltosas usando apenas $n \geq 2f + 1$ réplicas [Correia et al. 2004, Chun et al. 2007, Veronese et al. 2013]. Foram propostas também abordagens para executar apenas $f + 1$ réplicas, mantendo outras $2f$ *em espera*, ou seja, sem consumir tempo de CPU mas sendo ativadas em caso de falha [Distler et al. 2011, Wood et al. 2011].

Este artigo explora um outro ponto do espaço de projeto. O artigo apresenta uma nova arquitetura, chamada TwinBFT, para replicação de máquina de estados tolerante a faltas bizantinas eficiente baseada em virtualização. O objetivo é reduzir de $n \geq 3f + 1$ para $n \geq 2f + 1$ o número de máquinas físicas necessárias para tolerar f faltas. Além disso, o algoritmo apresentado reduz o número de passos de comunicação em funcionamento normal de 5 (do PBFT) para 3, sem a participação do cliente no acordo. Até onde sabemos, este é o primeiro algoritmo com este número de passos sem adotar uma abordagem especulativa [Kotla et al. 2008, Veronese et al. 2013], a qual envolve a participação do cliente no acordo.

Nossa proposta consiste na utilização de conjuntos de *máquinas virtuais gêmeas* executando o mesmo serviço da aplicação em cada uma das $n \geq 2f + 1$ máquinas físicas do sistema. Por simplicidade, neste artigo, adotaremos conjuntos de duas máquinas virtuais apenas. Cada máquina virtual executa o mesmo serviço, e cada conjunto de máquinas virtuais atua como uma réplica do esquema de replicação de máquina de estados. A ideia principal da proposta é utilizar cada máquina virtual como um detector de falhas para sua gêmea: ao enviar uma requisição para duas máquinas gêmeas, ambas devem fornecer a mesma resposta, caso contrário, ambas serão consideradas faltosas e sua resposta pode ser ignorada pelas outras réplicas. Assim, cada conjunto de máquinas virtuais gêmeas pode apenas funcionar corretamente ou omitir mensagens. No entanto, essas omissões são toleradas pela replicação de máquina de estados. Um *crash* corresponde também a omissões por tempo indeterminado, logo é também tolerado pelo uso de replicação.

A arquitetura e o algoritmo apresentados no artigo não pretendem oferecer a solução ideal para todos os casos. A utilização de máquinas virtuais é adequada para empresas que utilizam este tipo de tecnologia, como as de computação em nuvens. É também indicada quando não estejam disponíveis componentes confiáveis e ou não se possa esperar pela ativação de réplicas em espera em caso de falha. Nesses casos uma solução para replicação de máquinas de estados BFT eficiente – com apenas $2f + 1$ réplicas físicas ($4f + 2$ virtuais) – usando virtualização pode ser adequada.

Na Seção 2, serão mostrados alguns trabalhos relacionados, apresentando o estado da arte. A Seção 3 apresenta uma descrição do nosso modelo de sistema e suposições. Uma explicação detalhada do algoritmo será apresentada na Seção 4. Após isso, a Seção

5 apresenta algumas avaliações da implementação do algoritmo e a Seção 6 resume as conclusões.

2. Trabalhos Relacionados

Muitas propostas surgiram recentemente com o intuito de produzir serviços tolerantes a intrusão baseados em replicação de máquina de estados tolerante a faltas bizantinas (BFT). Sendo a primeira abordagem prática para protocolos BFT, o PBFT [Castro and Liskov 1999] é um dos protocolos mais bem sucedidos. Apesar de ser prático, os custos de implementação do PBFT são relativamente elevados, necessitando pelo menos 4 réplicas ($3f+1$ para $f=1$) e 5 passos de comunicação. Desta forma, surgiram vários algoritmos derivados do PBFT com dois objetivos: diminuir o número de réplicas e melhorar o desempenho.

Vários trabalhos propuseram alternativas para melhorar a resiliência reduzindo o número de réplicas. Yin *et al.* [Yin et al. 2003] introduziram uma arquitetura separando o serviço em duas camadas, uma responsável pelo acordo, contendo $3f+1$ réplicas, e outra responsável por executar as requisições com apenas $2f+1$ réplicas. Apesar de ainda serem precisas $3f+1$ réplicas, as réplicas utilizadas para a execução do serviço tendem a ter um custo bem mais elevados do que as réplicas de acordo.

Correia *et al.* [Correia et al. 2004] apresentaram a primeira solução para executar replicação de máquina de estados BFT com apenas $2f+1$ réplicas, utilizando um componente confiável distribuído. Mais tarde outro trabalho apresentou o primeiro algoritmo do gênero baseado num componente confiável local que implementa a abstração de *attested append-only memory* [Chun et al. 2007]. Veronese *et al.* [Veronese et al. 2013] propõem dois algoritmos que utilizam um componente confiável que fornece identificadores únicos para cada mensagem. O primeiro deles, MinBFT é capaz de reduzir o número de réplicas necessárias para $2f+1$ e o número de passos de comunicação para 4. O segundo é um algoritmo especulativo chamado MinZyzzyva que reduz o número de passos ainda mais, para 3, mantendo o número de réplicas em $2f+1$.

Em outra abordagem, Stumm *et al.* [Stumm et al. 2010] tiram vantagem de técnicas de virtualização para reduzir o número de réplicas necessárias para $2f+1$ desde que a VMM forneça uma forma de comunicação segura entre as réplicas. Esta abordagem, entretanto, requer que todas as réplicas estejam na mesma máquina e, portanto, não tolera faltas de *crash* na máquina física, ao contrário deste artigo.

Outra abordagem baseada na ideia de duas réplicas monitorando uma à outra é apresentada em [Mpoeleng et al. 2003], utilizando a abordagem *signal-on-fail*. Essa abordagem precisa de $4f+2$ máquinas físicas e requer comunicação síncrona e confiável entre cada par de réplicas, o que é um pressuposto difícil de garantir na prática. Nesta mesma linha, Inayat e Ezhilchevan apresentam um protocolo multicast BFT otimista com ordenação total baseado na abordagem *signal-on-fail*. Os autores demonstram que em uma execução livre de falhas, a abordagem tende a apresentar uma melhor performance do que outras abordagens BFT [Inayat and Ezhilchelvan 2006].

Vários trabalhos apresentaram soluções para melhorar o desempenho do PBFT. Cowling *et al.* [Cowling et al. 2006] apresentaram o HQ, um algoritmo baseado em quóruns e no PBFT que tem muito bom desempenho quando não há concorrência no acesso a dados. Kotla *et al.* [Kotla et al. 2008] apresentaram o Zyzzyva, um algoritmo que reduz o número de passos de comunicação na ausência de faltas. Ao invés de tentar chegar a um

acordo entre as réplicas antes de enviar a resposta, o serviço responde especulativamente ao cliente. O serviço precisa executar novamente a requisição e chegar a um acordo apenas no caso das respostas recebidas pelo cliente divergirem uma das outras. Esta abordagem se aproveita do fato de a maior parte das requisições ser livre de faltas, mas requer a habilidade de se reverter operações executadas previamente.

Como já mencionado na introdução, este artigo explora um outro ponto do espaço de projeto, usando virtualização para implementar replicação de máquina de estados BFT em apenas $2f + 1$ réplicas físicas (e o dobro de máquinas virtuais) e com apenas 3 passos de comunicação em funcionamento normal.

Muitos trabalhos usam virtualização para isolar componentes de software entre si. Dois dos primeiros usam virtualização para proteger um detector de intrusões dos próprios intrusos [Garfinkel and Rosenblum 2003, Laureano et al. 2004], enquanto outro mais recente usa a mesma ideia para proteger o monitor de um *honeypot* [Jiang and Wang 2007]. No entanto, a segurança do hipervisor é essencial para obter isolamento, por isso alguns trabalhos estudaram como melhorar essa segurança. Murray *et al.* [Murray et al. 2008] propuseram *desagregar* o sistema de virtualização como solução para diminuir a *trusted computing base* do sistema. O sistema NoHype vai mais longe retirando o hipervisor do caminho e executando as máquinas virtuais de modo nativo [Szefer et al. 2011]. O sistema HyperSafe usa outra abordagem: protege hipervisor de modo a detectar ataques que modifiquem o fluxo de controle, como *buffer overflows* [Wang and Jiang 2010].

3. Modelo de Sistema

Uma representação da arquitetura do sistema é mostrada na Figura 1. O sistema é composto por um conjunto de *n* máquinas físicas (ou *hosts*) *H* = $\{h_1, h_2, \ldots, h_n\}$ sendo que $n \geq 2f + 1$ e f é o número máximo de máquinas físicas faltosas. Cada *host* da Figura 1 contém um gerenciador de máquinas virtuais (VMM ou hipervisor) com duas máquinas virtuais (vm_i, vm'_i),chamadas gêmeas, executando emcada uma, uma réplica ou processo, respectivamente p_i e p'_i. Ambos os processos $\{p_i, p'_i\}$ executam o mesmo serviço e se comunicam entre si para validar cada mensagem de saída antes de enviar para outros processos.

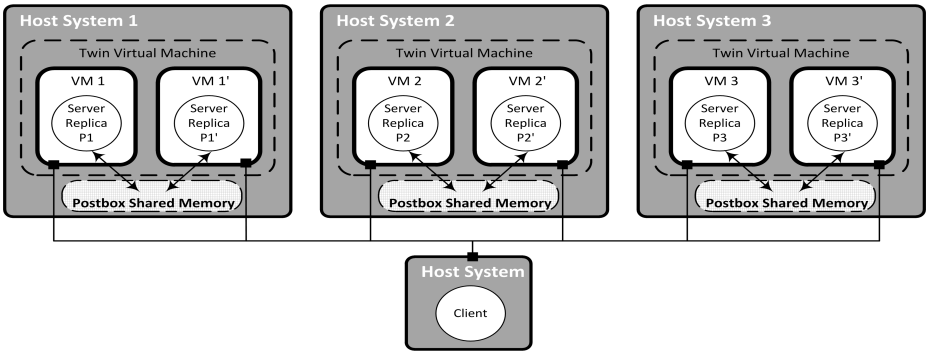


Figura 1. TwinBFT - Arquitetura com Máquinas Virtuais Gêmeas.

Assumimos que até f máquinas virtuais podem falhar de forma bizantina, ou arbitrária, mas apenas uma em cada máquina física. Quando uma máquina virtual falha arbitrariamente, o mecanismo de validação transforma essa falha numa omissão. Assim, assumimos também que até f máquinas físicas podem falhar por paragem ou omitindo mensagens, acidentalmente ou devido à falha de uma das suas máquinas virtuais. Para

substanciar o limite de f falhas é necessário recorrer a *diversidade*, ou seja, implementações diferentes dos processos e máquinas físicas [Bessani et al. 2009, Garcia et al. 2011, Gashi et al. 2007]. Essa diversidade reduz a chance de máquinas virtuais de um mesmo *host* sofrerem intrusão simultaneamente. Assumimos também que a virtualização fornece isolamento entre as máquinas virtuais e do próprio VMM / hipervisor. Como já explicado no final da Seção 2, técnicas como desagregação ou mecanismos como o HyperSafe e o NoHype podem ser usadas para tornar esta premissa mais realista.

Nenhuma suposição é feita sobre o tempo necessário para o sistema computar uma mensagem. A comunicação entre diferentes VMs dentro de um mesmo *host* é feita por um espaço de memória compartilhada, chamada *postbox*. Os processos nas VMs, em diferentes hosts, se comunicam pela rede, apenas por troca de mensagens. Esta rede pode falhar ao entregar mensagens, entregar fora de ordem, atrasar, ou duplicar mensagens.

Cada *host* pode assumir dois diferentes papéis: (1) *host* primário, sendo responsável por definir a ordem em que as requisições dos clientes serão executadas; e (2) *host* backup, que executa as requisições seguindo a ordem proposta pelo primário. Dentro de um *host* primário, um processo pode assumir dois diferentes papéis: (1) líder, que é responsável por atribuir um número de sequência para cada requisição recebida; e (2) seguidor, que executa as requisições seguindo a ordem definida pelo líder. Todos os processos em *hosts* backups são considerados seguidores. O *host* primário h_j é definido por $j = v \bmod |S|$, sendo *v* a visão atual, conforme definido na próxima seção. O processo líder primário em um *host* é, por definição, p_j.

Utilizamos técnicas criptográficas para autenticar mensagens e garantir sua autenticidade. Cada par de processos compartilha entre si uma chave secreta usada para gerar um vetor de MACs (*Message Authentication Code*) [Tsudik 1992] com um MAC válido para cada processo.

Por simplicidade, assumimos que cada máquina física comporta apenas duas máquinas virtuais (VMs), onde, para uma dada entrada, as respostas fornecidas por ambas têm que ser a mesma para que a máquina física não seja considerada faltosa. No entanto, o modelo pode facilmente ser estendido para mais VMs, segundo a condição $nVM \geq 2fVM + 1$, onde fVM é o número máximo de máquinas virtuais faltosas em uma máquina hospedeira. Neste caso, a máquina hospedeira não será considerada faltosa se a maioria das VMs ($fVM + 1$) fornecerem a mesma resposta.

4. Algoritmo TwinBFT

O algoritmo implementa uma replicação de máquina de estados no modelo de sistema que acabamos de apresentar. As réplicas se alternam seus papéis por uma sucessão de configurações chamadas de visão. Em cada visão, temos uma réplica primária que é responsável por definir a ordem das mensagens e encaminhar as requisições para todas as réplicas. Como mostrado por Schneider [Schneider 1990], a máquina de estados deve ser determinística e todas as réplicas precisam iniciar em um mesmo estado, caso contrário, a propriedade *safety* não pode ser garantida.

Nesta seção é apresentada nossa proposta de transformação das faltas bizantinas em faltas de omissão. Neste sentido, será apresentada uma adaptação do protocolo PBFT [Castro and Liskov 1999] para o modelo proposto. Em cada visão, apenas um réplica p_j é primária (líder), e é responsável por definir a ordem das mensagens e encaminhar as requisições para as réplicas do serviço. Se uma mensagem enviada por um *host* qualquer

for assinada por ambas as VMs deste *host*, ou seja, ambas resposta são iguais, assumimos esta mensagem como correta, já que apenas uma VM pode falhar ao mesmo tempo em um mesmo *host*.

4.1. Propriedades

Sendo uma Replicação de Máquina de Estados, é necessário assegurar as seguintes propriedades para garantir a corretude do serviço:

- **Ordem Total** (*safety*): cada requisição é executada sequencialmente na mesma ordem em cada réplica, i.e. apesar da replicação, as operações são executadas da mesma forma como seriam em um sistema centralizado;
- **Terminação** (*liveness*): qualquer requisição iniciada pelo cliente é terminada em algum momento, mesmo em caso de falha.

O algoritmo proposto fornece tanto *safety* quanto *liveness*, assumindo que não mais do que $f = \lfloor \frac{n-1}{2} \rfloor$ *hosts* são faltosos e, ao menos um processo *p* seja correto em cada *host* faltoso. Para garantir que as réplicas executarão as requisições na mesma ordem, todas as réplicas seguem a ordem definida pelo líder e esta ordem pode ser considerada correta desde que assinada por ambos os processos na réplica primária. Os algoritmos asseguram a corretude (ou *safety*) apesar do tempo levado para o processamento das requisições, porém uma certa sincronia é necessária para garantir as propriedades de progresso (i.e. *liveness*).

Como todas as réplicas seguem a ordem definida pelo líder primário, não é necessário um algoritmo de consenso pois pode-se confiar na ordem definida pela réplica primária, desde que as suposições prévias não sejam violadas. Isto ocorre porque quando o primário define a ordem, esta ordem apenas será seguida se ambos os processos no *host* primário tiverem acordo.

4.2. Detalhamento dos Algoritmos

Nesta seção o algoritmo será apresentado em detalhes. Primeiro, na Figura 2 é mostrado um diagrama com uma visão geral dos passos do protocolo. A configuração mostrada assume $f = 1$, sendo necessários três *hosts*, cada um com duas VMs. Cada par de VMs em um mesmo *host* se comunica através de um canal confiável FIFO chamado *postbox*. A *postbox* pode ser mais rápida do que a comunicação via rede, usando um espaço de memória compartilhado entre as VMs, fornecido pela VMM.

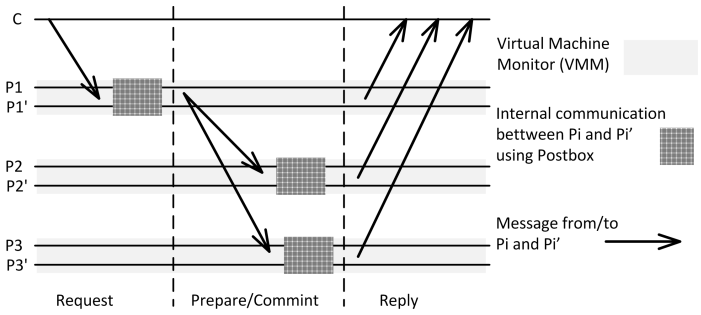


Figura 2. Passos do algoritmo em execução normal com $f = 1$.

O algoritmo funciona basicamente como segue:

1. Cliente envia a requisição para ambos os processos na réplica primária;

2. O líder primário p_i define um número de sequência e insere uma mensagem “ORDER"na *postbox*;
3. O seguidor primário p_i' lê a mensagem da *postbox*, pega o número de sequência e insere na postbox a mensagem “ORDER” contendo a requisição original e o número de sequência recebido;
4. Ambos os processos assinam a mensagem “ORDER” lida da *postbox* e enviam para todas as réplicas;
5. Assim que cada processo nas réplicas recebe a mensagem “ORDER”, executa a operação e insere na *postbox* uma mensagem “REPLY” assinada;
6. Quando um processo lê da *postbox* uma mensagem “REPLY", compara com a mensagem gerada localmente e, se todos os argumentos forem idênticos, adiciona sua assinatura e envia a resposta para o cliente;
7. Se o cliente receber ao menos $f + 1$ respostas corretamente assinadas de diferentes réplicas, aceita a resposta.

Como um exemplo de cliente, o Algoritmo 1 mostra uma execução normal em que o cliente envia uma requisição (linha 3) para o serviço e aguarda até receber ao menos $f + 1$ respostas válidas de diferentes réplicas (linhas 4-6). A requisição tem o formato $\langle$REQUEST, c, seq, op$\rangle_{\sigma_c}$ sendo que *c* é o identificador do cliente, *seq* é o número de sequência em relação ao cliente, e *op* é a operação a ser executada. Se o cliente não receber $f+1$ respostas em um determinado tempo, reenvia a requisição para todas as réplicas (linha 8).

Algoritmo 1 Algoritmo executado pelo cliente

```
1: procedure REQUEST                                          ▷ Envia uma nova requisição
2:     Δ_c ← default timeout
3:     multi_send(⟨REQUEST, c, seq, op⟩_σc)                   ▷ Envia a requisição para ambos os processos na réplica primária
4:     repeat
5:         buffer ← buffer ∪ recv()
6:     until f + 1 respostas correspondentes ∃ buffer /* Timer em uma thread separada */
7:     if Δ_c expired then
8:         multi_send(⟨REQUEST, c, seq, op⟩_σc)               ▷ Envia a mensagem para todas as réplicas
9:     end if
10: end procedure
```

4.3. Operação normal do protocolo

O Algoritmo 2, executado em cada um dos processos possui duas tarefas concorrentes. A Tarefa 1 é responsável por ler as mensagens recebidas através da rede. A Tarefa 2 é responsável por ler as mensagens recebidas a partir da *postbox*, inseridas por sua gêmea. O estado de cada processo é composto pelo estado do serviço, um *buffer* de mensagens e a visão atual. Este estado é compartilhado entre as tarefas.

Quando qualquer um dos processos $\{p_i, p_i'\}$ no *host* primário recebe a requisição do cliente, p_i gera um novo número de sequência *n* e cria uma mensagem $\langle\langle$ORDER, p_i, v, n, dm$\rangle_{\sigma_{p_i}}$, m$\rangle$, sendo *v* o número da visão atual, e *dm* o resumo da mensagem *m* (linhas 4-6). Assim que p_i' lê a mensagem “ORDER” inserida na postbox por p_i e possui sua respectiva mensagem “REQUEST"no buffer, pega o número de sequência proposto por p_i, cria uma mensagem “ORDER”, assina e insere na *postbox* (linha 23). Quando cada um dos processos lê a mensagem “ORDER” da *postbox*, verifica se todos os parâmetros correspondem aos processados localmente e, em caso positivo, adiciona sua própria assinatura à mensagem de sua gêmea e envia para todas as réplicas (linha 25).

Algoritmo 2 Algoritmo em operação normal

```
/* Tarefa 1:  rede */
1: loop
2:     msg ← receive()
3:     if received (REQUEST) then
4:         if é o líder primário then
5:             n ← n + 1
6:             postbox.append(⟨⟨ORDER, p_i, v, n, dm⟩_σpi , msg⟩)
7:         else if é o seguidor primário then
8:             buffer ← buffer ∪ msg
9:         else
10:            envia mensagem para primários
11:            inicia Δ_p
12:        end if
13:    else if received (ORDER) then
14:        termina Δ_p
15:        postbox.append(⟨REPLY, p_i, v, seq, c, res⟩ >_σpi )
16:    end if
17: end loop
    /* Tarefa 2:  postbox */
18: loop
19:    msg ← postbox.read()
20:    if received (ORDER) then
21:        if todos os parâmetros correspondem aos computados localmente then
22:            if is the primary follower then
23:                postbox.append(⟨⟨ORDER, p_i, v, m.n, dm⟩_σpi , msg⟩)
24:            end if
25:            multicast(⟨⟨⟨ORDER, p'_i, v, n, dm⟩_σp'i ⟩σp_i , msg⟩)
26:        end if
27:    else if received (REPLY) then
28:        if todos os parâmetros correspondem aos computados localmente then
29:            reply_to_client(⟨⟨REPLY, p'_i, v, seq, c, res⟩_σp'i ⟩_σpi )
30:        end if
31:    end if
32: end loop
```

Para cada mensagem "ORDER" recebida, as réplicas consideram válida caso as seguintes condições estejam cumpridas:

- A mensagem é corretamente assinada, i.e., se recebida pela rede deve estar assinada por ambas máquinas gêmeas no remetente, e se recebida pela *postbox* deve estar assinada pela sua própria gêmea;
- A visão na mensagem é a visão atual;
- Não aceitou outra mensagem "ORDER" com o mesmo número de sequência para uma requisição diferente;
- A número de sequência está entre um valor mínimo *h* e máximo *H* de possíveis números de sequência (na prática, se esta verificação for feita pelo seguidor primário quando lê a mensagem "ORDER"da *postbox* as réplicas *backup* nunca receberão um número de sequência fora de *h* e *H*).

Ao receber uma mensagem "ORDER" de ambos os processos em uma máquina física, cada um dos processos $\{p_i, p'_i\}$ verifica se a mensagem é válida e, em caso positivo, executa operação e cria a mensagem $\langle$REPLY, p_i, v, seq, c, res$\rangle_{\sigma_{p_i}}$, sendo *res* o resultado da operação executada, e insere na *postbox* (linha 15). Quando um processo lê a mensagem "REPLY" da *postbox*, compara os seus parâmetros e, se forem idênticos aos processados localmente, assina a mensagem e envia ao cliente (linha 29).

Quando o cliente recebe uma mensagem "REPLY", aceita como válida se as seguintes condições forem verdadeiras:

- Está assinada por ambos os processos $\{p_i, p'_i\}$ no *host* remetente.
- Ainda não aceitou uma mensagem válida remetida pela gêmea do *host* remetente.

O cliente aguarda até ter recebido ao menos $f + 1$ mensagens válidas das réplicas para aceitar o resultado. Se estas mensagens não forem recebidas em um determinado tempo, envia a requisição para todas as réplicas, podendo ocorrer uma troca de visão por suspeita do primário.

4.4. Maior Número de Máquinas Gêmeas

Conforme citado anteriormente, é possível que mais do que duas máquinas virtuais componham o grupo de máquinas gêmeas em cada *host*. Desta forma, o algoritmo confere uma maior resistência permitindo que uma ou mais máquinas virtuais se comportem de forma bizantina e ainda assim o *host* seja considerado correto, desde que exista uma maioria de respostas idênticas. Esta maioria é indicada pela quantidade de assinaturas presentes em cada mensagem.

Desta forma, ao criar uma nova mensagem, cada processo publica sua proposta de mensagem assinada na *postbox* (linhas 6 e 15). Ao ler as mensagens da postbox (linhas 21 e 28), é necessário verificar também se, juntamente com sua assinatura, a mensagem estará assinada por uma maioria das máquinas virtuais do *host*. Em caso positivo, assina e envia para o destinatário. Caso contrário, apenas insere novamente na *postbox* a mensagem com sua assinatura até que esteja assinada por uma quantidade suficiente de máquinas virtuais no *host*.

4.5. Coleta de Lixo

Para prevenir que ocorra um estouro na memória, é necessário um mecanismo que descarte as mensagens antigas armazenadas no *buffer*, isto é, que efetue uma coleta de lixo (i.e. *garbage collection*). Para isso, o algoritmo gera periodicamente um *checkpoint*, após algum número constante de requisições recebidas. Ao chegar em um *checkpoint*, cada processo cria uma mensagem $\langle$CHECKPOINT, p_i, v, n, d$\rangle_{\sigma_{p_i}}$, sendo que *n* é o número da última requisição processada e *d* é um sumário do estado atual de p_i, e insere na *postbox*.

Cada máquina gêmea lerá a mensagem da *postbox* e assim que atingir o mesmo *checkpoint*, confirma se o estado recebido é o mesmo estado local e, em caso positivo, adiciona sua própria assinatura à mensagem "CHECKPOINT" e envia para as outras réplicas. Quando um processo recebe $f + 1$ mensagens "CHECKPOINT" corretamente assinadas de diferentes réplicas, para o mesmo número de visão *v*, o mesmo número de sequência *n* e o mesmo estado *d*, aceita como último checkpoint válido e remove do buffer todas as mensagens com número de sequência menor do que *n*.

4.6. Protocolo de Troca de Visão

A principal função do protocolo de troca de visão é manter o serviço progredindo mesmo na presença de um primário faltoso. Se o primário é faltoso, as réplicas backup nunca receberão uma mensagem "ORDER" válida e, portanto, devem definir um novo primário. Se um cliente não receber respostas suficientes em um tempo hábil, envia a requisição para todos os processos do sistema. Se uma réplica backup recebe uma requisição diretamente do cliente, verifica se já processou esta requisição e, em caso positivo, reenvia a resposta ao cliente. Caso contrário, encaminha a requisição aos processos da réplica primária e inicia um contador Δ_p (linhas 6-11).

Ao receber a mensagem "ORDER" correspondente do primário, o contador Δ_p é cancelado (linha 14) e o algoritmo continua normalmente. Se nenhuma mensagem "ORDER for recebida até o contador expirar, o processo inicia o protocolo de troca de visão,

apresentado no Algoritmo 3, inserindo na *postbox* a mensagem $\langle$VIEW-CHANGE, p_i, v+1, n, C, P$\rangle_{\sigma_{p_i}}$, sendo *n* o número de sequência do último checkpoint válido, *C* um conjunto composto por $f + 1$ mensagens "CHECKPOINT" garantindo o último *checkpoint*, e *P* um conjunto com todas as requisições processadas após o último *checkpoint* (linha 2). Se sua gêmea concordar com a troca de visão, a partir da mensagem "VIEW-CHANGE" lida da *postbox*, adiciona sua própria assinatura e envia a mensagem para todos os processos (linha 5).

Algoritmo 3 Algoritmo do protocolo de troca de visão

```
/* Tarefa 1:  rede */
1: loop
2:     msg ← receive()
3:     if received (VIEW-CHANGE) then
4:         buffer ← buffer ∪ msg
5:         if buffer contém ao menos um VIEW-CHANGE com n = msg.n ∧ d = msg.d then
6:             envia mensagem aos primários
7:             if i = msg.v mod |S| then
8:                 postbox.append(⟨NEW-VIEW, p_i, msg.v, V, P⟩_σpi )
9:             end if
10:        end if
11:    else if received (NEW-VIEW) then
12:        buffer ← buffer ∪ msg
13:        for all req in msg.P do
14:            garante que req foi processada e armazenada no log.
15:        end for
16:    end if
17: end loop
/* Tarefa 2:  postbox */
1: loop
2:     msg ← postbox.read()
3:     if received (VIEW-CHANGE) then
4:         if all parameters corresponds the ones locally computed then
5:             multi_send(⟨⟨VIEW-CHANGE, p'_i, v+1, n, C, P⟩_σp'i ⟩_σpi )
6:         end if
7:     else if received (NEW-VIEW) then
8:         if todos os parâmetros correspondem aos computados localmente then
9:             multi_send(⟨⟨NEW-VIEW, p'_i, v+1, V, P⟩_σp'i ⟩_σpi )
10:        end if
11:    end if
12: end loop
/* Tarefa 3:  timeout */
1: procedure TIMEOUT_EXPIRE                          ▷ Ao expirar o timeout Δ_p
2:     postbox.append(⟨VIEW-CHANGE, p_i, v+1, n, C, P⟨_σpi )
3: end procedure
```

Ao receber $f+1$ mensagens "VIEW-CHANGE" válidas de diferentes *hosts*, ambos os processos $\{p_i, p'_i\}$ no *host* h_i verificam se h_i é primário. Se sim, *p* aceita a troca de visão criando e inserindo na *postbox* a mensagem $\langle$NEW-VIEW, p_i, v+1, V, P$\rangle_{\sigma_{p_i}}$, sendo *V* um conjunto contendo as mensagens "VIEW-CHANGE" que originaram a troca de visão, e *P* um conjunto contendo as mensagens "ORDER" enviadas após o último *checkpoint* válido (linhas 5-8). Quando um processo *p* lê da *postbox* uma mensagem "NEW-VIEW", verifica se seus parâmetros são idênticos aos processados localmente e, caso sejam, adiciona sua assinatura e envia a mensagem para todas as réplicas (linha 9).

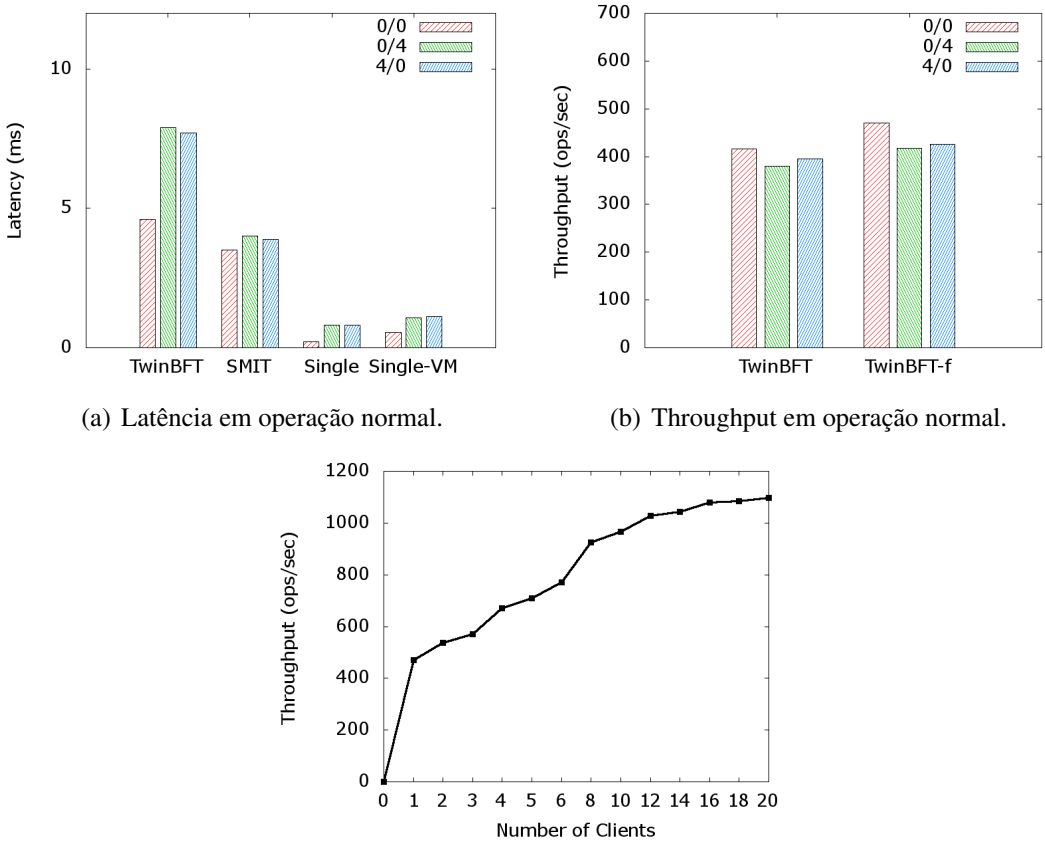
Quando qualquer processo *p* recebe uma mensagem "NEW-VIEW" do novo primário, verifica se: (1) a mensagem está devidamente assinada, (2) contém um conjunto *V* com $f + 1$ mensagens "VIEW-CHANGE" válidas. Se as condições forem satisfeitas, reexecuta todas as requisições contidas em *P* para a nova visão (linhas 11-14).

5. Implementação e Resultados

Para avaliar a performance da abordagem, foi escolhido um método experimental. O algoritmo foi implementado em Java, de acordo com as especificações da versão 1.6. Os canais de comunicação foram feitos pela utilização de *channels* do Java NIO, usando TCP com MACs (Códigos de Autenticação de Mensagem).

Executamos nossos experimentos em três servidores Intel®Core™i7 3.8Ghz com Debian 7.0 "wheezy" (Kernel 3.2.0 x86-64) e VMM Xen Hypervisor 4.1.3. Cada máquina virtual foi configurada com 2GB de memória e duas CPUs virtuais, equipadas com SUN's JDK 1.6.0_29.

Como medida de avaliação, foi adotado latência e *throughput*, por serem largamente utilizadas neste tipo de avaliação e porque permitem uma verificação simplificada da eficiência do sistema [Jain 1991]. Os resultados foram obtidos através de *microbenchmarks* em diferentes condições de carga. A latência foi obtida a partir de algumas requisições com um único cliente enviando uma requisição por vez, e o *throughput* medindo quantas requisições o sistema consegue responder em uma unidade de tempo. O sistema foi avaliado a partir de *microbenchmarks* para que o custo fosse avaliado sem a influência do serviço. Para estes *microbenchmarks*, foi utilizado um serviço *stateless* com uma operação nula, variando o tamanho das requisições e respostas entre 0KB e 4KB.



(a) Latência em operação normal.

(b) Throughput em operação normal.

(c) Throughput com múltiplos clientes.

Figura 3. Desempenho verificado para o *TwinBFT* em operação normal.

A fim de avaliar o desempenho da solução proposta, o algoritmo foi executado em condições, em que foram enviadas 10.000 requisições de um único cliente, em três

diferentes cargas: 0/0, 0/4 e 4/0. Eles representam, respectivamente, uma requisição e resposta nula, uma requisição nula e uma resposta de 4KB, e uma requisição de 4KB e resposta nula. Todos os tempos foram medidos pelo cliente, a partir da leitura de seu relógio local antes do envio da requisição e após o recebimento de uma resposta válida.

Na Figura 3(a), são mostradas as diferentes latências em cada carga. Para obter a latência, 10.000 requisições foram enviadas e executadas sequencialmente, de forma individual, sendo a latência o tempo médio destas requisições. A abordagem é comparada com o algoritmo SMIT [Stumm et al. 2010], que contém certas semelhanças pela utilização de máquinas virtuais e memória compartilhada, porém não suporta faltas de *crash*. *Single* se refere a uma implementação do serviço centralizado, sem máquinas virtuais enquanto *Single-VM* representa uma execução centralizada dentro do isolamento de uma máquina virtual. É esperado que a abordagem proposta apresente uma avaliação pior do que versões centralizadas pois estas não são tolerantes a faltas.

O *throughput*, como mostrado na Figura 3(b) foi calculado baseado no tempo total para execução das 10.000 requisições enviadas simultaneamente para o serviço. No primeiro grupo, são mostradas as medidas para o serviço em seu caso normal, sem falhas. Em TwinBFT-f mostramos o *throughput* do algoritmo em caso de faltas, sendo que em 1% das requisições a réplica primária se mostra faltosa, gerando uma troca de visão para manter a consistência. Para uma comparação do desempenho em diferentes cargas do algoritmo, a Figura 3(c) apresenta o *throughput* quando temos mais de um cliente fazendo requisições simultaneamente. O número de requisições por segundo se eleva com o aumento de clientes simultâneos até se estabilizar em torno de 1000 operações por segundo quando passa a ficar mais limitado pela capacidade das réplicas do que pela capacidade dos clientes.

Na Tabela 1 é mostrada uma comparação entre a abordagem proposta e outros algoritmos BFT na literatura. Todos os número consideram apenas execuções sem faltas. Os benefícios no uso de máquinas virtuais gêmeas são visíveis no número de réplicas e passos de comunicação. Enquanto nossa abordagem tem o menor número de réplicas, juntamente com [Correia et al. 2004, Chun et al. 2007, Veronese et al. 2013], ela tem o mesmo número de passos de comunicação de algoritmos especulativos [Kotla et al. 2008, Veronese et al. 2013], mesmo em caso de faltas. Algoritmos especulativos, entretanto requerem um número maior de passos em casos com faltas, além de envolverem o cliente no protocolo, perdendo transparência.

Tabela 1. Comparação entre as propriedades de protocolos BFT.

Protocolos Avaliados	*Propriedades e Características*				
	Número de réplicas	**Número de processos**	**Número de máquinas físicas**	**Passos de comunicação**	**Especulativo/ Otimista**
PBFT [Castro and Liskov 1999]	$3f+1$	$3f+1$	$3f+1$	5	não
Zyzzyva [Kotla et al. 2008]	$3f+1$	$3f+1$	$3f+1$	3 / 5[1]	sim
BFT-TO [Correia et al. 2004]	$2f+1$	$2f+1$	$2f+1$	5	não
A2M-PBFT-EA [Chun et al. 2007]	$2f+1$	$2f+1$	$2f+1$	5	não
MinBFT [Veronese et al. 2013]	$2f+1$	$2f+1$	$2f+1$	4	não
MinZyzzyva [Veronese et al. 2013]	$2f+1$	$2f+1$	$2f+1$	3 / 5[1]	sim
TwinBFT	$2f+1$	$4f+2$	$2f+1$	3	não

Como a abordagem proposta utiliza duas máquinas virtuais em cada réplica, possui um número maior de processos, apesar do número de máquinas físicas ser o mesmo de [Correia et al. 2004, Chun et al. 2007, Veronese et al. 2013].

[1]Necessita de dois passos adicionais para confirmar a requisição no caso de suspeita de falta.

6. Conclusões

A partir da exploração de algumas técnicas de virtualização, foi possível a proposta de um algoritmo BFT alternativo. Foi mostrado que é possível implementar um algoritmo RME confiável com $2f + 1$ réplicas físicas em um ambiente assíncrono. Apesar de necessitar de um canal de comunicação confiável para a comunicação entre as máquinas virtuais, acreditamos que a virtualização é vastamente disponível atualmente e pode fornecer um isolamento entre as réplicas e o mundo exterior. Além do mais, foi possível reduzir também o número de passos de comunicação, reduzindo o custo desta comunicação.

Referências

Bessani, A., Daidone, A., Gashi, I., Obelheiro, R. R., Sousa, P., and Stankovic, V. (2009). Enhancing fault / intrusion tolerance through design and configuration diversity. In *Proceedings of the 3rd Workshop on Recent Advances on Intrusion-Tolerant Systems*.

Castro, M. and Liskov, B. (1999). Practical Byzantine fault tolerance. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, pages 173–186.

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

Chun, B.-G., Maniatis, P., Shenker, S., and Kubiatowicz, J. (2007). Attested append-only memory: making adversaries stick to their word. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles*, pages 189–204.

Correia, M., Neves, N. F., and Verissimo, P. (2004). How to tolerate half less one Byzantine nodes in practical distributed systems. In *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*, pages 174–183.

Cowling, J., Myers, D., Liskov, B., Rodrigues, R., and Shrira, L. (2006). HQ-Replication: A hybrid quorum protocol for Byzantine fault tolerance. In *Proceedings of 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 177–190.

Distler, T., Popov, I., Schröder-Preikschat, W., Reiser, H. P., and Kapitza, R. (2011). SPARE: Replicas on hold. In *Proceedings of the 18th Network and Distributed System Security Symposium*, pages 407–420.

Doudou, A., Garbinato, B., Guerraoui, R., and Schiper, A. (1999). Muteness failure detectors: Specification and implementation. In *Proceedings of the Third European Dependable Computing Conference on Dependable Computing*, pages 71–87. Springer-Verlag.

Garcia, M., Bessani, A., Gashi, I., Neves, N., and Obelheiro, R. (2011). OS diversity for intrusion tolerance: Myth or reality? In *Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks*, pages 383 –394.

Garfinkel, T. and Rosenblum, M. (2003). A virtual machine introspection based architecture for intrusion detection. In *Proceedings of the Network and Distributed Systems Security Symposium*.

Gashi, I., Popov, P. T., and Strigini, L. (2007). Fault tolerance via diversity for off-the-shelf products: A study with SQL database servers. *IEEE Transactions on Dependable and Secure Computing*, 4(4):280–294.

Inayat, Q. and Ezhilchelvan, P. (2006). A performance study on the signal-on-fail approach to imposing total order in the streets of byzantium. In *In Proceedings of the 36th International Conference on Dependable Systems and Networks*, pages 578–587.

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

Jiang, X. and Wang, X. (2007). Out-of-the-box monitoring of VM-based high-interaction honeypots. In *Proceedings of the 10th International Symposium on Recent Advances in Intrusion Detection*.

Kihlstrom, K. P., Moser, L. E., and Melliar-Smith, P. M. . (2003). Byzantine fault detectors for solving consensus. *The Computer Journal*, 46.

Kotla, R., Clement, A., Wong, E., Alvisi, L., and Dahlin, M. (2008). Zyzzyva: speculative Byzantine fault tolerance. *Commun. ACM*, 51:86–95.

Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401.

Laureano, M., Maziero, C., and Jamhour, E. (2004). Intrusion detection in virtual machine environments. In *Proceedings of the 30th Euromicro Conference*, pages 520–525.

Mpoeleng, D., Ezhilchelvan, P., and Speirs, N. (2003). From crash tolerance to authenticated Byzantine tolerance: A structured approach, the cost and benefits. In *Proceedings of the IEEE/IFIP 33rd International Conference on Dependable Systems and Networks*, pages 227–236.

Murray, D. G., Milos, G., and Hand, S. (2008). Improving Xen security through disaggregation. In *Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 151–160.

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

Stumm, V., Lung, L. C., Correia, M., da Silva Fraga, J., and Lau, J. (2010). Intrusion tolerant services through virtualization: A shared memory approach. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 768 –774.

Szefer, J., Keller, E., Lee, R. B., and Rexford, J. (2011). Eliminating the hypervisor attack surface for a more secure cloud. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 401–412.

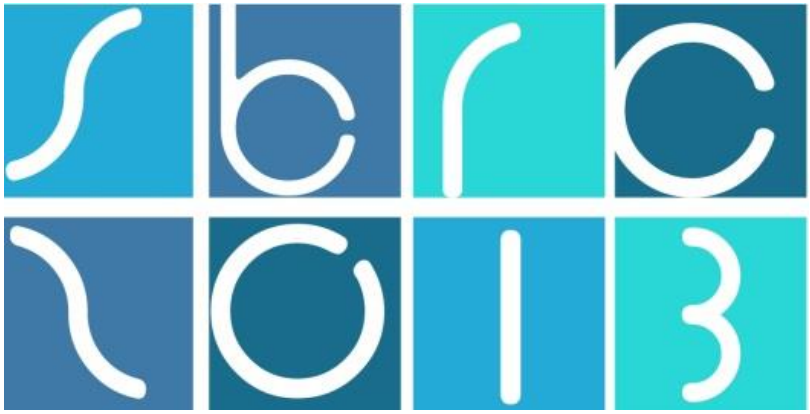
Tsudik, G. (1992). Message authentication with one-way hash functions. *SIGCOMM Comput. Commun. Rev.*, 22(5):29–38.

Veronese, G. S., Correia, M., Bessani, A. N., C., L., and Verissimo, P. (2013). Efficient Byzantine fault tolerance. *IEEE Transactions on Computers*, 62(1):16–30.

Wang, Z. and Jiang, X. (2010). HyperSafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In *Proceedings of the IEEE Security and Privacy Symposium*, pages 380–395.

Wood, T., Singh, R., Venkataramani, A., Shenoy, P., and Cecchet, E. (2011). ZZ and the art of practical BFT execution. In *Proceedings of the 6th ACM SIGOPS/EuroSys European Systems Conference*, pages 123–138.

Yin, J., Martin, J.-P., Venkataramani, A., Alvisi, L., and Dahlin, M. (2003). Separating agreement from execution for Byzantine fault tolerant services. *SIGOPS Oper. Syst. Rev.*, 37:253–267.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 23

Bancos de Dados Distribuídos

Processamento Justo de Transações em Bancos de Dados Tolerantes a Faltas Bizantinas

Aldelir Fernando Luiz[1,2], Lau Cheuk Lung[2,3], Miguel Correia[4]

[1]Câmpus Avançado de Blumenau - Instituto Federal Catarinense - Brasil

[2]Departamento de Automação e Sistemas - Universidade Federal de Santa Catarina - Brasil

[3]Departamento de Informática e Estatística - Universidade Federal de Santa Catarina - Brasil

[4]Instituto Superior Técnico - Universidade Técnica de Lisboa / INESC-ID - Portugal

aldelir@das.ufsc.br, lau.lung@inf.ufsc.br, miguel.p.correia@ist.utl.pt

Resumo. *Nos últimos tempos, tem havido um crescente interesse na investigação e proposição de mecanismos para tolerar faltas Bizantinas em transações. As soluções propostas recentemente são baseadas no método de processamento e concorrência otimista, porém, tal método permite que parcela de transações sejam suscetíveis ao fenômeno conhecido como inanição (ou starvation). Este artigo apresenta uma solução para este problema, em que é proposto um protocolo para o processamento justo de transações, por meio de uma abordagem adaptativa e tolerante a faltas Bizantinas. O protocolo apresentado é robusto e garante a terminação de uma transação em até $f+1$ tentativas, no melhor caso.*

Abstract. *Nowadays, there is an increasing interest on mechanisms for Byzantine fault tolerance in transaction processing. Several recently proposed mechanisms follow an optimistic approach. However, optimistic transaction processing is inherently susceptible to a phenomenon known as starvation. This paper presents a solution to the starvation problem: a fair and starvation-free protocol for an environment subject to Byzantine faults. Our protocol is adaptive and robust, guaranteeing the termination of a transaction in $f+1$ attempts in favorable conditions.*

1. Introdução

A replicação de dados tem sido amplamente empregada no contexto de sistemas computacionais, em especial em sistemas de computação distribuída, como ferramenta para permitir o gerenciamento confiável dos dados processados nas computações. Em geral, os dados são manipulados por meio de abstrações conhecidas na literatura como transações [Bernstein et al. 1987]. Uma transação consiste em uma unidade lógica de trabalho que assegura a execução atômica de uma sequência de operações de manipulações de dados em um sistema de computação, a qual garante que todas estas operações sejam refletidas corretamente no sistema, ou nenhuma o será. Neste sentido, a replicação de dados representa ainda, um desafio em se tratando do gerenciamento de transações em ambientes sujeitos a faltas de alguma natureza. Não obstante, a evolução de tecnologias e facilidades para o processamento de transações incorre no surgimento de novos modelos e classes de aplicações, que por sua vez trás novos desafios à confiabilidade.

Neste sentido, trabalhos recentes têm proposto soluções para o gerenciamento de transações sobre dados replicados, mais especificamente em sistemas

de bancos de dados (BD) [Gashi et al. 2007, Vandiver et al. 2007, Luiz et al. 2011, Garcia et al. 2011, Pedone et al. 2011]. Dentre os trabalhos encontrados na literatura, [Luiz et al. 2011, Garcia et al. 2011, Pedone et al. 2011] usam a abordagem otimista para o processamento de transações [Kung and Robinson 1981], enquanto [Gashi et al. 2007, Vandiver et al. 2007] são protocolos pessimistas. Em protocolos otimistas se pressupõe que durante o processamento das transações são raras as interferências e/ou conflitos, então a serialização é verificada apenas na confirmação da transação. Por outro lado, em protocolos pessimistas se admite que ocorrerão interferências e/ou conflitos entre transações, e por isso eles adotam mecanismos de bloqueios para garantir a serialização enquanto a transação está ativa. Um aspecto de grande importância em se tratando de transações concorrentes é a suscetibilidade destas a um fenômeno conhecido por **inanição** (do inglês *starvation*), em que parcela de transações tenta ser processada diversas vezes, mas nunca consegue ser finalizada. Em ambientes sujeitos a ações arbitrárias por parte dos processos, o problema torna-se ainda mais agravado, pois a ocorrência de faltas pode impedir que determinadas transações atinjam seu término, de modo que elas são reiniciadas infinitas vezes, sem obter êxito no processamento de suas operações.

Nenhum dos trabalhos propostos no contexto do gerenciamento de transações em ambientes sujeitos a faltas Bizantinas (BFT - *Byzantine Fault-Tolerance*) [Gashi et al. 2007, Vandiver et al. 2007, Luiz et al. 2011, Garcia et al. 2011, Pedone et al. 2011] tratam ou propõem soluções para o problema da inanição. Desta forma, em situações onde há a violação da consistência por parte das transações, não é realizado nenhum tratamento para garantir o término destas, e elas tem de ser canceladas não espontaneamente. Este artigo apresenta uma solução para o problema de inanição em transações sujeitas a faltas Bizantinas, e até onde sabemos é o primeiro trabalho a considerar uma solução para este cenário, e, portanto, representa uma contribuição na direção de protocolos de replicação BFT para transações. A característica inovadora introduzida por este trabalho é a possibilidade de se garantir a terminação e o progresso justo de uma transação, a despeito de faltas Bizantinas (i.e. benignas ou maliciosas) da parte dos clientes e das réplicas.

Nossa proposta visa prover um ambiente confiável para a execução de transações de longa e de curta duração (i.e. OLTP e OLAP [Hasse and Weikum 1997]). No entanto, a combinação destes dois tipos de transações em um mesmo ambiente facilmente causa inanição, onde em geral, transações de curta duração cancelam transações concorrentes de longa duração [Weikum and Vossen 2002]. Com isso, ao invés de utilizar um único mecanismo para o processamento de transações (i.e. pessimista ou otimista), propõe-se o uso de um mecanismo adaptativo [Tai and Meyer 1996] conjuntamente ao uso de um contador monotônico (i.e. um *ticket*). Para tanto, a estratégia proposta integra três elementos de controle de concorrência já conhecidos na literatura: controle de concorrência baseado em *ticket* [Georgakopoulos et al. 1994]; controle de concorrência pessimista [Bernstein et al. 1987] e; o controle de concorrência otimista [Kung and Robinson 1981]. Este mecanismo adaptativo visa permitir que as transações canceladas não espontaneamente em detrimento de outras, sejam novamente submetidas por meio de um algoritmo de processamento e terminação justa, a fim de garantir que as estas alcancem seu término em um número finito de tentativas de execução, evitando assim sua inanição.

2. Trabalhos Relacionados

Poucos são os trabalhos que investigam transações e faltas Bizantinas. O primeiro [Molina et al. 1986] apenas investigou o uso de acordo bizantino e replicação de máquina de estados no contexto de bancos de dados (BD), onde propôs a execução das operações em BDs de maneira sequencial, sem concorrência entre transações. Em [Gashi et al. 2007] foi apresentado o uso de diversidade como mecanismo para tolerar faltas em transações em BDs, onde os autores demonstraram que uma série de *bugs* podem induzir os BDs a manifestar faltas Bizantinas. Uma solução apresentada para o problema foi uma arquitetura de *middleware* que isolou as faltas observadas, por meio da submissão das operações das transações sequencialmente às réplicas, e da verificação da consistência dos resultados através de um mecanismo de voto majoritário; o que limitou em muito a concorrência entre as transações naquele ambiente.

Em [Vandiver et al. 2007] os autores apresentaram o HRDB, em que um protocolo de escalonamento por barreira de confirmação (do inglês - *commit barrier scheduling*) é a base para prover um controle de concorrência robusto, e permite assegurar o comportamento correto sistema e a consistência forte dos dados, ao mesmo tempo que obtém elevado grau de concorrência, a despeito de faltas bizantinas. Uma fraqueza observada neste trabalho é a dependência de um coordenador centralizado - uma entidade confiável do sistema, cujo papel é restringir a ordem em que as operações serão enviadas às réplicas, a fim de evitar conflito entre as transações e preservar a concorrência. Recentemente foi proposto o Byzantium [Garcia et al. 2011], um protocolo que provê um suporte à execução de transações em BDs sujeitos a faltas Bizantinas, que relaxa a consistência a fim de obter um maior grau de concorrência no processamento das transações. Em suma, o protocolo de acordo Bizantino é o PBFT [Castro and Liskov 1999], e o suporte de transações adota como semântica de consistência o *snapshot isolation* [Gray et al. 1996]. No entanto, o *snapshot isolation* não provê uma semântica equivalente à execução sequencial de transações, tal como ocorre na serialização [Bernstein et al. 1987]. Além disso, estudos evidenciaram que esta semântica é suscetível a anomalias, que afetam em potencial a integridade dos dados manipulados nas transações [Berenson et al. 1995]. Isso implica que o sistema pode vir a sofrer um colapso em decorrência da corrupção de dados e violação de integridade, causadas por uma entidade faltosa (benigna ou maliciosa).

Por fim, duas contribuições que visam a busca de soluções para as lacunas encontradas nos trabalhos mencionados até então, são os protocolos BFT-*Deferred Update* [Pedone et al. 2011] (ou BFT-DU) e o apresentado em [Luiz et al. 2011]. Ambos os trabalhos apresentam soluções baseadas no processamento otimista, em que os protocolos não dependem de uma entidade centralizada, tal como ocorre no HRDB, e ambos adotam como critério de consistência a serialização. A principal diferença entre eles, é que [Pedone et al. 2011] parte do pressuposto de que apenas as réplicas são suscetíveis a faltas Bizantinas. Com isso, o protocolo realiza as operações de leituras apenas sobre a réplica primária, e as operações de escritas são efetuadas primeiramente em um *buffer* local do cliente, sendo propagadas às réplicas apenas no pedido de confirmação da transação, que parte do cliente. Já o protocolo de [Luiz et al. 2011], embora apresente latência superior no processamento de uma transação, é o único que assume réplicas e clientes Bizantinos e é capaz de manter as propriedades do sistema em condições de faltas nestas entidades. Não obstante, embora os trabalhos aqui mencionados permitam gerenciar transações em ambientes Bizantinos, todos os protocolos que fazem parte destes podem sofrer por inanição.

3. Contextualização do Problema

Existem dois tipos de aplicações que fazem uso de transações em bases de dados: OLAP (*Online Analytical Processing*) e OLTP (*Online Transactional Processing*) [Hasse and Weikum 1997]. De um lado, estão as aplicações OLAP, que são orientadas à síntese, análise e consolidação de dados, onde efetuam o processamento analítico e on-line dos dados, por meio de transações que realizam diversas operações de leitura, de longa duração (p. ex. *business intelligence*). De outro lado, as aplicações OLTP são orientadas à manipulação de dados operacionais (p. ex. transações bancárias), em que as transações são de curta duração, e em geral contêm operações de leitura e de escrita.

A conciliação das transações destes dois tipos de aplicações é uma tarefa difícil, principalmente porque a contenção de recursos da parte de uma ou outra pode implicar no cancelamento daquela que não detém a posse dos mesmos. Neste caso, privilegiar uma ou outra pode induzir o sistema a situações de inanição daquelas transações não privilegiadas. Este cenário é ainda mais agravado se considerado o comportamento arbitrário oriundo de faltas Bizantinas por parte das entidades do ambiente, o que é factível em nosso modelo/sistema. É digno de nota, que a inanição pode ocorrer em qualquer situação em que for verificada a existência de conflitos entre os itens de dados das transações em execução, independente do tipo de aplicação em uso (i.e. OLTP ou OLAP).

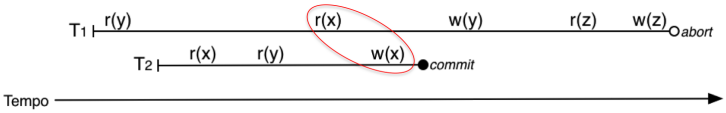


Figura 1. Cenário com transações de longa e de curta duração em conflito.

A Figura 1 ilustra um conflito entre uma transação de longa duração T_1 e uma transação de curta duração T_2. No caso, T_2 é iniciada após e concluída antes de T_1. Como T_2 escreve em um item de dados que foi lido por T_1, fica comprometida a confirmação de T_1, já que T_2 é bem sucedida e confirmada por não violar a serialização. T_1 por sua vez, se confirmada, violará o critério de serialização por ter realizado uma leitura-suja de x. Este cenário pode se repetir inúmeras vezes até que T_1 consiga, de fato ser terminada, e confirmada ou cancelada ao seu término. Uma solução trivial para o problema poderia ocorrer pela concessão de prioridades para as transações canceladas involuntariamente pelo protocolo, e também pelo uso dos esquemas clássicos para o tratamento de inanição em bancos de dados, tal como o *wound-wait* e *wait-die* [Weikum and Vossen 2002]. Porém, como estamos a lidar com faltas Bizantinas, o uso de prioridades seria algo um tanto delicado, já que uma transação privilegiada de um cliente faltoso poderia ficar indefinidamente em atividade, sem manifestar interesse na confirmação ou cancelamento, a fim de impedir o progresso de outras transações honestas que fazem referência aos mesmos itens de dados daquela privilegiada. Não obstante, se observa que os esquemas clássicos foram propostos sem hipóteses acerca da ocorrência de falhas [Weikum and Vossen 2002].

4. Visão Geral da Proposta

Conforme já citado, nossa proposta parte da integração dos elementos de controle de concorrência baseado em *ticket* [Georgakopoulos et al. 1994], pessimista [Bernstein et al. 1987] e otimista [Kung and Robinson 1981]. Contudo, diferente do propósito para o qual o método do *ticket* foi inicialmente definido

[Georgakopoulos et al. 1994], empregamos o uso do contador monotônico (i.e. um sequenciador) baseado em *ticket* para assegurar a ordem na qual uma transação poderá ter êxito em sua execução plena, independente se ao término ela seja confirmada ou cancelada. Em outros termos, o *ticket* é usado para determinar a ordem na qual as transações que são canceladas não espontaneamente devido à conflitos, faltas, etc., possam ser de fato, processadas e concluídas com êxito. A intenção em se evitar a inanição, é, principalmente, reduzir o desperdício de computações e recursos para impedir uma degradação de desempenho do sistema, em decorrência de reprocessamentos sucessivos de transações.

É importante destacar, que apenas o uso do *ticket* não permite resolver completamente o problema da inanição em transações. E para garantir o progresso e terminação justa das transações e evitar a inanição, a solução apresentada adota uma estratégia de escalonamento adaptativa e dinâmica de transações que, quando apropriado, alterna o método de controle de concorrência da transação de otimista para pessimista. Além disso, propomos a adaptabilidade não apenas para o controle de concorrência empregado, mas também para o tipo de escrita de transações, isto é, pré-declaradas e dinâmicas (vide Seção 4.3.).

4.1. Modelo de Sistema

O modelo de sistema adotado é composto por dois tipos de processos: as réplicas (ou servidores) $R = \{r_1, r_2, ..., r_n\}$ implementam o ambiente de BDs replicado, e os clientes $C = \{c_1, c_2, ..., c_n\}$ que fazem uso da base de dados. Assim, assumimos que C contém um número arbitrário (não infinito) de processos, e que R tem cardinalidade $|R| \geq 3f + 1$. Neste caso, um número ilimitado de clientes e até $f \leq \lfloor \frac{n-1}{3} \rfloor$ réplicas podem se desviar de forma arbitrária de suas especificações, com possibilidade de parar; omitir o envio e/ou a recepção ou ainda a entrega das mensagens; enviar respostas incorretas às operações dos clientes, bem como atuar em conluio com outros processos visando à corrupção do sistema. De outro modo, admitimos a independência de falhas por meio do uso de diversidade [Obelheiro et al. 2005], de tal forma que a ocorrência de uma falta em uma determinada réplica não necessariamente causa a mesma falta nas demais.

A saber que o modelo de interação assíncrono não permite assegurar a terminação de transações [Bernstein et al. 1987], adotamos o modelo de sincronismo terminal (*eventually synchronous system*) [Dwork et al. 1988]. A escolha por este modelo se justifica pela sua característica realista, onde o sistema se porta de forma assíncrona em grande parte do tempo, mas durante os períodos de estabilidade o tempo de transmissão de mensagens e das computações é limitado, porém, desconhecido. O critério de consistência adotado é o *one-copy serializability* (1-SR) [Bernstein et al. 1987], em que a execução concorrente de transações em uma base de dados "replicada" é equivalente à execução sequencial das mesmas transações em uma base de dados "não replicada". A despeito da diversidade, as réplicas são deterministas (i.e. todas as réplicas suportam o mesmo subconjunto de operações de manipulação de dados). Logo, o resultado de uma determinada operação é o mesmo para todas as réplicas. Por fim, nosso modelo de transações admite que um cliente submeta apenas uma transação por vez, e no contexto de uma transação, uma operação é enviada apenas se não há operações pendentes de execução para aquela transação.

Todas as comunicações entre os processos ocorrem por meio de canais ponto-a-ponto confiáveis e autenticados, e com ordenação FIFO; e o protocolo subjacente para difusão com ordem total adotado é o PAXOS Bizantino [Zielinski 2004], onde assumimos o uso de uma função de resumo criptográfico resistente à colisão para assegurar a integridade,

bem como códigos de autenticação de mensagens (p. ex. vetores de MACs) para assegurar a autenticidade das comunicações. Este mecanismo também é empregado para prover o controle de acesso dos clientes à base de dados, de modo que apenas mensagens oriundas de clientes devidamente autenticados são recebidas e entregues pelas réplicas. Por fim, o acesso aos objetos da base de dados é regulado por um mecanismo de controle de acesso discricionário, assim como ocorre em listas de controle de acesso (ACL).

4.2. Preliminares

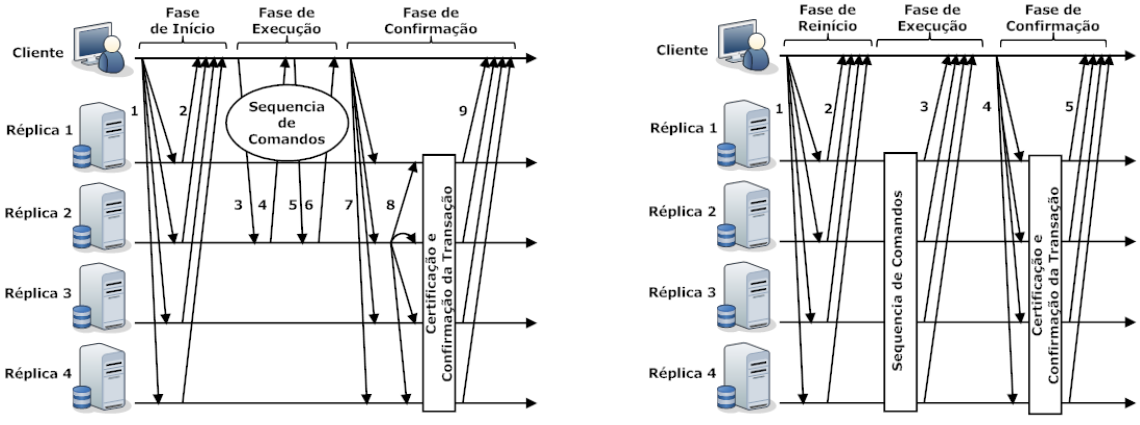
Conforme preconiza o modelo de transações [Bernstein et al. 1987], uma transação consiste em uma sequencia finita de operações de leitura e escrita, iniciada por uma instrução ***begin*** e finalizada por uma instrução ***commit*** ou ***abort***. As operações de uma transação devem satisfazer as propriedades de atomicidade, isolamento e durabilidade [Bernstein et al. 1987]. Para tanto, nossos algoritmos devem atender as condições de correção (*safety*) e vivacidade (*liveness*), e satisfazer as seguintes propriedades:

- **Consistência (*safety*):** o protocolo assegura o critério de consistência 1-SR (*one-copy serializability*) e o mantém para cada transação confirmada.
- **Terminação (*liveness*):** em circunstâncias normais, onde há uma transação em confirmação e transações concorrentes estão em situação de conflito, a transação em estado de confirmação sempre termina.
- **Terminação Justa (*fairness*):** uma transação que sofre um cancelamento não espontâneo é executada em completude, em no mínimo $f + 1$ e no máximo $2f + 1$ tentativas de execução.

Note que a diferença entre as propriedades de "terminação" e "terminação justa" reside no fato de que, na primeira, uma transação em execução pode sofrer cancelamentos não espontâneos, sempre que verificado conflito com uma transação em confirmação. Já na segunda, uma transação em execução tem seu término assegurado a despeito de conflitos, e do êxito ou não de sua execução, em um número finito de tentativas de execução.

4.3. Dinâmica de Funcionamento do Protocolo

O protocolo proposto adota uma estratégia adaptativa para o processamento de transações, a fim de obter a capacidade de assegurar a terminação justa das transações submetidas ao sistema. Neste sentido, o protocolo faz uso dos métodos pessimista e otimista de processamento de transações, de acordo com as condições verificadas no ambiente.



(a) Execução normal (otimista/dinâmica). (b) Execução justa (pessimista/pré-declarada).

Figura 2. Funcionamento básico do protocolo.

Primeiramente, a transação é executada de maneira tentativa (caso normal da Figura 2(a), onde o método otimista com escrita dinâmica (operações enviadas/processadas uma a uma, conforme a necessidade de interação dos clientes com as réplicas) é usado. Se porventura a transação vier a sofrer alguma falha ou conflito durante a execução otimista, em que a situação ocasione no seu cancelamento não espontâneo (o que pode se repetir indefinidas vezes), ela é novamente enviada para processamento por meio do protocolo de execução justa (Figura 2(b)). A execução justa é acionada para evitar que a transação fique indefinidamente em tentativa de processamento, e venha a sofrer por inanição. Neste caso, a transação é novamente submetida, mas por meio do método pessimista e com escrita pré-declarada, onde todas as operações são enviadas de uma única vez, no reinício da transação.

O propósito da execução justa é garantir a terminação de uma transação, de modo a evitar que sucessivas tentativas de processá-la causem a inanição da mesma, seja pelos conflitos verificados durante o processamento da transação ou pelo efeito transiente de faltas Bizantinas. No caso, o uso de escrita pré-declarada é proposital para evitar que um cliente com comportamento Bizantino possa manter uma transação em atividade sem nunca confirmá-la (o que é possível com escrita dinâmica), visando impedir o progresso das demais transações em execução normal ou justa. O uso de escrita pré-declarada é a única garantia de que a transação terá início e fim, já que todas as operações são enviadas de uma única vez, no momento do reinício da transação.

4.3.1. Gerenciamento e Execução de Transações

Nesta seção apresentamos a formalização dos algoritmos que compõem o protocolo proposto. Conforme especificado na Seção 4.1., a comunicação é confiável e autenticada, bem como o acesso aos objetos da base de dados é regulado por um mecanismo de controle de acesso discricionário. Todavia, para simplificar os códigos e facilitar o entendimento dos algoritmos por parte do leitor, os detalhes das operações de geração, verificação e controles criptográficos foram omitidas dos mesmos. Conforme descrito na Seção 4., nosso protocolo consiste na primeira e única proposta que resolve o problema da inanição em transações, mediante a faltas Bizantinas. Como os protocolos apresentados naquela seção são aptos a processar transações em bases de dados sujeitas a faltas Bizantinas, utilizamos como algoritmo para o processamento de transações em condições normais (Figura 2(a)) aquele apresentado no trabalho de Luiz et al. [Luiz et al. 2011]. Com isso, devido a restrições de espaço, nos limitados apenas em fazer um breve comentário a respeito de seu funcionamento, de modo que é facultada ao leitor a verificação destes na íntegra em [Luiz et al. 2011].

Em suma, o protocolo opera da seguinte maneira: quando um cliente tem a intenção de iniciar uma transação na base de dados, ele submete-a ao protocolo, que inicia a transação por meio do sub-protocolo de **execução normal** (Figura 2(a)). Ao iniciar a transação as réplicas definem dentre elas um líder, o qual irá realizar a execução da transação (passos 1 e 2 - Figura 2(a)). A partir dai, a interação do cliente com a base de dados ocorre de maneira otimista e dinâmica, isto é, a comunicação ocorre apenas com a réplica líder e as operações são enviadas de acordo com a necessidade (e vontade) do cliente (passos 3 a 6 - Figura 2(a)). Quando não há mais operações para uma transação, o cliente solicita a confirmação, por meio da difusão com ordem total de um pedido de confirmação à todas as réplicas (passo 7 - Figura 2(a)). Ao entregar esta mensagem, as réplicas não-líder passam a ter o conhecimento das operações que compõem a transação, que vão apensadas à mensagem que acabara de ser entregue. Neste ponto, a réplica líder

propaga às demais réplicas as operações por ela executadas para aquela transação, como forma de notificar que a transação enviada pelo cliente está em consonância com aquela executada de maneira otimista por ela (passo 8 - Figura 2(a)). A entrega desta última mensagem pelas réplicas, implica no inicio do processo de confirmação, em que as operações da transação são executas pelas réplicas não-líder, ou também pela líder, caso seja verificado que o cliente pediu a confirmação para uma transação diferente daquela executada por ela. Após estas verificações, a transação é submetida a um teste de certificação para determinar se ela cumpre os requisitos necessários à sua confirmação (i.e. serialização) e, ao passar pela certificação, a transação é aceita e se torna persistente na base de dados. Do contrário, se por alguma razão a transação não puder ser confirmada, ela é cancelada pelas réplicas, e é marcada localmente como "cancelada não espontaneamente" para indicar que é requerido o reprocessamento da mesma. No caso de um cancelamento unilateral por parte da réplica líder, esta envia uma mensagem de notificação de cancelamento às demais réplicas, por meio de difusão confiável, para que todas as réplicas tenham ciência de que a transação foi cancelada. Ao término do processo, as réplicas notificam o cliente quanto à situação da transação (confirmada ou cancelada), no passo 9 da Figura 2(a).

Variáveis:

1: $TICKET = \perp$ {Ticket a ser atribuído às transações (global)}
2: $ticket = 0$ {Ticket da última transação atendida (local)}
3: $last\_commited\_ticket = 0$ {Última transação confirmada pelo ticket}
4: $transaction\_data = \emptyset$ {Dados de controle das transações}
5: $commit\_data = \emptyset$ {Dados temporários para confirmação}
6: $\prod^{rst} = \emptyset$ {Conjunto que contém as transações reenviadas}
7: $\prod^{stv} = \emptyset$ {Conjunto que contém as transações suscetíveis à inanição}
8: $\prod^{act} = \emptyset$ {Conjunto que contém as transações atuamente em atividade}

upon: $TO\text{-}deliver(c_i, \langle RESTART, t_i, t_i^{tsb}, t_i^{ops}\rangle))$
9: **if** $\exists c_i \in \mathcal{C}$ **then**
10: **if** $((\exists t_i \in \prod^{stv}) \wedge (state(t_i) = aborted)) \wedge (check\_data(t_i, t_i^{tsb}) = true)$ **then**
11: $[\prod^{stv} \leftarrow \prod^{stv} \setminus \{t_i\}\,;\, \prod^{rst} \leftarrow \prod^{rst} \cup \{t_i\}\,]$
12: $TICKET \leftarrow TICKET + 1$ {Incremento do ticket global}
13: $ticket(t_i) \leftarrow TICKET$ {Ticket atribuído à transação t_i}
14: $state(t_i) \leftarrow active$
15: $transaction\_data \leftarrow transaction\_data \cup \{\langle ticket(t_i), t_j, t_i^{tsb}, t_i^{ops}\rangle\}$
16: $send(s_i, \langle ACTIVE, t_i, ticket, (ticket(t_i) - ticket)\rangle)\ to\ c_i$
17: **end if**
18: **else**
19: $discards\ the\ transaction$
20: **end if**

Figura 3. Algoritmo da fase de reinício de transações da Figura 2(b).

Por outro lado, quando a transação não puder ser concluída em sua execução normal, seja pelo cancelamento em detrimento de outras ou pela verificação de situação de falha, o protocolo executa os passos ilustrados na Figura 2(b). Neste caso, a transação passa a ser executada de maneira pessimista, e é aí que ocorre a adaptação do protocolo à condição do ambiente. A execução justa de transações, conforme apresentada na Figura 2(b), é realizada pelo sub-protocolo por meio de três algoritmos, sendo um para cada fase. Os algoritmos para as fases são apresentados nas **Figuras** 3, 4 e 5, respectivamente. A primeira fase do protocolo, que é o reinício da transação, é formalizada no algoritmo da **Figura** 3. Neste, é possível observar a existência de um *TICKET* global, que é o principal mecanismo usado para assegurar o atendimento em plenitude das transações. Para tanto, primeiramente o *TICKET* é inicializado por todas as réplicas do sistema como nulo (linha 1), e é incrementado a cada mensagem *RESTART* entregue pelo protocolo, o que indica o reenvio de uma transação. Também é usado um *ticket* local para o controle interno de atendimento de cada réplica (linha 2), pois durante o processamento da transação, não há qualquer tipo de sincronização entre as réplicas.

Assim, quando uma transação é novamente submetida após uma tentativa de execução mal sucedida, primeiramente as réplicas verificam se a transação está sendo soli-

citada por um cliente já conhecido no sistema (linha 9). Esta verificação é necessária para impedir que clientes Bizantinos que entram no sistema, tentem iniciar uma transação com privilégios/prioridade em relação às honestas. Também é realizada uma verificação se a transação já foi anteriormente executada e sofreu um cancelamento, bem como se os dados de controle da transação recebida são consonantes com os daquela cancelada. Se qualquer uma das verificações falhar, as réplicas simplesmente descartam o pedido de reinício para a transação. Por outro lado, se a transação é aceita, ele é incluída em um conjunto de transações em atividade prioritária (linha 11), e o protocolo atribui a ela um valor de *ticket*, que indica a ordem na qual a transação terá a prioridade para sua execução em completude (linhas 12 e 13). Por fim, as réplicas inserem os dados de controle da transação em uma estrutura de dados do tipo "tabela de dispersão" (*hashtable*), em que a chave de cada entrada é o próprio *ticket*. A finalização da fase se dá quando as réplicas enviam ao cliente a notificação de que a transação foi iniciada, além de uma previsibilidade para sua execução, isto é, quantas transações estão à sua frente aguardando a execução (linha 16).

task TRANSACTION_PROCESS:
1: **while** $true$ **do**
2: **wait until** $|transaction\_data| > 0$
3: $ticket \leftarrow ticket + 1$
4: $\langle ticket(t_i), t_i, t_i^{tsb}, t_i^{ops} \rangle \leftarrow get\_by\_ticket(transaction\_data, ticket)$
5: $request\_priority\_locks(\langle t_i^{ops}, ticket(t_i) \rangle))$
6: $\langle RS, WS \rangle \leftarrow get\_readwrite\_Sets(t_i)$
7: **for each** $t_k \in \prod^{act} : \{ticket(t_k) = \perp \wedge (RS(t_k) \cap WS \neq \emptyset \vee WS(t_k) \cap WS \neq \emptyset \vee WS(t_k) \cap RS \neq \emptyset)\}$ **do**
8: **if** $(state(t_k) = active \vee state(t_k) = preparing)$ **then**
9: $abort\_transaction(t_k)$
10: **else if** $state(t_k) = ready$ **then**
11: $undo\_transaction(t_k)$
12: $has\_redo(t_k) \leftarrow true$
13: **end if**
14: **end for**
15: **wait until** $get\_priority\_locks(ticket(t_i)) =$ **true**
16: **for each** $op_i \in t_i^{ops}$ **do**
17: $result \leftarrow execute(op_i)$
18: $t_i^{res} \leftarrow t_i^{res} \cup \{result\}$
19: **end for**
20: **if** $\exists$ **exception** $\in t_i^{res}$ **then**
21: $commit\_data \leftarrow commit\_data \cup \{\langle t_i, ticket, H(t_i^{ops}), H(t_i^{res}), RS, WS, \text{“}not - OK\text{“} \rangle\}$
22: **else**
23: $commit\_data \leftarrow commit\_data \cup \{\langle t_i, ticket, H(t_i^{ops}), H(t_i^{res}), RS, WS, \text{“}OK\text{“} \rangle\}$
24: **end if**
25: $state(t_i) \leftarrow preparing$
26: $\nexists$ **exception** $\in t_i^{res}$? $status \leftarrow \text{“}OK\text{“} : status \leftarrow \text{“}not - OK\text{“}$
27: $transaction\_data \leftarrow transaction\_data \setminus \{\langle ticket(t_i), t_j, t_i^{tsb}, t_i^{ops} \rangle\}$
28: $send(s_i, \langle END, t_i, t_i^{ops}, t_i^{res}, status \rangle)\ to\ p_i$
29: **end while**

Figura 4. Algoritmo da fase de execução de transações da Figura 2(b).

Após o reinício da transação e da obtenção de seu *ticket* para execução, a próxima etapa do protocolo consiste na entrada na fase de execução da transação, por parte das réplicas. A tarefa que realiza a execução justa das transações, baseado no *ticket* a ela atribuído, é formalizado no algoritmo da **Figura** 4. O algoritmo da Figura 4 é bastante simples, onde inicialmente a tarefa entra em um laço infinito, em razão da necessidade desta permanecer em execução enquanto a réplica estiver em atividade. É digno de nota, que o valor inicial do *ticket* local é 0 (linha 2 do algoritmo da Figura 3). A cada interação do laço, o algoritmo primeiramente verifica se há alguma transação pronta para ser executada com base na tabela de dispersão (linha 2), e caso houver, o valor do *ticket* para controle de atendimento local é incrementado em uma unidade, e então a tarefa recupera da tabela de dispersão os dados da transação, cuja chave de inserção é igual ao valor atribuído ao *ticket* (linhas 3 e 4).

O passo seguinte consiste na requisição dos bloqueios sobre os itens de dados a que a transação faz referência (linha 5), e na sequência são obtidos os conjuntos de leituras e de escritas da transação (RS e WS - linha 6), estes necessários para verificar possíveis con-

flitos que possam estar impedindo a aquisição dos bloqueios. A concessão dos bloqueios para as transações em fase de execução justa ocorre de forma prioritária, e com base no *ticket* obtido para a transação. Neste caso, se durante a aquisição dos bloqueios houver alguma transação em **execução normal** (i.e. sem um *ticket*), e que esteja impedindo a concessão dos bloqueios, esta transação é cancelada de forma compulsória (condição indicada na linha 7) para liberar os bloqueios e então permitir a concessão para a transação indicada pelo *ticket*. Não obstante, as transações canceladas não espontaneamente recebem um tratamento adequado, de acordo com o estado em que elas se encontram (linhas 8 a 13). Ao obter a concessão dos bloqueios, a transação inicia a execução das operações que compõem sua unidade atômica e, cada resultado obtido é armazenado em um *buffer* de retenção de resultados para uso posterior (laço entre as linhas 16 a 19).

É importante salientar, que o mecanismo de bloqueio adotado é aquele que opera em duas fases de maneira conservadora (*strict two-phase locking* ou 2PL), de modo que, após a aquisição dos bloqueios estes são mantidos pela transação até o seu término, a fim de evitar a preempção po interferências alheias. Embora este mecanismo seja essencialmente pessimista, por meio de seu uso é possível assegurar o término da transação, além de evitar a ocorrência de *deadlocks* envolvendo-a [Weikum and Vossen 2002]. Ao término da execução das operações da transação, é verificado se dentre as operações executadas, alguma delas produziu uma exceção ou um erro. Se a condição não for verificada, a transação é marcada como "*OK*" (linha 23), e do contrário é marcada como "*not - OK*" (linha 21). Em ambos os casos, a transação e seus dados de controle são inseridos em um *buffer* de dados temporários de confirmação, para uso posterior durante a fase de confirmação. Note que uma decisão pelo cancelamento da transação, poderia ser tomada com base na(s) exceção(ões) verificada(s). Todavia, como uma exceção também pode caracterizar uma falta Bizantina (p. ex. um *bug*) isolada em uma réplica, por esta razão, a decisão é deixada para o cliente na fase de confirmação. A fase de execução se encerra com o envio do resultado da transação, ao cliente responsável pela mesma (linha 28).

Por fim, o algoritmo para a última fase do protocolo que é a confirmação, é formalizado na **Figura** 5. Neste caso, diferente do que ocorre na execução normal, onde a confirmação, de fato, é enviada pela réplica líder da transação, aqui ela é enviada diretamente pelo cliente - em ambos os casos enviado por meio de difusão com ordem total. Com isso, quando as réplicas entregam uma mensagem de confirmação (i.e. *COMMIT*), o primeiro passo do algoritmo é a verificação acerca da identificação do cliente, pois um cliente faltoso recém-chegado ao sistema pode vir a tentar confirmar uma transação espúria, o que por conseguinte deve ser rejeitado. Se o cliente já é conhecido no sistema, indica que pelo menos uma transação dele já foi submetida para processamento, neste caso, resta verificar se os dados enviados na mensagem de confirmação são consistentes, o que é efetuado na linha 2. Na condição da linha 2 são verificadas duas questões, primeiro se a transação se encontra em atividade e se, além disso, a transação está no estado de preparação - único estado possível para se confirmar uma transação, conforme o autômato que define as transições de estado do protocolo [Luiz et al. 2011]. Se as condições forem satisfeitas, as réplicas verificam se os dados entregues junto à mensagem estão em consonância com os dados contidos no conjunto de informações temporárias de confirmação, inseridos na fase de execução (linha 3). E em sendo, o procedimento segue adiante, ou do contrário, a transação é descartada.

No procedimento de confirmação, o protocolo recupera os dados de controle da

transação, bem como os dados temporários de confirmação (linhas 4 e 5). Na sequência, é verificado se o *ticket* daquela transação é igual ao valor *ticket* da última transação atendida, acrescido em uma unidade (i.e. o próximo a ser atendido). Se o *ticket* da transação for o próximo da sequência, é verificado se naquela réplica a transação teve sua execução sem a ocorrência de exceções (linha 7), pois como já foi dito, uma exceção pode ser oriunda de uma réplica faltosa. Do contrário, se a transação não for a próxima ela é simplesmente descartada (linha 23). Caso não tenha havido nenhuma exceção, a transação é marcada para confirmação, ou do contrário, para cancelamento (linhas 8 e 10, respectivamente). Neste sentido, se a transação puder ser confirmada, a réplica então verifica se os dados enviados pelo cliente às réplicas no pedido de confirmação, são exatamente iguais àqueles que foram executados para a transação, na fase anterior. E se assim o for, a transação é confirmada e os dados tornam-se permanentes na base de dados (linhas 12 a 14). Por outro lado, caso a verificação não se confirme, a transação é cancelada e o efeito (ou estado) desta é descartado da base de dados (linhas 15 a 17). Por fim, as réplicas enviam ao cliente o resultado final da transação, liberam os bloqueios mantidos sobre os itens de dados e incrementam o *ticket* de controle de transações confirmadas em uma unidade (linhas 19 a 21). O cliente por sua vez, aceita o resultado da transação se recebe $f + 1$ respostas iguais de diferentes réplicas, o que indica que pelo menos uma réplica correta concluiu com êxito a transação.

upon: *TO-deliver*$(p_i, \langle COMMIT, t_j, ticket(t_j), \perp, \perp, H(t_j^{ops}), H(t_j^{res})\rangle)$

```
1:  if ∃p_i ∈ C then
2:      if ∃t_j ∈ ∏^act ∧ state(t_j) = preparing then
3:          if ⟨t_j, ticket(t_j), H(t_j^ops), H(t_j^res)⟩ ∈ commit_data then
4:              ⟨t_i, ticket(t_i), t_i^ops, t_i^res⟩ ← get_context(∏^act, t_j)
5:              ⟨status, RS, WS⟩ ← get_commit_data(commit_data, t_i)
6:              if ticket(t_i) = last_commited_ticket + 1 then
7:                  if status = "OK" then
8:                      can_commit ← true
9:                  else
10:                     can_commit ← false
11:                 end if
12:                 if can_commit = true ∧ matches(⟨H(t_i^ops), H(t_i^res)⟩, ⟨H(t_j^ops), H(t_j^res)⟩) then
13:                     [ outcome ← COMMITED ; T_i ← t_i ; ∏^c ← ∏^c ∪{T_i} ; ∏^act ← ∏^act \{t_i} ]
14:                     [ state(t_i) ← commited ; commit_transaction(T_k) ]
15:                 else
16:                     ∏^act ← ∏^act \{t_i}
17:                     [ outcome ← ABORTED ; abort_transaction(t_i) ]
18:                 end if
19:                 send(s_k, ⟨outcome, t_i⟩) to c_i
20:                 release_locks(t_i)
21:                 last_commited_ticket ← last_commited_ticket + 1
22:             else
23:                 discards the transaction until their turn comes
24:             end if
25:         end if
26:     end if
27: end if
```

Figura 5. Algoritmo da fase de confirmação de transações da Figura 2(b).

Note que um cliente faltoso pode vir a executar apenas as duas primeiras fases do protocolo de execução justa (reinício e execução), e nunca enviar o pedido de confirmação às réplicas, no intuito de fazer com que a transação mantenha os bloqueios sobre os itens de dados, e impedir que o sistema tenha progresso. Este problema é resolvido da seguinte maneira, após o envio do resultado do processamento da transação pelas réplicas ao cliente (ao término da fase de execução), é iniciado um temporizador que é dinamicamente ajustado de acordo com as condições verificadas no ambiente (p. ex. tempo de transmissão e recepção). Se este temporizador vier a se esgotar, as réplicas descartam a transação para liberar os bloqueios e assegurar a progressão do sistema. É importante ressaltar esta medida é adotada em decorrência do fato de que, o próprio modelo de transações não assegura a terminação destas em ambientes assíncronos.

5. Avaliação, Implementação e Resultados

Esta seção apresenta uma análise acerca do desempenho do protocolo proposto, de maneiras analítica e experimental. A análise se dá através da comparação dos custos associados ao processamento de uma transação, onde em particular, estamos interessados em observar a eficiência de nossa solução em relação aos trabalhos correlacionados, no que tange ao caso normal do protocolo, já que os demais não garantem a liberdade de inanição. No caso, a avaliação analítica é apresentada na Tabela 1, onde é realizada uma análise dos custos envolvidos no processamento de uma transação em situação *normal*, o que compreende às fases de início, execução e terminação, dos protocolos. O protocolo proposto neste trabalho aparece na Tabela 1 com o nome BFT-SF (de *Byzantine Fault-Tolerant - Starvation Free Transactions*), os demais protocolos são oriundos dos trabalhos relacionados (vide Seção 2.). Para o caso da latência, as equações ali apresentadas se referem ao número de mensagens requerido para todas as fases da transação, em que o termo s indica o número de operações executadas para a transação.

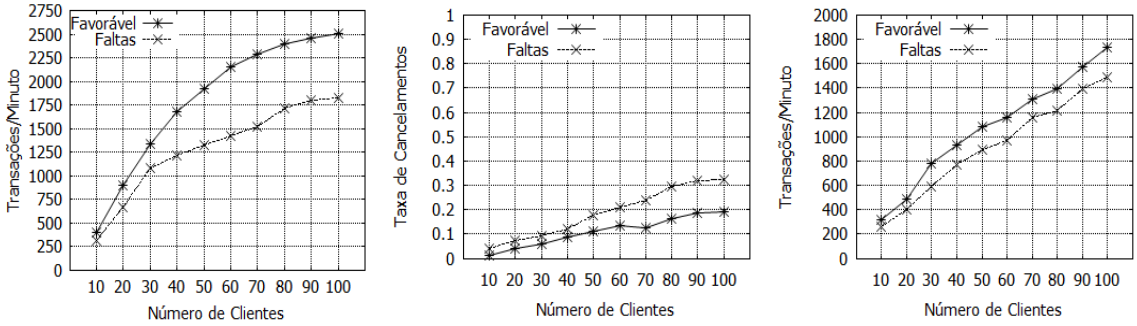
Tabela 1. Propriedades e custos associados ao processamento de uma transação.

Características e Propriedades	***Protocolos Avaliados***			
	BFT-SF	**HRDB**	**Byzantium**	**BFT-DU**
# Replicas	$3f + 1$	$2f + 1 + controlador$	$3f + 1$	$3f + 1$
Latência Normal	$s + 3(TOMCast) + 2$	$s + 4$	$s + 2(TOMCast) + 2$	$s + (TOMCast) + 1$
Consistência	Forte	Forte	Relaxada	Forte
Controle	Distribuído	Centralizado	Distribuído	Distribuído
Faltas	Réplicas e Clientes	Réplicas e Clientes	Réplicas e Clientes	Apenas Réplicas
Livre de inanição	Sim	Não	Não	Não
Latência Justa	$2(TOMCast) + 3$	–	–	–

Como se pode notar, o BFT-SF apresenta latência superior a todos os demais trabalhos. Isto advém da necessidade de se efetuar controles e verificações adicionais em relação aos demais protocolos, já que o BFT-SF é o único que permite o processamento confiável de transações de maneira totalmente distribuída, com o critério de consistência serializável, e sobretudo, com faltas oriundas de réplicas e clientes. Outro aspecto importante decorre do fato de que o BFT-SF é o único livre de inanição, e neste caso, o custo em termos de latência para o protocolo em execuções justas (i.e. latência justa) é inferior ao verificado para as execuções normais. Isto se explica pelo simples fato de que na execução justa, ao invés da transação ser executada de maneira dinâmica, ela é pré-declarada, o que implica na necessidade de apenas uma única mensagem para o envio de todas as operações da transação. Além disso, como na execução justa todas as réplicas executam a transação, não há a necessidade da réplica líder enviar sua afirmação para o pedido de confirmação do cliente, o que reduz em muito o número de mensagens do protocolo.

No intuito de analisar BFT-SF experimentalmente, foi implementado um protótipo para o mesmo. Assim, realizamos alguns experimentos em um ambiente composto por 14 máquinas HP 6005 *Pro Microtower* (AMD Phenom II^{TM}X4 3.2 GHz; 4GB RAM; Ethernet Gigabit), sendo todas conectadas em um ambiente de rede local por meio de um *switch* D-Link DGS-3100. Destas máquinas, 4 foram usadas para as réplicas e 10 para os clientes (i.e. com 10 clientes por máquina). O ambiente de software utilizado foi o Ubuntu Server 12.04.1 LTS, com a JVM Sun 1.6.0_29 sendo ativado o compilador Just-In-Time (JIT). Como SGBD, utilizamos o MySQL 5.5.8, sendo que a base de dados foi populada conforme o *benchmark* TPC-C (*http://www.tpc.org/tpcc*), que simula um ambiente de trabalho do tipo fornecedor por atacado. A escolha pelo TPC-C se deu não apenas para medirmos a sobrecarga, mas também para avaliar a escalabilidade do protocolo. Este *benchmark* produz uma carga de elevado nível de concorrência, por meio da mistura de transações

de intensivas atualizações com aquelas de somente-leitura, a fim de simular as atividades encontradas em ambientes transacionais complexos. Cabe ressaltar, que no ambiente do TPC-C predominam as transações de atualização (p. ex. com operações de leitura e escrita), o que compõe 92% da carga de trabalho do ambiente simulado, enquanto apenas 8% compreende a transações somente de leitura.



(a) Vazão (execução normal). (b) Cancelamentos (exec. normal). (c) Vazão (execução justa).

Figura 6. Desempenho verificado para o BFT-SF no *benchmark* TPC-C.

Visto que o BFT-SF é a única solução que previne a inanição, decidimos por não compará-lo com os trabalhos relacionados, já que a partir destes não é possível mostrar os resultados que se pretende avaliar. Assim, os experimentos consideraram apenas o BFT-SF em condições favoráveis (i.e. livres de faltas) e execuções com faltas, em processamento de maneiras normal e justa. A Figura 6(a) apresenta o resultado da vazão do protocolo de execuções normais, com e sem faltas. Note que a ocorrência de faltas incorre na degradação de aproximadamente 25% a 30% no desempenho de execuções normais, sendo que a taxa de cancelamentos não espontâneos é de aproximadamente 20% em condições favoráveis e de 30% em cenários com faltas (conforme a Figura 6(b)). Além disso, como o propósito do protocolo é evitar inanição de transações, avaliamos também a execução justa do protocolo. Esta avaliação foi realizada, a fim de verificar tanto a efetividade como o desempenho do mesmo, em condições favoráveis e com faltas. Estes resultados são reportados na Figura 6(c), onde se observa que o desempenho do protocolo com transações em execução justa é afetado em aproximadamente 30%, para ambos os cenários (com e sem faltas). Note que nesta situação, o protocolo continua a operar de forma normal para as transações concorrentes àquela em execução justa. Assim, se considerado o benefício que a execução justa oferece ao modelo de transações, o desempenho se torna bastante aceitável. É digno de nota que esta degradação já era prevista, justamente porque o processamento das transações em execução justa ocorre de maneira sequencial, a fim de evitar interferências à elas, para impedir o seu cancelamento não espontâneo (i.e. para preservar a execução da transação até sua conclusão). Não obstante, a despeito da execução justa, nesta situação a taxa de cancelamentos se mantém, pois as transações concorrentes em execução normal, quando em conflito, são canceladas para que as em execução justa possam ser concluídas.

6. Conclusão

Neste trabalho apresentamos um novo protocolo para o processamento e terminação justa de transações em ambientes sujeitos a faltas Bizantinas, o primeiro livre de inanição. A despeito do desempenho verificado, acreditamos que os benefícios associados com a possibilidade de evitar as transações de sofrer por inanição justificam os custos, se levado em consideração que as transações em situação normal podem executar infinitas vezes sem êxito algum, incorrendo em um custo mais elevado que o obtido. Também demonstramos,

que apesar do desempenho a eficiência do algoritmo de execução justa é superior a do algoritmo normal de processamento de transações, de todos os trabalhos. Além do mais, como a solução integra mecanismos alternativos para o controle de concorrência de transações, ela é de possível adaptação nos protocolos já existentes (i.e. dos trabalhos correlacionados), sem muita complexidade, bem como da necessidade de despender grande esforço.

Referências

Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., and O'Neil, P. (1995). A critique of ANSI SQL isolation levels. In *SIGMOD'95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 1–10, New York, NY, USA. ACM.

Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.

Castro, M. and Liskov, B. (1999). Practical Byzantine fault tolerance. In *OSDI '99: Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, pages 173–186. USENIX Association.

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

Garcia, R., Rodrigues, R., and Preguiça, N. (2011). Efficient middleware for byzantine fault-tolerant database replication. In *Proceedings of the 6th European Conference on Computer Systems - EuroSys'11*. ACM.

Gashi, I., Popov, P. T., and Strigini, L. (2007). Fault tolerance via diversity for off-the-shelf products: A study with SQL database servers. *IEEE Transactions on Dependable and Secure Computing*, 4(4):280–294.

Georgakopoulos, D., Rusinkiewicz, M., and Sheth, A. P. (1994). Using tickets to enforce the serializability of multidatabase transactions. *Knowledge and Data Engineering, IEEE Transactions on*, 6(1):166–180.

Gray, J., Helland, P., O'Neil, P., and Shasha, D. (1996). The dangers of replication and a solution. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 173–182, New York, NY, USA. ACM.

Hasse, C. and Weikum, G. (1997). Inter- and intra-transaction parallelism for combined OLTP/OLAP workloads. In Jajodia, S. and Kerschberg, L., editors, *Advanced Transaction Models and Architectures*, pages 279–302. Springer-Verlag.

Kung, H. T. and Robinson, J. T. (1981). On optimistic methods for concurrency control. *ACM Transactions on Database Systems*, 6:213–226.

Luiz, A. F., Lung, L. C., and Correia, M. (2011). Protocolo tolerante a faltas bizantinas para bases de dados transacionais. In *Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 559–572. SBC.

Molina, H. G., Pittelli, F., and Davidson, S. (1986). Applications of byzantine agreement in database systems. *ACM Transactions on Database Systems*, 11(1):27–47.

Obelheiro, R. R., Bessani, A. N., and Lung, L. C. (2005). Analisando a viabilidade da implementação prática de sistemas tolerantes a intrusões. In *Anais do V Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg 2005*.

Pedone, F., Schiper, N., and Armendáriz-Iñigo, J. (2011). Byzantine fault-tolerant deferred update replication. In *Proceedings of the 5th Latin-American Symposium on Dependable Computing - LADC'11*. SBC.

Tai, A. T. and Meyer, J. F. (1996). Performability management in distributed database systems: An adaptive concurrency control protocol. In *Proceedings of the 4th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, MASCOTS '96, pages 212–216, Washington, DC, USA. IEEE Computer Society.

Vandiver, B., Balakrishnan, H., Liskov, B., and Madden, S. (2007). Tolerating byzantine faults in transaction processing systems using commit barrier scheduling. In *SOSP'07: Proceedings of 21st ACM Symposium on Operating Systems Principles*.

Weikum, G. and Vossen, G. (2002). *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Zielinski, P. (2004). Paxos at war. Technical Report UCAM-CL-TR-593, University of Cambridge Computer Laboratory, Cambridge, UK.

Model Checking the Deferred Update Replication Protocol

Odorico Machado Mendizabal[1,2], **Fernando Luís Dotti**[2]

[1]Universidade Federal do Rio Grande – FURG
Rio Grande – RS – Brazil

[2]Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre – RS – Brazil

odoricomendizabal@furg.br, fernando.dotti@pucrs.br

Abstract. *As the number of distributed applications and the volume of generated data increase, support from robust data management systems becomes even more necessary. Since these management systems possibly have to deal with heavy workloads, in this paper we analyze the deferred update replication, a successful technique to implement highly available and performing transactional databases. Although it offers a strong consistency semantics, no enough efforts have been invested to prove its properties. Due to its critical requirements, in this paper we verify the deferred update replication protocol using model checking. A model of the deferred update replication protocol and a comprehensive set of safety and liveness properties are presented. According to our investigation, all properties hold for the presented model, leading to a higher confidence in the protocol's correctness.*

1. Introduction

The increase in the number of distributed applications and the high volume of data being generated demand support from robust data management systems. Besides providing high availability, these data management systems must be capable to scale up without affecting performance nor consistency. Underlying protocols for reliable communication and replication techniques are largely adopted in the development of those kind of systems. Of special interest in this case is the deferred update technique, that aims to achieve both high availability and performance of transactional dependable data management systems [Pedone et al. 1997].

Like distributed algorithms in general, guaranteeing that update replication protocols run correctly and efficiently is not trivial. Although this class of protocols suggests a strong consistency semantics, little efforts have been invested to formally prove their properties. Most of the authors in the literature argue about the correctness of the deferred update protocol reasoning in natural language [Garcia et al. 2011, Pedone and Schiper 2012]. Even though this approach gives a general understanding of the reasons behind design aspects of the implemented protocol, it hardly will detect non trivial failures presented in the solution.

A few works introduce some kind of formal proof of correctness for the deferred update replication technique [Pedone et al. 1997, Kemme and Alonso 2000, Garcia et al. 2011], and a few contributions use the theorem proving approach

This work is partially sponsored by FAPERGS and CNPq projects SAN Model Checking.

[Schmidt and Pedone 2007, Armendáriz-Iñigo et al. 2009]. However we could not identify the use of semi-automated reasoning in the later cases. For the theorem proving approach, a formal description of the model is needed for the reasoning process and the reasoning itself can be conducted in a semi-automated or non automated way, the last one being more error prone than the first. In either case, the verification process would be arduous and time consuming.

This paper presents a formal verification of the deferred update replication protocol by using model checking. The model checking technique is very attractive due to its simplicity of use combined with a solid theoretical foundation on verification approach. While it does not require high skilled specialists as usually needed for theorem proving, it offers a very expressive resource for properties specification through temporal logic. Since the deferred update protocol strongly relies on the atomic broadcast protocol to ensure correct progress of the servers, we first present properties and a model for the atomic broadcast which will be used as building block for the specification of the deferred update protocol. We took Promela [Holzmann 1991] and Spin [Holzmann 1997] as modeling language and model checking tool respectively. Promela offers abstractions very close to the ones typically used while building distributed algorithms, such as sequential processes, messages and channels.

The rest of the paper is structured as follow: Sections 2 and 3 illustrate the models and verification for atomic broadcast and deferred update replication protocols. A discussion about related work is presented in Section 4 and Section 5 concludes this paper.

2. A Promela Model for the Atomic Broadcast Protocol

The atomic broadcast protocol provides reliable communication channels for message exchange in distributed environments. It guarantees that a group of communicating processes delivers the same set of messages to every process following the same delivery order [Défago et al. 2004]. Due to the ordering guarantees, this protocol is often used by distributed algorithms such as replication algorithms. The total ordering delivery provides deterministic update on database replicas [Agrawal et al. 1997, Pedone et al. 1997, Kemme and Alonso 2000].

Delivery guarantees are preserved since the protocol satisfies the following properties [Hadzilacos and Toueg 1994]:

- *Validity*: If a correct process broadcasts a message m, then it eventually delivers m.
- *Uniform Agreement*: If a process delivers a message m, then all correct processes eventually deliver m.
- *Uniform Integrity*: For any message m, every process delivers m at most once, and only if m was previously broadcast by $sender(m)$.
- *Uniform Total Order*: If both processes p and q deliver messages m and m', then p delivers m before m', if and only if q delivers m before m'.

Next, we describe a simple model for the atomic broadcast and verify the protocol's properties using model checking. In our model, every process p_i running the protocol uses a channel $abcast_i$ for reliable communication. External processes that do not participate in the protocol have no access to $abcast$ channels.

Algorithm 1 shows the primitive *send* modeled in Promela [Holzmann 1991]. For representation of message passing, operators ! and ? are used for write and read over channels, respectively. An atomic block is used to ensure that no other process will execute while the send operation in execution has not finished. The number of channels used by the protocol is represented by $num\_servers$.

Algorithm 1 Send(m)

```
1:  atomic {
2:      do
3:          :: i < num_servers →
4:          assert(nfull(abcast));
5:          abcast[i]!m;
6:          i++;
7:          :: i == num_servers →
8:          i = 0;
9:          break;
10:     od
11:     abcast_send_count++;
12: }
```

Since Promela channels are bounded, write operation in a full channel causes the waiting process to block and the atomic block, if any, looses atomicity. The assert command (line 4) is used to check if the *abcast* channel is not full before writing in the channel. This means that the channel size has to be calculated such that it does not become full during verification. With this the sender process will not block and atomicity is assured. If the channel is full, the assert statement will produce an error during the verification. It is important because atomicity is preserved only if no blocking commands are executed in the atomic block. In case of assertion error, it is necessary to set a larger buffer size to the channel and resume the verification.

Primitive *deliver* performs a read on $abcast$ channel. According to Promela semantics, reading messages from a channel follows a FIFO order. Thus, during a reading operation, p_i consumes the oldest message from channel $abcast_i$. As described in Algorithm 2, an array of channels allows processes p_0 to p_{n-1} to read messages from channels $abcast[0]$ to $abcast[n-1]$, respectively.

Algorithm 2 Deliver(m)

```
1: do
2:     :: abcast[id]?m →
3:     skip
4: od
```

In this example, the content of a message is represented by m. Although it has not been described in the algorithm, right after delivering a message, a process $p[i]$ copies m to a list of received messages ($p[i].msg\_list[]$). That step just mimics an application adding each delivered message to a list. Moreover, in order to differentiate messages, an incremental monotonic counter is used to generate exclusive message's content.

We set up a scenario with 3 processes executing and a maximum number of 8 atomic broadcast messages being sent by them. Due to the exhaustive combination among processes execution, the model checker tool creates a total state space containing all possible execution behaviors for this model. That is, traces considering every possible order of execution among the processes are represented.

In order to specify atomic broadcast properties we use Linear Temporal Logic (LTL) formulas [Manna and Pnueli 1991]. LTL allows representation and reasoning about propositions qualified in terms of time. Thus, it is possible to express formulas considering the future of the paths. Next we state atomic broadcast properties using LTL formulas:

Validity: This property states that if a correct process p_i broadcasts a message m, then m will be eventually delivered to p_i. We added arrays $sent[]$ and $delivered[]$ to our model for verification purposes. $sent[i]$ ($delivered[i]$) is updated whenever a new message is sent (delivered) by the i^{th} process. Propositions $p1\_send\_m1$ and $p1\_deliver\_m1$ are defined as $sent[1] = m1$ and $delivered[1] = m1$, respectively. Thereafter, we write validity property in LTL as follows: $\Box(p_1\_send\_m1 \rightarrow \Diamond p_1\_deliver\_m1)$.

Operators $\Box$ and $\Diamond$ represent the globally and eventually conditions. Thus, for all states in generated traces, if $p_i\_send\_m1$ is true, then in a future state $p_i\_deliver\_m1$ must be true.

Although we have illustrated formulas for individual instances m_1 and m_2, we also checked the formulas for general messages m_i and m_j with i and j representing other scenarios. The processes in this model are identical, that means there is no difference on their behaviors. Therefore, we do not need to check every property for processes individually. Once a property holds for p_1, it also holds for other processes p_i.

Uniform Agreement: If a process delivers a message m, then all correct processes eventually deliver m. In other words, uniform agreement states that a message m will be delivered to every process sooner or latter. Thus, we can write uniform agreement formula in LTL as follow: $(\Diamond p1\_deliver\_m1) \rightarrow (\Diamond p2\_deliver\_m1 \wedge \Diamond p3\_deliver\_m1)$.

The use of operator $\Diamond$ (eventually) indicates that there is a time in the future in which message $m1$ will be delivered to each process.

Uniform Integrity: For any message m, every process delivers m at most once, and only if m was previously sent by a process. To check this property, we first verify that a message m is delivered iff m was previously sent by a process using the precedence pattern described in [Salamah et al. 2005]: $\neg p1\_deliver\_m1\ U\ (m1\_sent \vee \Box\neg p1\_deliver\_m1)$. That means $m1$ will never be delivered until $m1$ has been sent by some process ($m1\_sent$ is defined as $sent[0] = m1 \vee sent[1] = m1 \vee sent[2] = m1$).

In order to check that every process delivers m at most once, we used Promela's assert statements which verify whether a boolean condition specified holds. Since we have distinct messages being sent, and each process has a list with delivered messages ($msg\_list[]$), we express the following boolean condition: $\forall 0 \leq i, j < total\_messages \wedge i \neq j, (p[id].msg\_list[i] \neq p[id].msg\_list[j]) \vee p[id].msg\_list[i] = \varnothing$. The assert statement checks the boolean condition whenever a message is delivered. As expected, the assertion is always true, *i.e.* the same message is not delivered twice or more.

Uniform Total Order: If processes p_i and p_j both deliver messages m and m', then p_i delivers m before m', iff p_j delivers m before m'. That means all processes must deliver all messages at the same order.

In addition to proposition $p1\_deliver\_m1$, we define $p1\_deliver\_m2$ as $delivered[1] = m2$, $p2\_deliver\_m1$ as $delivered[2] = m1$, and $p2\_deliver\_m2$ as $delivered[2] = m2$. Thus, we can state the uniform total order property through the LTL formula: $\Box(p1\_deliver\_m1 \rightarrow \Diamond p1\_deliver\_m2) \rightarrow \Box(p2\_deliver\_m1 \rightarrow \Diamond p2\_deliver\_m2)$.

All formulas presented above are satisfied by our model. That means the specification holds atomic broadcast properties. The state space generated by this model was easily tractable by Spin and the executions took less than 1 minute and allocated at most 400 MB of memory.

3. Deferred Update Replication

In this paper we focus on the deferred update replication technique. When compared to primary-backup or state-machine replication, this technique presents better performance [Pedone et al. 1997, Kemme and Alonso 2000, Garcia et al. 2011, Luiz et al. 2011]. Its success stems from the independence of coordination between servers during the execution phase. Initially, a single server is chosen to serve a given transaction and is responsible for the execution of the transaction's operations locally. Only when a client requests the commit of a transaction, the request and some additional transaction's information are broadcast to all servers for certification. This optimistic concurrency control reduces dramatically the communication and synchronization between replicated servers.

3.1. Deferred Update Replication Model

Our model is based on algorithms defined in [Pedone and Schiper 2012] and represents transaction and server processes. The transaction process models a transaction being executed by a client, while the server process reproduces a server replica.

Figure 1 depicts a single transaction execution. A transaction life cycle is separated in two phases: execution and termination. *Execution Phase* encompasses all read and write operations, whilst *Termination Phase* certificates a commit request.

Each transaction keeps a read and a write set. The write set (ws) is a set of tuples with $<item, value>$ and the read set (rs) is a set of tuples with $<item, value, version>$. Write operations are executed locally by the transaction. Every updated item is kept local to the client in the ws until the transaction enters the termination phase. If a read operation accesses an item already in ws, the data value is copied directly from ws. Otherwise, if read item is not in ws, the read operation requests the item value from a server replica.

After executing all read and write operations, the *Termination Phase* starts by sending a *commit request* to all replicated servers. Besides containing client and transaction identifiers, the *commit request* also propagates the rs and ws. This information is used by the server in the certification test. If the transaction's rs contains stale information, then the server decides to abort the transaction.

Differently from messages sent during the *Execution Phase*, the *commit request* is sent through the atomic broadcast protocol (solid lines in Figure 1). That is necessary to

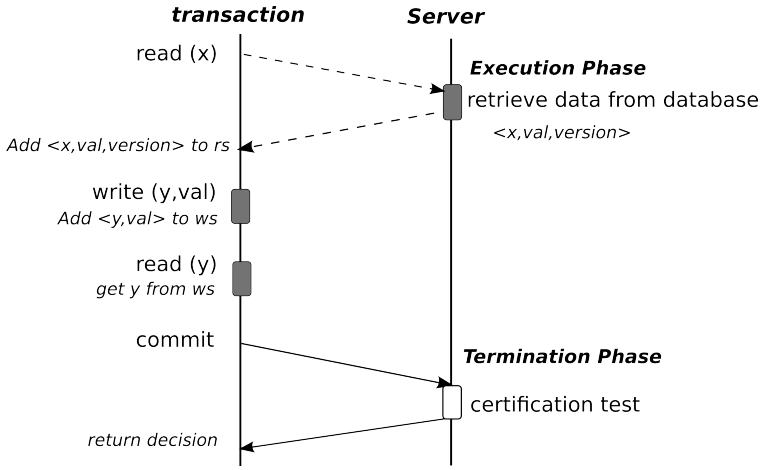


Figure 1. Transaction phases

enforce replicas to receive ongoing transactions in the same order. Once all servers are fully replicated and receive commit messages in the same order, correct servers will take the same decisions in the same order, committing or aborting transactions.

Algorithms 3 and 4 are high level descriptions of transaction and server processes. Message passing over common channels are represented by operators ! and ?. Notice, though, atomic broadcast messages are sent through the *abcast* building block described in previous section. The reuse makes the modeling process simpler and the generated model relies on the building block properties previously verified.

Algorithm 3 T(cid, t)

1: $ws \leftarrow \varnothing$; $rs \leftarrow \varnothing$; $i \leftarrow 0$;
2: choose randomly one of the replica servers s
3: **while** $t.getOp(i) \neq commit \wedge t.getOp(i) \neq abort$ **do**
4: **if** $t.getOp(i) = write$ **then**
5: $ws \leftarrow ws \cup (t.getItem(i), t.getValue(i))$
6: **if** $t.getOp(i) = read$ **then**
7: **if** $t.getItem(i) \in ws$ **then**
8: return v, s.t. $(t.getItem(i), v) \in ws$
9: **else**
10: $c2s[s]!read, t.getItem(i), cid$
11: $s2c[cid]?v, version$ from s
12: $rs \leftarrow rs \cup (t.getItem(i), v, version)$
13: i++;
14: **if** $t.getOp(i) = commit$ **then**
15: $abcast.send(com\_req, cid, t.id, rs, ws)$
16: $s2c[cid]?outcome, s$
17: $t.result = outcome$
18: **else**
19: $t.result = abort$

Before executing operations, one server is randomly selected (l. 2 of Algorithm

3). Write operations do not require communication with the server initially. Instead, they are stored in the write set (ws) (l. 3-4). If a read operation accesses a data item previously updated by the current transaction, then the data value is retrieved from ws (l. 7-8). Otherwise, a read request is sent to the selected server and the received value is added to rs (l. 10-12). When there is no additional read or write operations to execute, the transaction either requests for commit or abort (l. 14 and 18). Further, a commit request message is sent to all replicated servers by atomic broadcast (l. 15). This optimistic approach avoids several communication's rounds throughout the transaction execution.

The server side awaits for messages *read request* or *commit request* (l. 4 and 6 of Algorithm 4). Upon receiving a read request, the server retrieves its value and version for the data item requested (l. 5). For simplicity and reduction of the final size of generated state space, there are only two items in our model (x and y).

Upon receiving a commit request, a certification test verifies if the ongoing transactions ensures serializability [Bernstein et al. 1987]. The server checks if the transaction's read set contains stale items comparing each item's version from rs to the respective item's version in the database. If at least one received item is out of date, the transaction must be aborted (l. 8-11). Otherwise the server decides to commit the transaction. That consists in update item's local version in database, performs all updates according to ws and sent a commit outcome to the requesting transaction (l. 12-19).

Algorithm 4 Server(id)

```
1:  lastCommitted ← 0
2:  db[id].setVersion(x, 0); db[id].setVersion(y, 0)
3:  while true do
4:      :: c2s[id]?read, item, cid →
5:      s2c[cid]!db[id].getVal(item), db[id].getVersion(item)
6:      :: abcast.deliver(com_req, cid, t.id, rs, ws) →
7:      i ← 0; j ← 0;
8:      while rs[i].getItem() ≠ ∅ do
9:          item = rs[i].getItem()
10:         if db[id].getVersion(item) > rs[i].getVersion() then
11:             s2c[cid]!abort, t.id
12:         else
13:             lastCommitted++
14:             db[id].addVersion(item)
15:             while ws[j].getItem() ≠ ∅ do
16:                 item = ws[i].getItem()
17:                 db[id].setItem(item, ws[j].getVal())
18:                 j++
19:             s2c[cid]!commit, t.id
20:         i++
```

Although Pedone *et. al* [Pedone and Schiper 2012] propose optimizations for execution of read only transactions, this aspect has not been yet analyzed thoroughly in our research. Thus, we use the certification test to ensure serializability without distinction between read only or updating transactions.

3.2. Properties Verification

The correct behavior of the protocol can be described by a set of properties. In our investigation, properties were separated in two groups. The first one refers to aspects of the replication approach, such as termination, consistency and agreement in replicated databases. The second specifies scenarios with particular conflicting transactions in order to check if the protocol preserves conflict serializability isolation level.

The scenarios for verification are set up with 2 replicated database servers and 3 transactions t_1, t_2, and t_3, all executing concurrently. Read and write operations are given by $r_i(item)$ and $w_i(item, value)$, respectively, and they are followed by a commit (c_i) or abort (a_i) operation. Transactions t_1 and t_2 are used to specify a particular case of concurrency, while t_3 exploit the non-determinism of model checking to generate any combination of read and write operations over data items x and y. Therefore, during the verification, the transaction t_3 will execute any possible sequence of 3 operations before requesting for commit or abort.

Although t_1 and t_2 express conflicting transactions, the non-determinism of model checking combined with the random generation of t_3 increases the concurrency among transactions' operations. It is a very effective approach, once it increases the verification coverage by inserting automatically execution histories that could hardly be perceived.

3.2.1. Replication Properties

The properties introduced in this section refer to the correct behavior expected by the replication protocol. These safety and liveness properties express termination of transactions, consistency among replicated databases and agreement in transaction's result by the replicas.

For verification we set up transactions t_1 and t_2 as $w_1(x, 11), r_1(y), w_1(y, 21), c_1$, and $r_2(y), r_2(x), w_2(x, 12), c_2$, respectively and t_3 executes up to 3 operations chosen non deterministically before request for commit or abort.

T1 – Transaction Termination: If a transaction is started, then it eventually is decided, *i.e.* it receives a result (commit or abort) from servers.

For verification purposes, we first define propositions $t1\_started = true$ as $t1.started = true$ and $t1\_decided$ as $t1.result \neq 0$, where $t1.started$ is a boolean variable that is set up to true when t_1 starts. Then we state the property T1 as an LTL formula in the form of: $\Box(t1\_started \rightarrow \Diamond t1\_decided)$

The formula specifies that whenever t_1 is started it is eventually decided.

T2 – Uniform Total Order: If two servers s_i and s_j execute transactions t and t', then s_i executes t before t', if and only if s_j executes t before t'.

A transaction finishes its execution in a server once the server answers a commit or abort to that transaction in response to a commit request. Proposition $s_i\_finishes\_t_j$ is defined as $s_i.decided[j] = commit \vee s_i.decided[j] = abort$ and it represents the server decision for a given transaction. Then, the property T2 is specified by the formula $\Box(s1\_finishes\_t1 \rightarrow \Diamond s1\_finishes\_t2) \rightarrow \Box(s2\_finishes\_t1 \rightarrow \Diamond s2\_finishes\_t2)$.

Despite the formula presented above, the use of compositional reasoning would provide a systematic approach to modular verification. Once the deferred update replication model reuses the atomic broadcast model to broadcast messages, the properties verified in previous section would be valid in deferred update replication model as well.

DB1 – Uniform Consistency (Versions): Whenever a server s_i updates an item x to a version v, then every replicated server s_j updates the item x to version v in its instance of the database.

Database replicas consistency is provided once all replicas eventually update the same data items in the same order. In order to check consistency among replica's versions, we verify if (i) a given item x can be updated in a single replica db_i; (ii) for those cases where x is updated, there is no other version of x between versions v_i and v_{i+1}; (iii) all replicated servers execute the same sequence of updates over an item x.

We first need to prove that there are some traces in which the item x is updated to versions v_1 and v_2 (all data items start with version v_0 in our model). Once the model checker generates traces for every possible concurrent execution among transactions as well as all combinations of read and write operations for transaction t_3, it is expected to have situations in which x is updated to v_1 and v_2.

The demonstration of a witness path showing both updates is done by contradiction. Let's say that data item x in replica db_1 will never be updated to versions v_1 and v_2. We can describe it in LTL using the absence pattern [Salamah et al. 2005]: $\Box\neg(\Diamond db1xv1 \wedge \Diamond db1xv2)$, where propositions $db1xv1$ and $db1xv2$ are defined as $db[0].x.version[1] \neq \varnothing$ and $db[0].x.version[2] \neq \varnothing$, respectively. As expected, this property does not hold and a counterexample showing a trace with item x being updated to versions v_1 and v_2 is generated by the model checker tool.

From now on we can check properties (ii) and (iii) against those traces where item x is updated. The correct order for successive updates is verified by the formula $(\Diamond db1xv1 \wedge \Diamond db1xv2) \rightarrow \Box(db1xv1 \rightarrow \Diamond db1xv2)$. Proposition on the left hand side of the logical implication guarantees that traces in which x is not updated twice will not invalidate the formula. On the right hand side of implication we use a response pattern [Salamah et al. 2005], where $db1xv2$ comes after $db1xv1$ globally.

After checking that a single replica is able to update items and that items' version increments in a sequential order, we verify that different replicas perform the same updates in the same order. We state this property as: $(\Box(db1xv_i \rightarrow \Diamond db1xv_{i+1})) \rightarrow \Box(db2xv_i \rightarrow \Diamond db2xv_{i+1})$. That means always the replica db_1 updates x from version v_i to v_{i+1}, then the replica db_2 also updates x in the same order.

DB2 – Uniform Consistency (Values): Two replicated database db_i and db_j have the same value for a same item's version v_i.

That means the value updated in a replica must be the same for the respective version in other replicas. Once we have already verified that different replicas perform the same sequence of updates (property DB1), we check whether the updated values are the same. We use contradiction to show that two replica servers db_1 and db_2 eventually update an item x to the same version. The formula $\Box(db1xv_i \rightarrow \Box\neg dbx\_same\_version)$ states that if db_1 has x at version $v_i : 1 \leq i \leq 3$, then both replicas db_1 and db_2 will

never have item x at the same version. As expected this property is invalid and the model checker shows a witness path with cases where both replicas update x to the same version.

Thereafter, we can check whether item values for the same version in different databases are equal: $\Box(dbx\_same\_version \rightarrow dbx\_same\_value)$, where $dbx\_same\_version$ is defined as $db[0].x.current\_version = db[1].x.current\_version$ and dbx_same_value is defined as $db[0].x.current\_value = db[1].x.current\_value$.

TDB1 – Agreement: Whenever a transaction t has been decided, all replicated servers s_i have previously decided t with the same result.

The transaction result observed by the client must be the same decided by all replicas. In order to check this formula, the following statements must be valid: (i) if replica $db1$ decides to commit, then replica $db2$ also decides to commit; and (ii) if any replica decided to commit, then transaction's result must be *commit*. Before describing LTL formulas, we define the propositions $s_i\_commits\_t_j$ as $s_i.decided[j] = commit$, and $t1\_commit$ as $t1.result = commit$. Then we split property TDB1 in two LTL formulas:

TDB1(i): $\Box(\neg s1\_finishes\_t1 \rightarrow \Diamond s1\_commits\_t1) \rightarrow \Box(\neg s2\_finishes\_t1 \rightarrow \Diamond s2\_commits\_t1)$;

TDB1(ii): $\Box(s1\_commits\_t1 \rightarrow \Diamond t1\_commit)$

3.2.2. Isolation Level Properties

The set of properties presented so far focused mainly on verification of termination and consistency guarantees for the database replicas. However, transactions with conflicting operations would still incur in inconsistencies due to wrong data being manipulated by conflicting operations. Those inconsistencies may happen depending on isolation level provided by the concurrency management protocol [Adya et al. 2000, Kemme and Alonso 2000].

A conflict serializability isolation level requires a history to be conflict–equivalent to a serial history. Lower levels of isolation are less restrictive in terms of concurrency, but they may present inconsistencies. Berenson *et al.* [Berenson et al. 1995] presents a study of the isolation level semantics based on observations of some phenomena, like *dirty read*, *lost update*, *non repeatable read*, and *read/write skew*. Less restrictive isolation levels are susceptible to anomalies caused by some of those phenomena.

The verification scenarios discussed next were devised in order to validate the serializability isolation level provided by the deferred update replication protocol. The model was set up with 3 concurrent transactions. Transactions t_1 and t_2 are set up with conflicting operations that might lead to a specific phenomenon. Transaction t_3 can execute every possible sequence of 3 operations over x or y before requesting for commit or abort. Next we describe some phenomena and then verify if the deferred update protocol disallows those anomalous effects.

Non repeatable read: Transaction t_1 reads a data item. A transaction t_2 then modifies that data item and commits. If t_1 then attempts to reread the data item, it receives a modified value. Transactions are set up as $t_1 : r_1(x), w_1(y, 21), r_1(x), c_1$ and $t_2 : w_2(x, 12), r_2(y), w_2(y, 22), c_2$. A possible history that exemplifies this phenomenon

is $h : r_1(x)..w_2(x,12)..r_1(x)..$ (c_1 and c_2 occur in any order)[1].

In order to avoid this anomaly, t_1 must abort whenever it reads two different versions for a same data item. We check this behavior with an LTL formula in the form of: $\Diamond(t1rx\_v1 \wedge \Diamond t1rx\_v2) \rightarrow \Diamond t1abort$ where $t_krx\_v_j$ is true iff $\exists i \mid i \in t_k.rs \wedge i.item = x \wedge i.version = j$. The property holds, *i.e.* all histories in which a transaction reads two different versions for a same data item result in an abort decision.

Lost Update: Transaction t_1 reads a data item and then t_2 updates the data item. Based on its earlier read value, t_1 updates the data item and commits. The value updated by t_2 will be lost. Transactions are set up as $t_1 : r_1(x), w_1(x,11), w_1(y,21), c_1$ and $t_2 : w_2(x,12), r_2(y), r_2(x), c_2$. A possible history that exemplifies this phenomenon is $h : r_1(x)..w_2(x,12)..w_1(x,11)..c_1..c_2$.

By keeping updated items in the write set, each transaction isolates its local updates from potential updates being performed by other transactions until the transaction terminates. We verify this isolation property through the LTL formula $\Box(t2wx\_val12 \rightarrow \Diamond t2rx\_val12)$. The formula is true once $w_2(x,12)$ happens before $r_2(x)$ and values wrote by other transactions (e.g. $w_1(x,11)$ or any update from t_3) should not be visible by t_2 during its execution. Propositions $t2wx\_val12$ and $t2rx\_val12$ are given by $\exists i \mid i \in t_2.ws \wedge i.item = x \wedge i.value = 12$, and $\exists j \mid j \in t_2.rs \wedge j.item = x \wedge j.value = 12$.

Dirty read: Transaction t_1 modifies a data item. A transaction t_2 then reads that data item before t_1 finishes the transaction. If t_1 then aborts, t_2 has read a data item that had never really existed. Transactions are set up as $t_1 : w_1(x,11), r_1(y), a_1$ and $t_2 : r_2(y), r_2(x), r_2(x), c_2$. A possible history that exemplifies this phenomenon is $h : w_1(x,11)..r_2(x)..a_1..c_2$.

Once transactions keep their updated values locally, there is no way to interfere in the values read by other transactions. The updated values will just be perceived by others after the transaction commits. Considering transactions t_1, t_2, and t_3, we can check that t_2 will never read a value 11 for x describing the absence pattern in the formula: $\Box\neg(t2rx\_val11)$, where the proposition $t2rx\_val11$ is given by $\nexists i \mid i \in t_2.rs \wedge i.item = x \wedge i.value = 11$.

Write skew: Transaction t_1 reads x and y. Another transaction t_2 reads x and y, writes x, and commits. Then t_1 writes y. If there were a constraint between x and y, it might be violated. Transactions are set up as $t_1 : r_1(x), r_1(y), w_1(y,21), c_1$ and $t_2 : r_2(x), r_2(y), w_2(x,12), c_2$. A possible history that exemplifies this phenomenon is $h : r_1(x)..r_2(y)..w_2(x,12)..w_1(y,21)..$ (c_1 and c_2 occur in any order).

Once the deferred update replication protocol preserves serializability isolation level, it must avoid this anomaly by, for example, aborting one of the conflicting transactions. Whenever the operation $r_2(y)$ happens before c_1, and c_1 happens before t_2 request for commit, then servers s_1 and s_2 must, obligatorily, decide to abort t_2. This can be checked with the formula $(\Diamond(s1\_commits\_t1 \wedge \neg s1\_finishes\_t2 \wedge \neg s2\_finishes\_t2) \wedge ((\neg s1\_commits\_t1 \wedge \neg s2\_commits\_t1)\ U\ t2ry)) \rightarrow \Diamond t2abort$. The formula shows that if t_1 is committed in s_1, and t_2 not yet committed, and t_2 reads y before t_1 is committed,

[1]For the sake of simplicity, irrelevant operations for illustration of the phenomenon are omitted in h. The .. in the history suppresses any possible sequence of interleaved operations from t_1 and t_2.

then t_2 will abort.

Except by contradiction formulas (DB1(i) and DB2(i)), which were intentionally used to expose witness paths for desirable behaviors, all other specified formulas hold. That means the deferred update replication model satisfied the properties enunciated above. The model checker Spin was set up with partial order reduction and compression enabled. The experiments were performed in a computer with CPU of 6 cores and 2.66 GHz, and 32 GB of main memory. Table 1 exhibits the verification results.

Table 1. Resources allocated for Deferred Update Replication model

Formula	Stored States	Memory (MB)	Time
T1	50453993	3670.071	12 min
T2	60330993	4204.511	16 min
DB1(i)	3551454	350.263	33 sec
DB1(ii)	55748067	3959.893	13 min
DB1(iii)	68001328	4612.778	21 min
DB2(i)	47	129.106	1 sec
DB2(ii)	26440649	1898.649	4 min
TDB1(i)	53556032	3842.877	12 min
TDB1(ii)	32962609	2743.100	6 min
Non Repeatable Read	63838571	5032.491	13 min
Lost Update	40962043	3575.372	8 min
Dirty Read	20564107	1476.934	3 min
Write Skew	53473325	4162.453	8 min

4. Related Work

Most of the authors in literature justify correctness of replication protocols in a rather informal way without support of formal methods. Specially for the deferred update replication, most researchers sketch proofs of their designs in a natural language description [Kemme and Alonso 2000, Garcia et al. 2011, Luiz et al. 2011, Pedone and Schiper 2012]. However, due to the high concurrency level and inherent complexity of environments in which those protocols execute, a formal verification of those protocols is required.

Armedáriz-Iñigo *et al.* [Armendáriz-Iñigo et al. 2009] present a formal specification and correctness proof for replicated database systems. They carefully describe database replicas and the underlying replication protocol. Similar to our work, they check properties for atomic broadcast communication as well as database consistency provided by a certification-based protocol. A small difference between their and our work is that the protocol they analyzed assumes snapshot isolation level of consistency while the replication protocol we presented preserves serializability. They used I/O automata to describe system's behavior and verified system's properties through theorem proving.

Schmidt *et al.* [Schmidt and Pedone 2007] formally proved that a generic deferred update protocol preserves the serializability property. First they modeled a serial database and the termination phase of the protocol using TLA+. Then, by using refinement mapping, they proved that the states generated by the termination phase of the protocol are

equivalent to those generated by the serial database model. The authors described a deductive correctness proof, combining theorem proving and refinement mapping techniques.

In this paper we illustrated how useful model checking is for verification of data replication management protocols. Compared to [Schmidt and Pedone 2007] and [Armendáriz-Iñigo et al. 2009], the main advantages of this approach is the automatic verification and a very expressive description of properties by using of temporal logic. Moreover, due to its abstraction level, Promela models are closer to implementation and more familiar to distributed system developers if compared to I/O automata and TLA$^+$. This can help to better bridge the gaps between model and implementation.

5. Conclusion

This paper illustrated the use of model checking for database replication techniques. Specially, we recall the deferred update replication algorithm, that has been successfully adopted to increase availability with good performance. The experiments presented in this paper consolidate a first step on formal verification of deferred update replication protocol using model checking.

The use of Promela language provides a concise specification of the protocol in an algorithmic style. Properties are represented in LTL and their verification allowed us to observe the correct behavior expected by the protocol. In fact, temporal logic provided a very natural way to specify temporal relationships among operations executed throughout the complete history of the model computation. Section 3.2 demonstrated how to specify some common concurrency scenarios and how to check if undesirable phenomena affect the correctness of the protocol.

Another subject addressed by this paper is how to create models from smaller models previously verified. Modular approaches are common in distributed systems development, where specialized modules are coupled to a same system. For instance, a reliable system would be equipped with failure detectors, reliable channels (e.g. implementing atomic broadcast), or security components. Although we checked the uniform total order property in both, atomic broadcast and deferred update replication models, further development through compositional reasoning [Dotti et al. 2006] would provide a systematic approach to modular verification using building block models.

Extensions of the deferred update protocol tolerate crash or byzantine failure models [Luiz et al. 2011, Pedone and Schiper 2012]. In this direction, a next step for our work is to check protocol correctness under certain failure behaviors. It can be done by automatic insertion of failures to the model, followed by model checking [Dotti et al. 2005]. Fault injection combined to model checking is very attractive, since it is capable to represent non trivial faulty scenarios and disclosure misbehaviors hardly perceptible.

References

Adya, A., Liskov, B., and O'Neil, P. (2000). Generalized Isolation Level Definitions. In *Data Engineering, 2000. Proceedings. 16th International Conference on*.

Agrawal, D., Alonso, G., El Abbadi, A., and Stanoi, I. (1997). Exploiting atomic broadcast in replicated databases. In *Euro-Par'97 Parallel Processing*, volume 1300 of *Lecture Notes in Computer Science*.

Armendáriz-Iñigo, J. E., González, D. M., Garitagoitia, J. R., and Muñoz-escoí, Francesc, D. (2009). Correctness proof of a database replication protocol under the perspective of the I/O automaton model. *Acta Informatica*, 46(4).

Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., and O'Neil, P. (1995). A critique of ANSI SQL isolation levels. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, SIGMOD '95. ACM.

Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.

Défago, X., Schiper, A., and Urbán, P. (2004). Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv.*, 36(4).

Dotti, F. L., Mendizabal, O. M., and dos Santos, O. M. (2005). Verifying Fault-Tolerant Distributed Systems Using Object-Based Graph Grammars. In *LADC*, volume 3747 of *Lecture Notes in Computer Science*. Springer.

Dotti, F. L., Ribeiro, L., Santos, O. M., and Pasini, F. (2006). Verifying object-based graph grammars. *Software & Systems Modeling*, 5.

Garcia, R., Rodrigues, R., and Preguiça, N. (2011). Efficient middleware for byzantine fault tolerant database replication. In *6th Conference on Computer systems*, EuroSys '11. ACM.

Hadzilacos, V. and Toueg, S. (1994). A modular approach to fault-tolerant broadcasts and related problems. Technical report, Ithaca, NY, USA.

Holzmann, G. (1991). *Design and Validation of Computer Protocols*. Prentice Hall.

Holzmann, G. J. (1997). The model checker SPIN. *Software Engineering, IEEE Transactions on*, 23(5):279–295.

Kemme, B. and Alonso, G. (2000). A new approach to developing and implementing eager database replication protocols. *ACM Trans. Database Syst.*, 25(3).

Luiz, A., Lung, L. C., and Correia, M. (2011). Byzantine fault-tolerant transaction processing for replicated databases. In *Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on*.

Manna, Z. and Pnueli, A. (1991). Completing the temporal picture. *Theoretical Computer Science*, 83(1).

Pedone, F., Guerraoui, R., and Schiper, A. (1997). Transaction Reordering in Replicated Databases. In *16th IEEE Symposium on Reliable Distributed Systems*.

Pedone, F. and Schiper, N. (2012). Byzantine fault-tolerant deferred update replication. *Journal of the Brazilian Computer Society*, 18.

Salamah, S., Gates, A., Roach, S., and Mondragon, O. (2005). Verifying pattern-generated LTL formulas: A case study. In *Model Checking Software*, volume 3639 of *Lecture Notes in Computer Science*.

Schmidt, R. and Pedone, F. (2007). A formal analysis of the deferred update technique. In *11th International Conference on Principles of distributed systems*, OPODIS'07, Berlin, Heidelberg. Springer-Verlag.

Processamento Distribuído de Operações de Junção Espacial com Bases de Dados Dinâmicas para Análise de Informações Geográficas

Sávio S. T. de Oliveira[1], **Vagner J. do Sacramento Rodrigues**[1], **Anderson R. Cunha**[1], **Everton L. Aleixo**[1], **Thiago B. de Oliveira**[1], **Marcelo de C. Cardoso**[1], **Roberto R. Junior**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Bloco IMF I, Campus II - Samambaia – Goiânia – GO – Brasil

{savio.teles,vagner.sacramento,anderson.cunha,everton.lima,
thiago.borges,marcelo.cardoso,roberto.junior}@lupa.inf.ufg.br

Abstract. *This paper presents the proposal of a new data distribution technique, called Proximity Area, that optimizes the processing of distributed spatial join operations for analysis of a large volume of spatial data in dynamic datasets. The techniques found in the literature, for processing distributed spatial join, perform data distribution in static datasets, where it is necessary redistribute the objects on cluster for each dataset update. This becomes unfeasible for datasets with large volume of data and constant updates. The experiments have shown the efficiency of the Proximity Area and the impact of the data distribution on distributed spatial join.*

Resumo. *Este artigo apresenta a proposta de uma nova técnica de distribuição de dados, denominada Proximity Area, que otimiza o processamento de operações de junção espacial distribuída para análise de um grande volume de dados geográficos em bases de dados dinâmicas. As técnicas encontradas na literatura, para processamento da junção espacial, realizam a distribuição de dados em bases de dados estáticas, onde é necessário redistribuir os objetos pelo cluster a cada atualização da base de dados. Isto se torna inviável para bases de dados com grande volume de dados e com constantes atualizações. Os experimentos realizados demonstraram a eficiência da Proximity Area e apresentaram o impacto da distribuição de dados sobre a junção espacial distribuída.*

1. Introdução

Se a resposta para a pergunta "Onde?" é importante para o seu problema, um Sistema de Informação Geográfico (SIG) é a solução. No entanto, só visualizar dados em um mapa não supre a demanda de informações para tomada de decisão que precisam fazer análises de grandes volumes de dados em tempo real. Em muitos casos, é necessário realizar uma operação de junção espacial, correlacionando diferentes camadas de dados. Por exemplo: encontrar as áreas desmatadas no Brasil que estão próximas de leitos de rios.

Os algoritmos de processamento da junção espacial apresentam alto custo computacional [Mutenda and Kitsuregawa 1999]. Em função disto, as soluções de análise de dados geográfico que manipulam grande volume de dados geográficos devem ser capazes

de paralelizar o processamento da junção espacial entre os computadores de um *cluster*. Com isso, alguns desafios são identificados: i) distribuição dos dados pelo *cluster* e ii) processamento paralelo e distribuído da junção espacial.

O objetivo deste trabalho é apresentar uma nova técnica de distribuição de dados, denominada *Proximity Area*, que otimiza o processamento de operações de junção espacial distribuída para análise de grande volume de dados geográficos em bases de dados dinâmicas. Os trabalhos na literatura têm explorado a distribuição em bases de dados estáticas, onde qualquer atualização da base de dados requer que todos os dados sejam novamente distribuídos pelo *cluster*. Isto se torna inviável em bases de dados com grandes volumes de dados e que sofrem constantes atualizações. *Proximity Area* consegue atualizar a base de dados sem que haja a necessidade de redistribuir os objetos pelo *cluster*. Esta técnica foi implementada sobre a plataforma de *middleware* para geoprocessamento distribuído, DistGeo, baseada no modelo peer-to-peer para comunicação entre os servidores.

As principais contribuições deste trabalho são:

- Implementação de um algoritmo de junção espacial distribuída para análise de dados geográficos;
- Uma nova técnica de distribuição de dados para bases de dados dinâmicas.

O restante do trabalho está organizado como segue. A Seção 2 apresenta uma revisão das estruturas de dados espaciais existentes e a visão geral da operação de junção espacial. A Seção 3 apresenta a arquitetura da plataforma DistGeo e a técnica de distribuição de dados *Proximity Area*. A Seção 4 descreve a metodologia e os resultados dos testes executados na plataforma. A Seção 5 compara a plataforma DistGeo com as estratégias encontradas na literatura para processar a junção espacial distribuída. A Seção 6 apresenta as conclusões deste trabalho e uma breve descrição dos trabalhos futuros.

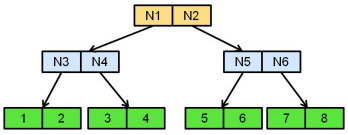
2. Processamento de Dados Espaciais

A indexação de dados espaciais vetoriais vem sendo pesquisada desde 1975, o que ocasionou o surgimento de diversas estruturas de dados, sendo as principais: KD-Tree [Bentley 1975], Hilbert R-Tree [Kamel and Faloutsos 1994] e a R-Tree [Guttman 1984].

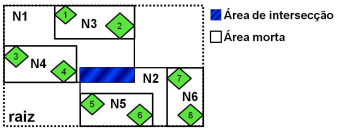
A R-Tree é uma árvore balanceada por altura, semelhante a B^+-Tree, com ponteiros para objetos espaciais nos nós folhas. É uma estrutura de dados hierárquica que utiliza retângulos para organizar um conjunto dinâmico de objetos espaciais, de maneira que objetos co-localizados fiquem armazenados próximos uns dos outros e que haja uma redução no espaço de busca a cada nível da árvore [Guttman 1984]. Estes retângulos, chamados de MBRs, *Minimum Bounding Rectangle*, possuem área menor possível para envolver as geometrias dos filhos. Na Figura 1(b), o MBR de N3 é o menor possível para envolver os filhos 1 e 2.

A Figura 1(a) ilustra a estrutura hierárquica da R-Tree com um nó raiz, nós internos (N1..2 $\subset$ N3..6) e um último nível de nós folha (N3..6 $\subset 1..8$). Cada nó armazena no máximo M e no mínimo $m \leq \frac{M}{2}$ entradas [Guttman 1984]. A Figura 1(b) retrata o desenho dos MBRs agrupando os objetos espaciais de 1 a 8 em subconjuntos, de acordo com sua co-localização.

Entre as várias extensões propostas para a R-Tree, a R*-Tree [An et al. 2003] foi escolhida para implementação da arquitetura proposta neste trabalho. Esta variante propõe mecanismos para melhorar o tempo de busca [Beckmann et al. 1990] como reduzir espaço morto e áreas sobrepostas entre os MBRs. Espaço morto é a área adicional do MBR necessária para cobrir o polígono como um todo. Na Figura 1(b), a área de N1 não preenchida pelas suas entradas é um exemplo de espaço morto. Áreas sobrepostas são regiões de interseção entre polígonos. Um exemplo de sobreposição é ilustrado na Figura 1(b), entre N1 e N2.



(a) Índice espacial *R-Tree*

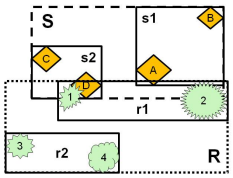


(b) Disposição espacial dos objetos geográficos

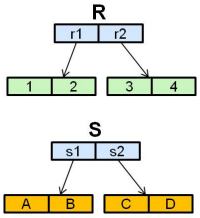
Figura 1. Estrutura de dados espacial

2.1. Junção espacial

A junção espacial pode ser definida a partir de duas relações R = $r_1, ..., r_n$ e S = $s_1, ..., s_m$, onde r_i e s_j são objetos espaciais, $1 \leq i \leq n$ e $1 \leq j \leq m$. A operação verifica todos os pares (r_i, s_j) que satisfazem o predicado de um operador topológico, por exemplo a interseção, isto é, $r_i \bigcap s_j \neq \emptyset$ [Jacox and Samet 2007]. Este trabalho utiliza um algoritmo de junção espacial que caminha por duas relações R e S indexadas por *R*-Trees* [Brinkhoff et al. 1993].



(a) Disposição espacial das relações R e S



(b) Indexação das relações R e S na *R*-Tree*

Figura 2. Junção espacial entre as relações R e S

O processamento da junção é realizado em duas etapas: etapa de filtragem e etapa de refinamento [Patel and DeWitt 1996]. A etapa de filtragem inicia na raiz das duas relações R e S e é realizada nos nós internos da *R*-Tree*. Esta etapa utiliza aproximações das geometrias dos objetos [1] na operação de intersecção para gerar um conjunto de possíveis respostas à consulta. Observando a Figura 2(a), percebe-se que o MBR de r1 intersecta

[1] Aproximações são polígonos utilizados para reduzir a complexidade das geometrias para poucos pontos geométricos [Brinkhoff et al. 1994]. Por exemplo: uma geometria com milhares de pontos pode ser representada por um MBR com 2 pontos. Estas aproximações geram espaços mortos e, por isso, são utilizadas na operação de intersecção para descartar resultados incorretos. A intersecção entre duas aproximações não significa que suas respectivas geometrias também intersectam.

com os MBRs de s1 e s2, mas que o MBR de r2 não intersecta com nenhum item na relação S. Por isso, o conjunto de saída da etapa de filtragem é formado pelos pares (r1, s1) e (r1, s2). A fase de refinamento é realizada nas folhas e remove deste conjunto os resultados incorretos utilizando as geometrias reais de cada objeto. Observando a Figura 2(b), a etapa de refinamento irá analisar os nós filhos de (r1, s1) e (r1, s2). Nesta fase houve a necessidade de verificar os seguintes conjuntos de pares de candidatos {(1, A), (1, B), (1, C), (1, D), (2, A), (2, B), (2, C), (2, D)} e apenas (1, D) fez parte do resultado final por apresentar intersecção entre suas respectivas geometrias.

3. Processamento da junção espacial distribuída

Os algoritmos de junção espacial apresentam alto custo de processamento e, por isso, as pesquisas têm se concentrado em resolver o problema de forma distribuída. A Figura 3 ilustra as duas *R*-Trees*, R e S, apresentadas na Figura 2(b) distribuídas em um *cluster* de computadores. Os algoritmos de inserção e junção espacial executados na *R*-Tree* centralizada podem ser processados de forma semelhante na versão distribuída, exceto pela i) necessidade de troca de mensagens para acessar os objetos distribuídos e ii) pelo tratamento de concorrência e consistência necessário para ao processamento paralelo e distribuído dos algoritmos [de Oliveira et al. 2011].

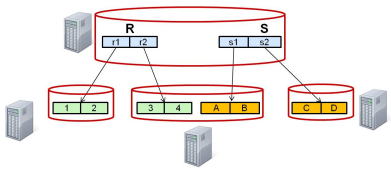


Figura 3. R-Tree distribuída

Nem sempre os dados necessários para a junção entre as bases de dados estão disponíveis localmente. No exemplo da Seção 2.1, a junção espacial entre R e S teve como resultado o par (1, D). Observando a Figura 3, percebe-se que os objetos 1 e D estão localizados em máquinas diferentes. Portanto, para processar a junção espacial entre os dois objetos, um deles deve ser trafegado na rede até o local em que está armazenado o outro objeto. Para reduzir o tráfego de dados na rede, o algoritmo de processamento da junção espacial da plataforma DistGeo trafega o objeto com menor número de pontos.

3.1. DistGeo: Plataforma de Geoprocessamento Distribuído de Operações Espaciais

Como pode ser visto na Figura 4, a plataforma DistGeo é constituída pelas aplicações clientes, servidores que executam as operações espaciais e um serviço de nomes replicado. O serviço de nomes armazena informações sobre os servidores ativos e as bases de dados que foram inseridas no sistema. As aplicações clientes se comunicam com a plataforma através de uma API cliente, que disponibiliza métodos para atualização e consultas nas bases de dados. O serviço de nomes é consultado pela API cliente para decidir em qual servidor executar a operação espacial desejada. A API cliente envia uma requisição e recebe uma resposta do um servidor escolhido e repassa o resultado da operação para a aplicação cliente. Cada servidor é responsável por executar as operações espaciais requisitadas pelas aplicações clientes.

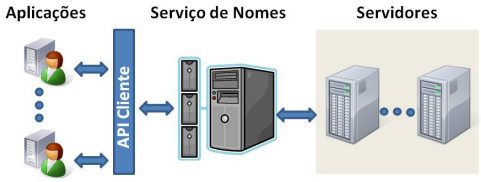


Figura 4. Plataforma DistGeo

A arquitetura da plataforma DistGeo é baseada no modelo *peer-to-peer* híbrido, na qual os servidores trocam mensagens entre si, mas necessitam do serviço de nomes para obterem o endereço IP dos outros servidores. Cada servidor guarda em uma *cache* os endereços já obtidos, para que não seja sempre necessário acessar o serviço de nomes para se comunicar com outro servidor. A plataforma possui tratamentos de concorrência e consistência semelhantes aos apresentados em [de Oliveira et al. 2011].

O primeiro servidor que inicia no *cluster* se cadastra no serviço de nomes como monitor. Este servidor tem ciência de quais objetos foram inseridos e em quais máquinas foram armazenados. Para cada objeto novo no sistema, deve-se requisitar as informações do *cluster* ao servidor monitor para decidir para qual máquina enviar o novo objeto. Depois de alocado, deve-se informar ao monitor em qual servidor este objeto foi armazenado. O serviço de nomes monitora cada servidor e, quando o monitor cai, ele fica ciente e avisa aos outros servidores do *cluster*. Neste caso, os outros servidores tentam se cadastrar como monitor e o primeiro que conseguir, assume a função utilizando as últimas informações obtidas do antigo monitor.

3.2. Estratégia de distribuição em bases de dados dinâmica

A distribuição de dados pelas máquinas do *cluster* é o fator que mais influencia no paralelismo em um ambiente *clusterizado* [Mutenda and Kitsuregawa 1999] e possui dois requisitos principais [Patel and DeWitt 2000]: a) os dados devem ser distribuídos de forma balanceada pelas máquinas do *cluster* e b) uma máquina deve possuir a maior parte dos dados que precisa para processar uma operação localmente, ou seja, não é necessário obter dados de outra máquina.

Este trabalho apresenta um nova técnica de distribuição para bases de dados dinâmicas denominada *Proximity Area*. Esta busca manter objetos, de bases de dados diferentes, espacialmente próximos na mesma máquina (**co-localizar**) para que ocorra menos tráfego na rede nas operações de junção espacial. Para que os dados fiquem distribuídos de forma balanceada, foi criado um fator de balanceamento **k** - entre 0 e 1 - que limita a diferença na quantidade de objetos entre os servidores. Manter o *cluster* balanceado aumenta o grau de paralelismo na execução da operação de junção espacial distribuída e, consequentemente, é possível aproveitar melhor os recursos disponíveis no *cluster*.

O Algoritmo 1 apresenta a descrição da técnica *Proximity Area*. Este algoritmo tem como objetivo escolher o servidor S no qual o novo objeto O será alocado e recebe três parâmetros: a) MBR de O, b) fator de balanceamento k e c) $lista$, obtida no monitor, que contém as informações dos servidores do *cluster*. O servidor monitor armazena duas informações de cada máquina: i) o MBR que engloba os objetos de todas as bases de dados naquela máquina; ii) quantidade de objetos. Cada elemento I em $lista$ possui

como atributos a referência para o servidor S correspondente a I, o número de objetos em S e o MBR que representa a área espacial que engloba todos os objetos das diferentes bases de dados alocados em S.

Algoritmo 1: $ProximityArea(M, k, lista)$

Entrada: M MBR do objeto alocado, k fator de balanceamento, $lista$ informações de distribuição dos servidores do cluster
Saída: Referência para o servidor escolhido

```
1  min ← inteiro máximo
2  para cada elemento I em lista faça
3      se I.numObjetos = 0 então
4          retorna I.referenciaServidor
5      fim
6      se I.numObjetos < min então
7          min ← I.numObjetos
8      fim
9  fim
10 minArea ← número de ponto flutuante máximo
11 referenciaServidor ← null
12 para cada elemento I em lista faça
13     se (min/I.numObjetos) > k então
14         area ← aumento de área de I.MBR para inserir M
15         se area < minArea então
16             minArea ← area
17             referenciaServidor ← I.referenciaServidor
18         fim
19     fim
20
21 fim
22 retorna referenciaServidor
```

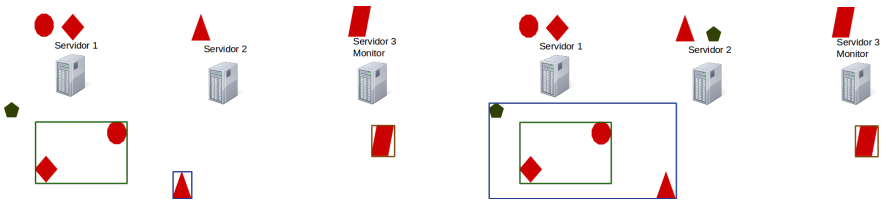
O Algoritmo 1 realiza um primeiro laço entre as linhas 2 e 9 que verifica se existe algum servidor que não tenha nenhum objeto alocado. Neste caso, o algoritmo retorna este servidor como resposta. Caso não exista nenhum servidor vazio, ao final do laço a variável min guardará o número de objetos do servidor com menor quantidade de dados alocados.

No segundo laço, na linha 13 é verificado se o servidor S em questão contém um número de objetos que não obedecem ao fator de balanceamento k. Para isso, a divisão entre min e o número de objetos $I.numObjetos$ em S deve ser maior que k. Se, por exemplo, $min = 1$, $k = 0,5$ e $I.numObjetos = 2$, então $(min/I.numObjetos) = 0,5$ não é maior que k e, neste caso, devemos descartar este servidor na alocação de O. Entre os servidores que obedecerem o fator de balanceamento, é escolhido aquele cujo MBR está mais próximo espacialmente do MBR do novo objeto O.

Objetos espaciais têm características de distribuição não uniforme e, como não é possível controlar a ordem com que os objetos espaciais de uma base de dados são inseridos, as técnicas de distribuição dinâmicas de dados não conseguem alocar estes

objetos da melhor forma possível [Zhou et al. 2011]. A técnica *Proximity Area* também apresenta este comportamento, como pode ser visto na Figura 5 na inserção de um novo objeto (pentágono) com $k = 0,5$. O objeto deveria ser alocado no Servidor 1, que possui área espacial mais próxima do pentágono. Mas, como $min = 1$ e o Servidor 1 possui 2 objetos, então o número de objetos no Servidor 1 não obedece o fator de balanceamento. Com isto, o objeto será alocado no Servidor 2 que obedece o fator de balanceamento e está mais próximo espacialmente do objeto que o Servidor 3.

Assim, observando a Figura 5, percebe-se que a área do Servidor 2 não contém objetos co-localizados e, isto, gera um aumento no número de mensagens na rede na operação de junção espacial distribuída. Este problema será mitigado em trabalhos futuros, através da redistribuição dos dados entre os servidores em determinados intervalos de tempo.



(a) Antes da inserção (b) Depois da inserção

Figura 5. Inserção de um objeto utilizando a técnica *Proximity Area* com k = 0,5

4. Avaliação de Performance

Para analisar o desempenho da plataforma com a técnica de distribuição de dados *Proximity Area*, foram medidos o tempo de resposta e a quantidade de bytes transferidos na rede para realizar a operação de junção espacial. A quantidade de bytes transferidos na rede é importante para verificar o quanto os dados estavam co-localizados. Quanto mais co-localizados, menor é o tráfego de rede, mas em contrapartida, a distribuição do processamento pelas máquinas do *cluster* fica prejudicada.

4.1. Ambiente Experimental e Bases de Dados

Foram utilizadas bases de dados geográficas disponibilizadas pelo LAPIG (Laboratório de Processamento de Imagens e Geoprocessamento)[2], que permitiram avaliar a plataforma DistGeo em um ambiente de *cluster* com dados geográficos reais. Para estudar o impacto das peculiaridades de cada base de dados na performance da plataforma, foram avaliadas bases de dados com características diferentes em relação a: a) Tipo de geometria: polígonos, linhas e pontos; b) Número de itens e c) Tamanho em disco. As bases de dados utilizadas estão descritas na Tabela 1.

Para avaliar o desempenho da plataforma na execução da junção espacial distribuída, foram realizadas as seguintes junções: a) Bioma do Cerrado e Desmatamento do Cerrado, que retorna 60798 resultados; b) Bioma da Caatinga e Localidades, que retorna 3934 resultados; c) Rodovias e Hidrografia, que retorna 55764 resultados. Essa combinação de experimentos permite avaliar a junção espacial com bases de dados com características diferentes, conforme pode ser observado na Tabela 1.

[2]www.lapig.iesa.ufg.br

Tabela 1. Descrição das bases de dados

Base de Dados	Geometria	Número de itens	Tamanho(MB)
Desmatamento do Cerrado	Polígono	32578	11,2
Bioma do Cerrado	Polígono	151986	411,3
Bioma da Caatinga	Polígono	10994	275,3
Hidrografia	Linha	226963	64,5
Rodovias	Linha	51645	15,2
Localidades	Ponto	21840	1,4

Foram encontradas na literatura duas técnicas de distribuição de dados para bases dinâmicas. Em [Zhou et al. 2011], os objetos espacialmente próximos são enviados para máquinas diferentes. Esta técnica é uma estratégia interessante para paralelizar consultas em apenas uma base de dados. Para a junção espacial se torna inviável, pois gera um aumento na quantidade de tráfego na rede por não manter os dados co-localizados. Em [de Oliveira et al. 2011] é apresentado uma técnica semelhante a Round Robin, onde os dados são distribuídos de forma balanceada pelo *cluster*. Esta técnica, entretanto, não foi testada em operações de junção espacial. Por isso, o desempenho da plataforma foi analisado utilizando duas técnicas de distribuição de dados: i) *Proximity Area* e ii) *Round-Robin*.

Para avaliar a técnica *Proximity Area*, foram utilizados como fator de balanceamento os valores 0.1 (PA 0.1), 0.5 (PA 0.5) e 0.9 (PA 0.9). Esta variação no fator de balanceamento permite investigar o impacto do balanceamento e a co-localização dos dados. Geralmente, quanto menor o fator de balanceamento mais co-localizados estão os dados, entretanto menos balanceado fica o *cluster*.

Os testes foram realizados com máquinas Optiplex 780 Intel Core 2 Quad 2.83GHz, com 4 Gb de memória RAM e foram configurados da seguinte forma: doze Servidores, um Cliente e um Servidor de Nomes. As máquinas foram conectadas por uma rede Ethernet 1Gbit/segundo e um switch Dell PowerConnect 6248P.

A junção espacial distribuída foi avaliada com 1, 4, 6, 8, 10 e 12 servidores. Cada junção descrita anteriormente foi realizada cinco vezes, em cada configuração do *cluster*, onde foi descartada a junção de maior tempo de execução e a de menor tempo e calculada uma média aritmética dos tempos das três restantes.

4.2. Avaliação

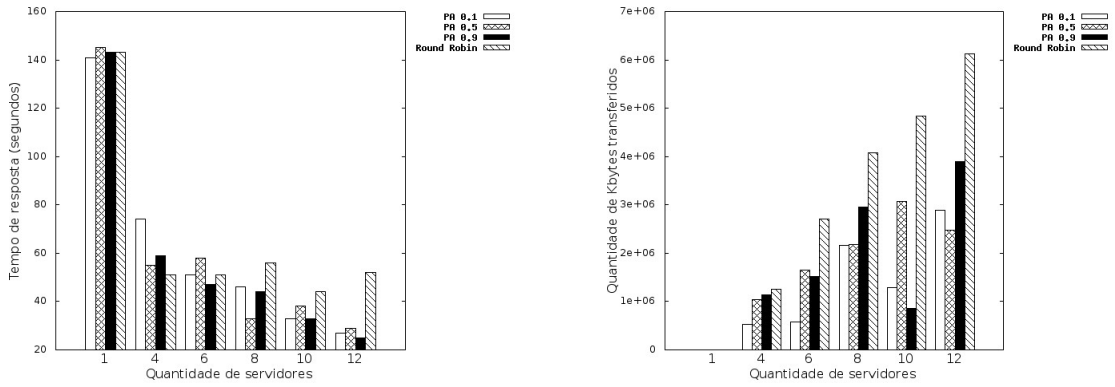
A Figura 6 apresenta o resultado das junções espaciais realizadas entre Desmatamento do Cerrado e Biomas do Cerrado. Estas duas bases de dados possuem geometrias complexas (grande número de pontos) e, por isso, a junção espacial apresenta um alto custo de processamento e tráfego de dados na rede. A Figura 6(a) demonstra que a plataforma DistGeo apresentou escalabilidade com o acréscimo de máquinas no *cluster* devido a alta demanda de processamento desta junção.

Como pode ser visto na Figura 6(b), a diferença na quantidade de bytes transferida na rede entre *Round Robin* e as outras técnicas é grande, pois *Round Robin* não consegue manter os itens co-localizados. Por isso, a técnica *Proximity Area* apresentou um desempenho melhor que *Round Robin* com o acréscimo de máquinas, pois a diferença no tráfego

de dados na rede aumenta consideravelmente.

Como a diferença de tráfego de dados na rede entre PA 0.9 e as técnicas PA 0.1 e PA 0.5 foi pequena, PA 0.9 apresentou melhor desempenho, pois consegue além de co-localizar os dados, distribuir o processamento entre as máquinas do *cluster*. A técnica PA 0.9 apresentou o melhor desempenho entre todas as técnicas, sendo, na média, 21% melhor que *Round Robin*.

Assim, em junções que apresentam um alto custo de processamento e tráfego de dados na rede, um fator de balanceamento alto consegue paralelizar o processamento e, ao mesmo tempo, manter uma co-localização satisfatória dos dados se comparado ao *Round Robin*.



(a) Tempo de resposta (b) Quantidade de Kbytes tranferidos na rede

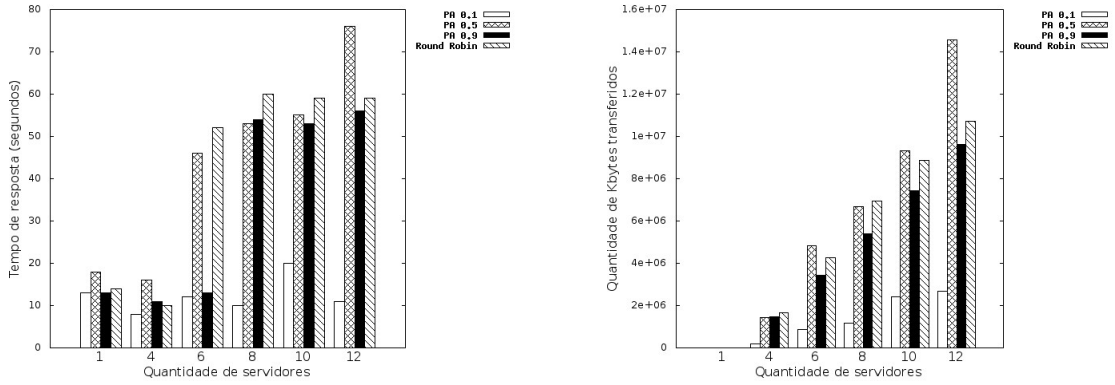
Figura 6. Junção espacial entre Bioma do Cerrado e Desmatamento

A junção entre as bases de dados Rodovias e Hidrografia analisa uma grande quantidade de pares de objetos que não fazem parte do resultado da consulta. Isso porque estas bases de dados possuem geometrias de linhas que geram MBRs com grande espaço morto. Este espaço morto exige que uma grande quantidade de pares de objetos das bases de dados sejam analisados para gerar o resultado da consulta. Para analisar estes pares, devem ser realizadas cópias de objetos entre as bases de dados para processar a junção espacial. Isto resulta em uma grande quantidade de tráfego de dados na rede.

Métricas coletadas nos testes mostraram que o uso da CPU no processamento desta junção é pequeno (em média 30%) porque as geometrias são simples e, com isto, o tráfego de dados se torna dominante. Por isso, pode ser observado na Figura 7 que o tempo de resposta da junção espacial foi proporcional ao tráfego de dados na rede. As técnicas não se beneficiaram com a adição de máquinas, já que a quantidade de bytes transferidos aumenta com um número maior de servidores - Figura 7(b). Como pode ser visto na Figura 7(a), a técnica *Proximity Area* apresentou na média desempenho melhor que *Round Robin*, pois consegue co-localizar os dados.

PA 0.1 teve o melhor desempenho (4 vezes melhor que *Round Robin*), pois consegue concentrar objetos espacialmente próximos na mesma máquina. Portanto, para junções onde o tráfego de dados na rede é dominante, um fator de balanceamento menor gera um melhor resultado, pois reduz o tráfego na rede através de uma melhor co-localização dos dados. Entre as restantes, PA 0.9 teve o melhor desempenho, sendo 29% melhor que a técnica *Round Robin*.

PA 0.5 apresentou um comportamento estranho neste teste, pois a quantidade de tráfego de dados trafegados na rede foi maior que as técnicas PA 0.9 e *Round Robin*, apesar de teoricamente PA 0.5 conseguir co-localizar melhor os dados. Por isso, PA 0.5 teve o pior desempenho entre todas as técnicas. Como foi dito na Seção 3.2, estas situações podem ocorrer devido a ordem de inserção dos itens e serão investigadas em trabalhos futuros.



(a) Tempo de resposta (b) Quantidade de Kbytes tranferidos na rede

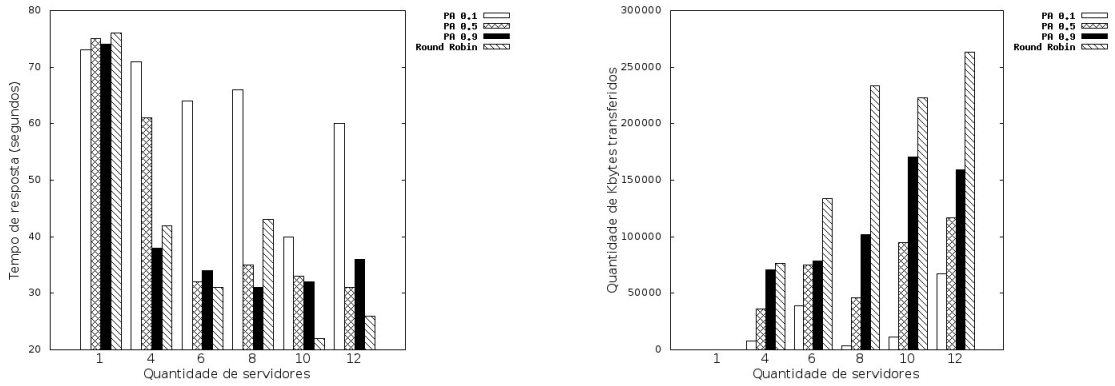
Figura 7. Junção espacial entre Hidrografia e Rodovias

Conforme apresentado na Seção 3, a plataforma DistGeo trafega na junção espacial os objetos com menor número de pontos para reduzir a quantidade de dados na rede. Na junção espacial entre Bioma da Caatinga e Localidades, a base de dados de Localidades é transferida, já que apresenta geometrias de pontos, enquanto Bioma da Caatinga apresenta geometrias complexas. Por isso, pode ser observado na Figura 8(b) que esta junção apresentou uma baixa quantidade de dados transferidos na rede, já que a base de dados Localidades possui pouco tamanho em disco (Tabela 1).

Métricas coletadas no testes mostraram que o uso da CPU no processamento desta junção não foi pequeno (em média 60%), pois a base de dados de Bioma da Caatinga apresenta geometrias muito complexas e, que requerem considerável uso da CPU mesmo em uma junção com uma base de dados com geometrias de Ponto (neste caso Localidades). Como o tráfego de dados na rede foi baixo, o tempo de processamento tornou-se dominante sobre o tráfego de dados na rede.

PA 0.9 e *Round Robin* apresentaram os melhores desempenhos entre todas as técnicas, pois conseguem distribuir o processamento da junção espacial entre as máquinas do *cluster*. PA 0.9 teve desempenho 3% pior que a técnica *Round Robin*. A técnica PA 0.1 apresentou o pior desempenho entre todas as técnicas, pois concentra o processamento em um pequeno conjunto de máquinas, subutilizando os recursos disponíveis no *cluster*. Portanto, para junções onde o processamento é dominante, um fator de balanceamento alto é melhor, pois balanceia os dados pelo *cluster* e, consequentemente, aumenta o paralelismo do processamento da junção espacial distribuída.

Os testes demonstraram que a técnica de distribuição de dados proposta neste trabalho, *Proximity Area*, apresentou um desempenho, no geral, melhor que *Round Robin*. Nos casos em que o trafego de dados na rede influenciaram de forma significativa no tempo de resposta, a diferença no tempo de resposta foi maior, pois a técnica *Proximity*



(a) Tempo de resposta (b) Quantidade de Kbytes tranferidos na rede

Figura 8. Junção espacial entre Bioma da Caatinga e Localidades

Area consegue co-localizar os dados, reduzindo assim o tráfego de dados na rede. A técnica *Round Robin* teve um bom desempenho quando houve pouco tráfego de dados na rede, apresentando um tempo de resposta próximo a PA 0.9. No geral, PA 0.9 apresentou o melhor desempenho entre todas as técnicas analisadas sendo, na média entre todos os testes, 16% melhor que *Round Robin*.

5. Trabalhos Correlatos

[Koudas et al. 1996] apresenta a proposta de uma arquitetura, onde o índice fica armazenado um uma máquina - *master* - e os dados são distribuídos pelas outras máquinas disponíveis - clientes. Esta abordagem gera um gargalo na máquina que armazena o índice e [Schnitzer and Leutenegger 1999] tenta resolver esse problema criando índices nos clientes para os dados armazenados localmente e deixando no *master* um índice para os clientes. Esta abordagem de [Schnitzer and Leutenegger 1999] também gera um gargalo no *master* para uma grande quantidade de consultas. Em [An et al. 1999] é proposta uma arquitetura semelhante a [Koudas et al. 1996] e, por isso, apresenta os mesmos problemas descritos anteriormente. Para resolver o problema de gargalos no sistema, [Mutenda and Kitsuregawa 1999] propôs uma arquitetura, onde o índice é replicado entre todas as máquinas do *cluster* e os dados são distribuídos segundo uma política circular (*Round-Robin*). Entretanto, a replicação do índice gera grande redundância de dados.

[Tan et al. 2000] analisa o impacto da cardinalidade de R e S no processamento da junção espacial distribuída. [Ramirez and de Souza 2001] avalia o impacto de aproximações mais complexas no processamento da junção espacial que, geralmente, utiliza o MBR como aproximação. [Kang and Choy 2002] propõe alguns modelos de custo para o processamento da junção espacial. [Karam and Petry 2005] também propõe vários modelos de custo detalhados para processamento da junção espacial distribuída utilizando *R-Trees*. Estas arquiteturas armazenam as duas bases de dados R e S envolvidas na junção espacial em apenas duas máquinas, subutilizando os recursos disponíveis em um *cluster* com vários computadores.

Em [Chung et al. 2005], cada base de dados é replicada em algumas máquinas do *cluster*. Cada consulta é enviada a um servidor central e espalhada pelas máquinas que contém replicas das bases de dados envolvidas na junção espacial. [Wei et al. 2008]

replica os índices de cada base de dados entre todas as máquinas do *cluster*. Os blocos são divididos em tamanhos iguais e distribuídos uniformemente pelo *cluster*. [Xie et al. 2008] propõe um *framework* de balanceamento de carga em duas fases. A primeira fase envolve a distribuição uniforme dos dados pelas máquinas do *cluster* e a segunda fase visa balancear a carga de processamento entre os computadores. [Zhang et al. 2009] apresenta uma arquitetura que utiliza o modelo *MapReduce* para processar operações espaciais. A distribuição de dados pelas máquinas é realizada pelo *HDFS (Hadoop's Distributed FileSystem)*, que não utiliza nenhuma informação espacial para realizar esta operação. [Zhong et al. 2012] também apresenta uma arquitetura utilizando o modelo *MapReduce*. Esta arquitetura divide os dados em vários blocos, onde cada bloco contém dados próximos espacialmente, e os blocos são distribuídos no *cluster* pelo HDFS.

Todos os trabalhos apresentados anteriormente possuem técnicas de distribuição de dados para bases de dados estáticas. Estas técnicas são inviáveis para bases de dados com grandes volumes de dados e com alta frequência de atualizações. Para mitigar este problema, [Zhou et al. 2011] propôs uma técnica híbrida de distribuição de dados: a base de dados é particionada utilizando uma estratégia de distribuição estática e atualizações são tratadas com uma estratégia de distribuição dinâmica. Esta estratégia visa manter objetos próximos espacialmente em máquinas diferentes e, como dito na Seção 4.1, gera grande tráfego de dados em junções espaciais distribuídas. Em [de Oliveira et al. 2011] é proposta uma estratégia de distribuição dinâmica de dados semelhante ao algoritmo *Round-Robin*. Entretanto, esta estratégia não tenta co-localizar os dados para diminuir o tráfego de dados na rede.

Nenhuma proposta encontrada na literatura possui uma solução para processamento da junção espacial com técnicas de distribuição de dados para bases de dados dinâmicas. Os trabalhos [Zhou et al. 2011] e [de Oliveira et al. 2011] apresentam técnicas de distribuição com bases de dados dinâmicas para consultas em apenas uma base de dados.

6. Conclusões

Conforme apresentado, numa plataforma de sistemas distribuídos para processamento de grande volume de dados geográfico, a técnica de distribuição de dados é um dos principais fatores que determina a eficiência dos serviços implementados sobre uma plataforma de *middleware* para geoprocessamento distribuído. Os trabalhos da literatura apresentam propostas para bases de dados estáticas. Para sistemas com bases dinâmicas em que os dados são atualizados ou novos dados são inseridos, a técnica *Proximity Area* proposta neste artigo é uma das precursoras que apresenta bons resultados na execução de operações de Junção Espacial Distribuída num *cluster* de computadores.

Para manter o *cluster* balanceado, foi criado um fator de balanceamento, onde quanto maior o fator de balanceamento mais balanceado fica o *cluster*. Os testes demonstraram que quando o processamento é dominante, um fator de balanceamento maior (PA 0.9) permite que os recursos do *cluster* sejam aproveitados e o tempo de resposta reduzido. Quando o tráfego de dados na rede se torna dominante, um fator de balanceamento menor (PA 0.1) reduz a quantidade de mensagens trocadas na rede e, consequentemente, o tempo de resposta da junção espacial. A técnica *Proximity Area* com fator de balanceamento 0.9 apresentou o melhor desempenho entre as técnicas testadas sendo, na média, 16% melhor

que *Round Robin*.

Como trabalho futuro, será investigada uma solução híbrida, que leva em conta dois fatores de balanceamento: dados e geometrias. Esta solução híbrida visa distribuir os dados de forma com que: i) uma máquina não fique com poucos dados, pois estes servidores têm menor chance de participar da operação de junção espacial, ficando assim subutilizados; ii) uma máquina não acomode muitos objetos com geometrias complexas, pois esses servidores com muitas geometrias grandes e complexas têm maior probabilidade de serem sobrecarregados com o processamento de algoritmos de operadores topológicos. O algoritmo *Proximity Area* irá redistribuir os objetos, em determinados intervalos de tempo, para que estes sejam alocados nos servidores mais próximos espacialmente segundo os critérios de balanceamento.

Referências

An, N., Kanth, R., Kothuri, V., and Ravada, S. (2003). Improving performance with bulk-inserts in Oracle R-trees. In *VLDB-Volume 29*, page 951. VLDB Endowment.

An, N., Lu, R., Qian, L., Sivasubramaniam, A., and Keefe, T. (1999). Storing spatial data on a network of workstations. *Cluster Computing*, 2(4):259–270.

Beckmann, N., Kriegel, H., Schneider, R., and Seeger, B. (1990). The R*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, 19(2):322–331.

Bentley, J. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):517.

Brinkhoff, T., Kriegel, H., Schneider, R., and Seeger, B. (1994). *Multi-step processing of spatial joins*, volume 23. ACM.

Brinkhoff, T., Kriegel, H., and Seeger, B. (1993). *Efficient processing of spatial joins using R-trees*, volume 22. ACM.

Chung, W., Park, S., and Bae, H. (2005). Efficient parallel spatial join processing method in a shared-nothing database cluster system. *Embedded Software and Systems*, pages 81–87.

Comer, D. (1979). Ubiquitous B-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137.

de Oliveira, T., Sacramento, V., Oliveira, S., Albuquerque, P., Cardoso, M., Bloco, I., and Campus, I. (2011). DSI-Rtree - Um Índice R-Tree Escalável Distribuído. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *ACM Sigmod Record*, 14(2):47–57.

Jacox, E. and Samet, H. (2007). Spatial join techniques. *ACM Transactions on Database Systems (TODS)*, 32(1):7.

Kamel, I. and Faloutsos, C. (1994). Hilbert R-tree: An Improved R-tree using Fractals. In *VLDB 20th*, page 509. Morgan Kaufmann Publishers Inc.

Kang, M. and Choy, Y. (2002). Deploying parallel spatial join algorithm for network environment. In *High Speed Networks and Multimedia Communications 5th IEEE International Conference on*, pages 177–181. IEEE.

Karam, O. and Petry, F. (2005). Optimizing distributed spatial joins using r-trees. In *Proceedings of the 43rd annual Southeast regional conference-Volume 1*, pages 222–226. ACM.

Koudas, N., Faloutsos, C., and Kamel, I. (1996). Declustering spatial databases on a multi-computer architecture. *Advances in Database Technology-EDBT'96*, pages 592–614.

Mutenda, L. and Kitsuregawa, M. (1999). Parallel r-tree spatial join for a shared-nothing architecture. In *Database Applications in Non-Traditional Environments, 1999.(DANTE'99) Proceedings. 1999 International Symposium on*, pages 423–430. IEEE.

Patel, J. and DeWitt, D. (1996). Partition based spatial-merge join. In *ACM SIGMOD Record*, volume 25, pages 259–270. ACM.

Patel, J. and DeWitt, D. (2000). Clone join and shadow join: two parallel spatial join algorithms. In *Proceedings of the 8th ACM international symposium on Advances in geographic information systems*, pages 54–61. ACM.

Ramirez, M. and de Souza, J. (2001). Distributed processing of spatial join. In *Proc. of the Anais do III Workshop Brasileiro de GeoInformáticaâ¯ GeoInfo*, volume 2001, pages 1–8.

Schnitzer, B. and Leutenegger, S. (1999). Master-client r-trees: A new parallel r-tree architecture. In *Scientific and Statistical Database Management, 1999. Eleventh International Conference on*, pages 68–77. IEEE.

Tan, K., Ooi, B., and Abel, D. (2000). Exploiting spatial indexes for semijoin-based join processing in distributed spatial databases. *Knowledge and Data Engineering, IEEE Transactions on*, 12(6):920–937.

Wei, H., Wei, Z., and Yin, Q. (2008). A new parallel spatial query algorithm for distributed spatial databases. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 3, pages 1570–1574. IEEE.

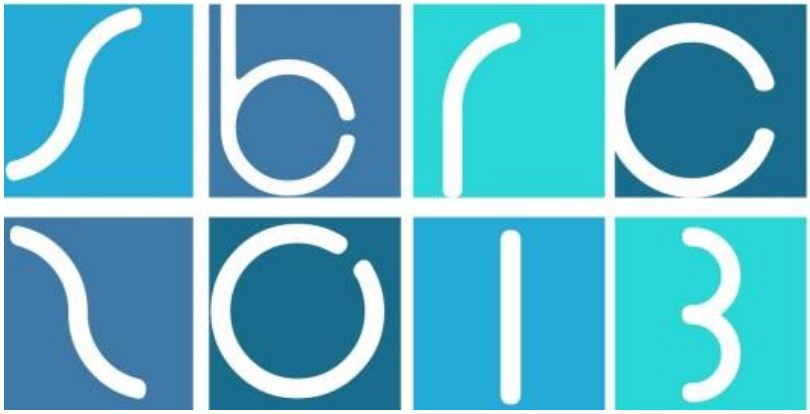
Xie, Z., Ye, Z., and Wu, L. (2008). A two-phase load-balancing framework of parallel gis operations. In *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, volume 2, pages II–1286. IEEE.

Zhang, S., Han, J., Liu, Z., Wang, K., and Feng, S. (2009). Spatial queries evaluation with mapreduce. In *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on*, pages 287–292. IEEE.

Zhong, Y., Han, J., Zhang, T., Li, Z., Fang, J., and Chen, G. (2012). Towards parallel spatial query processing for big spatial data. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 2085–2094. IEEE.

Zhou, X., Abel, D., and Truffet, D. (1997). Data partitioning for parallel spatial join processing. In *Advances in Spatial Databases*, pages 178–196. Springer.

Zhou, Y., Zhu, Q., and Zhang, Y. (2011). Spatial data dynamic balancing distribution method based on the minimum spatial proximity for parallel spatial database. *Journal of Software*, 6(7):1337–1344.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Artigos Completos do SBRC 2013

Sessão Técnica 24

Redes Veiculares 2

Um novo Algoritmo Geográfico Ciente de Partições na Rede para Disseminação de Dados em Redes Veiculares

Leandro A. Villas[1,4], Azzedine Boukerche[2], Antonio A. F. Loureiro[3] e Jo Ueyama[4]

[1]Instituto de Computação – UNICAMP
[2]PARADISE Research Laboratory - University of Ottawa
[3]Departamento de Ciência da Computação - UFMG
[4]Instituto de Ciências Matemáticas e de Computação - USP
leandro@ic.unicamp.br, boukerch@site.uottawa.ca
loureiro@dcc.ufmg.br e joueyama@icmc.usp.br

Abstract. *Vehicular Ad hoc Networks (VANETs) have attracted the attention of the research community recently as they have opened up a myriad of on the road applications and increased their potential by providing accident-free and intelligent transport systems. The envisaged applications, as well as some inherent VANET characteristics such as intermittent connectivity, a highly dynamic topology, and a different and dynamic network density, make data dissemination an essential service and a challenging task in these networks. Although they have been extensively studied in the literature, the existing solutions for data dissemination do not effectively address broadcast storm and network partition problems when considered together. To tackle these problems, we propose a novel geographical data dissemination of alert information and aware of network partition, named as DRIVING, which eliminates the broadcast storm and maximizes data dissemination capabilities across network partitions with short delays and low overhead. The simulation results show that the data dissemination performed by DRIVING provides better efficiency than other algorithms, outperforming them in different scenarios in all the evaluations carried out.*

Resumo. *As Redes Ad hoc Veiculares (VANETs) têm atraído a atenção da comunidade de pesquisa e tem permitido uma infinidade de aplicações em veículos nas rodovias e ampliou o seu potencial, provendo sistemas de transporte inteligentes e ao mesmo tempo livres de acidentes. As aplicações destes ambientes, bem como algumas características das VANETs, como conectividade intermitente, topologias altamente dinâmicas, e uma densidade de rede incomum e dinâmica, fazem com que a disseminação de dados transformem-se em um serviço essencial e desafiadora nas VANETs. Apesar de terem sido amplamente estudadas na literatura, as soluções existentes para a disseminação de dados não tratam efetivamente os problemas de tempestades de broadcast e partições da rede considerados em conjunto. Para solucionar estes problemas, propomos um novo algoritmo geográfico ciente de partições na rede para* ***D****isseminação de dados em* ***R****edes* ***V****eiculares* ***IN****teli****G****entes (DRIVING) . O algoritmo proposto, elimina a tempestade de broadcast e maximiza a capacidade de disseminação de dados entre as partições de rede com pequenos atrasos e baixo overhead. Os resultados da simulação mostram que a disseminação de dados realizada por DRIVING proporciona uma melhor eficiência do que outros algoritmos, superando-os em cenários diferentes em todas as avaliações realizadas.*

1. Introdução

Em redes Ad hoc Veiculares (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* (RSU), permitindo a criação de uma rede "espontânea" enquanto o veículo locomove-se nas rodovias [Luo and Hubaux 2006, Li and Wang 2007, Toor et al. 2008, Wang and Li 2009].

O uso de sistemas de comunicações móveis em veículos deve se tornar uma realidade nos próximos anos, já que a indústria, o mundo acadêmico e os governos do mundo inteiro estão dedicando consideráveis recursos para a implantação de redes veiculares no intuito de garantir uma sólida infra-estrutura e um sistema de transporte mais seguro. Este novo paradigma que permite que as informações sejam compartilhadas entre veículos, vai permitir uma ampla gama de aplicações de segurança, de assistência ao condutor, sensoriamento urbano, eficiência do tráfego, informação e entretenimento entre outras que serão incorporados em projetos de veículos modernos.

A disseminação dos dados é uma das tarefas mais desafiadoras e indispensáveis em redes veiculares. Ela é particularmente desafiadora devido a algumas características inerentes das VANETs, como por exemplo, a existência de uma topologia altamente dinâmica, frequentes desconexões de rede e uma densidade de rede com facetas diferentes, além de ser dinâmica. 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.

Para a criação de protocolos de disseminação de dados eficiente para VANETs, dois problemas chaves devem ser considerados:

1. **Tempestade de *Broadcasts:*** ocorre quando vários veículos que estão perto um do outro tentam transmitir ao mesmo tempo, causando um alto tráfego de dados, congestionamento na rede, colisões entre pacotes e um atraso adicional no controle de acesso ao meio (camada MAC).
2. **Partição de 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. Neste cenário, se o veículo não estiver ciente de que a rede está desconectada, quando recebe uma nova mensagem, ele simplesmente irá retransmitir em *broadcast* a mensagem e descartá-la na sequência. Uma vez que não há nenhum outro veículo dentro da área de interesse para receber esta mensagem, ela será simplesmente perdida para sempre. O problema da partição da rede é muito comum em uma rede VANET; isso 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 nas VANETs [Korkmaz et al. 2004, Durresi et al. 2005, Joshi et al. 2007, Chen et al. 2009, Tonguz et al. 2010, Bakhouya et al. 2011]. Entretanto, as soluções existentes foram projetadas para lidar com o problema da tempestade de *broadcasts* ou o problema da partição da rede e não os dois em conjunto. Argumentamos que estes dois problemas não devem

ser tratadas separadamente e salientamos que até o presente momento de nossa pesquisa, não encontramos trabalhos na área de disseminação de dados que endereçam estes dois problemas de maneira eficiente e em conjunto.

Para superar os desafios citados acima, propomos um novo algoritmo geográfico ciente de partições na rede para **D**isseminação de dados em **R**edes **V**eiculares **IN**teli**G**entes (*DRIVING*). Esta proposta de disseminação elimina a tempestade de *broadcast* e maximiza a capacidade de disseminação de dados entre partições de rede com pequenos atrasos e baixo *overhead*. A nossa proposta possui três vantagens: *primeiro*, ela realiza a disseminação de dados de maneira reativa, isto é, ela não requer a construção e manutenção da tabela de vizinhos para disseminar dados dentro da área de interesse. As VANETs possuem uma topologia altamente dinâmica devido à alta velocidade de mobilidade dos veículos. Por isso, o custo da construção e manutenção da tabela dos vizinhos normalmente é alto. Entretanto, a maior parte das soluções discutidas na literatura requer o uso das tabelas dos vizinhos [Durresi et al. 2005, Joshi et al. 2007, Chen et al. 2009, Tonguz et al. 2010]. *Segundo*, propomos o uso da zona de preferência e a utilizamos para disseminar dados de maneira eficiente dentro da área de interesse; e isso leva a um baixo *overhead*, baixos atrasos e uma alta cobertura. *Terceiro*, a nossa proposta supera o problema da partição da rede sem a necessidade de utilizar a zona de encaminhamento *zone of forwarding* (ZoF) como em [Joshi et al. 2007, Chen et al. 2009], pois o uso de uma área fixa para a ZoF traz alguns problemas. Se a área é maior do que o ideal, alguns veículos irrelevantes encaminharão mensagens, fazendo o uso desnecessário de recursos do canal. Por outro lado, se esta área é menor do que o ideal, a partição da rede poderá ocorrer. Consequentemente, o cálculo da área do ZoF é complexo e torna-se um aspecto chave para os protocolos propostos.

O restante deste artigo é organizado da seguinte forma. Na próxima seção, provemos uma visão geral das abordagens existentes para a disseminação de dados em VANETs. A nossa proposta de um protocolo de disseminação de dados eficiente e robusta é descrita na Seção 3, enquanto que uma avaliação detalhada da performance e dos resultados de simulação são apresentados na Seção 4. Finalmente, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O mecanismo mais simples de realizar a disseminação é através de *flooding* (inundações), o que significa que os dados são enviados a todos os veículos vizinhos, que por sua vez irá armazenar e transmitir os dados para os seus vizinhos também. Essa abordagem garante um bom desempenho para redes esparsas, mas logo encontra o problema de tempestade de broadcast quando a densidade da rede aumenta. Em vista disso, as propostas atuais para a disseminação de dados dá o foco principalmente na forma como os pacotes serão encaminhados; várias propostas podem ser encontradas para isso – as baseados em posição, baseados estatisticamente, baseados em distância, baseados na topologia local, baseados nos temporizadores e as baseados em mapas.

O trabalho *Adaptive approach for Information Dissemination* (AID) [Bakhouya et al. 2011] é uma abordagem descentralizada e adaptável para a disseminação de informação em VANETs. Nesta 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 da tempestade de *broadcast*. No entanto, o protocolo AID não trata o problema de partição da rede.

O trabalho *Distance Based Relay Selection* (DBRS) [Sun et al. 2000, Kim et al. 2007] é uma estratégia simples e eficiente para disseminar informações 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 que o mesmo já está programado para ser transmitido, cancelam a sua transmissão para evitar a tempestade de *broadcast*. Esta abordagem é eficaz no sentido de que ela pode evitar a tempestade de *broadcast*, mas ela pode trazer dois problemas: (i) o atraso pode ser elevado, pois não há garantia de 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.

A nossa proposta é baseado também na distância, porém utilizamos a região da zona de preferência para priorizar veículos situados no interior dessa região, além de permitir disseminar os dados, com um baixo atraso. Ademais, a estratégia de cancelamento da nossa proposta é utilizada somente em veículos dentro do mesmo quadrante da área de transmissão, aumentando assim a sua cobertura.

3. Algoritmo Proposto

O objetivo principal do algoritmo proposto DRIVING é realizar a disseminação de dados dentro de uma área de interesse (AOI), eliminando assim a tempestade de *broadcast* e maximizando a capacidade da disseminação de dados entre partições de rede, além de prover tudo isso com uma baixa sobrecarga, pequenos atrasos e uma alta cobertura. Além do que foi mencionado, os veículos não necessitam de uma tabela de vizinhos como a maioria das propostas encontradas na literatura. O DRIVING usa uma zona de preferência para eliminar o problema da tempestade de broadcast e maximizar a disseminação de dados.

Definition 3.1 (Zona de Preferência) *Definimos uma zona de preferência como uma área onde os veículos dentro dela são considerados os mais adequados para continuar realizando a disseminação de dados. Isto significa que, entre todos os veículos que receberam os dados, a transmissão de um único veículo dentro da zona de preferência é suficiente para realizar a disseminação de dados eficientemente. Os veículos localizados dentro das zonas de preferência são os mais indicados para "espalhar"a informação mais longe, assim como para alcançar um maior número de vizinhos que não puderam ser alcançados pelo último transmissor.*

A Figura 1 ilustra a zona de preferência proposta. Por uma questão de simplificação, a forma da área de comunicação é considerada como sendo um círculo, onde o veículo transmissor se encontra no centro do círculo. Mas convém salientar que qualquer formato (além do círculo) funciona para a solução proposta. A área de comunicação é decomposta em quatro quadrantes, e em cada quadrante, uma sub-área do quadrante é definida como uma zona de preferência.

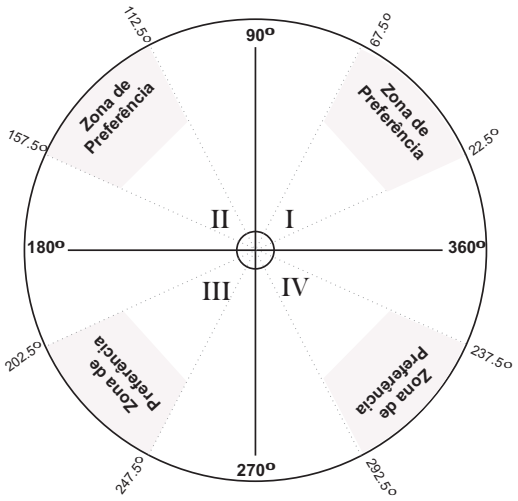


Figura 1. Zona de Preferência.

No nosso algoritmo DRIVING, a disseminação de dados é realizada de duas maneiras distintas. No primeiro caso, não existe uma partição de rede dentro da área de interesse - como mostrado na Seção 3.1. No segundo caso, a área de interesse é dividida - como mostrado na Seção 3.2.

O algoritmo DRIVING elimina o problema da tempestade de *broadcast*, porque ele requer menos transmissões para realizar a disseminação de dados. A razão para isto é que os veículos dentro da zona de preferência transmitem os dados 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 véiculo transmissor irá transmitir primeiro e abortar as transmissões desnecessárias dos outros veículos que estão dentro da zona de preferência. No caso de não haver veículos dentro da zona de preferência, o veículo mais distante de cada quadrante irá retransmitir a mensagem. Através das zonas de preferência, podemos impedir a transmissão da mesma mensagem dos veículos que estão próximos uns dos outros e pertençam a quadrantes diferentes. Assim evitamos que as mensagens alcancem áreas similares, o que seria uma redundância desnecessária. Além disso, DRIVING utiliza os veículos fora da área de interesse (sem há necessidade de definir uma zona de encaminhamento como nas soluções apresentadas na literatura) para realizar a disseminação de dados para os veículos que estão em partições de rede distintas e que estão dentro da mesma área de interesse.

3.1. Disseminação de Dados

No DRIVING, quando ocorre um evento, o veículo que detecta o evento começa a disseminar os dados sobre o evento dentro da área de interesse. Durante este processo, os veículos no interior da zona de preferência (ver Figura 1) são preferencialmente escolhidos para continuar o processo de disseminação de dados e eliminar a tempestade de *broadcast*. Este processo é descrito na Figura 2 (retângulo pontilhado à esquerda). Subsequentemente, o veículo fonte cria uma mensagem de informações sobre o evento e transmite a mesma para os seus vizinhos. Ao receber tal mensagem, cada veículo verifica se está dentro da área de interesse (AOI), conforme exibido no ponto E da Figura 2. Se

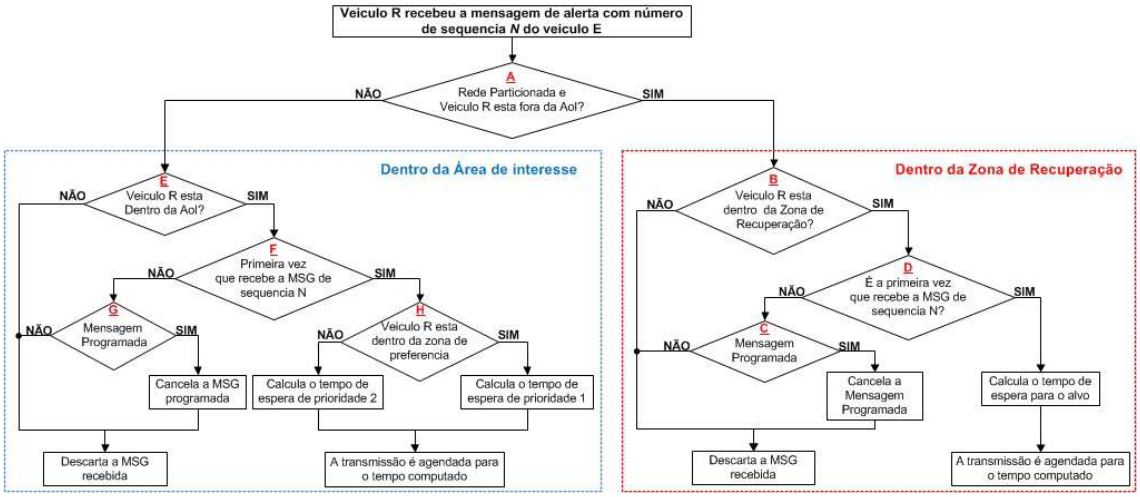


Figura 2. DRIVING - Fluxograma da disseminação de dados.

esta condição não for cumprida, o veículo descarta a mensagem recebida. Caso contrário, a condição no ponto F da Figura 2 é verificada e se satisfeita, o veículo calcula o tempo de espera e escalona a transmissão que é baseada na zona de preferência e na sua posição. O Algoritmo 1 provê mais detalhes a respeito da zona de preferência e a maneira como o tempo de espera é calculado.

3.2. Disseminação de Dados entre as Partições de Rede

No DRIVING, quando uma partição de rede é detectada por um veículo fonte, alguns veículos localizados fora da área de interesse são utilizados para disseminar os dados sobre o evento dentro da área de interesse. Estes veículos localizados fora da área de interesse formam uma zona de recuperação dinâmica, onde não há a necessidade de definir uma área fixa como nas soluções apresentadas na literatura. O principal objetivo da utilização da zona de recuperação é a realização da disseminação de dados para os veículos que estão em partições de rede distintas, mas que estão dentro da mesma área de interesse. Além disso, superar as limitações das soluções que utilizam uma zona de encaminhamento de área fixa, conforme já foi discutido na Seção 1.

Este processo é exibido na Figura 2 (retângulo pontilhado à direita). Subsequentemente, o veículo fonte cria uma mensagem de informações sobre o evento e transmite a mesma para os seus vizinhos. Ao receber tal mensagem, cada veículo verifica se ele encontra-se fora da área de interesse, como mostrado no ponto A da Figura 2. Se este não for o caso, o veículo executa o processo apresentado na Seção 3.1. Caso contrário, se o veículo está dentro da zona de recuperação (ponto B da Figura 2), a condição no ponto D da Figura 2 é verificada e caso ela seja verdadeira, o veículo calcula o tempo de espera e escalona a transmissão baseada no alvo e na sua posição. Onde o alvo é o extremo da área de interesse. O Algoritmo 2 apresenta mais detalhes sobre o cálculo do tempo de espera.

4. Avaliação de Desempenho

Nessa seção, nós avaliamos o desempenho do algoritmo proposto DRIVING. Nós também comparamos o desempenho com três outros conhecidos algoritmos para disseminação de dados em redes veiculares: Flooding, AID and DBRS.

Algoritmo 1: Verifica se o veiculo receptor está dentro da zona de preferência e calcula o tempo de espera para agendar a transmissão.

Entrada: (x_t, y_t) and (x_r, y_r) `// Coordenadas dos Veiculos Transmissores e Receptores.`
Saida: Atraso `// Tempo de espera computado para agendar a transmissão.`
`// A função atan2 com dois argumentos é a variação da função arco-tangente.`
$angulo = atan2(y_s - y_r, x_s - x_r)$;
$distancia = \sqrt{(x_s - x_r)^2 + (y_s - y_r)^2}$;
$Atraso = 0.01 \times (\frac{distancia}{raio_{comunicacao}})$;
se $(|angulo| \leq \frac{\Pi}{2})$ **então**
    se $(\ (|angulo| > \frac{\Pi}{8})$ *and* $(|angulo| < \frac{\Pi}{8} + \frac{\Pi}{4})\ )$ **então**
        `// Calcula o tempo de espera com prioridade` 1
        $Atraso = Atraso + Aleatorio(0, 0.01)$
    fim se
    senão
        `// Calcula o tempo de espera com prioridade` 2
        $Atraso = Atraso + Aleatorio(0.02, 0.05)$
    fim se
fim se
senão
    se $(\ (|angulu| > \frac{\Pi}{2} + \frac{\Pi}{8})$ *and* $(|angulo| < \frac{\Pi}{2} + \frac{\Pi}{8} + \frac{\Pi}{4})\ )$ **então**
        `// Calcula o tempo de espera com prioridade` 1
        $Atraso = Atraso + Aleatorio(0, 0.01)$
    fim se
    senão
        `// Calcula o tempo de espera com prioridade` 2
        $Atraso = Atraso + Aleatorio(0.02, 0.05)$
    fim se
fim se

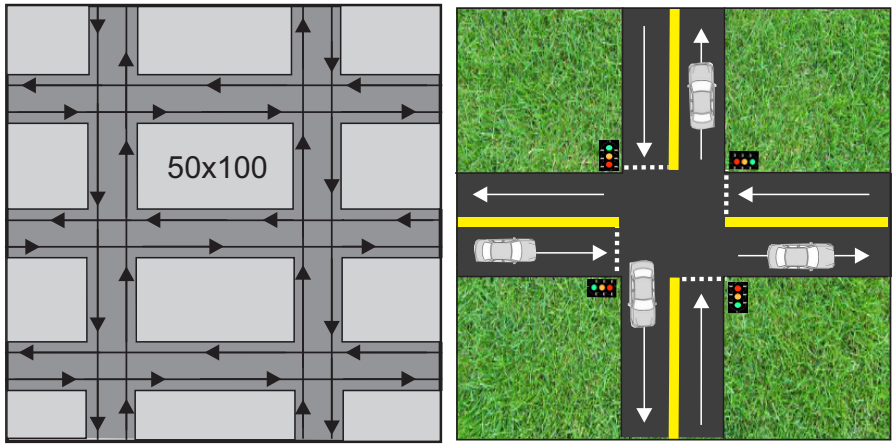
Algoritmo 2: Computa o tempo de espera para agendar a transmissão.

Entrada: $geometriaParticionamento$ e $fontePosicao$ `// Coordenadas da área de interesse particionada e veiculo fonte.`
Saida: Atraso `// Tempo de espera computado para agendar a transmissão.`
`// Alvo é o centro do lado oposto da área de interesse particionada.`
se $(fontePosicao_x > geometriaParticionamento_{x1})$ **então**
    $Alvo_x = geometriaParticionamento_{x1} - AoI/raioComunicacao$;
fim se
senão
    $Alvo_x = geometriaParticionamento_{x2} + AoI/raioComunicacao$;
fim se
$Alvo_y = (geometriaParticionamento_{y2} - geometriaParticionamento_{y1})/2$;
$distanciaAlvo = \sqrt{(atualPos_x - Alvo_x)^2 + (atualPos_y - Alvo_y)^2}$;
$Atraso = 0.01 \times (\frac{distanciaAlvo}{Alvo_y})$;

4.1. Metodologia

A avaliação é realizada através de simulações utilizando o *OMNeT++ Network Simulation Framework v.4.1* [OMNET++], *Inetmanet framework*, and *SUMO-Simulation of Urban MObility* [SUMO]. 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 3(a).

Nos cenários experimentais, o tráfego de veículos é gerado aleatoriamente. A velocidade dos veículos podem variar de 30 até 50 quilômetro por hora, o número de veículos é fixado para $800, 1000, 1200, 1400,$ e 1600 veículos e a área de interesse é fixada para $500 \times 500, 550 \times 550, 600 \times 600$ e $650 \times 650\, m^2$. Este cenário, é gerado aleatoriamente e cada um contém 60 estradas, 800 interseções e 120 pontos de cruzamento na



(a) Cenário de mobilidade (b) Estrutura de cada interseção

Figura 3. Cenário de Avaliação

fronteira. Como mostrado na Figura 3(a), 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 indicadas pelas setas. Em um cruzamento na fronteira, os veículos escolhem virar ou manter se movendo na mesma direção com igual probabilidade, 0.5. Em um cruzamento de uma rua vertical e uma horizontal (veja Figura 3(b)), cada véiculo escolhe virar à direita ou esquerda com uma probabilidade de 0.25 e manter se movendo na mesma direção com uma probabilidade de 0.5.

O tempo de simulação utilizado é de 100 s, que é suficientemente longo para avaliar os protocolos de dissmeinação através das variações da velocidade e da densidade dos veículos. Durante a simulação, um veículo arbitrário é escolhido para causar um acidente, e depois de bater, os veículos envolvidos divulgam as informações sobre o acidente para todos os veículos dentro da área de interesse. Cada veículo usa um protocolo MAC IEEE 802.11, operando a 2 Mbps, para enviar e receber mensagens. Nós usamos um modelo de propagação de rádio *two-ray ground* e o raio de transmissão é 200 m.

Cada simulação foi replicada 33 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 parâmetros de simulação são mostrados na Tabela 1.

Tabela 1. Parâmetros de Simulação

Parâmetros	Valores
Duração da disseminação (segundos)	60
Número de veiculos	800, 1000, 1200, 1400, 1600
Velocidade do veículo (Quilômetros por hora)	30, 35, 40, 45, 50
Área de Interesse (m^2)	500x500, 550x550, 600x600, 650x650
Taxa de disseminação (por segundo)	1
Raio de comunicação (metros)	200
Tempo de simulação (segundos)	100

As seguintes métricas foram utilizadas na avaliação de desempenho:

- **Custo**: quantidade de transmissões realizadas durante a disseminação dos dados;
- **Colisões**: número de colisões de pacotes durante a disseminação dos dados;

- **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;
- **Atraso**: tempo médio para disseminar os dados dentro da área de interesse.

Para as métricas avaliadas, nós variamos o número e velocidade dos veículos e o tamanho da área de interesse (apresentados na Tabela 1) para analisar o desempenho do algoritmo proposto. Nós avaliamos cenários sem partições de rede dentro da área de interesse (Seção 4.2) e cenários onde ocorrem partições de rede dentro da área de interesse (Seção 4.3).

4.2. Rede não Particionada dentro da Área de Interesse

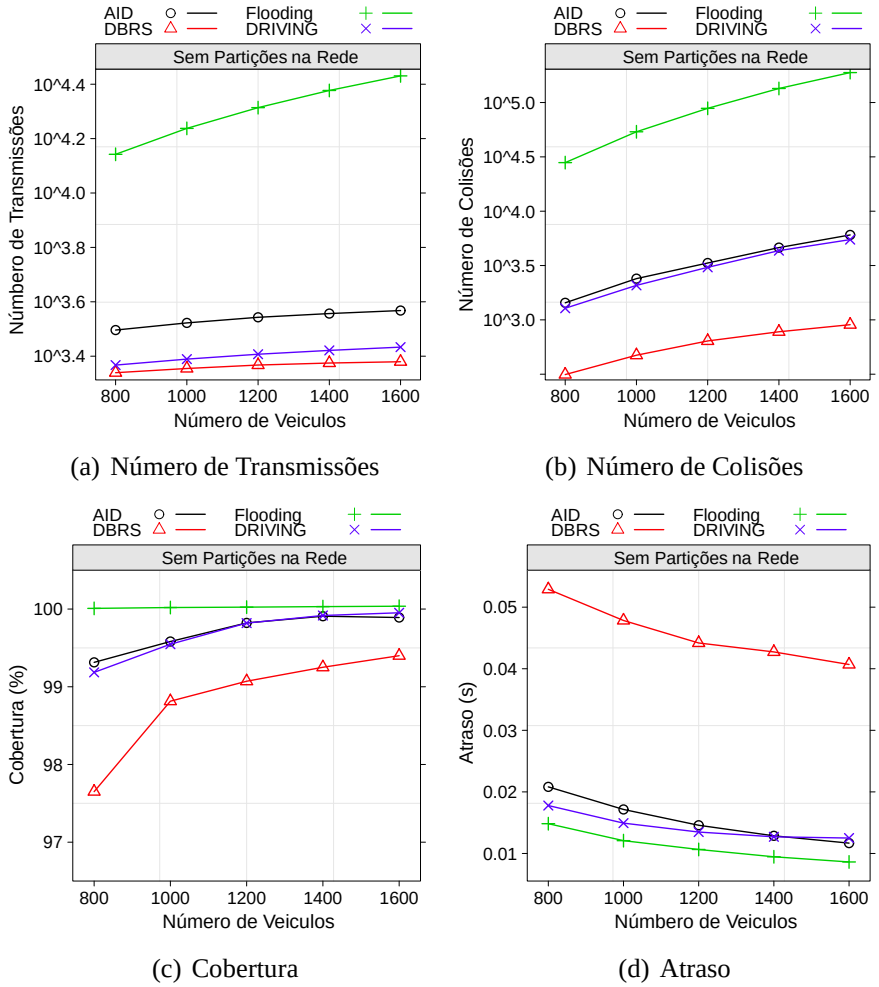
O objetivo principal de avaliar cenários que não têm partições de rede dentro da área de interesse é mostrar que a solução proposta elimina a tempestade de *broadcast* e tem baixa sobrecarga para realizar a disseminação dos dados, com pequenos atrasos.

4.2.1. Custo

Figura 4(a) mostra que o *DRIVING* é mais eficiente do que os algoritmos *Flooding, AID* e *DBRS*, uma vez que necessita menos transmissões para realizar disseminação de dados dentro da área de interesse. Na média, o número de transmissões do *DRIVING* é de cerca de uma ordem de grandeza menor do que *Flooding*. Este resultado é esperado uma vez que nenhum mecanismo para reduzir a tempestade *broadcast* é implementado juntamente com o *Flooding*. *DRIVING, AID* e *DBRS* apresentam a mesma ordem de grandeza do número de transmissões, mas em média, *DRIVING* requer mais transmissões do que o *DBRS* e menos transmissões do que o *AID*. Embora *DBRS* requer menos transmissões para realizar a disseminação dos dados, ele deixa de cobrir toda área de interesse (veja Figura 4(c)), e causa alto atraso (veja Figura 4(d)), quando comparado com *DRIVING*.

4.2.2. Colisões

Soluções de disseminação de dados baseadas em *flooding* tem uma alta taxa de colisões e sofrem do problema de tempestade de *broadcast*. Figura 4(b) mostra que o *DRIVING* elimina o problema de tempestade de *broadcast*, uma vez que necessita menos transmissões para realizar a disseminação de dados (veja Figura 4(a)). Podemos observar que *DRIVING* transmite um número semelhante de mensagens para cobrir a totalidade da área de interesse como *AID*. Novamente, *Flooding* não é eficiente porque não tenta resolver o problema de tempestade de *broadcast*. *DBRS* é a abordagem que apresentou o menor número de colisões de pacotes, mas vale a pena mencionar novamente que não consegue cobrir toda a área de interesse e apresenta alto atraso. O número de colisões de pacotes é outra métrica para avaliar o desempenho dos algoritmos para lidar com o problema de tempestade de *broadcast*.



(a) Número de Transmissões (b) Número de Colisões

(c) Cobertura (d) Atraso

Figura 4. Resultados de Simulações para Redes não Particionadas

4.2.3. Cobertura

Flooding é usado como linha de base para avaliar a cobertura, uma vez que se espera que o *Flooding* realiza a disseminação dos dados para todos os veículos dentro da área de interesse, que estão ligados ao mesmo componente que o veículo que inicia o processo de disseminação de dados. Assim, pode ser visto na Figura 4(c) que usando *Flooding*, nós poderíamos alcançar 100% de cobertura para todos os cenários que nós avaliamos. *DRIVING* e *AID* produziu resultados muito semelhantes e foram muito próximos de 100% de cobertura. Em cenários esparsos, ambos *DRIVING* e *AID* alcançaram cobertura ligeiramente inferior de 100% (aproximadamente 99%). A razão para isso foi que o esquema de supressão de tempestade de *broadcast* (o processo de cancelamento) às vezes, poda um ramo de transmissão, que não pode ser alcançado por um outro caminho. Entretanto, em geral, ambos *DRIVING* e *AID* alcançam uma alta cobertura que é muito próximo ao conseguido pelo *Flooding*, mas com a vantagem de suprimir a tempestade de *broadcast*. *DBRS* não abrange a área de interesse porque seu esquema de supressão é muito agressivo e poda muitos ramos de transmissões que não podem ser alcançados por outros caminhos.

4.2.4. Atraso

O tempo decorrido até que os veículos que encontram-se dentro da área de interesse recebam o dado disseminado é um valor importante para avaliar o desempenho dos algoritmos de disseminação de dados porque muitas aplicações, como o sistema de aviso de colisão, por exemplo, tem requisitos de tempo real para funcionar corretamente. Figura 4(d) mostra o comportamento dos quatro algoritmos, com o *Flooding* sendo a linha de base. Podemos observar que *DRIVING* comporta-se de forma semelhante quando comparado com a AID, ainda que utilize o atraso proporcional ao inverso da distância do ultimo veículo transmisssor. Como pode ser observado, o esquema de zona de preferência é capaz de diminuir o atraso para quase o mesmo nível do *AID*. *DBRS* é o algoritmo que causa o maior atraso. isto ocorre porque porque o DBRS impõe um atraso adicional para cada salto, o atraso com base na distância. Apesar do DRIVING também impor um atraso baseado na distância, a zona de preferência alivia este efeito no atraso total.

4.3. Rede Particionada dentro da Área de Interesse

O objetivo principal de avaliar cenários que têm partições de rede dentro da área de interesse é mostrar que a solução proposta maximiza a capacidade de disseminação de dados entre partições de rede.

4.3.1. Custo e Colisões

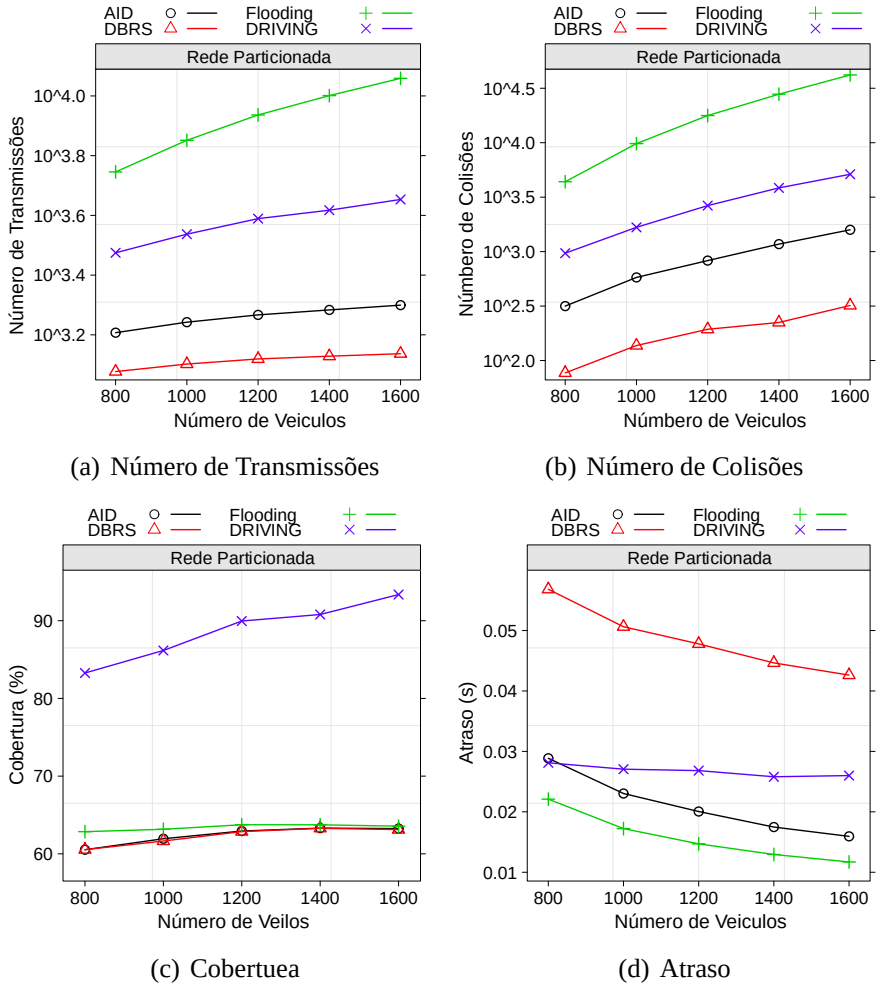
Figuras 5(a) e 5(b) mostram que o *DRIVING* aumenta ligeiramente o número de transmissões e colisões quando comparado com os cenários sem partições de rede (veja Figura 4(a) e 4(b)). Isto ocorre porque *DRIVING* é capaz de contonar partições de rede utilizando as zonas de recuperação e, consequentemente, maximiza a disseminação dos dados. Nos outros algoritmos, o número de transmissões e colisões são mais baixos porque eles não realizam disseminação dos dados em toda a área de interesse (ver Figura 5(c)).

4.3.2. Cobertura

Figuras 5(c) mostra que o *DRIVING* maximiza a capacidade de disseminação de dados entre partições de rede. Em contraste, *Flooding, AID* e *DBRS* não pode executar de forma eficiente a disseminação de dados sobre redes particionadas. Em média, *DRIVING* alcança uma cobertura de 89% dos veículos enquanto os outros algoritmos alcançam uma cobertura de 61%. Isto ocorre porque o *DRIVING* é capaz de contonar partições de rede utilizando as zonas de recuperação e, consequentemente, maximiza a disseminação dos dados.

4.3.3. Atraso

O tempo decorrido até que os veículos dentro da área de interesse recebam o dado disseminado, é uma métrica importante para avaliar o desempenho dos algoritmos de disseminação de dados. Figura 4(d) mostra que a zona de preferência proposta é eficiente para diminuir o atraso. No caso de cenários que têm partições de rede dentro da área de



(a) Número de Transmissões (b) Número de Colisões

(c) Cobertuea (d) Atraso

Figura 5. Resultados de Simulações para Redes Particionadas

interesse (Figura 5(d)), o atraso é um pouco maior, mas garante uma melhor cobertura (o que e muito importante para aplicações críticas) em entregar os dados divulgados dentro da área de interesse.

5. Conclusões

Nós apresentamos o DRIVING, um novo algoritmo geográfico de disseminação de dados em redes veiculares inteligentes. Criamos *DRIVING* para atender os problemas difíceis de tempestade de *broadcast* e partições de rede para realizar disseminação de dados em VANETs. *DRIVING* usa o conceito de zona de preferência para selecionar o veículo retransmissor que maximiza a cobertura e suprime o efeito de tempestade de *broadcast* . Além disso, *DRIVING* usa zonas de recuperação dinâmicas para maximizar a disseminação de dados em redes particionadas. *DRIVING* foi comparado com os algoritmos *Flooding*, *AID* e *DBRS*. Os resultados mostram que o *DRIVING* realiza supressão eficiente da tempestade de *broadcast*, reduz o atraso e maximiza a cobertura durante a disseminação de dados. A zona de preferência permite o *DRIVING* selecionar o veículo retransmissor de forma mais eficiente alcançando resultados melhores do que os algoritmos comparados. Como trabalhos futuros pretendemos avaliar a performance do DRIVING sob cenários de

rodovias, caso o DRIVING não apresentar um bom desempenho, pretendemos investigar as modificações necessárias para tornar *DRIVING* eficiente para cenários de rodovias.

6. Agradecimentos

Os autores gostariam de expressar a sua gratidão pelo apoio concedido pelo CNPq, FAPESP para o INCT-SEC (processos 573963/2008-9 e 2012/12061-1) e FAPEMIG.

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*, In Press, Uncorrected Proof:–.

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.

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. (2007). An effective data dissemination in vehicular ad-hoc network. volume 5200 of *Lecture Notes in Computer Science*, pages 295–304, Berlin/Heidelberg. Springer.

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.

Luo, J. and Hubaux, J.-P. (2006). A survey of research in inter-vehicle communications. *Embedded Security in Cars*, pages 111–122.

OMNET++. Omnet++ network simulation framework.

SUMO. Sumo - simulation of urban mobility.

Sun, M.-T., Feng, W.-C., Lai, T.-H., Yamada, K., Okada, H., and Fujimura, K. (2000). Gps-based message broadcast for adaptive inter-vehicle communications. In *Vehicular Technology Conference, 2000. IEEE VTS-Fall VTC 2000. 52nd*, volume 6, pages 2685 –2692 vol.6.

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.

Wang, Y. and Li, F. (2009). Vehicular ad hoc networks. In *Guide to Wireless Ad Hoc Networks*, Computer Communications and Networks, pages 503–525.

Roteamento Baseado na Trajetória para Redes Veiculares Desconectadas com Múltiplos Gateways

Vitor Borges C. da Silva, Fábio Oliveira B. da Silva, Miguel Elias M. Campista e Luís Henrique M. K. Costa

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - DEL/POLI

{borges,fabio,miguel,luish}@gta.ufrj.br

Resumo. *Este trabalho propõe uma arquitetura para cenários formados por redes sem-fio veiculares desconectadas e pontos de acesso à rede cabeada distribuídos (drive-thru Internet). A arquitetura proposta lida com a mobilidade dos usuários com conexão intermitente utilizando um protocolo de encaminhamento de mensagens tolerante a atrasos e desconexões. Para evitar replicações de mensagens na rede cabeada, é proposto um protocolo de roteamento que combina roteamento estático e epidêmico, assim como um mecanismo baseado na trajetória dos veículos. Experimentos mostram que a proposta reduz o tráfego na rede híbrida e é transparente para as aplicações dos usuários em relação à mudança de ponto de acesso à rede.*

Abstract. *This paper proposes an architecture for scenarios composed of disconnected vehicular wireless networks and distributed access points connected to the wired infrastructure (drive-thru Internet). The proposed architecture deals with users' mobility with intermittent connectivity using a message forwarding protocol which tolerates delays and disconnections. To avoid message replication in the wired network, we propose a routing protocol that combines static with epidemic routing, as well as a trajectory-based mechanism. Experimental results show that the proposal reduces the traffic in the hybrid network and is transparent for users' applications concerning changes to the network access point.*

1. Introdução

Atualmente, os usuários desejam ficar conectados à rede em todo lugar e a todo o momento. O aumento mundial das vendas de dispositivos móveis revela tal fenômeno [Weissberger, 2012]. A tecnologia mais usada para usuários móveis acessarem à Internet é a telefonia celular 3G/4G. No entanto, a alta utilização sobrecarrega a rede celular, fazendo que as operadoras adotem técnicas para descarga de dados. Nessa direção, o uso de redes oportunistas torna-se uma alternativa promissora [Han et al., 2010].

Entre as redes oportunistas, o IEEE 802.11 é uma tecnologia de baixo custo com ampla aceitação. As redes IEEE 802.11 podem ser usadas em redes veiculares, nas quais os usuários móveis obtém conectividade com a Internet através de comunicações por múltiplos pontos de interconexão (*gateways*). Tal cenário é conhecido também por *drive-thru* Internet [Ott e Kutscher, 2005]. Os usuários dentro de veículos se conectam às unidades de acostamento (UAs), que oferecem serviços de rede através de *gateways* para a infraestrutura cabeada. Embora muito benéfica, a adoção das redes veiculares em larga

escala ainda é um desafio. A arquitetura de rede é composta por ilhas de conectividade, delimitadas pelo alcance do rádio de cada UA. Com isso, a conectividade dos veículos é intermitente e de curta duração. Assim, os usuários podem experimentar interrupções de serviço, e comunicações fim-a-fim podem não ser estabelecidas, compondo um cenário de redes tolerantes a atrasos e desconexões (DTN) [Scott e Burleigh, 2007].

As DTNs usam comutação de mensagens com transferência de custódia para contornar a conectividade intermitente. Ao invés de encaminhar as mensagens recebidas, os nós intermediários as armazenam e as replicam ao encontrar um contato oportunista. Na abordagem trivial, os nós replicam as mensagens epidemicamente para todos os nós encontrados até alcançar o destino ou expirar um temporizador. Essa abordagem é válida em cenários onde nenhuma informação está disponível, mas é ineficiente sob o ponto de vista da sobrecarga de mensagens [Spyropoulos et al., 2008a, Spyropoulos et al., 2008b]. Soluções menos triviais reduzem a sobrecarga de mensagens, usando de estratégias estatisticas [Burgess et al., 2006] ou geográficas [Cheng et al., 2010]. Em cenários *drive-thru*, se os usuários móveis não permanecerem conectados a uma UA por tempo suficiente, o problema de conectividade ocorrerá devido a mobilidade dos nós. Deve-se considerar a infraestrutura híbrida para prover um serviço de qualidade aos usuários móveis.

As principais contribuições desse trabalho são: (i) uma arquitetura híbrida para lidar com a mobilidade dos usuários em cenários *drive-thru*, incluindo uma versão adaptada das DTNs na infraestrutura cabeada; (ii) um protocolo de roteamento DTN adaptado que combina o uso dos roteamentos estático e epidêmico nas redes cabeada e sem-fio, respectivamente; e (iii) um mecanismo baseado na trajetória para reduzir o tráfego de dados na infraestrutura cabeada e na rede sem-fio. Os resultados mostram que é possível prover serviços transparentes aos usuários móveis usando encapsulamento DTN na rede cabeada. A arquitetura híbrida reduz o tráfego nas redes cabeada e sem-fio, na rede cabeada através do roteamento estático e na sem-fio através da diminuição do tráfego de controle.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta o cenário abordado. A Seção 3 apresenta as contribuições deste trabalho, enquanto a Seção 4 fornece os detalhes da arquitetura proposta. A Seção 5 apresenta o ambiente de testes. Os experimentos conduzidos e os resultados obtidos são descritos na Seção 6. Finalmente, a Seção 7 conclui este trabalho e apresenta as direções futuras.

2. Problema Abordado

As redes veiculares baseadas no padrão IEEE 802.11 com conexão intermitente utilizam frequentemente as DTNs como solução para lidar com a ausência de caminhos fim-a-fim. Nas DTNs, os nós mantêm a custódia das mensagens e as transferem em *bundles* ao encontrar outros nós. Independentemente se há alguma seletividade na operação, o procedimento não é apropriado para suportar requisitos de interatividade. Ao adicionar a infraestrutura representada pelos múltiplos pontos de interconexão (*gateways*), os veículos podem ter acesso aos serviços de rede se o cenário em questão for explorado de maneira que os usuários não percam a conectividade. Neste trabalho, os termos mensagens e *bundles* são usados de forma intercalada.

2.1. Cenário híbrido

Os protocolos DTN não foram projetados originalmente para redes híbridas, onde as redes com e sem-fio estão interconectadas. Portanto, em cenários *drive-thru*, os

usuários executam os procedimentos de descoberta e de associação à infraestrutura cabeada ao mudar de ilha de conectividade. Caso uma conexão esteja ativa, ela é interrompida assim que o usuário móvel mudar de rede. Esse problema é uma consequência do modelo da Internet, que não levou em conta a mobilidade dos nós em seu projeto original [Saucez et al., 2009].

As redes tolerantes a atrasos e desconexões podem prover serviços de rede para usuários sem-fio móveis, desde que o serviço não exija conexão fim-a-fim. Consequentemente, a execução dos protocolos das redes tolerantes a atrasos e desconexões na rede cabeada pode ser considerada uma solução para lidar com a mobilidade. No entanto, a desvantagem dessa solução é o uso de protocolos do tipo epidêmico na rede cabeada, que leva a replicações desnecessárias de mensagens. Nesse cenário, se um usuário móvel enviar uma requisição à Internet, essa seria replicada por todos os nós a partir da UA mais próxima. Logo, uma alternativa que considere a mobilidade dos usuários bem como a combinação de redes sem-fio sem garantias de conectividade e utilização eficiente da infraestrutura cabeada ainda é uma questão em aberto.

3. Drive-thru Internet em Redes Híbridas

Este artigo propõe um protocolo de roteamento para redes híbridas que combinam redes sem-fio sem garantias de conectividade com redes infraestruturadas com múltiplos *gateways*. O objetivo é oferecer suporte à mobilidade dos usuários nas mudanças de ponto de acesso e períodos de desconexão. Para tal, estende-se o uso das redes DTN, inicialmente usadas na rede sem-fio, para a infraestrutura cabeada. A arquitetura resultante emprega roteamento epidêmico na parte sem-fio e roteamento estático com informações da trajetória dos usuários móveis na parte cabeada. A combinação de roteamento estático e informações sobre trajetória evita a replicação de mensagens (*bundles*) na rede cabeada.

3.1. Arquitetura proposta

A Figura 1 descreve a arquitetura proposta onde usuários móveis dentro de veículos usam conexões oportunistas para troca de dados através de redes IEEE 802.11. A arquitetura é composta de quatro entidades: o usuário móvel no interior do veículo, o veículo com o equipamento sem-fio, a Unidade de Acostamento (UA) e a Central. O usuário e o veículo estão na rede sem-fio, enquanto a Central está na parte cabeada. A UA é responsável por interconectar a rede sem-fio com a infraestrutura cabeada.

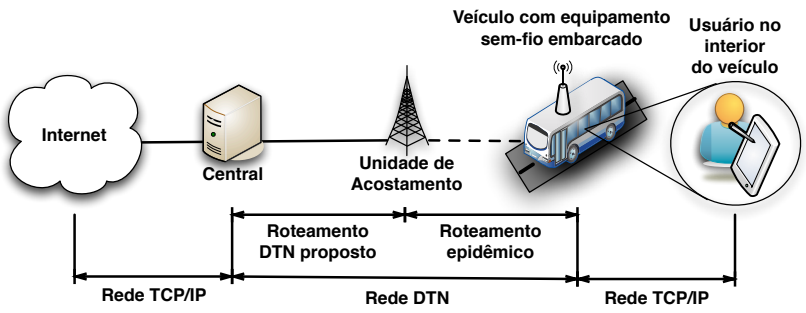


Figura 1. A arquitetura proposta: usuário móvel dentro do veículo, veículo com equipamento sem-fio embarcado, Unidade de Acostamento (UA) e Central.

Assume-se que os usuários executam aplicações com algum nível de interatividade, mas não de tempo real. As aplicações não precisam ser modificadas, sendo mantida compatibilidade com aplicações atuais da Internet com transparência aos usuários, graças ao equipamento sem-fio embarcado no veículo que desempenha o papel de entrada na rede DTN, encapsulando e desencapsulando os dados enviados e recebidos pelos usuários, respectivamente. A premissa da existência de equipamentos capazes de interconectar as redes TCP/IP com as redes DTN é mais simples que a implementação de aplicações de usuários específicas para cenários com conexão intermitente. No caso de transporte público como ônibus, um único roteador sem-fio embarcado poderia intermediar as comunicações de diversos usuários com a Internet.

Uma entidade que intermedeie a comunicação entre a rede DTN e a Internet também é necessária. Essa é a tarefa da Central, que recebe os *bundles* de todos os usuários móveis, os desencapsula e os envia para a Internet. No sentido inverso, a Central recebe as respostas da Internet, as encapsula e as envia para os usuários móveis. Usando a rede DTN, é possível lidar com a mobilidade em cenários *drive-thru*. O problema da correlação do endereço IP com a topologia da rede é contornado pelo encaminhamento de dados baseado em identificadores DTN. Para a rede DTN, os nós de origem e destino são os veículos e não os usuários dentro deles. Assim, localizar um usuário móvel na rede sem-fio significa determinar o veículo no qual ele está se deslocando. A comunicação do roteador sem-fio do veículo com o usuário é feita usando a pilha TCP/IP, de forma transparente para os usuários.

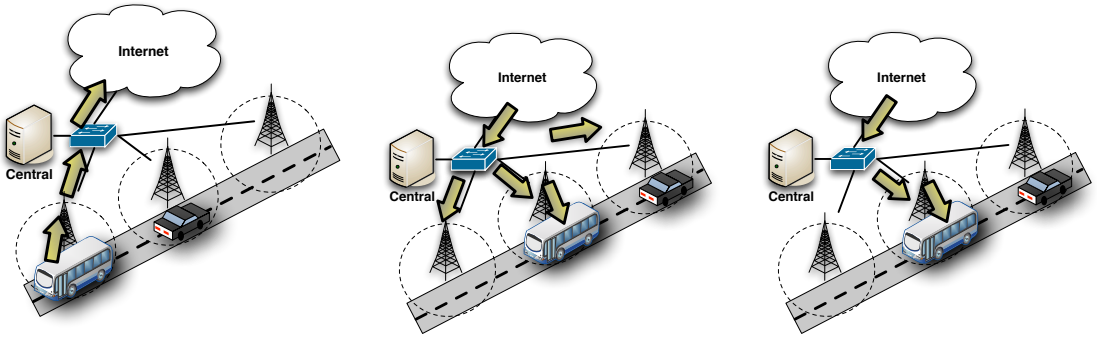
3.2. Roteamento híbrido

O roteamento epidêmico é adequado a cenários onde não há nenhuma informação sobre vizinhança dos nós. Entretanto, na parte cabeada da arquitetura híbrida, os nós são estáticos. Logo, um esquema de roteamento estático é aplicado para conectar a Central às UAs. De fato, implementações DTN oferecem tal opção, que é utilizada para evitar replicação desnecessária de mensagens em redes estáticas. Assim, pode-se ajustar as configurações do roteamento de acordo com a rede. Uma observação importante é a preservação do identificador DTN em ambas as redes, isto é nas redes sem-fio e cabeada, que é importante para a mobilidade. Na arquitetura proposta, cada UA deve lidar com o roteamento estático e epidêmico, em suas interfaces cabeada e sem-fio, respectivamente. A implementação será detalhada na Seção 4.

3.3. Redução da replicação de mensagens

Embora a utilização do roteamento estático possa reduzir a replicação de mensagens na rede cabeada, a posição do usuário móvel na rede ainda é desconhecida. Assim, mesmo usando o roteamento estático, no pior dos casos, a Central teria que enviar a mesma mensagem para todas as UAs. A estimativa da posição do veículo pode, portanto, reduzir ainda mais o número de mensagens transmitidas na rede cabeada, já que a Central se torna capaz de identificar as UAs mais próximas ao destino móvel. Portanto, as mensagens destinadas aos nós na rede móvel não precisam ser encaminhados a todas as UAs, mas apenas a um subconjunto de UAs selecionadas conforme a trajetória estimada dos veículos.

Nas redes DTN, os nós trocam informações sobre as mensagens (*bundles*) que já possuem antes da troca efetiva para evitar replicação. Já que a Central encaminha as



(a) Requisição. (b) Resposta com roteamento estático simples. (c) Resposta com roteamento estático baseado na trajetória.

Figura 2. Entrega de mensagens com o roteamento baseado em trajetória: (a) um usuário móvel envia uma requisição à rede; (b) a resposta é enviada para todas as UAs caso o encaminhamento seja simplesmente estático; (c) a resposta é entregue à UA mais próxima do usuário móvel de acordo com a previsão baseada na trajetória conhecida.

mensagens apenas para as UAs mais próximas ao destino móvel, a troca de mensagens de controle, mesmo na rede DTN, pode ser reduzida. Assim, encontrar mais cedo a provável posição do destinatário evita replicações de mensagens na rede sem-fio. A desvantagem é a possibilidade de o veículo realizar uma mudança de trajetória não prevista, levando à perda de dados. Em caso de ônibus, esse comportamento é pouco provável.

Na arquitetura proposta, os veículos enviam informações de localização à Central toda vez que se conectam a uma nova UA usando uma mensagem DTN específica. Assumindo que a trajetória dos veículos é conhecida, após receber as mensagens de controle, a Central pode estimar sua futura localização. A trajetória pode ser predeterminada, em veículos tais como ônibus e trens ou informada por usuários cooperativos. O banco de dados de posicionamento mantido pela Central é permanentemente atualizado com a informação sobre as futuras UAs no caminho dos nós móveis. Portanto, a Central obtém no banco de dados a próxima UA ao qual um veículo irá se conectar para enviar uma mensagem que lhe é destinada. Assim, reduz-se o número de UAs encarregadas de encaminhar as mensagens. Para melhorar a confiabilidade, a Central poderia enviar a mensagem à UA mais provável e às UAs vizinhas. Assim, caso o veículo mudasse de rota, o mecanismo ainda teria chance de entregar a mensagem corretamente.

A Figura 2 ilustra o roteamento baseado em trajetória proposto. Na Figura 2(a), um usuário móvel envia uma requisição à rede. Já na Figura 2(b), a resposta é encaminhada utilizando o encaminhamento estático sem conhecimento da trajetória. Note que a Central desconhece a posição do destino e, por isso, encaminha a resposta para todas as UAs. A Figura 2(c) ilustra o efeito da proposta já que a resposta é encaminhada pela Central somente à UA mais próxima ao usuário móvel de acordo com a previsão da trajetória.

4. Implementação do Roteamento Híbrido

No cenário híbrido, é necessário configurar dois modos diferentes de roteamento nas UAs, o modo estático na interface de rede cabeada e o epidêmico na interface de rede sem-fio. Entretanto, a implementação de DTN utilizada,

IBR-DTN [Doering et al., 2008], permite apenas um modo de roteamento por nó da rede, independente do número de interfaces. Nesse caso, uma alternativa para contornar essa limitação deve ser desenvolvida. Na implementação realizada, as UAs foram configuradas com o roteamento epidêmico e regras de *firewall* foram adicionadas na interface cabeada, de maneira que ela se comporte como se estivesse no modo estático. Cada UA só possui, então, a Central como vizinho na rede cabeada.

Outra questão sobre a implementação das UAs atuando como ponto de interconexão entre a rede epidêmica e a rede estática é a mudança no identificador DTN do destinatário. Para forçar as UAs a se comportarem como pontes, é adicionada a informação do identificador do destinatário final dentro do *bundle* DTN. Essa estratégia se faz necessária já que a Central não é vizinha dos usuários móveis e, portanto, não possui entradas configuradas em sua tabela de encaminhamento. Assim, se a Central quiser se comunicar com um veículo, ela deve adicionar o identificador desse veículo como destino do *bundle* e então enviar à UA correspondente. A UA usa esse identificador para enviar a mensagem ao veículo correto.

4.1. Encaminhamento baseado em trajetória

O mecanismo baseado em trajetória é implementado via uma "aplicação de localização" que provê à Central informações sobre os veículos e suas posições. Consequentemente, o equipamento sem-fio dentro dos veículos se conecta a uma rede sem-fio usando o aplicativo de localização desenvolvido. Essa conexão é efetuada por um *script* que primeiro procura por identificadores de redes sem-fio (SSIDs) conhecidos e então se conecta àquela com a maior potência de sinal. Para fins de confiabilidade, testa-se a conectividade entre o veículo e a Central de tempos em tempos, se a conexão não for boa o suficiente, o *script* tenta se conectar a outra UA, reiniciando todo o procedimento. Assim que a conexão entre um veículo e a Central é estabelecida, o veículo cria uma mensagem de localização e a envia à Central. Esse procedimento usa a aplicação `dtnsend` disponível na implementação usada neste trabalho (`IBR-DTN`) [Doering et al., 2008]. O `dtnsend` é usado para enviar arquivos na rede DTN. A mensagem contém um arquivo escrito em `XML`, que pode ser visto na Lista 1.

Na Central, a aplicação efetua um laço infinito recebendo mensagens dos nós móveis que executam a aplicação de localização desenvolvida. A Central utiliza o `dtnrecv`, que é uma aplicação do `IBR-DTN` para receber arquivos enviados com o `dtnsend`. A mensagem recebida é analisada e a informação de posição é extraída. A aplicação usa o identificador do ônibus para procurar sua trajetória predeterminada. A partir da trajetória e do identificador da última UA que o ônibus esteve, a aplicação prevê as próximas UAs que o veículo passará. As informações obtidas sobre a atual e as futuras posições do ônibus são escritas em um arquivo nomeado de acordo com o identificador do veículo. Esses arquivos podem ser usados como entrada de outras aplicações que precisem da posição do veículo. Neste trabalho, o roteamento estático pode estar ciente da posição de um dado veículo pela leitura de arquivo de registros (*log*) correspondente.

Quando a Central envia uma mensagem a um veículo, ela a envia à última UA onde o veículo foi visto e às próximas UAs da trajetória. Neste artigo, experimentos foram realizados variando o número de UAs intermediárias, a fim de observar o impacto desse parâmetro na rede.

```
<LOCBUNDLE>
    <APPLICATIONID>
        LOCALIZATION
    </APPLICATIONID>
    <APPLICATIONHEAD>
        <TIME> Hora-minuto </TIME>
        <BID> Número aleatório </BID>
    </APPLICATIONHEAD>
    <APPLICATIONBODY>
        <VEHICLE> Identificador do veículo </VEHICLE>
        <BUSTOP> Identificador da UA </BUSTOP>
    </APPLICATIONBODY>
</LOCBUNDLE>
```

Lista 1. Formato da mensagem de localização.

5. Cenário de Testes

O cenário de testes consiste de um computador e seis roteadores sem-fio. O computador faz o papel da Central, enquanto os roteadores sem-fio fazem ou o papel das UAs ou dos equipamentos sem-fio dentro dos veículos. Os roteadores agindo como equipamentos sem-fio dos veículos também assumem o papel de clientes móveis.

O computador possui um processador Intel Pentium D de 3,20 GHz e 4 GB de memória RAM. O sistema operacional utilizado é o Debian Etch [Debian, 2012] e a implementação do protocolo DTN é a `IBR-DTN` [Doering et al., 2008]. O protocolo de roteamento padrão é o estático, o que requer configuração prévia de todas as UAs. As UAs são roteadores sem-fio D-Link DIR-320 com 32 MB de memória RAM e 4 MB de memória flash. Esses roteadores utilizam como sistema operacional o OpenWrt [OpenWrt, 2012]. O protocolo de roteamento é configurado de acordo com o tipo de interface de rede: interfaces de rede sem-fio são configuradas em modo epidêmico enquanto as interfaces de rede cabeada são configuradas em modo estático usando as regras de *firewall* como recurso disponível. Todas as UAs anunciam uma rede sem-fio para conexão dos veículos. Adicionalmente, as UAs podem armazenar *bundles* em seus *buffers* enquanto a conexão com o destinatário não for possível. Por isso, todos os roteadores possuem um dispositivo de armazenamento USB externo.

O cenário de rede utilizado pode ser visto na Figura 3. Existem cinco UAs no ambiente de testes, a Central e um roteador representando, ao mesmo tempo, o equipamento sem-fio dentro de um veículo e o usuário móvel. Sempre que necessário, um *script* que emula a movimentação do veículo é utilizado, conectando e desconectando o veículo das UAs como aconteceria com o veículo em um deslocamento real.

6. Resultados

Os testes realizados contemplam as seguintes métricas de desempenho: tempo de localização do veículo, número de replicações de mensagens, taxa de entrega de mensagens e tráfego na rede sem-fio. Para avaliar a queda da taxa de entrega imposta pela solução, os testes comparam a proposta com o protocolo de roteamento epidêmico, que é o melhor em termos de taxa de entrega e ainda é um dos poucos a disponibilizar uma implementação conhecida para uso em redes de testes. Uma vez que a abordagem de

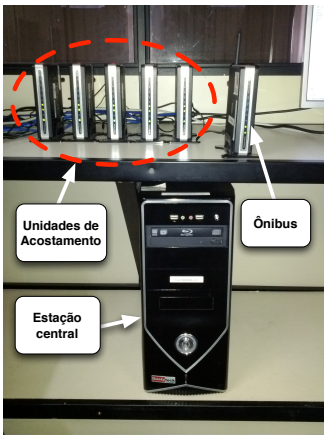


Figura 3. Rede de testes utilizada nos experimentos.

localização depende de trajetórias predeterminadas, os testes foram conduzidos considerando o veículo como um ônibus universitário. Assim, assume-se as paradas de ônibus como o local mais adequado para o posicionamento das UAs. Consequentemente, uma avaliação preliminar foi realizada para medir o tempo médio que um ônibus permanece conectado com uma mesma UA. Foi avaliado o caso onde o ônibus não para e o caso em que ele para no ponto de ônibus para pegar passageiros. Ambas as avaliações foram conduzidas na própria universidade (Universidade Federal do Rio de Janeiro - UFRJ), para permitir resultados mais próximos da realidade.

De acordo com as medidas feitas, os ônibus permanecem conectados em média por 65 e 36 segundos, respectivamente, caso parem ou não para pegar passageiros. Ainda conforme testes realizados, as regiões de cobertura das UAs são de aproximadamente 200 metros na via. Além disso, o tempo de desconexão entre UAs consecutivas é de 90 segundos em média. Isso é calculado considerando que a distância entre as paradas de ônibus e a velocidade máxima permitida na via do campus são de 1 km e 40 km/h, respectivamente. Ao longo dos experimentos são comparadas a abordagem proposta e a abordagem puramente epidêmica.

6.1. Tempo de localização dos veículos

O cenário híbrido proposto requer mensagens de localização dos ônibus para a Central. Portanto, toda a comunicação entre essas duas entidades deve ser realizada em um tempo menor que o tempo mínimo de conexão com uma UA. Se esse requisito for garantido e a trajetória do ônibus seguir o esperado, a localização correta do ônibus pode ser assumida.

Neste teste, mediu-se o tempo necessário para todo o processo, desde a associação à rede sem-fio até a recepção da informação correspondente na Central. Para tal, foram criados dois *scripts*: um executado pelo roteador do ônibus e outro pela Central. O *script* do ônibus é responsável por conectá-lo às UAs, escrevendo a hora de inicialização do *script* em um arquivo e enviando a mensagem de localização para a Central. A Central aguarda a mensagem de localização e ao recebê-la, escreve a hora da chegada em um arquivo de registro próprio. Os relógios do roteador do ônibus e da Central estão sincronizados através de um único servidor NTP (*Network Time Protocol*). Embora seja utilizado

o sincronismo dos relógios na localização dos ônibus, pequenas diferenças são toleráveis, já que a escala de tempo é da ordem de segundos ou minutos dependendo do cenário.

O teste foi repetido dez vezes e os seus resultados estão na Tabela 1. A última coluna é a diferença entre a hora de recebimento da mensagem pela Central e a inicialização do *script* do ônibus. O resultado revela que o tempo médio necessário para este procedimento é de 15,6 segundos, com desvio padrão de 3,98 segundos. Como pode ser observado, o tempo necessário para que um ônibus se conecte a uma UA e a Central receba sua localização é menor que o pior tempo de permanência do ônibus conectado a uma UA, que é de 36 segundos. Assim, pelos testes realizados, pode-se assumir que a mensagem de localização é entregue e, consequentemente, os nós móveis podem ser localizados.

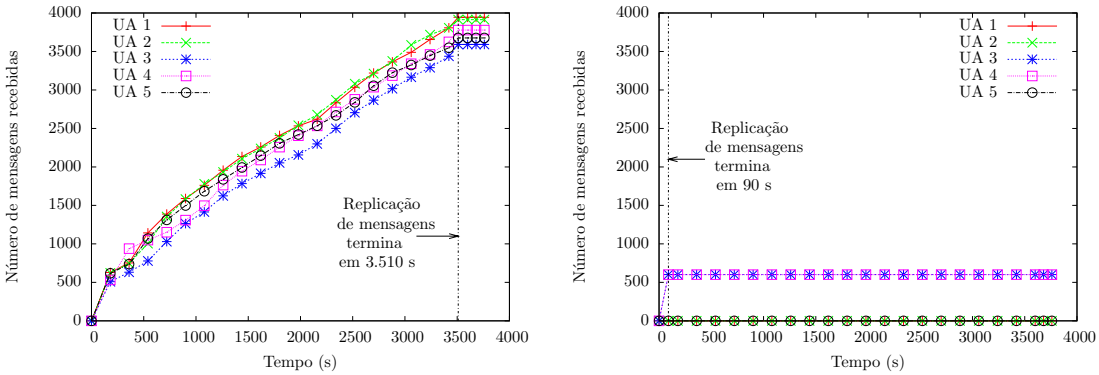
Tabela 1. Tempo necessário para a localização do veículo.

Identificador da mensagem	Inicialização do *script* no ônibus	Recepção na UA	Recepção na Central	Tempo total da localização (s)
1	13:45:58	13:46:08	13:46:11	**13**
2	13:46:50	13:47:01	13:47:05	**15**
3	13:47:42	13:47:53	13:48:00	**18**
4	13:48:34	13:48:45	13:48:47	**13**
5	13:49:26	13:49:37	13:49:41	**15**
6	13:50:18	13:50:29	13:50:31	**13**
7	13:51:10	13:51:21	13:51:25	**15**
8	13:52:02	13:52:13	13:52:17	**15**
9	13:52:54	13:53:05	13:53:20	**26**
10	13:53:46	13:53:57	13:53:59	**13**

6.2. Replicação das mensagens

No teste de replicação das mensagens, o roteador agindo como ônibus emula a movimentação entre as diferentes UAs. O período que o ônibus permanece conectado à UA é de 65 segundos, que é o tempo médio requerido para um ônibus pegar passageiros. Já o tempo de desconexão é de 90 segundos conforme discutido no início da Seção 6. No teste, o ônibus inicia seu movimento na UA 1 e percorre todas as UAs em ordem crescente de identificador. A Central envia uma mensagem diferente a cada 0,1 segundos para o ônibus durante 1 minuto. Como o tempo em que o ônibus permanece conectado à mesma UA neste teste é maior que o tempo de envio de mensagens pela Central, o ônibus permanece conectado à mesma UA durante este experimento.

A Tabela 2 expressa o percentual de mensagens recebidas por cada UA, considerando as mensagens enviadas pela Central, e as mensagens de localização apenas na abordagem proposta. Devido à replicação de mensagens, a quantidade total recebida pelas UAs pode passar de 100%. Na Tabela 2, observa-se que a abordagem proposta reduz em mais de quinze vezes o número de mensagens replicadas na rede cabeada em comparação com o cenário de roteamento epidêmico. Além disso, como o encaminhamento proposto usa a abordagem baseada na trajetória, as mensagens são encaminhadas apenas à última UA onde o ônibus esteve conectado e à próxima UA da trajetória do ônibus. Pode-se, então, inferir que o ônibus estava conectado à rede através da UA 3. O valor percentual maior que 100% ocorre, pois as mensagens de localização também são enviadas pelo encaminhamento proposto.



(a) Roteamento epidêmico. (b) Roteamento proposto.

Figura 4. Replicação de mensagens na rede cabeada.

No roteamento epidêmico, as réplicas das mensagens são recebidas por todas as UAs. A considerável quantidade de réplicas ocorre, pois cada UA, tendo recebido a mensagem, a replica para todos os seus vizinhos, isto é para todas as outras UAs. Por isso, a vantagem de utilizar o roteamento proposto é proporcional ao número de UAs já que o número de réplicas de mensagens é constante no protocolo proposto, enquanto é crescente com o número de UAs no caso epidêmico.

Tabela 2. Replicação de mensagens nas UAs.

Identificador da UA	Roteamento proposto	Roteamento epidêmico
1	0%	657,3%
2	0%	652,3%
3	100,2%	598,5%
4	100%	629,7%
5	0%	612,7%
Total	**200,2%**	**3150,5%**

Durante o teste, todas as mensagens foram enviadas pela Central em um minuto. Entretanto, as abordagens de roteamento levam tempo adicional para terminar a replicação de mensagens na rede cabeada. Na abordagem proposta, o procedimento de replicação leva 90 segundos para entregar todas as mensagens às UAs selecionadas de acordo com o encaminhamento proposto baseado na trajetória, como visto na Figura 4(b). Em oposição, o roteamento epidêmico leva 58 minutos e 12 segundos para entregar todas as mensagens e suas réplicas para todas as outras UAs, como visto na Figura 4(a). Em ambas as figuras, o platô somente é alcançado quando a replicação de mensagens termina. Isso claramente mostra o impacto negativo do roteamento epidêmico nas redes cabeadas, que é evitado em nossa arquitetura híbrida.

6.3. Entrega de mensagens

No teste de entrega de mensagens, o ônibus circula pelas cinco UAs parando para pegar passageiros. Os testes foram realizados durante 24 horas e a Central envia 500 mensagens durante o experimento. Esse teste avalia a comunicação do ponto de vista do roteador do ônibus.

A Tabela 3 mostra os resultados do número de mensagens perdidas, do percentual de mensagens entregues ao roteador do ônibus, do número médio de réplicas de mensagens recebidas pelo roteador do ônibus e do maior número de réplicas recebidas pelo ônibus. Analisando os resultados, observa-se um compromisso da abordagem baseada na trajetória. O número médio de réplicas de mensagens entregues ao ônibus e o número de mensagens perdidas são maiores que o do roteamento epidêmico. Contudo, o número de réplicas é uma função do número de UAs usadas durante o procedimento de localização. Embora, a Central escolha apenas a UA atual e a próxima na trajetória do ônibus, isso pode ser ajustado, podendo variar desde apenas uma UA (p. ex. a UA atual) até todas as UAs em uma determinada área. Isso vai depender do nível de confiança desejado frente o número tolerado de réplicas.

Tabela 3. Estatísticas de entregas de mensagens.

Estatísticas	Roteamento epidêmico	Roteamento proposto
Número de mensagens perdidas	0,00	3,00
Taxa de entrega de mensagens	100,00%	99,60%
Número médio de réplicas recebidas	1,36	1,79
Número máximo de réplicas recebidas	2,00	2,00

6.4. Análise do tráfego na rede sem-fio

No teste de análise do tráfego na rede sem-fio, a quantidade de dados transferidos entre o ônibus e todas as UAs por onde o ônibus passou em seu percurso foi medida. Durante o experimento, a Central envia 160 mensagens igualmente distribuídas ao longo de duas horas. Tais mensagens são recebidas em 55 oportunidades de contato do ônibus com as UAs, das quais as seis primeiras e as seis últimas são usadas apenas para estimar o tráfego das mensagens de controle. Para uma análise mais completa, foi analisada também a taxa de entrega das mensagens. Assim, pretende-se identificar as causas de uma possível redução na taxa de entrega, seja ela devido à carga da rede ou ao efeito da aplicação de localização.

O ônibus mantém o mesmo padrão de movimentação do experimento anterior, ou seja, 65 segundos de conexão com as UAs e 90 segundos de desconexão entre UAs. Esse teste foi realizado para quatro diferentes configurações: a epidêmica e três possibilidades de configurações híbridas. A diferença entre as configurações híbridas é o número de UAs servindo como nós intermediários entre a Central e os destinatários móveis. Foram testadas as configurações nas quais a Central envia mensagens para apenas a última UA onde o ônibus foi encontrado (Configuração Híbrida 1UA); para a última e a próxima UA na trajetória do ônibus (Configuração Híbrida 2UA); e para a última e as duas próximas UAs na trajetória do ônibus (Configuração Híbrida 3UA). Espera-se que o aumento do número de UAs tenha como consequência um aumento do tráfego na rede sem-fio e um aumento da taxa de entrega.

A Tabela 4 apresenta, para cada cenário, a taxa de entrega e o número de mensagens perdidas, desconsideradas as réplicas. A replicação média de uma mensagem,

ou seja, o número médio de mensagens iguais que o ônibus recebe para cada mensagem entregue, e o máximo de réplicas recebidas de uma mesma mensagem são também apresentadas. A taxa de entrega é menor comparada a resultados anteriores (Tabela 3) pois os experimentos consideram um cenário móvel com maior carga.

A Tabela 4 mostra que a taxa de entrega na configuração Híbrida 1UA foi de 31,250% que é baixa já que as mensagens eram entregues apenas às UAs nas quais os ônibus estavam conectados. Isso significa que apenas as mensagens geradas pela Central no intervalo de tempo entre receber a mensagem de apresentação e a saída do ônibus do ponto podiam ser recebidas. Observa-se que a taxa de entrega da configuração Híbrida 3UA e da configuração epidêmica são próximas, demonstrando que a configuração proposta consegue aumentar a taxa de entrega com o aumento do número de UAs intermediárias. Além disso, pode-se concluir que com três UAs, já é possível obter uma taxa de entrega próxima a da configuração epidêmica. Vale ainda mencionar que o número de cópias recebidas pelo ônibus reflete o número de mensagens geradas na Central, no caso do cenário epidêmico; e o número de mensagens geradas nas UAs, no caso híbrido.

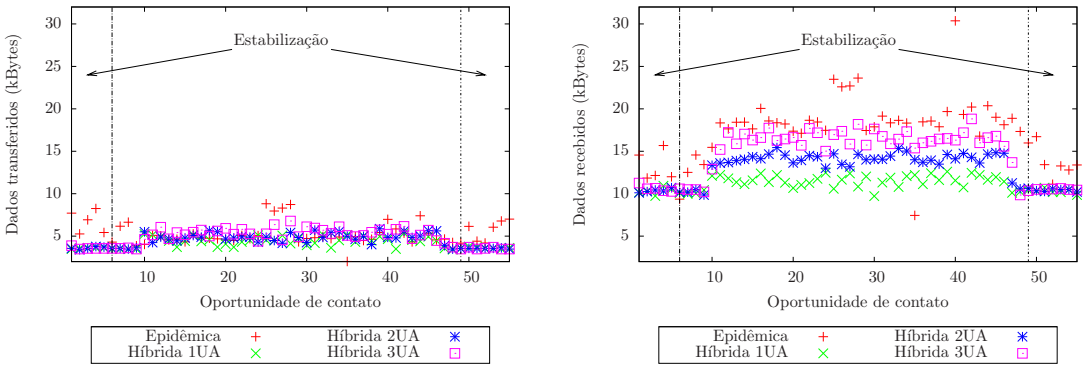
Tabela 4. Entrega de mensagens.

Cenário	Taxa de entrega	Número de mensagens perdidas	Replicação média de uma mensagem	Maior número de réplicas
Epidêmico	80,625%	31	1	1
Híbrido 1UA	31,250%	110	1	1
Híbrido 2UA	59,375%	65	1,484	2
Híbrido 3UA	80,000%	32	1,880	3

A Figura 5 apresenta a quantidade de dados transmitidos e recebidos pelo ônibus nas oportunidades de contato com as UAs. Os resultados mostram que o número de UAs intermediárias influencia diretamente no tráfego da rede sem-fio. Comparando a configuração epidêmica com as configurações híbridas, nota-se que a configuração epidêmica possui, em média, maior tráfego do que qualquer uma das configurações híbridas. Tal efeito é mais evidente nos dados recebidos pelo ônibus, já que os dados transmitidos pelo ônibus para as UAs são apenas de controle, conforme o experimento realizado. É importante perceber que a configuração Híbrida 3UA apresenta resultados de taxa de entrega próximos aos da epidêmica e, mesmo com maior replicação de mensagens, insere uma menor quantidade de dados na rede sem-fio. Isso é consequência da aplicação de localização que, além de reduzir a carga de dados na rede cabeada, ainda diminui a carga de controle na rede sem-fio. Tal carga de controle é usada pela rede DTN antes da troca de mensagens para evitar réplicas.

A Tabela 5 apresenta a média com o desvio padrão da carga transferida (dados mais controle). São apresentadas as médias ao longo das oportunidades de contato dos dados transmitidos e recebidos pelo ônibus, além do total. Novamente, verifica-se que a configuração epidêmica é a que insere maior quantidade de dados na rede sem-fio, seguida pela configuração Híbrida 3UA. Vale mencionar que o tráfego torna-se mais estável com a redução do número de UAs intermediárias.

Um resultado adicional pode ser verificado na Figura 5 durante os períodos de estabilização do sistema – seis primeiras e seis últimas oportunidades de contato. Nesses



(a) Dados transmitidos. (b) Dados recebidos.

Figura 5. Dados transmitidos e recebidos pela interface sem-fio do ônibus.

Tabela 5. Carga de dados e controle transferidos na rede sem-fio.

Configuração	Dados transmitidos (kBytes)	Dados recebidos (kBytes)	Total (kBytes)
Epidêmica	5,27±1,35	16,83±3,71	22,10±4,41
Híbrida 1UA	4,11±0,61	10,91±0,83	15,03±1,40
Híbrida 2UA	4,41±0,75	12,65±1,81	17,05±2,51
Híbrida 3UA	4,71±0,95	14,23±2,85	18,94±3,76

períodos, todo o tráfego na rede sem-fio é devido aos processos de controle. No cenário epidêmico, onde todas as UAs e a Central se tornam vizinhas indiretas do ônibus, ele é resultante da troca periódica de informações de vizinhança. Todavia, nas configurações híbridas, ele é resultante tanto da troca periódica de informações de vizinhança quanto do envio das mensagens de localização. Como somente a Central é vizinha indireta do ônibus nos casos híbridos, há uma redução da carga da informação de vizinhança. Nota-se nesses períodos uma maior sobrecarga inserida pelo cenário epidêmico, o que confirma os resultados anteriores. Enquanto todos os cenários híbridos possuem taxa de controle entre 12 e 15 kBytes, o cenário epidêmico possui uma taxa que varia entre 12 e 24 kBytes.

De forma geral, pode-se concluir que o cenário proposto pode diminuir o tráfego de controle da rede sem-fio e ainda alcançar taxas de entrega similares à configuração epidêmica. Tanto a taxa de entrega quanto a carga de dados são dependentes das configurações realizadas, como o número de UAs intermediárias utilizadas. Acredita-se que o desempenho da proposta é ainda mais evidente em cenários em maior escala onde o número de UAs é maior que cinco.

7. Conclusões e Trabalhos Futuros

Este trabalho propôs um protocolo de roteamento para cenários híbridos que combinam redes sem-fio veiculares com conexão intermitente e redes infrastruturadas com acesso à Internet através de múltiplos pontos de interconexão (*gateways*). O protocolo proposto combina a utilização de roteamento estático na parte cabeada e epidêmico na parte sem-fio. Como as redes DTN não foram projetadas para redes cabeadas, um mecanismo baseado em trajetória foi desenvolvido para evitar replicação de mensagens na rede cabeada e ainda reduzir a carga de controle da rede sem-fio. Os resultados experimentais

mostraram que o mecanismo proposto evita, de fato, a replicação de mensagens na rede cabeada e ainda oferece mobilidade transparente para os usuários sem-fio. Ainda, a proposta se mostrou capaz de reduzir a carga de controle da rede sem-fio, podendo manter a mesma taxa de entrega da configuração epidêmica.

Como trabalhos futuros, planeja-se propor um mecanismo para mudar dinamicamente a trajetória esperada dos veículos e ainda estender os experimentos realizados para uma rede de testes maior.

Agradecimentos

Este trabalho foi financiado pelo CNPq, CAPES, CTIC/RNP e FAPERJ.

Referências

Burgess, J., Gallagher, B., Jensen, D. e Levine, B. N. (2006). MaxProp: Routing for vehicle-based disruption-tolerant networks. Em *IEEE INFOCOM*, p. 1–11.

Cheng, P.-C., Lee, K. C., Gerla, M. e Härri, J. (2010). GeoDTN+Nav: Geographic DTN routing with navigator prediction for urban vehicular environments. *Mobile Networks and Applications*, 15:61–82.

Debian (2012). The universal operating system. `http://www.debian.org/`.

Doering, M., Lahde, S., Morgenroth, J. e Wolf, L. (2008). IBR-DTN: an efficient implementation for embedded systems. Em *ACM CHANTS*, p. 117–120.

Han, B., Hui, P., Kumar, V. S. A., Marathe, M. V., Pei, G. e Srinivasan, A. (2010). Cellular traffic offloading through opportunistic communications: a case study. Em *ACM CHANTS*, p. 31–38.

OpenWrt (2012). Openwrt: Wireless freedom. `https://openwrt.org`.

Ott, J. e Kutscher, D. (2005). Exploiting regular hot-spots for drive-thru Internet. Em *Kommunikation in Verteilten Systemen (KiVS)*, p. 218–229.

Saucez, D., Iannone, L. e Bonaventure, O. (2009). OpenLISP: An open source implementation of the Locator/ID separation protocol. Em *ACM SIGCOMM Demos Session*, p. 1–2.

Scott, K. e Burleigh, S. (2007). Bundle protocol specifications. RFC 5050.

Spyropoulos, T., Psounis, K. e Raghavendra, C. S. (2008a). Efficient routing in intermittently connected mobile networks: The multiple-copy case. *IEEE/ACM Transactions on Networking*, 16(1):77–90.

Spyropoulos, T., Psounis, K. e Raghavendra, C. S. (2008b). Efficient routing in intermittently connected mobile networks: The single-copy case. *IEEE/ACM Transactions on Networking*, 16(1):63–76.

Weissberger, A. (2012). Strong growth in WiFi LAN equipment and WiFi phone sales as total market nears $1B! `http://community.comsoc.org/blogs/alanweissberger/strong-growth-wifi-lan-equipment-and-wifi-phone-sales-total-marketnears-1b`.

Uma Política de *Handover* de Gerência de Mobilidade de Fluxo baseada em Lógica *Fuzzy*

Rodolfo I. Meneguette[1], **Luiz F. Bittencourt**[1], **Edmundo R. M. Madeira** [1]

[1]Instituto de Computação - Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 - Cidade Universitária - Campinas/SP - Brasil

{ripolito,bit,edmundo}@ic.unicamp.br

Abstract. *Applications of vehicular ad hoc networks can take advantage of the use of simultaneous connections, thereby maximizing the throughput of the network and reducing network latency. In order to take advantage of all radio interfaces in the vehicle and to provide good quality of service for vehicular applications, we developed a handover selection policy based on fuzzy logic, which indicates the best interface for a particular flow. Each flow is the aggregation of traffic of applications in the same class. Besides indicating the best network interface of a given flow, the policy aims to minimize unnecessary changes between interfaces and data flow. We use the simulator NS3 to assess the policy performance against other approaches. For this, we compare our policy with policy-based thresholds, a policy based on game theory and a policy that also is based on fuzzy logic. It was observed that the proposed scheme showed a low flow switching time, few packet losses, and low delay.*

Resumo. *Aplicações das redes veiculares ad hoc podem tirar vantagem da utilização de conexões simultâneas, e com isso maximizam a vazão e diminuem a latência da rede. A fim de tirar proveito de todas as interfaces de rádio do veículo e para dar boa qualidade de serviço para aplicações veiculares, desenvolvemos uma política de seleção de handover baseada na lógia fuzzy , que indica qual é a melhor interface a um determinado fluxo. Cada fluxo é a agregação do tráfego de aplicações de uma mesma classe. Além de indicar a melhor interface de rede para um determinado fluxo, a política visa minimizar trocas desnecessárias entre as interfaces e os fluxos de dados. Utilizamos o simulador NS3 para analisar a política de seleção de interface. E compará-la a uma política baseada em limiares, uma política baseada na teoria dos jogos e uma política que também é baseada na lógica fuzzy. Observamos que o mecanismo proposto apresentou um tempo de troca de fluxo baixo, com menor perda de pacotes e menor atraso.*

1. Introdução

Redes ad hoc veiculares (VANET) são uma subclasse das redes móveis ad hoc que prevê a comunicação sem fio entre veículos, bem como entre veículos e dispositivos ao longo de rodovias. Em VANETs, cada veículo pode ter múltiplas interfaces de rádio, sendo assim capaz de se conectar simultaneamente a diferentes domínios e tecnologias de acesso à rede de rádio [Blanchet and Seite 2011]. Embora estes veículos possam se conectar a diferentes tecnologias de rede simultaneamente, hoje em dia os veículos estão limitados

a escolher uma interface padrão para envio e recebimento de informações. Essa limitação está relacionada com o atual modelo de gestão de múltiplas interfaces, onde várias interfaces estão ligadas ao sistema operacional [Wasserman and Seite 2011]. Normalmente, os sistemas operacionais utilizam arquivos de configuração do usuário, ou baseiam-se no tipo de aplicação para selecionar uma interface de rede padrão para enviar e receber dados [Makaya et al. 2012].

Para permitir a utilização de mais do que uma interface de rede simultaneamente, o *Internet Engineering Task Force* (IETF) tem desenvolvido a tecnologia de mobilidade de fluxo de IP, que permite a divisão de Fluxo IP entre vários enlaces de acordo com os requisitos das aplicações e preferências do usuário. Existem alguns grupos do IETF estudando as extensões de mobilidade para o IPv6 (MEXT) [Tsirtsis et al. 2011] considerando as extensões da rede de mobilidade (NETEXT) [Bernardos 2012]. Estes grupos trabalham no desenvolvimento e elaboração de um protocolo que permite o uso de mais de uma interface simultaneamente. Embora esses protocolos lidem com várias interfaces ao mesmo tempo, nenhum desses grupos de desenvolvimento especifica um protocolo ou uma política de selecão da mobilidade do fluxo, ou seja, para qual interface um determinado fluxo deverá ser mapeado e quando essa mudança deve ocorrer.

Neste trabalho propomos e avaliamos uma política de seleção utilizando lógica *fuzzy* para o mapeamento dinâmico entre as classes de aplicação das redes veiculares e as interfaces de rede ativas no veículo. A política proposta precisa lidar com as características de cada fluxo, e além disso conhecer o estado atual de cada tecnologia de rede ativa no ambiente para indicar qual a melhor interface de rede para um determinado fluxo de dados. Para isso dividimos as aplicações de redes veiculares em três classes, sendo essa divisão de acordo com os objetivos gerais de cada aplicação: *segurança*, *conforto*, e de *usuário*. Além disso, esse modelo considera que os veículos estão se movendo em uma cidade ou em uma estrada, e que os condutores ou passageiros estão executando mais de uma classe de aplicativo ao mesmo tempo. O objetivo dessa política é diminuir o tempo de troca de um fluxo de uma interface para outra, com isso diminuindo a perda de pacotes e aumentando a vazão da rede.

Este artigo está organizado como segue. Na Seção 2 descrevemos o protocolo 802.21, o protocolo *Proxy Mobile IP* versão 6 e também introduzimos conceitos de sistemas *fuzzy*. Na Seção 3 são discutidos os trabalhos relacionados. A Seção 4 apresenta a proposta de política de *handover* para a gerência de mobilidade de fluxo baseada em lógica *fuzzy*. A Seção 5 apresenta uma análise dos resultados obtidos, seguido pela conclusão na Seção 6.

2. Fundamentação Teórica

Essa seção apresenta alguns conceitos básicos utilizados neste artigo.

2.1. Protocolo 802.21

O IEEE 802.21 [Dutta et al. 2007] é um esforço recente de especificação do IEEE, que visa permitir a transferência e interoperabilidade entre redes heterogêneas, incluindo os padrões 802 e redes não 802. Uma das principais idéias do IEEE 802.21 é fornecer uma interface comum para a gestão de eventos e mensagens de controle trocadas entre dispositivos de redes que possuem tecnologias diferentes. O objetivo do IEEE 802.21 é melhorar

e facilitar o uso dos nós móveis, proporcionando transmissão ininterrupta em redes heterogêneas. Para este fim, os procedimentos de entrega podem utilizar as informações recolhidas a partir do terminal móvel e/ou infraestrutura de rede. Ao mesmo tempo, diversos fatores podem determinar a decisão de entrega: serviço de continuidade, classe de aplicações, qualidade de serviço, negociação de qualidade de serviço, segurança, etc. As tarefas mais importantes do IEEE 802.21 são: a descoberta de novas redes no ambiente e seleção da rede mais apropriada para uma determinada necessidade. A descoberta de rede e processo de seleção são facilitados pelo intercâmbio de informações da rede que ajudam os dispositivos móveis a determinar quais as redes ativas ao seu redor, permitindo assim que o dispositivo móvel se conecte à rede mais apropriada com base em suas próprias políticas [Meneguette et al. 2012]. Entretanto, esse protocolo não especifica nenhum algoritmo de seleção de rede.

O núcleo do 802.21 é a *Media Independent Handover Function* (MIHF). O MIHF terá de ser implementado em todo dispositivo compatível com o IEEE 802.21 (em *hardware* ou *software*). Essa função é responsável pela comunicação com diferentes terminais, redes e MIHFs remotos e também pelo oferecimento de serviços de informações para as camadas superiores [Marquez-Barja et al. 2011]. O MIHF define três serviços diferentes: *Independent Event Service* (MIES), *Media Independent Command Service* (MICS) e *Media Independent Information Service* (MIIS). Esses serviços permitem a obtenção e armazenamento de informações relevantes sobre o estado da rede tais como perda, vazão, e quais são as sub-redes [Meneguette and Madeira 2011]. Detalhes sobre o protocolo 802.21 podem ser encontrados em [Dutta et al. 2007].

2.2. Proxy Mobile IPv6

Proxy Mobile IPv6 (PMIPv6), conforme especificado em [Gundavelli et al. 2008], fornece um gerenciador de mobilidade baseado na rede para conectar hosts a um domínio PMIPv6. PMIPv6 introduz duas novas entidades funcionais: o *Local Mobility Anchor* (LMA) e o *Mobile Access Gateway* (MAG). MAG é a primeira camada que detecta um nó móvel (MN) associando-se a esse nó e oferecendo uma conectividade IP. O LMA é a entidade que irá atribuir um ou mais *Home Network Prefixes* (HNPs) para o nó móvel.

A base fundamental do PMIPv6 está no MIPv6 no sentido em que ele estende o MIPv6 utilizando conceitos tais como a funcionalidade do *home agent* (HA). O LMA e o MAG estabelecem um túnel bidirecional para encaminhamento de todo o tráfego de dados pertencente aos nós móveis. A gerência de mobilidade suporta uma liberdade de mobilidade dentro do domínio do PMIPv6, ou seja, um host móvel pode circular livremente dentro do domínio PMIPv6 sem alterar o seu endereço IP [Bernardos et al. 2012].

2.3. Lógica Fuzzy

Sistemas Fuzzy podem ser definidos como sistemas que utilizam a teoria de conjuntos *fuzzy* proposta por Lofti A. Zadeh em 1965 para representar pelo menos uma de suas variáveis, permitindo a representação e o processamento de informações imprecisas e incertas, abundantes no mundo real.

Os sistemas baseados em lógica *fuzzy*, ou Sistemas *Fuzzy* (SF), usam um mecanismo de raciocínio baseado no raciocínio aproximado que possui grande habilidade para expressar a ambigüidade e subjetividade presentes no raciocínio humano [Uesu et al. 2011].

As bases de regras dos SF armazenam conhecimento representado, geralmente, por meio de regras do tipo:

$$IF \rightarrow THEN$$

Os antecedentes das regras relacionam as entradas do sistema, enquanto os conseqüentes relacionam as saídas, usando operadores lógicos. Um sistema de inferência, baseado em graus de pertinência e operadores de associação, é utilizado para a obtenção de uma saída a partir de dados de entrada.

3. Trabalhos Relacionados

Esta seção apresenta alguns trabalhos relacionados a políticas de decisão no momento da troca de uma interface de rede para outra. Alguns desses trabalhos utilizam a teoria de jogos como mecanismo de decisão e outros utilizam a lógica *fuzzy*.

Patil e Kolte [Patil and Kolte 2011] desenvolveram um mecanismo de otimização de algoritmo de *handover* utilizando a lógica *fuzzy*. O algoritmo é utilizado para *handover* vertical e utiliza 3 métricas: *Signal to Interference Ratio* (SIR), a velocidade do dispositivo móvel e o tipo de tráfego da rede. Esses parâmetros são utilizados para a tomada de decisão em realizar um *handover* ou não. Essa decisão é feita pelo sistema lógico *fuzzy* que utiliza 7 tipos de saída: *Highest, Higher, High, Normal, Low, Lower,* ou *Lowest*.

Dhar e colaboradores [Sourav Dhar and Bera 2011] implementam um mecanismo de *handover* vertical inteligente baseado no *Analytic Hierarchy Process* (AHP) e de processos de decisão *fuzzy* para *Intelligent Transportation System* (ITS). O mecanismo utiliza o custo, a velocidade do veículo e a carga de tráfego da rede. Além disso, utiliza 2 tipos de saída na tomada de decisão do *handover*: *low* e *high*. Esse trabalho utiliza números *fuzzy* triangulares.

Dhar e colaboradores [Dhar et al. 2012] propuseram um mecanismo de Handover, o *Cognitive Vertical Handover* (CVHO), para assegurar a conectividade. Para isso o mecanismo utiliza rede neural artificial e o *Analytic Hierarchy Process* (AHP) para a escolha da melhor rede no momento do handover. Para a realização da escolha da rede a política utiliza múltiplos critérios, tais como, velocidade, largura de banda, tráfego da rede, atraso e custo.

Bin Ma e colaboradores [Ma and Liao May] propuseram um algoritmo de handover vertical, utilizando lógica fuzzy tipo 2 e levando em consideração a velocidade adaptativa do veículo na fase da descoberta de redes ativas no ambiente. Esse algoritmo adaptativo atualiza o conjunto de redes candidatas e ajusta o tempo handover de acordo com a velocidade do usuário. A melhor rede que faz parte desse conjunto é selecionada pela lógica Fuzzy tipo 2.

Nossa política utiliza lógica *fuzzy* para indicar a necessidade de troca de um fluxo de uma interface para outra e qual será a melhor interface. Para isso, utilizamos três variáveis lógicas: perda de pacote, vazão da rede e atraso das mensagens. Além disso, utilizamos cinco tipos de saída: muito alta, alta, normal, baixa e muito baixa, para evitar que a troca de uma interface de rede para outra seja precipitada, com isso diminuindo a quantidade de trocas executadas.

4. Política de *Handover* de Gerência de Mobilidade de Fluxo Baseada em Lógica *Fuzzy*

A política de gerenciamento de mobilidade de fluxo proposta trata do mapeamento entre o fluxo das classes de aplicações das redes veiculares e as interfaces de rede ativas no ambiente onde o veículo está transitando. Essa política tem por objetivo selecionar a melhor interface que atenda os requistos mínimos de cada fluxo de dados. Para atingir esse objetivo utilizamos a lógica *fuzzy* como política de decisão. As regras *fuzzy* levam em conta a vazão da rede, perda de pacote e o atraso.

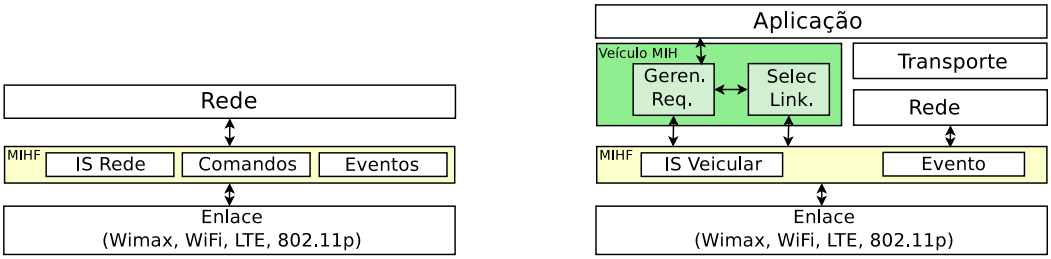
Essa política foi acrescentada em uma arquitetura previamente desenvolvida [Meneguette et al. 2013], que foi a base (indispensável) para desenvolvimento e avaliação desse trabalho. A Seção 4.1 apresenta uma descrição dessa arquitetura.

4.1. Arquitetura Utilizada

Essa política de seleção de fluxo foi incorporada na arquitetura de gerência de fluxo de mobilidade desenvolvida [Meneguette et al. 2012]. Essa arquitetura consiste de uma infraestrutura comum para tecnologia multi-acesso de forma transparente em redes sem fio, trabalhando com tecnologias como WiMax e LTE, além de tecnologias sem fio para redes veiculares, provendo uma conexão contínua e transparente para as aplicações veiculares.

O objetivo dessa arquitetura é maximizar a vazão da rede, mantendo a latência e a perda de pacotes dentro dos requisitos mínimos das aplicações veiculares. Para isso ela utiliza um gerenciador de fluxo baseado nas classes de aplicação das redes veiculares e no estado de cada rede ativa no ambiente.

A arquitetura utiliza o protocolo 802.21 para capturar o estado das redes ativas no ambiente e o protocolo PMIPv6 estendido. Além disso, a arquitetura está dividida em dois módulos: um módulo que está no MAG e no LMA (Figura 1(a)), e outro módulo que está embarcado no veículo (Figura 1(b)).



(a) Arquitetura embarcada no MAG e no LMA.

(b) Arquitetura veicular.

Figura 1. Componentes da arquitetura.

Na arquitetura do MAG e do LMA (Figura 1(a)), a camada de rede contém o protocolo PMIPv6 para lidar com o endereçamento e com os prefixos dos pontos de acesso requeridos para o roteamento das mensagens. Esses módulos também possuem o módulo MIHF, com funções estendidas do protocolo 802.21. Essa extensão faz com que o LMA tenha uma visão global do estado da rede que está conectada a ele, também permitindo que o MAG possa monitorar seus próprios enlaces, dando ao MAG uma visão do estado de sua rede local.

A arquitetura do veículo contém um módulo de gestão de mobilidade, chamado Veículo MIH, que compreende um módulo de gerenciamento de requisitos que recebe os requisitos mínimos de rede para que a aplicação possa ser executada. O Veículo MIH também tem um módulo de seleção de enlace, o qual recebe a informação do estado da rede e decide se vai efetuar uma transferência e, em caso afirmativo, a qual rede deve se conectar. Além disso, o módulo de seleção de enlace ajuda a interface lógica a decidir qual será o enlace usado para enviar uma mensagem específica. Essa decisão é feita através da lógica *fuzzy* proposta. Tanto o módulo de gerenciamento de requisitos quanto o módulo de seleção de enlace enviam comandos para o módulo MIHF. O módulo MIHF é uma extensão das funções do protocolo 802.21. Essa extensão permite a criação de um gerenciamento de fluxo com base nos requisitos e características de cada aplicação de redes veiculares, tais como o atraso e taxa de transferência, além de informar ao nó quais são as interfaces ativas em um determinado momento. Parte do detalhamento dessa arquitetura pode ser encontrado em [Meneguette et al. 2012].

4.2. Divisão do Fluxo de Informação

Agrupamos aplicações das redes veiculares em três classes de acordo com seus objetivos gerais: *segurança*, *conforto* e *usuário*. A classe de segurança compreende aplicações destinadas a ajudar motoristas a lidar com eventos imprevisíveis ou perigos das vias públicas através do monitoramento do tráfego próximo ao veículo através de mensagens [Singh et al. 2011]. A classe de conforto inclui aplicações que se concentram no conforto e na eficiência do fluxo de carros nas ruas e estradas. Em outras palavras, esses aplicativos aumentam o grau de conveniência de motoristas e eficiência do tráfego através da troca de informação de tráfego entre as infraestruturas de acostamento e veículos [Singh et al. 2011]. A classe de usuário é composta de aplicações que estão focadas na interação entre os ocupantes do veículo e informações, anúncios, entretenimento e vários tipos de serviço de comunicação [Meneguette et al. 2013].

Essa classificação das aplicações da rede veicular permite a divisão do fluxo de informação da rede em três fluxos distintos: o primeiro fluxo para a classe de segurança, o segundo fluxo para classe de conforto e o terceiro fluxo para a classe de usuários. Para diferenciar estes fluxos utilizamos uma 2-tupla, que consiste no protocolo usado para transmissão e a porta de destino. Em resumo, o fluxo é a agregação do tráfego das mensagens das aplicações de uma mesma classe.

4.3. Lógica *Fuzzy*

Embora o protocolo 802.21 tenha como objetivo otimizar o *handover* entre redes heterogêneas, esse protocolo não especifica um algoritmo de seleção de rede. Para resolver esse problema, desenvolvemos um mecanismo de troca de interface baseado nos requisitos mínimos de cada fluxo. Esse mecanismo utiliza lógica *fuzzy* para decidir qual interface atribuir a cada grupo e indicar se em um determinado momento algum fluxo deve mudar de interface.

Para o desenvolvimento desse mecanismo utilizamos uma técnica de número *fuzzy* triangular baseado no trabalho de Dhar e colaboradores [Sourav Dhar and Bera 2011]. A

estrutura do número *fuzzy* triangular é:

$$u_a = \begin{cases} \frac{x-1}{m-l} & l \leq x \leq m \\ \frac{u-x}{u-m} & m \leq x \leq u \\ 0 & \text{caso contrário} \end{cases}$$

sendo as variáveis l, m e u os limiares inferior, médio e superior de uma variável fuzzy. A definição acima resulta no gráfico triangular indicado na Figura 2(a), onde o eixo *y* representa a pertinência do valor fuzzy representado no eixo *x*.

Para a criação das regras *fuzzy*, utilizamos três variáveis *fuzzy*: vazão da rede, perda de pacotes e atraso das mensagens. Cada variável possui três conjuntos *fuzzy*: alta, normal e baixa:

- Vazão: alta de 30 até 100 Kbps; normal de 10 até 50 Kbps; baixa de 0 até 20 Kbps.
- Perda: alta de 15 até 30 pacotes por segundo; normal de 5 até 20 pacotes por segundo; baixa de 0 até 10 pacotes por segundo.
- Atraso: alta de 0.3 até 1s; normal de 0.06 até 0.6s; baixa de 0 até 0.09s.

Por exemplo, os números *fuzzy* triangulares para a variável *vazão* são representados como mostra a Figura 2(b).

(a) Número triangular Fuzzy.

(b) Números triangulares fuzzy para a variável vazão.

Figura 2. Representação de números triangulares Fuzzy.

A escolha desse conjunto fuzzy e seus valores baseou-se nos padrões de avaliação de aplicações em redes veiculares da ETSI [ETSI TR 102 638 2009] e também em experimentos realizados na arquitetura ([Meneguette et al. 2012] e [Meneguette et al. 2013]).

Ao contrário do trabalho [Sourav Dhar and Bera 2011], que utiliza três classificações de saída (baixo, médio e alto), nós propomos a utilização de cinco variáveis de saída: muito alto, alto, normal, baixo e muito baixo. Optamos por um número maior de classes de saída para evitar uma precipitação no momento de realizar uma troca de interface e também fornecer um indicativo que uma interface está muito distante de atender as necessidades de um fluxo de informação. Através das combinações das variáveis de entrada com seus respectivos conjuntos, obtivemos 27 regras *fuzzy* para cada fluxo. Cada regra possui uma classificação de saída para cada tecnologia de rede. A geração dessa classificação de saída foi realizada através do fator de influência que as variáveis têm umas sobre as outras no uso de uma determinada tecnologia de rede. Para o cálculo desse fator de influência nos baseamos em [Sourav Dhar and Bera 2011]. A Tabela 1 apresenta um excerto das regras resultantes para o fluxo de segurança.

Tabela 1. Regras do fluxo de segurança

	Entrada			Saída	
Regras	Vazão da rede	Perda de pacote	Atraso	WiFi	LTE
1	Alta	Alta	Alta	Alta	Muito Baixa
2	Alta	Alta	Normal	Alta	Muito Baixa
....	...	....	.....	.....	....
9	Alta	Baixa	Baixa	Alta	Normal
10	Normal	Alta	Alta	Normal	Muito Baixa
....	...	....	.....	.....	....
18	Normal	Baixa	Baixa	Normal	Muito Alta
19	Baixa	Alta	Alta	Muito Alta	Muito Baixa
....	...	....	.....	.....	....
26	Baixa	Baixa	Normal	Alta	Muito Alta
27	Baixa	Baixa	Baixa	Alta	Muito Alta

5. Resultados

Nesta seção apresentamos as políticas utilizadas na comparação da proposta, os cenários de simulação e as análises dos resultados obtidos.

5.1. Políticas utilizadas na comparação

Utilizamos 4 políticas diferentes em nossa avaliação:

- **Proposta**: Mecanismo proposto neste trabalho (Seção 4.3).
- **Limiar**: A decisão de mudar um fluxo de uma interface para outra é realizada através de um único limiar para cada variável, ou seja, um limiar para a perda de pacote, um limiar para vazão e um limiar para o atraso dos pacotes. Quando o nó percebe que pelo menos um desses valores está passando do seu limiar, o mecanismo realiza a troca do fluxo que foi detectado com excesso para outra interface que atenda aos requisitos da aplicação.
- **Lógica Fuzzy**:A decisão de realizar a mudança de um fluxo de uma interface para outra ocorre através de inferências *fuzzy*. Para a criação desse cenário nos baseamos no trabalho de Dhar e colaboradores [Sourav Dhar and Bera 2011] utilizando somente 3 classificações de saída. Nesse cenário também utilizamos os números triangulares *fuzzy*, além das faixas das variáveis *fuzzy* de baixo, meio e alto, para melhor adequar a saída.
- **Teoria dos jogos**:Em um jogo, cada participante possui um conjunto de estratégias que podem ser usadas. Quando cada jogador escolhe qual será a sua estratégia, obtemos um perfil, ou seja, um espaço que contém todas as possíveis situações que podem ocorrer. Cada jogador tem interesses ou preferências para cada situação no jogo. Em termos matemáticos, cada jogador tem uma *função utilidade* que atribui um número real (o ganho, ou *payoff*, do jogador) a cada situação do jogo.
  Mais especificamente, um jogo tem os seguintes elementos básicos: um conjunto finito de jogadores, representado por $G = \{g_1, g_2, ..., g_n\}$, onde cada jogador $g_i \in G$ possui um conjunto finito $S_i = \{s_{i1}, s_{i2}, ..., s_{im_i}\}$ de opções, denominadas estratégias do jogador $g_i (m_i \geq 2)$, onde, portanto, o jogador g_i possui m_i estratégias. Um vetor $s = (s_{1j_1}, s_{2j_2}, ..., s_{nj_n})$, onde s_{ij_i} é uma estratégia para o

jogador $g_i \in G$, é denominado um perfil de estratégia. O conjunto de todos os perfis de estratégia pura forma o produto cartesiano:

$$\prod_{i=1}^{n} S_i = S_1 * S_2 * .... * S_n,$$

denominado espaço de estratégia do jogo. Para jogadores $g_i \in G$, existe uma função utilidade

$$u_i : S \rightarrow R$$

$$s_i \rightarrow u_1(s)$$

que associa o ganho (*payoff*) $u_i(s)$ do jogador g_1 a cada perfil de estratégia $s \in S$. A política de comparação que utiliza teoria de jogos é baseada no dilema do prisioneiro. Temos 3 jogadores (os 3 fluxos de dados: segurança, conforto e usuário) e temos duas estratégias, ou seja, cada estratégia seleciona uma das interfaces de rede que o usuário possui. Assumimos que um fluxo pode selecionar apenas uma interface a cada rodada. Assim temos: $G = \{g_1, g_2, g_3\}; S_1 = \{s_{11}; s_{12}\}; S_2 = \{s_{21}; s_{22}\}$; $S_3 = \{s_{31}; s_{32}\}$. A cada estratégia do jogador é atribuído um *payoff*. Para o cálculo do *payoff*, utilizamos a função de satisfação considerando os requisitos mínimos da rede, ou seja, a porcentagem que uma determinada tecnologia de rede está atendendo num determinado fluxo.

$$Payoff = \sum_{i=1}^{3} w_i * \text{parâmetro}$$

O *payoff* de cada estratégia consiste na média normalizada dos parâmetros da rede tais como atraso, vazão e perda de pacotes. Os pesos $w_i, 1 \leq i \leq 3$ variam no que diz respeito aos parâmetros e são calculados com base no desempenho da rede, utilizando o trabalho [Charilas et al. 2008].

5.2. Cenários de simulação e análise dos resultados

As políticas foram implementadas no Network Simulator (NS-3.13). Utilizamos o modelo PMIPv6 que foi implementado por Hyon-Young Choi [Choi et al. 2010], e também o modelo 802.21 [Salumu 2012]. A finalidade das simulações foi verificar o impacto que o nosso mecanismo de escolha causaria na rede e nas aplicações. Com isso, pretendemos verificar se o mecanismo está realizando um bom mapeamento dos fluxos de informação das classes de aplicativos para as interfaces de rede sem sobrecarregar nenhuma rede. Para isso, nós utilizamos cinco métricas para avaliar o nosso mecanismo: tempo de *handover*, perda de pacotes, atraso, atraso por classe de aplicação e vazão.

Em nosso cenário de simulação, cada veículo (ou seus ocupantes) está executando uma aplicação de cada classe, ou seja, uma aplicação da classe de segurança, uma aplicação da classe de conforto e uma aplicação da classe de usuário. A freqüência de envio de mensagens de cada aplicação é baseada no padrão ETSI (European Telecommunication Standardization Institute) [ETSI TR 102 638 2009], onde a aplicação da classe de segurança envia uma mensagem a cada 0.1s, a aplicação da classe de usuário envia uma mensagem a cada 1s, e da aplicação da classe de conforto envia uma mensagem a cada 0.5s.

Conduzimos as simulações com 50 veículos que transitavam no mapa, dentre os quais selecionamos uma série de veículos para enviar e receber mensagens dos 3 tipos diferentes de aplicação. Variamos o número de veículos que estavam executando as três classes de aplicação ao mesmo tempo no conjunto $\{10, 20, 30, 40, 50\}$. Enquanto tais veículos enviavam e recebiam informações, os outros veículos apenas trafegavam nas vias sem enviar e receber mensagens.

A topologia de rede era constituída de um nó cabeado, cinco nós de backbone, um ponto de acesso LTE e três pontos de acesso 802.11p, como podemos ver na Figura 3. Todos os veículos estavam dentro da área de cobertura do LTE, porém a tecnologia WiFi não cobria todo o mapa. Para a simulação das redes 802.11p e LTE foram utilizados módulos padrões do ns3 (tal como wifi.SetStandard(WiFi_PHY_STANDARD_80211p_SCH) e WifiMacHelper::Default()). Foram realizadas 30 simulações para cada cenário para as quais calculamos intervalos de confiança de 95%.

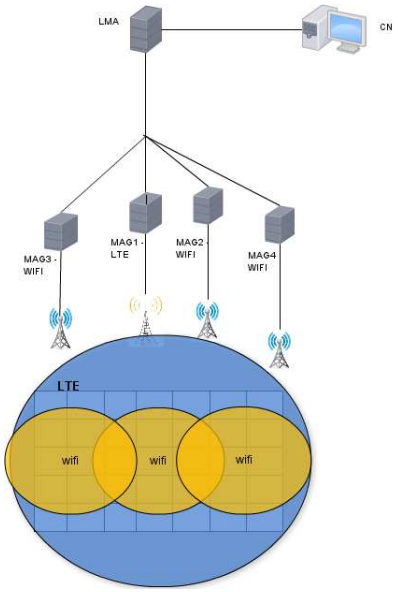
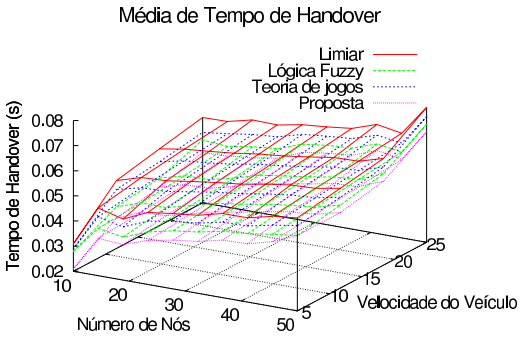


Figura 3. Topologia da simulação.

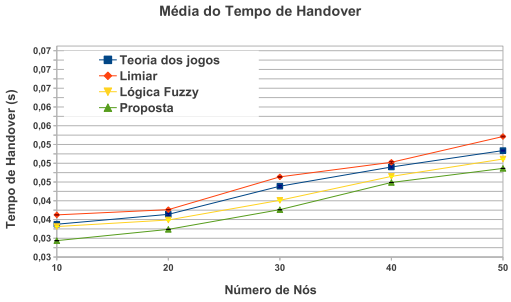
Foram utilizados dois mapas para modelar a mobilidade dos veículos. O primeiro é o modelo de mobilidade Manhattan e o segundo é um trecho de um mapa real. Para criar o modelo de mobilidade Manhattan usamos o *bonmotion*, ferramenta que criou uma grade com quatro linhas e seis colunas. Neste mapa variamos a velocidade dos veículos, entre 5, 10, 15, 20 e 25 m/s, a fim de verificar se há qualquer impacto sobre o desempenho do mecanismo proposto quando os veículos possuem diferentes velocidades. Como segundo mapa utilizamos um bairro na cidade de Campinas, no estado de São Paulo, onde o simulador de mobilidade urbana (SUMO) foi usado para converter o mapa extraído do OpenStreetMap para um formato compatível com o simulador SUMO.

Comparamos as quatro políticas descritas na Seção 5.1 (limiar, lógica fuzzy, teoria de jogos, e a nossa proposta). A Figura 4 apresenta a média dos tempos de *handover*. Analisando a Figura 4(a) podemos observar que quanto maior a velocidade do veículo maior será o tempo de *handover*. Isso ocorre devido ao aumento do número de ocorrências de *handover*. Com velocidades entre 10m/s até 25m/s todos os cenários tiveram um au-

mento significativo em seu tempo médio de *handover*. A política proposta evitou uma precipitação em trocar a interface de um determinado fluxo para outra interface, apresentando melhor desempenho. Analisando o mapa real com 30 veículos, a política *limiar* teve 32 *handovers* a mais que a política proposta. Já a política da *teoria dos jogos* teve 26 *handovers* a mais que a proposta, enquanto a política *fuzzy* fez 10 *handovers* a mais que a proposta. Comparando o mapa real com o Manhatan com 30 nós e com uma velocidade de 15 m/sm a política proposta teve uma redução de 20% no tempo de *handover* comparado com o mapa Manhattan, com a proposta mantendo um desempenho melhor que as outras políticas.



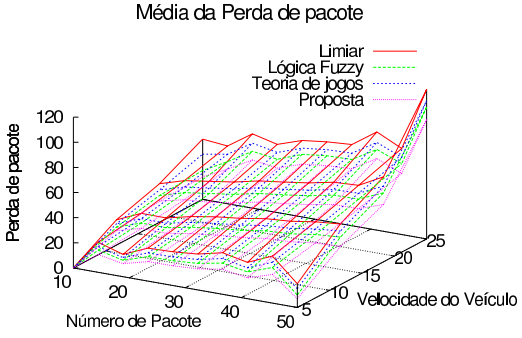
(a) Mobilidade mapa Manhattan.



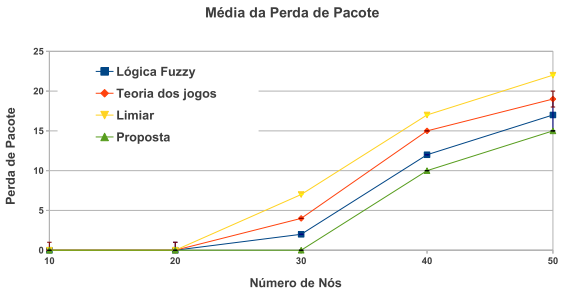
(b) Mobilidade mapa real.

Figura 4. Média do Tempo de Handover

Podemos observar na Figura 5 que a política proposta teve uma menor perda de pacotes para ambos os mapas, devido a um baixo número de *handovers* e também por melhor distribuir os fluxos entre as interfaces. A política proposta obteve uma redução total de 18% a 38% comparada às outras políticas. Considerando 50 veículos e uma velocidade de 15 m/s, a política proposta obteve uma redução de 25% nos cenários no mapa do Manhattan e uma redução de 22% no mapa real. As taxas de perda de pacote não tiveram um impacto relevante na vazão da rede como mostra a Figura 6. Para o mapa Manhattan também não observamos impacto na vazão da rede.



(a) Mobilidade mapa Manhattan



(b) Mobilidade mapa real.

Figura 5. Média da perda de pacote

A Figura 7 apresenta a média de atraso de todas as aplicações. Na Figura 7(a) observamos que a política proposta forneceu uma redução de 6% no atraso de envio da mensagem comparado com a política *fuzzy* e uma redução de 11% comparada com a

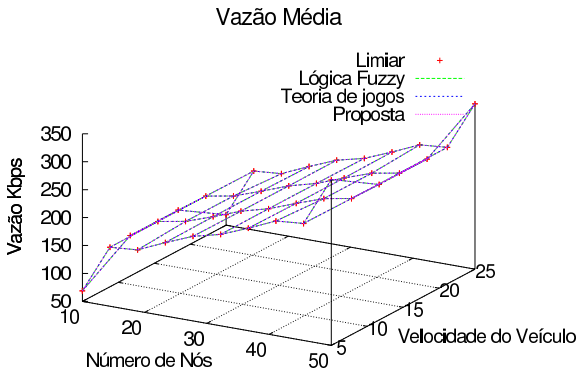
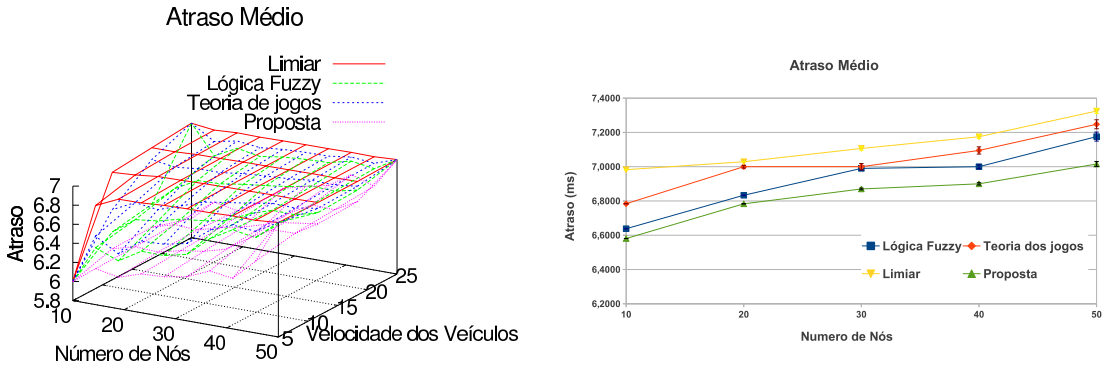


Figura 6. Vazão da rede no mapa do Manhattan em Kbps.

política *limiar* quando analisamos o gráfico com 10 veículos e com uma velocidade de 10 m/s. Na Figura 7(b), quando há 20 carros participando, a política proposta reduziu o tempo de atraso em 5% comparado à política *limiar*. Além disso, a proposta ofereceu uma redução de 67% sobre todos os cenários no gráfico 7(b). A política proposta manteve todos os atrasos por aplicação dentro de um tempo aceitável, como podemos ver na Figura 5.2. As simulações mostraram que tanto para o mapa real como para o Manhattan o atraso de todas as aplicações ficaram abaixo do padrão estabelecido pelo ETSI.



(a) Mobilidade mapa Manhattan. (b) Mobilidade mapa real

Figura 7. Média do Atraso

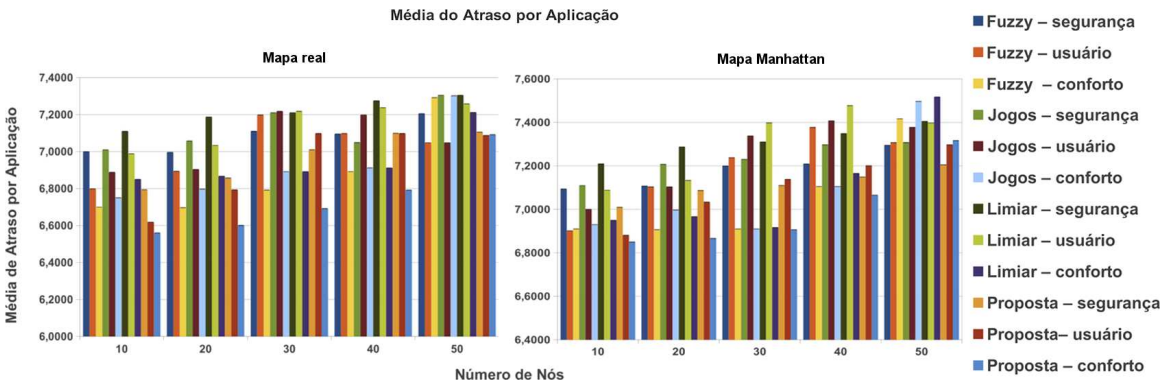


Figura 8. Média do Atraso por Aplicação.

Os resultados sugerem que a política proposta possui um bom desempenho em redes veiculares. A política proposta conseguiu uma melhoria média de 34% na perda de pacotes e de 2.8% em atraso no mapa real, enquanto para o mapa Manhattan esses números atingiram 27% e 4.95%, respectivamente. A política proposta apresentou uma baixa perda de pacote, mantendo uma boa vazão na rede e também um baixo atraso na entrega dos pacotes, não excedendo os padrões estabelecidos pelo ETSI.

6. Conclusão

Este trabalho explora políticas de decisão em redes veiculares para tratar com mudanças de fluxo de uma interface para outra num nó, considerando os requisitos de cada fluxo. A política proposta lida com o controle das interfaces de redes, buscando maximizar a vazão da rede, diminuir o tempo de troca de um fluxo de uma interface para a outra e satisfazer requisitos mínimos de perda de pacotes e atraso para cada classe de aplicação da rede veicular. Simulações mostraram que a política proposta apresentou um melhor desempenho comparado com outras políticas da literatura. O uso de mais de três classificações para a saída *fuzzy* permitiu uma melhor decisão no momento da escolha da troca do fluxo, evitando uma troca precipitada de interface e diminuindo a quantidade de *handovers* realizados na rede. Além da diminuição do tempo de *handover* a política proposta obteve uma baixa perda de pacotes com um baixo atraso quando há um grande número de participantes. Nenhum atraso das aplicações excedeu o tempo padrão estabelecido pelo ETSI.

Trabalhos futuros incluem levar consideração não somente o envio de pacote entre o veículo e a infraestrutura de acostamento, mas também entre os veículos.

Agradecimentos

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

Referências

Bernardos, C. J. (2012). Proxy mobile ipv6 extensions to support flow mobility. draft-ietf-netext-pmipv6-flowmob-03.

Bernardos, C. J., Calderon, M., and Soto, I. (2012). PMIPv6 and network mobility problem statement. draft-bernardos-netext-pmipv6-nemo-ps-02.

Blanchet, M. and Seite, P. (2011). Multiple interfaces and provisioning domains problem statement. IETF RFC 6418.

Charilas, D., Markaki, O., Nikitopoulos, D., and Theologou, M. (2008). Packet-switched network selection with the highest qos in 4g networks. *Computer Networks*, 52(1):248 – 258.

Choi, H.-Y., Min, S.-G., Han, Y.-H., Park, J., and Kim, H. (2010). Implementation and evaluation of proxy mobile IPv6 in NS-3 network simulator. In *5th Intl. Conference on Ubiquitous Information Technologies and Applications*, pages 1 –6.

Dhar, S., Ray, A., and Bera, R. (2012). Cognitive vertical handover engine for vehicular communication. *Peer-to-Peer Networking and Applications*, pages 1–20.

Dutta, A., Das, S., Famolari, D., Ohba, Y., Taniuchi, K., Fajardo, V., Lopez, R. M., Kodama, T., and Schulzrinne, H. (2007). Seamless proactive handover across heterogeneous access networks. *Wirel. Pers. Commun.*, 43:837–855.

ETSI TR 102 638 (2009). Intelligent transport systems (its); vehicular communications; basic set of applications; definitions.

Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K., and Patil, B. (2008). Proxy mobile IPv6. http://tools.ietf.org/html/rfc5213.

Ma, B. and Liao, X. (May). Speed-adaptive vertical handoff algorithm based on fuzzy logic in vehicular heterogeneous networks. In *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 371–375.

Makaya, C., Das, S., and Lin, F. (2012). Seamless data offload and flow mobility in vehicular communications networks. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 338 –343.

Marquez-Barja, J., Calafate, C. T., Cano, J.-C., and Manzoni, P. (2011). An overview of vertical handover techniques: Algorithms, protocols and tools. *Computer Communications*, 34(8):985 – 997.

Meneguette, R. and Madeira, E. (2011). An architecture for mobility management in vehicular networks with support to collaborative virtual environments. In *Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2011)*.

Meneguette, R. I., Bittencourt, L. F., and Madeira, E. R. M. (2012). User-centric mobility management architecture for vehicular networks. In *4th International Conference on Mobile Networks and Management (MONAMI 2012)*.

Meneguette, R. I., Bittencourt, L. F., and Madeira, E. R. M. (2013). A seamless flow mobility management architecture in vehicular communication networks. *Journal of Communications and Networks (Aceito para publicação)*.

Patil, C. G. and Kolte, M. T. (2011). An approach for optimization of handoff algorithm using fuzzy logic system. *J. of Comp. Science and Communication*, 2(1):113–118.

Salumu, M. (2012). ns3 - 802.21. Available at: http://code.nsnam.org/salumu/ns-3-mih/. Accessed in 2012.

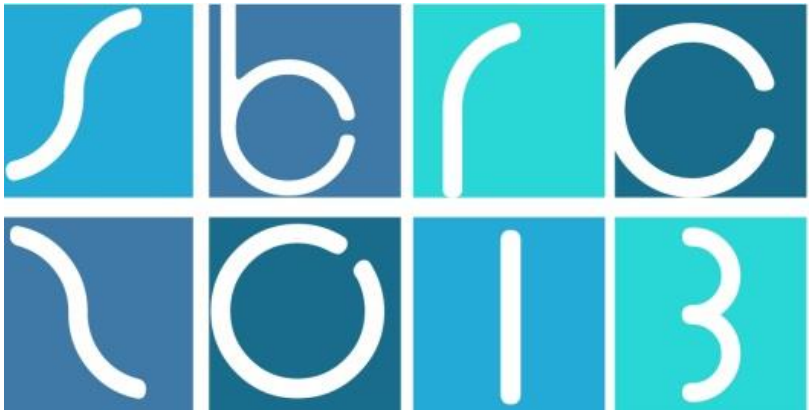
Singh, A., Kumar, M., Rishi, R., and Madan, D. K. (2011). A relative study of manet and vanet: Its applications, broadcasting approaches and challenging issues. In *Advances in Networks and Communications*, volume 132, pages 627–632. Springer Berlin Heidelberg.

Sourav Dhar, Amitava Ray, S. C. and Bera, R. N. (2011). Intelligent vertical handover scheme for utopian transport scenario. *Academic Journals Inc*.

Tsirtsis, G., Soliman, H., Montavont, N., Giaretta, G., and Kuladinithi, K. (2011). Flow bindings in mobile IPv6 and network mobility (nemo) basic support. IETF RFC 6089.

Uesu, H., Nagashima, K., Chung, H., and Tsuda, E. (2011). Relational structure analysis of fuzzy graph and its application: For analyzing fuzzy data of human relation. In *2011 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1593 –1597.

Wasserman, M. and Seite, P. (2011). Current practices for multiple-interface hosts. IETF RFC 6419.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Salão de Ferramentas

Sessão Técnica 1

Computação Móvel e Mobilidade

extMobilisTTS: Uma Arquitetura de Aplicação Móvel para Suporte a Fóruns usando Text-to-Speech em Ambientes Virtuais de Aprendizagem

Wellington W. F. Sarmento[1], Gabriel A. L. Paillard[1], Wedson de S. Lima[1], Katryne F. Rabelo[1], Andrei B. B. Torres[1], Humberto O. O. Gomes[1], Paulo M. B. Costa[1], Mauro C. Pequeno[1]

[1]Instituto Universidade Virtual – Universidade Federal do Ceará (UFC)
CEP 60440-554 – Fortaleza – CE – Brazil

{extmobilistts} @virtual.ufc.br

Abstract. *This paper presents an architecture for communication between mobile devices and a Virtual Learning Environment (VLE) called extMobilisTTS, and a client application (Solar Mobilis) which uses the hardware features of the device and the usage of external services, such as a text-to-Speech (TTS) module, favoring the interaction between teachers and students in discussion forums of the VLE.*

Resumo. *Este trabalho apresenta uma arquitetura de comunicação entre dispositivos móveis e um Ambiente Virtual de Aprendizagem (AVA), chamada extMobilisTTS, bem como uma aplicação cliente (Solar Mobilis) que se utiliza de recursos do* hardware *do dispositivo e de serviços externos à aplicação, como um módulo Text-to-Speech (TTS) para conversão de texto para voz, favorecendo a interação entre professores e alunos em fóruns de discussões de um AVA.*

1. Introdução

As atuais Redes Telemáticas modificaram o ritmo de vida das pessoas e levaram à necessidade cada vez maior de acesso à informação mesmo estando fora de suas instituições de trabalho ou estudo, ou ainda, em trânsito. Hoje, existem diversos tipos de dispositivos móveis que possibilitam a comunicação e a execução de aplicações computacionais. Assim como aconteceu com outras tecnologias, como é o caso dos computadores, os dispositivos móveis também estão influenciando a maneira como se faz educação, desta forma disponibilizando novas ferramentas que possibilitam o processo de aprendizagem, tanto no ensino presencial quanto à distância.

A utilização de dispositivos móveis para o processo de aprendizagem é chamada *Mobile Learning* (m-Learning) [Trifonova 2003][Lee et al. 2005]. É neste cenário de *m-Learning* que se insere a proposta de integração de um AVA, em nosso estudo de caso, o Solar versão 2.0, e de aplicações móveis de suporte à Educação a Distância. A arquitetura proposta neste trabalho é uma extensão do Mobilis, apresentado em [Sarmento 2007]. A arquitetura Mobilis descreve como um conjunto de aplicações móveis pode acessar um AVA, estendendo suas funcionalidades tradicionais e acrescentando funções adaptadas à realidade dos dispositivos móveis, mantendo o foco do uso de acordo com as potencialidades destes dispositivos. O que se propõe é um uso específico dos recursos essenciais de

um dispositivo móvel, tais como microfone, conexão com internet, reprodução de vídeo, áudio e *touchscreen* para acessar informações e interagir com um AVA.

A extensão do Mobilis explora o uso de tecnologias atuais como as plataformas móveis Android e iOS. Desta forma, alunos e professores terão formas alternativas de interagir com fóruns de AVA, utilizando o recurso de áudio, por exemplo. Como primeira aplicação a compor esta arquitetura, foi criado o Solar Mobilis, para acesso a fóruns através de áudio. Uma primeira versão desta aplicação foi apresentada no artigo [Paillard et al. 2012] e sua evolução continuou até o presente trabalho.

A presente pesquisa apresenta a arquitetura estendida do Mobilis - Extended Mobilis TTS ou extMobilisTTS -, que utiliza como base os recursos de reprodução de textos via *Text-to-Speech* (TTS - texto para fala) e de gravação de voz, bem como um estudo de caso feito através da aplicação Solar Mobilis de comunicação com o AVA Solar 2.0. A estrutura deste trabalho é composta pela seção 2, que apresentará os trabalhos relacionados ao tema abordado; seção 3, que expõe de forma mais detalhada o ambiente extMobilisTTS; na seção 4 será apresentado o estudo de caso com o SOLAR versão 2.0 e, por fim, as conclusões do estágio atual desta pesquisa.

2. Trabalhos Relacionados

Com a chegada dos *smartphones*, surgiu a necessidade de adequar o conteúdo dos AVA às novas formas de comunicação móvel. Antes, tal comunicação era feita unicamente pelo navegador do celular. Hoje, é possível que seja realizada por meio de aplicações específicas que possibilitam o aproveitamento total dos recursos do dispositivo.

Existem trabalhos que começam a explorar as funcionalidades dos dispositivos móveis. Em [Al Tunaiji and Zemerly 2010] é apresentada a proposta de uma aplicação chamada *M-learning and University Student Organizer* (MUSO) que permite ao estudante verificar atualizações referentes a sua vida acadêmica como notas, faltas, data de provas e notícias da universidade. Já em [Blackboard 2011] é descrita uma aplicação voltada para *smartphones* contendo notícias, blogs, notas, visualização de vídeo e áudio, *roster* e discussão. O sistema [Mobl21 2011] possui mais funcionalidades voltadas para *m-Learning* e contém ferramentas de avaliação como *Quizzes*, *Flashcards* e guias de estudo tanto *online* quanto *offline*, abrangendo, além dos *smartphones*, os *tablets*. Tais aplicações normalmente fazem uso de meios textuais para a comunicação entre os AVA e as aplicações móveis. Neste caso, recursos multimídia dos *smartphones* são relegados a segundo plano. É neste contexto que a extMobilisTTS se insere.

3. Descrição da Arquitetura

3.1. Visão Geral do Sistema

Esta seção mostra a arquitetura geral do extMobilisTTS, bem como a aplicação criada a partir desta, o Solar Mobilis. O código fonte da aplicação cliente e servidor, bem como o executável da aplicação cliente e a documentação, encontram-se no servidor `http://www.github.com/wwagner33/extmobilis`.

3.1.1. O que o Sistema faz

O Mobilis, especificado por [Sarmento 2007], define uma arquitetura de integração entre AVA e dispositivos móveis através do uso de *web services*, utilizando para a construção destes tecnologias como *Simple Object Access Protocol* (SOAP), .NET e/ou JAVA.

Para uma maior facilidade de integração entre as aplicações utilizadas e uma maior qualidade na troca de dados entre o dispositivo móvel e o *web service*, propomos utilizar uma nova tecnologia para a criação deste último. Assim, está-se propondo uma arquitetura que estende o Mobilis, chamada extMobilisTTS.

O extMobilisTTS é uma arquitetura Cliente-Servidor na qual a aplicação servidora é um AVA que fornece acesso a seu conteúdo por meio de uma API (*Application Programming Interface*) de acesso.

A aplicação cliente se utiliza dos recursos nativos das plataformas em que foram criadas para aprimorar a forma na qual um usuário pode interagir com o AVA (Fóruns, Avisos, Mensagens etc.). Está previsto, também, nessa arquitetura, o uso de API externas que fornecem serviços de terceiros, que não se comunicam diretamente com o AVA e são acessadas pelo lado do cliente, como o acesso a um módulo TTS, por exemplo.

3.2. Descrição do Cliente

Foi desenvolvido um sistema cliente chamado Solar Mobilis para acessar os fóruns do AVA Solar 2.0, a fim de validar o funcionamento da arquitetura. Para tanto, foram realizados testes para verificar a viabilidade de uso em diversos tipos de dispositivos e conexões. Estes testes são detalhados na seção 4.2.

As principais características da aplicação são: recuperar as novas mensagens dos fóruns para leitura; responder com texto, com a possibilidade de anexar arquivos de áudio gravado pela própria aplicação; e ler as mensagens recebidas para o usuário a partir de um módulo TTS (*text-to-speech*).

Os dados (postagens de usuários e informações de cursos) são recebidos no formato *JavaScript Object Notation* (JSON) e decodificados para serem armazenados no dispositivo móvel usando SQLite.

Para o envio de dados, estes são convertidos para objetos JSON e enviados para o servidor Solar 2.0 através de chamadas a API.

O módulo TTS faz uso da API *Speak* do Microsoft Bing que recebe como parâmetro um *array* de caracteres e retorna um arquivo de áudio que contém o texto falado. Esse arquivo é salvo temporariamente no dispositivo durante a sua reprodução. As chamadas são feitas na forma de blocos onde uma postagem é dividida em várias chamadas para que o usuário não espere muito tempo até que o áudio inicie.

3.3. Descrição da Comunicação com o Servidor

O AVA Solar 2.0, construído com o *framework Ruby On Rails* e utilizando as rotas baseadas no REST (definido em [Fielding 2000]), disponibiliza acesso a seus fóruns de discussão através de uma API (documentação disponível em `https://github.com/wwagner33/solar/wiki/API-Forum`). Isso facilita a comunicação com

aplicações externas e diminui a quantidade de código escrito, tornando-o mais legível e facilitando sua manutenção.

Após uma etapa obrigatória de *login*, utilizando os mesmos dados de usuário e senha do AVA, o aplicativo recebe um *token – hash* aleatório para ser utilizado nas próximas requisições. Esse *hash* identifica o usuário no sistema e é renovado a cada nova etapa de *login*. Requisições ao AVA, pela aplicação móvel, não são aceitas sem este *hash*.

Na troca de informações entre a aplicação móvel e o Solar 2.0, são utilizados objetos JSON no formato `{chave:valor}`.

Essa forma de comunicação permite uma otimização na quantidade de caracteres trocados entre a aplicação móvel e o servidor, pois assim não temos um *overhead* de cabeçalhos ou caracteres desnecessários, como as *tags* em uma aplicação que utiliza *eXtensible Markup Language* (XML) e o tempo de desserialização é menor [Hameseder et al. 2011].

Utilizando o *Ruby On Rails* com o que é disponibilizado por esta plataforma de arquitetura REST e considerando os métodos HTTP, a aplicação servidora é capaz de mapear a requisição e redirecionar para a classe e método adequados. Desta forma, podemos manter a aplicação com código reduzido e mais simples, tornando fácil a manutenção e criação de novas funcionalidades, o que agiliza o processo de desenvolvimento.

3.4. Interface e Usabilidade

O modelo de *design* do aplicativo móvel do sistema Solar Mobilis foi focado nas *guidelines* do Android 4.0, plataforma da Google [Google 2011]. As *guidelines* são guias atualizadas periodicamente com regras de bom uso de desenvolvimento e *design* de interfaces para a plataforma, de modo a padronizar alguns itens e tornar os aplicativos fáceis de se utilizar e ao mesmo tempo esteticamente agradáveis.

A interface do Solar Mobilis tem como objetivo principal, do ponto de vista pedagógico, tornar o uso da aplicação fácil e objetiva para os alunos do AVA Solar 2.0. É importante que eles consigam acessar uma versão móvel e simplificada do fórum que não provoque muitas distrações, visto que o foco é o conteúdo e a aprendizagem.

Partindo destas necessidades, dividimos a interface em 3 áreas distintas: barra de ação principal: exibe o título da tela atual e botões de ações relevantes (ex.: Responder a Todos); área de título: uma região fixa que identifica o título do fórum e o período de postagem; conteúdo principal (postagens): exibe os posts dos usuários, identificados por avatares personalizáveis, *nickname* e data de envio. O conteúdo da postagem é limitado a exibir as 5 primeiras linhas.

O objetivo da tela é facilitar a visualização e participação do usuário no fórum. Na barra de ação principal, o único botão disponível diretamente é o de responder ao fórum, que será a função mais utilizada pelos usuários e que é o foco da atividade.

Ao tocar sobre uma postagem, uma barra de ação contextual é exibida no lugar da barra de ação principal (ver Figura 1), e nela são exibidas funções específicas à postagem selecionada: expandir (caso ela tenha sido truncada, por passar do limite de 5 linhas), reproduzir via TTS, marcar como favorita, responder e ver mais informação.

A ação de gravação de uma postagem foi desenvolvida de modo a ficar claro para

Figura 1. Tela padrão de postagens (esquerda) e com barra contextual (direita)

o usuário que ele pode enviar uma mensagem textual e/ou anexar a gravação desta mensagem. Por isso, o ícone do áudio está claramente inserido na mensagem, podendo o usuário, ao clicar nesse ícone, editar ou excluir a gravação.

4. Estudo de Caso

4.1. Solar Mobilis

O aplicativo cliente foi desenvolvido utilizando um conjunto de ferramentas e bibliotecas que serão descritas a seguir.

Como ambiente de desenvolvimento foi utilizado o Eclipse IDE versão 3.7. Para que o IDE esteja apto para desenvolvimento de aplicativos para Android, é necessário instalar o *Android Development Tools* (ADT), *plugin* para Eclipse IDE fornecido pelo Google, que habilita o Eclipse para gerenciar projetos que utilizam o Android SDK e gere os arquivos binários para o sistema Android [Google 2012a].

O *Android Software Development Kit* (SDK) fornece as bibliotecas de API e as ferramentas de desenvolvimento necessárias para construir, testar e depurar aplicativos para Android [Google 2012b].

A *Microsoft Translator API* fornece uma série de serviços que podem ser acessados por diversos meios [Microsoft 2012]. No caso do Solar Mobilis, utilizamos o serviço de TTS para realizar a leitura de cada uma das postagens de um fórum para o usuário, de forma que o mesmo possa executar outras tarefas enquanto toma conhecimento do que foi discutido.

Foram utilizadas bibliotecas adicionais para facilitar certos aspectos da implementação: *JSON.simple* foi utilizada na versão 1.1.1 [Fang 2012]; *Lightweight Object Relational Mapping* (ORM) *Java Package* (OrmLite) na versão 4.4.1 [Watson 2012]; Apache Commons IO 2.1 [Apache 2012b]; HttpMime 4.1.3 [Apache 2012a]; ActionBarSherlock 4.1.0 [Wharton 2012].

4.2. Testes e Análise de Resultados

Testes preliminares foram realizados com o objetivo de verificar o desempenho do aplicativo em situações próximas as de uso diário. Tais testes foram realizados pela própria equipe de desenvolvimento e alguns colaboradores externos, a fim de que se pudesse ter

Duração total da leitura	903s
Tempo de espera total	46,047s
Tempo de Espera Médio por postagens	1,535s

Tabela 1. Teste de leitura de 30 postagens usando TTS e rede Wi-Fi (10 mbps)

Duração total da leitura	965s
Tempo de espera total	99,844s
Tempo de Espera Médio por postagens	3,328s

Tabela 2. Teste alternando uso de TTS em rede HSPA, GPRS e EDGE

uma noção sobre a conectividade do sistema, a estabilidade da informação frente a um ambiente real de uso de *smartphones* e o comportamento da interface com o usuário.

4.2.1. Teste de Conexão

Para aferir a conexão entre a aplicação Solar Mobilis e o servidor de AVA, foram realizados testes de comunicação usando diferentes tecnologias de celular e redes móveis locais.

Os testes foram realizados em um fórum contendo 30 postagens, com uma média de 300 caracteres. Esta seria a carga a ser recebida pelo usuário do AVA através do Solar Mobilis. Características dos testes: os testes foram realizados em redes Wi-Fi, HSPA, GPRS e EDGE e foram utilizados três parâmetros. A Duração Total de Leitura é o tempo que o aplicativo Solar Mobilis leva para realizar sua comunicação com o servidor de TTS, transformar todas as mensagens (postagem) em áudio e vocalizá-las para o usuário. O Tempo de Espera Total é o tempo que o sistema levou para iniciar todas as 30 postagens para o usuário (*delay* total de leitura). Por fim, a Tempo de Espera Médio por postagens é o *delay* que uma postagem leva para que seja iniciada sua vocalização.

O que se pode notar nas tabelas 1 e 2 é que o *delay* de início da leitura da mensagem é muito menor (54%) em redes Wi-Fi (10 mbps) do que com as tecnologias de comunicação por pacotes em celulares. Este resultado era esperado devido a diferença da taxa de transferência das tecnologias usadas. No entanto, ao receber o arquivo de áudio e começar a reproduzi-lo para o usuário, o tempo total de leitura é próximo (6,4%), pois depende somente do codificador de áudio do próprio celular e variações do processamento.

4.2.2. Teste em Ambiente Móvel

Teste de acesso ao fórum para leitura e ditado (gravação) feito em um automóvel com mudança de células de telefonia (percorrendo uma distância de aproximadamente 5 quilômetros). Foram realizadas 10 postagens em um fórum e leitura de outras 10. Este teste levou em consideração a satisfação do usuário e sua percepção do texto ditado.

Tivemos como resultado:

- *3G*: A leitura foi feita de maneira fluida, sem apresentar qualquer tipo de interrupção, mesmo transitando entre as células.

- *2,5G*: Verificou-se um pequeno atraso entre a leitura do cabeçalho da postagem, que informa o nome do aluno e o horário de envio, e o conteúdo. O mesmo atraso foi percebido entre o final da leitura de uma postagem e a seguinte.

Novamente, pode-se perceber que a modificação na tecnologia de transmissão de pacotes usadas no 3G (HSPA) e no 2,5G (EDGE) influencia a usabilidade da aplicação.

4.2.3. Adequação da Interface em Diferentes Resoluções

A plataforma Android foi criada para ser utilizada em uma ampla gama de dispositivos e formatos. Portanto, os aplicativos devem ser flexíveis e escalonáveis de maneira a se ajustar em tais dispositivos e manter um *layout* de fácil uso e agradável ao interagente.

O objetivo deste teste foi verificar como o aplicativo Solar Mobilis se comporta em 3 resoluções de tela com diferentes densidades, representadas em termo de DPI (*dots per inch* - pontos por polegada), especificadas pelo Google como LDPI, MDPI e HDPI, nos seguintes dispositivos: Sony Xperia X10 Mini: 240x320 *pixels*; LG Optimus One P500: 320x480 *pixels*; Motorola Droid 2: 480x854 *pixels*.

Após utilizar o aplicativo nos três dispositivos, verificou-se que ao seguir as recomendações de espaçamento de elementos, *grid* (disposição dos elementos em áreas proporcionais da tela) e tamanhos de fonte do *guideline* [Google 2011] provido pelo Google, todos os elementos mantiveram uma boa legibilidade e que os elementos da interface foram de fácil interação, independentemente da resolução da tela.

5. Conclusões

O objetivo principal da arquitetura extMobilisTTS é a utilização dos recursos dos celulares e *smartphones* em todo seu potencial para auxiliar o acesso a ferramentas de comunicação de AVA. Esta arquitetura foi desenvolvida na forma da aplicação Solar Mobilis que, em um primeiro momento, contará somente com acesso a fóruns, porém, pretende-se incluir outros mecanismos de comunicação que sejam favorecidos pelo uso dos dispositivos móveis (vídeo, gravação de imagens etc.). Assim, com os testes realizados pode-se verificar que a aplicação Solar Mobilis se comporta de forma satisfatória em seus primeiros testes relacionados a conectividade, satisfação do usuário e adequação da interface gráfica. Um ponto importante é que se utilizou a técnica de TTS baseada em servidor remoto, o que diminuiu o gasto de processamento dos equipamentos móveis utilizados, embora tenha refletido no aumento do uso da banda de comunicação.

Como trabalhos futuros, pretende-se portar a aplicação para Apple iOS e Microsoft Windows Phone, abrangendo assim os principais sistemas móveis em uso. Também almeja-se expandir a aplicação permitindo o acesso às outras funcionalidades de comunicação do AVA Solar 2.0.

6. Demonstração

A aplicação será apresentada no evento da SBRC 2013 através do uso do Solar Mobilis em tempo real e demonstradas suas funcionalidades quanto ao envio de mensagens gravadas e vocalização de textos postados em fórum do AVA. A tecnologia de comunicação usada será o HSPA ou Rede Wi-Fi disponível no evento. Para melhor apreciação da platéia, o processo será transmitido através de câmera de vídeo para um projetor multimídia.

7. Agradecimentos

À Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico.

Referências

Al Tunaiji, A. and Zemerly, M. (2010). Muso: An m-learning amp; university student organizer platform. In *Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on*, pages 155 –160.

Apache (2012a). Apache commons HttpMime. http://hc.apache.org/httpcomponents-client-ga/httpmime/.

Apache (2012b). Apache commons IO. http://commons.apache.org/io/.

Blackboard (2011). Blackboard móbile - aplicação para android. http://www.blackboard.com/platforms/mobile/overview.aspx.

Fang, Y. (2012). JSON simple - a simple java toolkit for JSON. http://code.google.com/p/json-simple/.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California.

Google (2011). Android design guidelines. http://developer.android.com/design/index.html.

Google (2012a). ADT plugin - Android Development Tools. http://developer.android.com/tools/sdk/eclipse-adt.html.

Google (2012b). Android SDK. http://developer.android.com/sdk/index.html.

Hameseder, K., Fowler, S., and Peterson, A. (2011). Performance analysis of ubiquitous web systems for smartphones. In *Performance Evaluation of Computer Telecommunication Systems (SPECTS), 2011 International Symposium on*, pages 84 –89.

Lee, V., Schneider, H., and Schell, R. (2005). *Aplicações Móveis – Arquitetura, Projeto e Desenvolvimento*. Editora Pearson Education do Brasil e Makron Books.

Microsoft (2012). Microsoft Translator API. http://api.microsofttranslator.com.

Mobl21 (2011). Mobl21. http://www.mobl21.com/.

Paillard, G. A. L., Costa, P. M. B., Rabelo, K. F., Sarmento, W. W. F., and Harriman, C. L. S. (2012). Extended mobilis: a integration of learning management system with mobile application to m-learning environment. *6th Euro American Conference on Telematics and Information Systems, 2012, Valência. EATIS '12 Proceedings of the 6th Euro American Conference on Telematics and Information Systems*.

Sarmento, W. W. F. (2007). Integração de um ambiente virtual de aprendizagem com aplicações móveis de suporte a educação à distância. Master's thesis, Programa de Pós-graduação do Departamento de Teleinformática da Universidade Federal do Ceará.

Trifonova, A. (2003). Mobile learning - review of the literature. *Technical Report DIT-03-009*. http://eprints.biblio.unitn.it/archive/00000359/01/009.pdf.

Watson, G. (2012). ORM Lite - lightweight object relational mapping (ORM) java package. http://ormlite.com.

Wharton, J. (2012). ActionBarSherlock. http://actionbarsherlock.com/.

Projeto Maritaca: Arquitetura e Infraestrutura para Coleta Móvel de Dados Usando *Smartphones* *

Bruno G. dos Santos, Alvaro H. Mamani-Aliaga, Jimmy V. Sánchez, Matheus F. Mendonça, Tiago Barabasz e Arlindo F. da Conceição[1]

[1] Instituto de Ciência e Tecnologia (ICT)
Universidade Federal de São Paulo (UNIFESP)
Rua Talim, 330. São José dos Campos - SP

arlindo.conceicao@unifesp.br

Abstract. *This paper presents the architecture of Maritaca, an infrastructure for data gathering using Android mobile devices. The system creates applications for mobile devices without need of knowledge about programming techniques. The system is based on free software and can be accessed in the following address:* `maritaca.unifesp.br`.

Resumo. *Este trabalho apresenta a arquitetura do projeto Maritaca, uma infraestrutura para coleta de dados a partir de dispositivos móveis Android. O sistema permite a criação de aplicações para a coleta móvel de dados sem a necessidade de conhecimentos de técnicas de programação. O sistema é baseado em tecnologias abertas e pode ser acessado a partir do seguinte endereço:* `maritaca.unifesp.br`.

1. Introdução

Nos últimos anos, o mercado de comunicação pessoal móvel evoluiu rapidamente, devido a três fatores: a queda dos preços, o lançamento de dispositivos móveis com alta capacidade de processamento e o surgimento de novas tecnologias para o desenvolvimento de Aplicações Móveis (Apps). Pode-se citar como exemplos destes avanços o lançamento de processadores *multicore* para dispositivos móveis e o amadurecimento das plataformas de programação para sistemas móveis Android [Pereira and Da Silva 2009]. Estes fatores, combinados, criaram condições para o surgimento de uma nova categoria de aplicações: a **Coleta Móvel de Dados** (CMD) [Rezende et al. 2010].

Entretanto, a criação de aplicações para coleta móvel de dados continua exigindo o trabalho de programadores, pois ainda é preciso programar os questionários eletrônicos. Atualmente, essa é a principal limitação para a ampla utilização de CMDs, pois nem toda empresa possui recursos humanos ou financeiros, para elaborar aplicações móveis.

Para enfrentar ou reduzir estas limitações e, desse modo, contribuir para a ampla utilização de ferramentas móveis para Coleta de Dados, desenvolvemos o **Projeto Maritaca**, que visa prover soluções tecnológicas abertas e infraestrutura para criação de aplicações para Coleta Móvel de Dados (CMD). O nome Maritaca[1] vem do acrônimo *MARitaca Is a Tool to creAte Cellular phone Applications*.

*Este trabalho recebeu o apoio da FINEP, edital Telessaúde e Telemedicina, processo 04.11.0077.00, referência 1488/10. Cabe mencionar os apoios recebidos por FAPESP, CNPq, IBOPE e FAP-UNIFESP, que indiretamente colaboraram para a execução deste projeto.

[1] *Maritaca* é o nome popular utilizado para designar diversas espécies de aves *psitaciformes*, da família

2. Trabalhos Relacionados

Existem algumas ferramentas com propósitos similares aos do Maritaca. A ferramenta *App Inventor*[2], permite construir visualmente aplicativos para a plataforma Android. Concentra-se no desenho passo-a-passo de elementos de interface, conectando-os aos respectivos eventos. O diferencial positivo do projeto Maritaca em relação ao *App Inventor* consiste em permitir o desenho mais simples e intuitivo das interfaces, isso é possível porque concentra-se em aplicações de CMD.

O *Nokia Data Gathering*[3] é um sistema para criação de questionários móveis que, colocados em um servidor na Internet, podem ser acessados pelos dispositivos móveis com acesso a rede, onde os dados são coletados e armazenados nos celulares e podem ser transmitidos para um servidor. Recentemente, a solução passou a ser distribuída como software livre.

Um projeto similar que devemos citar é o DoForms[4], um sistema para criação de questionários móveis multiplataforma, cuja proposta é semelhante ao Projeto Maritaca, porém, é de código fechado e possui limite de uso grátis.

3. Funcionalidades do Sistema

O sistema permite usuários construam aplicações para coleta de dados, que podem ser instaladas em quaisquer dispositivos móveis compatíveis com Android 2.2, ou superior. No dispositivo móvel, a App permite a coleta de dados utilizando interfaces amigáveis, onde os dados são armazenados no dispositivo móvel até serem transferidos para o servidor. Para realizar a coleta não é necessário que o usuário esteja conectado à Internet.

A seguir são apresentados os principais passos para a utilização da plataforma.

3.1. Passo-a-passo para utilização da solução

Os passos a seguir exemplificam a utilização da plataforma:

1. O sistema está disponível em `http://maritaca.unifesp.br`. Para utilizá-lo, o usuário necessita autenticar-se na tela inicial, o que pode ser realizado de duas maneiras: (*i*) utilizando usuário e senha registrados no sistema (*ii*) ou com uso de OpenID [Recordon and Reed 2006].
2. Após a autenticação, é apresentada ao usuário a tela principal do sistema, destinada ao gerenciamento de questionários (Figura 1). A lista de questionários está organizada entre formulários criados pelo próprio usuário (parte superior) e formulários com ele compartilhados (parte inferior). Há ainda um menu com abas na parte superior para o gerenciamento de grupos de usuários.
3. Para criar um novo formulário, basta pressionar o botão *New Form*, no canto inferior esquerdo da tela. A Figura 2 ilustra a interface para criação e edição de formulários contendo três perguntas como exemplo. A interface utiliza controles HTML5 arrastáveis (*Drag and Drop*), que permitem ao usuário "arrastar" as

dos *psitacídeos*, gênero *pionus*. Esta simpática ave foi escolhida como símbolo do projeto por sua capacidade de adaptação a diversos biomas, podendo ser encontrada em quase todo o território nacional.

[2] `http://appinventor.googlelabs.com/about`

[3] `http://projects.developer.nokia.com/ndg/wiki`

[4] `http://www.doforms.com/`

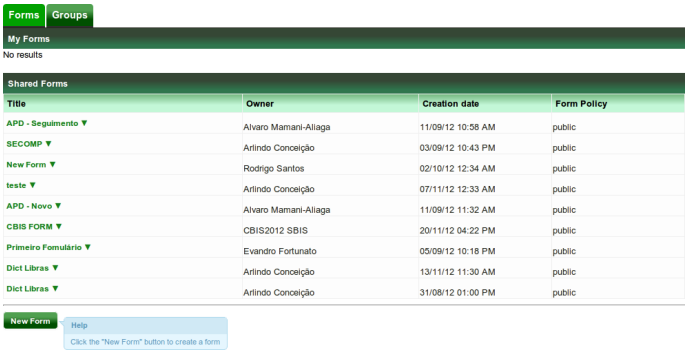


Figura 1. Tela inicial de gerenciamento de formulários.

componentes a serem utilizadas no questionário. O usuário pode escolher um tipo de pergunta que queira coletar dentre o conjunto de componentes disponíveis na caixa de ferramentas localizada no lado esquerdo da interface.

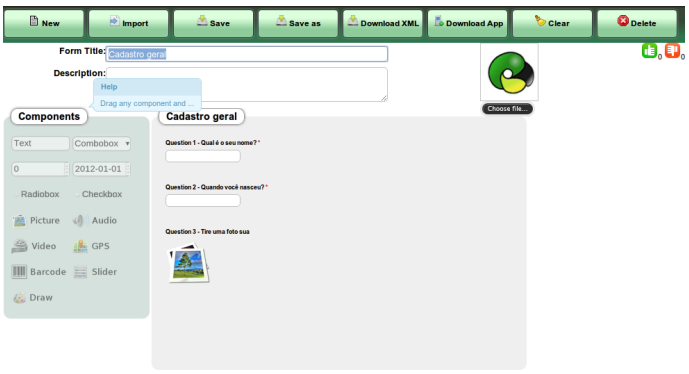
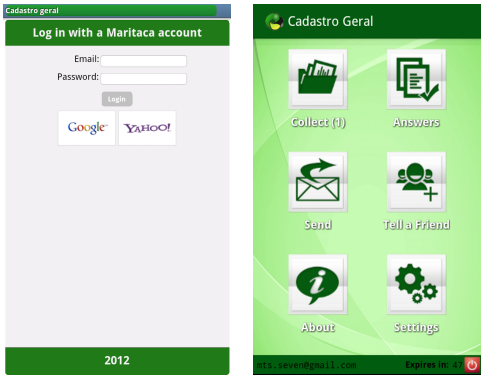


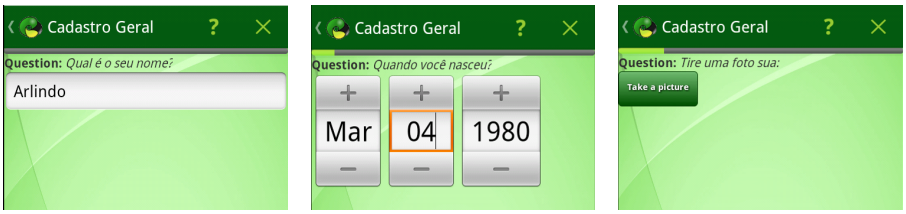
Figura 2. Editor de Formulários, preenchido.

4. Uma vez criado o formulário, o usuário pode descarregá-lo para um dispositivo móvel com sistema operacional Android. A instalação do aplicativo no dispositivo móvel é trivial. A Figura 3(a) mostra a tela de autenticação na aplicação móvel. A Figura 3(b) mostra o menu principal, a partir da qual inicia-se a coleta clicando-se no botão *Collect*.
5. A Figura 4 mostra a interface que foi especificada como exemplo na Figura 2 renderizada automaticamente no dispositivo móvel. Cada pergunta é mostrada em uma tela a qual é organizada em duas partes: acima, com o título do formulário, botões (*ajuda* e *cancelar*), e barra de progresso; e abaixo a questão em si. Os controles de navegação (*retornar* e *avançar*) podem ser executados através de comandos *touch*.
6. A coleta de dados pode ser realizada inúmeras vezes. Para coletar os dados não é necessário que o dispositivo móvel esteja conectado à Internet, pois uma vez coletados os dados, o usuário pode conectar-se à Internet e enviar as coletas realizadas para o servidor. Quando armazenados no servidor, os dados podem ser visualizados acessando-se a plataforma. Por exemplo, a Figura 5 ilustra a visualização de dados coletados com o formulario criado para este exemplo.



(a) Tela de autenticação (b) Tela inicial

Figura 3. Aplicação móvel



(a) Primeira questão, do tipo texto. (b) Segunda questão, do tipo data. (c) Terceira questão, do tipo foto.

Figura 4. Perguntas renderizadas no dispositivo móvel.

3.2. Os tipos de pergunta

Um questionário é composto por uma ou mais perguntas e cada pergunta pode ser de diferentes tipos: texto, numérico, data, múltipla escolha (*radio button*), seleção múltipla (*combo box* e *check bok*), foto, áudio, vídeo, localização, código de barra, porcentagem (controle deslizante ou *slider*) e desenho. Implementaremos ainda outros tipos, pois uma das características mais positivas da plataforma, graças ao seu desenho modular, é a facilidade para se criar e integrar ao sistema novas componentes.

O usuário pode personalizar valores padrão (*default*) para alguns tipos de campos a serem coletados, configurar validações para os valores inseridos e definir uma ordem de navegação entre as perguntas dependente dos valores inseridos.

3.3. Compartilhamento de dados

Atualmente, o sistema permite criar 3 classes de formulários: privados, públicos e compartilhados. O formulário **privado** pode ser utilizado apenas pelo seu criador e o formulário **público** pode ser visto por qualquer usuário da plataforma. Já os formulários compartilhados podem ser de dois subtipos: hierárquico e social. No formulário **compartilhado hierárquico**, o seu criador pode convidar, por exemplo, os usuários A e B, mas os dados coletados por A não são visíveis por B e vice-versa. Por sua vez, no formulário **compartilhado social** os usuários A e B podem ver os dados coletados um pelo outro.

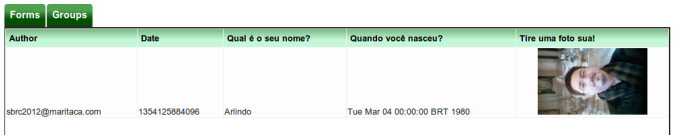


Figura 5. Editor de Formulários, preenchido.

4. Arquitetura da Solução

O projeto Maritaca substitui com vantagens os modelos de coleta tradicionais baseados em lápis e papel. O desenvolvimento de um aplicativo de coleta móvel de dados permite o armazenamento de informações em meio eletrônico, o que facilita a manipulação das informações, assim como o seu compartilhamento.

O projeto Maritaca foi desenvolvido como uma aplicação de nuvem, prevendo sua utilização como "software como serviço". Os dados coletados a partir de dispositivos móveis são armazenados na nuvem e são visualizados através de um navegador web.

A versão atual do projeto encontra-se disponível para utilização em `http://maritaca.unifesp.br`. O código fonte da solução e sua documentação associada está disponível, sob a licença GPL3, em `http://sourceforge.net/p/maritaca`.

Nesta seção será apresentada a arquitetura do projeto, suas componentes e o modelo de integração entre as componentes.

4.1. Componentes da Plataforma Maritaca

A Figura 6 apresenta as principais componentes da arquitetura da solução em nuvem e ilustra a relação entre essas componentes. A arquitetura é composta por:

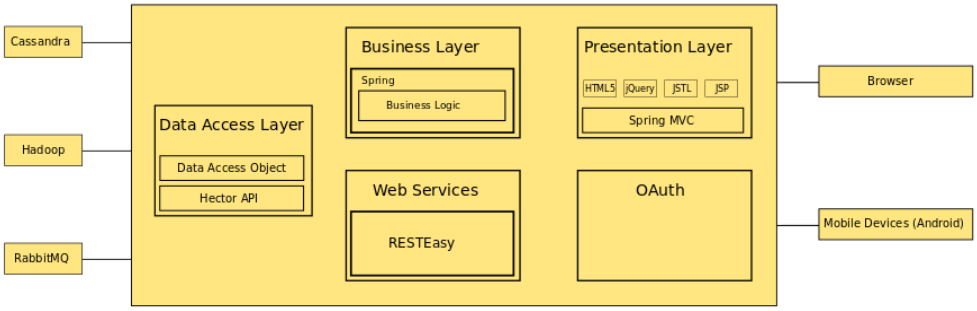


Figura 6. Componentes da solução em nuvem.

- **Servidor de Aplicações**: a parte servidora do projeto utiliza o servidor de aplicações *JBoss* onde hospedam-se os serviços e componentes Web. Toda a parte servidora, exceto alguns *scripts* de manutenção, foram implementados em Java utilizando o arcabouço *Spring* [Tate and Gehtland 2005]. Todos os *webservices* implementados utilizam a abordagem de serviços *RESTFul*.
- **Editor de Formulários**: apesar de estar embutido na plataforma Web, o Editor de Formulários é uma aplicação Web independente, escrita em HTML5 e Ajax. Esta componente gera como resultado um **descritor de questionário**, que é gravado em formato XML.

- **Componente móvel**: é uma aplicação Android que interpreta o arquivo XML e gera as interfaces automaticamente. Trata-se de uma *Engine* de interpretação, baseada no padrão de projeto *Interpreter* [Gamma and Helm 1994].
- **Servidor de Dados Cassandra**: componente para armazenamento escalável de dados estruturados, baseado no paradigma *NO-SQL*.
- **Sistema de arquivos Hadoop**: um sistema de arquivos distribuído e escalável utilizado para armazenar dados não estruturados (imagens, áudio, vídeo etc.).

4.2. Servidor de Aplicações

No módulo servidor encontram-se os serviços e funcionalidades Web do projeto, dentre os quais pode-se citar: armazenamento de dados estruturados e não estruturados (conteúdo multimídia e Apps), os serviços RESTFul utilizados para a comunicação entre o servidor e o dispositivo móvel, o editor de formulários etc. O diagrama da Figura 6 mostra os diferentes componentes e arcabouços utilizados no projeto e ilustra a relação entre eles.

A interação entre o usuário e a plataforma Web, via *browser*, é feita através da camada Web do sistema, que utiliza principalmente Spring MVC, JQuery e HTML5. A interação entre dispositivos móveis e a plataforma utiliza-se sempre de serviços RESTFul.

O desenho da arquitetura prevê a criação de instâncias dos componentes do sistema (JBoss, Cassandra e Hadoop) em um *cluster* computacional. O balanceamento de carga de requisições web será implementado utilizando-se o módulo mod_JK do servidor de páginas Apache.

4.3. Módulo Móvel

O usuário pode editar os questionários a partir de um navegador padrão e, uma vez editado, o questionário pode ser preenchido a partir do dispositivo móvel, onde os dados ficam temporariamente armazenados até serem transferidos para o servidor.

O Módulo Móvel consiste em uma *engine* que traduz o descritor do questionário (representado em formato XML) em uma hierarquia de objetos instanciados responsáveis pela renderização das interfaces e validação dos dados. O modelo computacional utilizado para representar os questionários é a parte mais sofisticada do componente móvel e foi o desenvolvimento tecnológico que viabilizou a solução.

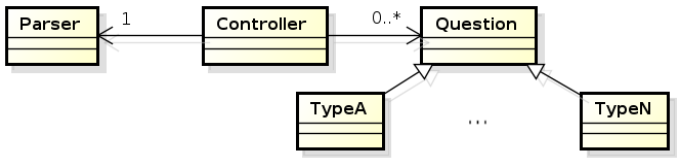


Figura 7. Arquitetura da análise do XML usando o padrão de projeto *Interpreter*

A técnica de mapeamento do XML em uma lista de objetos está inspirada no padrão de projeto *Interpreter* [Gamma and Helm 1994] (vide Figura 7). A aplicação móvel é uma controladora de contexto, que sempre aponta para um objeto em uso, no caso do Maritaca o objeto seria do tipo *Question*.

Para ser utilizada a partir do dispositivo móvel, a aplicação móvel requer que o usuário autentique-se no servidor, deste modo, certifica-se que o usuário tem autorização

para coletar dados para aquele formulário. Esse processo é feito utilizando o arcabouço OAuth [Tassanaviboon and Gong 2011] sobre o protocolo HTTPS.

A Figura 8 ilustra os passos para a autenticação segundo o padrão OAuth: (i) o usuário faz uma requisição de autorização para o servidor, onde é redirecionado para uma tela de autenticação; (ii) se o usuário é autenticado com sucesso, gera-se um código, esse passo é chamado de "confirmação da autorização"; (iii) com este código, o usuário gera um *token*, que necessita ser renovado após um período; (iv) por fim, de posse de um *token* válido para o formulário, o usuário pode coletar, salvar e visualizar os dados, onde a transferência de dados é feita utilizando-se de serviços RESTFul.

Figura 8. Modelo de autenticação entre o componente Móvel e o Servidor utilizando OAuth.

A interpretação do XML foi implementada baseando-se no uso do arcabouço Simple [5] para a serialização e deserialização de arquivos XML. Isto é, este arcabouço converte diretamente arquivos XML em objetos e vice-versa. A vantagem do uso deste arcabouço é devido a simplificação do mapeamento entre o formato XML e o formato dos objetos, simplificando a manutenção do código e criação de novas componentes.

4.4. Formato do arquivo XML de integração

O principal método de integração entre as componentes é o arquivo XML gerado pelo Editor de Formulários, a qual contém as questões do formulário representadas como *tags*. Cada *tag* possui os seguintes atributos básicos: *id*, *next*, *previous*, *required*, *label*, *help* e *type*.

Alguns tipos de *tag* podem possuir estruturas *conditions*, utilizadas para definir a navegação condicional entre as perguntas. A navegação entre as perguntas é definida no rótulo *condition*, onde a resposta da pergunta atual é utilizada para determinar a próxima pergunta a ser exibida. Por exemplo, considere a seguinte questão: "*Qual a sua idade?*" Se a resposta for um valor inferior a 18 anos, a próxima questão poderia ser: "*Qual o nome do seu responsável?*" Caso contrário, essa pergunta poderia ser omitida.

Alguns tipos de questões possuem validadores para os dados coletados. Por exemplo, em uma questão numérica, pode-se definir limites mínimos e máximos para as entradas. Desse modo, em uma questão sobre a idade do entrevistado, os valor mínimos e máximos para as respostas podem ser definidos, respectivamente, como 0 e 100. Essa validadação evita a coleta de dados errados.

4.5. Captura de dados não usuais: multimídia, localização geográfica etc.

Além de prover a coleta de dados usuais, tais como textos e números, a solução permite também a coleta de dados não usuais, tais como multimídia (aúdio, vídeo e imagens) [da Conceição et al. 2008], informações de localização geográfica, desenhos e

[5] Serialização para XML, mais informações em `http://simple.sourceforge.net/`.

códigos de barra etc. Em suma, o questionário poderá conter perguntas tais como: *Qual a sua localização atual? Tire uma foto! Registre o áudio!*

4.6. Construção automática de aplicações

Sempre que um formulário é salvo, o sistema gera um novo aplicativo Android (arquivo em formato APK) e o armazena no sistema de arquivos distribuído Hadoop. Esse processo de compilação e montagem leva alguns segundos, mas como é realizado em *background* não afeta a percepção de usabilidade do usuário. Essa não foi a primeira abordagem adotada, inicialmente planejava-se criar uma única aplicação Android na qual os descritores XML seriam carregados. Entretanto, notou-se que esse método dificultava o processo de divulgação de aplicações e a manutenção das versões das Apps.

5. Descrição da demonstração planejada

Para apresentação do Projeto *Maritaca* é necessário um computador com acesso à Internet para confecção de um formulário. Após criado o formulário, é necessário transferir a aplicação gerada para um dispositivo móvel com suporte a plataforma Android.

6. Conclusões

Este trabalho apresentou o Projeto Maritaca, uma solução aberta para a Coleta Móvel de Dados. Iniciado a mais de quatro anos, o projeto evoluiu de um gerador automático de aplicações para uma solução completa que permite armazenamento e compartilhamento de dados. O Maritaca busca resolver um problema prático: o elevado custo de produção de sistemas móveis. Esperamos que ele seja amplamente utilizado na resolução de problemas cotidianos, contribuindo, assim, para a utilização de sistemas móveis por não especialistas. Atualmente, trabalhamos para que ele permita não apenas a configuração dos questionários, mas também das respostas. Isto é, o usuário poderá configurar os dados a serem coletados e também a forma como estes dados serão exibidos.

Referências

da Conceição, A., Pereira, R., Rezende, J., Silva, B., Correia, R., Domingues, H., Kon, R., and Kon, F. (2008). Projeto Borboleta: Ferramentas Móveis e Multimídia para Atenção Básica Domiciliar. In *Congresso Brasileiro de Informática em Saúde. Artigo curto*.

Gamma, E. and Helm, R. (1994). *Design Patterns*. Addison-Wesley Professional.

Pereira, L. and Da Silva, M. (2009). *Android para desenvolvedores*. Brasport.

Recordon, D. and Reed, D. (2006). OpenID 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, New York, NY, USA. ACM.

Rezende, J. V. P., Silva, B. N. M., and da Conceição, A. F. (2010). Plataforma para desenvolvimento simples e flexível de questionários para Coleta Móvel de Dados (CMD). In *I Workshop de Pesquisa e Desenvolvimento em Software Livre (WPeDSL)*, Natal-RN.

Tassanaviboon, A. and Gong, G. (2011). OAuth and ABE based authorization in semi-trusted cloud computing: aauth. In *Proceedings of the second international workshop on Data intensive computing in the clouds*, DataCloud-SC '11, pages 41–50, New York, NY, USA. ACM.

Tate, B. and Gehtland, J. (2005). *Spring: a developer's notebook*. O'Reilly Media, Incorporated.

Desenvolvendo Aplicações de Rastreamento e Comunicação Móvel usando o Middleware SDDL

Igor Vasconcelos, Rafael Vasconcelos, Gustavo Baptista, Caio Seguin, Markus Endler

Departamento de Informática - Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

Rio de Janeiro – RJ – Brazil

{ivasconcelos,rvasconcelos, gbaptista, cseguin, endler}@inf.puc-rio.br

Abstract. *This article explains, and gives some details, of how an application prototype for mobile tracking and communication among roadside inspectors and vehicles called Acompanhamento Remoto de Fiscais e Frotas (ARFF) was developed using the Scalable Data Distribution Layer(SDDL) middleware.*

Resumo. *Este artigo explica, e dá alguns detalhes, de como um protótipo de aplicação para monitoramento móvel e comunicação entre fiscais e veículos em uma estrada chamado Acompanhamento Remoto e Fiscais Frotas (ARFF) foi desenvolvido utilizando middleware Scalable Data Distribution Layer (SDDL).*

1. Introdução

Os avanços nos dispositivos e redes de comunicação móveis têm possibilitado o desenvolvimento de aplicações em várias áreas como, por exemplo, transporte e logística, monitoramento ambiental e segurança. Contudo, estes aplicativos devem permitir a comunicação e coordenação entre os nós móveis, os quais podem ser pessoas, veículos ou robôs móveis. Tais nós móveis requerem tecnologias que ofereçam baixa latência e alta taxa de disseminação de dados, juntamente com estabilidade, escalabilidade e alta disponibilidade [Corradi et al. 2010; David, L et al. 2012].

O presente artigo tem por objetivo introduzir os passos básicos para o desenvolvimento de aplicações de rastreamento e comunicação utilizando o *middleware* SDDL (Scalable Data Distribution Layer), por meio do protótipo de Acompanhamento Remoto de Fiscais e Frotas - ARFF, que foi desenvolvido em apenas três meses, principalmente devido ao suporte de comunicação móvel do SDDL e seu intrínseco modelo *Publish-Subscribe* centrado em dados. Inicialmente, na Seção 2, são descritos sucintamente os conceitos e arquitetura do SDDL. A Seção 3 discute alguns trabalhos relacionados sobre *middleware* para aplicações móveis. Já a Seção 4, apresenta o ARFF, suas funcionalidades e a implementação de seus principais módulos. Por fim, a Seção 5 apresenta as considerações finais e possíveis evoluções do sistema.

2. SDDL

O SDDL (*Scalable Data Distribution Layer*) [David, L et al. 2012] é um middleware de comunicação que conecta nós estacionários em uma rede central (*core*) cabeada de alta velocidade com comunicação baseada na especificação Data Distribution Service for Real-time Systems (DDS), à nós móveis com uma conexão sem fio baseada em IP (*IP*-

based). O SDDL utiliza dois protocolos de comunicação: *Real-Time Publish-Subscribe* (RTPS) *Wire Protocol* [OMG 2010] para a comunicação cabeada dentro da rede central do SDDL, e *Mobile Reliable UDP* (MR-UDP) *Protocol* [David, L et al. 2012] para a comunicação entre os nós móveis e a rede central.

Os elementos da rede central baseiam-se no modelo DCPS (*Data-Centric Publish-Subscribe*) do DDS, onde os Tópicos DDS são definidos para serem usados para comunicação e coordenação entre tais elementos da rede central. Como parte da rede central, alguns nós do SDDL possuem funções distintas, tais como: (i) *Gateway*, (ii) *Controller* e (iii) *GroupDefiner*.

O *Gateway* (GW) define um único ponto de acesso (*Point of Attachment* – PoA) para as conexões com os nós móveis. O *Gateway* é responsável por gerenciar uma conexão MR-UDP separada para cada um dos nós móveis, encaminhando qualquer mensagem específica da aplicação ou informação de contexto para a rede central, e em direção oposta, convertendo mensagens DDS para mensagens MR-UDP e as entregando de maneira confiável aos nós móveis correspondentes.

O *Controller* é uma aplicação para a visualização da posição de todos os nós móveis (ou qualquer outra informação de contexto), além de permitir o gerenciamento de grupos e o envio de mensagens *unicast*, *groupcast* ou *broadcast* para os nós móveis.

O *GroupDefiner* que é responsável por avaliar as associações de grupo dos nós móveis. Para tal, o *GroupDefiner* se subscreve a um Tópico DDS onde as mensagens ou dados de contexto dos nós móveis são disseminados e então mapeia cada nó móvel em zero, um ou mais grupos, de acordo com alguma lógica de processamento de grupos específica da aplicação. Esta informação de associação de grupo é então compartilhada com todos os *Gateways* na rede central usando um Tópico DDS específico para este controle.

3. Trabalhos Relacionados

Na literatura pode ser encontrada uma grande quantidade de soluções de comunicação, entretanto, poucos demonstram como os middlewares podem ser utilizados na construção de aplicativos. Na prática, ocorre o desenvolvimento de estudos de caso para avaliação dos *middlewares:*

SIENA [Caporuscio et al. 2002] é um serviço de notificação de eventos *publish/subscribe* que possui duas entidades distintas: clientes e servidores. Os servidores são interconectados (a interconexão representa o serviço de eventos) como uma rede distribuída objetivando escalabilidade, e fornecem aos clientes pontos de acesso por meio da interface do sistema *Publish/Subscribe.* Em [Caporuscio et al. 2002], com o intuito de estudar o desempenho do SIENA em uma rede *wireless* de baixo desempenho e alta taxa de erro, foi desenvolvido um sistema de leilão ponto-a-ponto (P2P) que permitisse a compra e venda de produtos.

Outros artigos como em [Corradi et al. 2010; Rowstron and Druschel 2001], explicam a arquitetura dos *middlewares* e desenvolvem aplicações para validar questões de desempenho. Contudo, explicações de como desenvolver aplicações utilizando tais *middlewares* são negligenciadas.

4. ARFF

O sistema de Acompanhamento Remoto de Fiscais e Frotas (ARFF) é um protótipo desenvolvido sobre a API do SDDL, cujo objetivo é gerenciar e agilizar o processo de fiscalização de frotas de ônibus por parte dos órgãos responsáveis, bem como demonstrar a viabilidade e simplicidade no desenvolvimento de aplicações que distribuam dados de contexto de forma escalável para centenas de milhares de nós móveis (ônibus). Uma demonstração do funcionamento do ARFF pode ser visto no site do LAC (*Laboratory for Advanced Collaboration*) através do link: http://www.lac.inf.puc-rio.br/arff/.

Fiscais podem estar em comandos, ou seja, em lugares pré-definidos ao longo de estradas/rodovias, onde fiscalizam veículos (ônibus) em uma *blitz*, ou podem estar em centrais de controle/monitoramento para gerenciar eventos e auxiliar outros fiscais que estão realizando as fiscalizações em campo. Desta forma, as funcionalidades principais são: (i) identificar no mapa a posição atual de cada ônibus/fiscal; (ii) Enviar mensagens em *unicast, groupcast* e *broadcast*; (iii) Acompanhar, a partir da central de monitoramento, as inspeções em andamento; (iv) Delegar uma inspeção a um fiscal; e (v) Cadastrar regiões que os ônibus não podem entrar e avisar quando um ônibus entrar ou sair dessa região.

Os eventos de troca de informação entre o ARFF e o SDDL são assíncronos, ou seja, a aplicação não precisa a todo o momento verificar se há novos dados de contexto a serem exibidos. Assim que uma mudança de contexto ocorre, o SDDL dispara um evento que é tratado pela aplicação. A Figura 1 mostra, em alto nível, a arquitetura do ARFF.

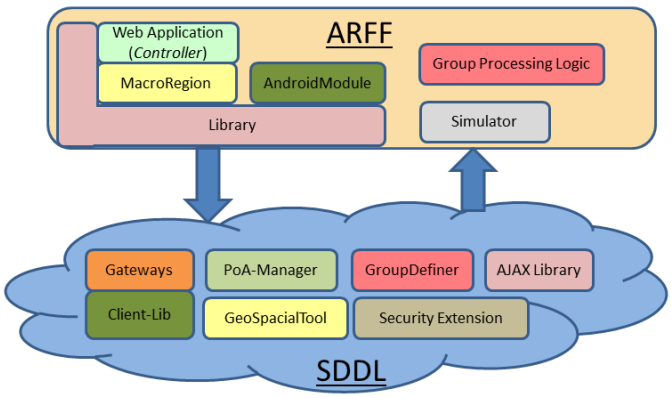


Figura 1 - Arquitetura do ARFF

As seções a seguir explicam o funcionamento de cada módulo do ARFF e como foi feita a implementação dos principais componentes usando a API do SDDL. Mais detalhes podem ser encontrados em [David, L et al. 2012].

4.1. Web Application

Da central de monitoramento do ARFF (*controller*) é possível saber a localização de cada ônibus e fiscal, bem como suas respectivas informações. Cada cor tem um significado, por exemplo, um ônibus em vermelho significa que o veículo foi reprovado na última inspeção. Caso haja algum tipo de conexão com a internet no momento de uma inspeção, é possível que a central acompanhe todos os procedimentos que o fiscal

está realizando, já sendo informados os itens inspecionados aprovados ou reprovados. A Figura 2 exibe a tela da central de monitoramento.

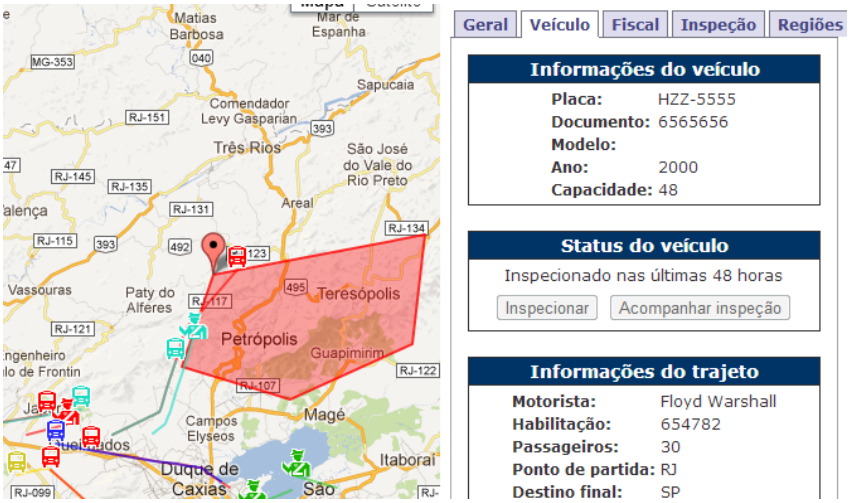


Figura 2 - Central de Monitoramento

Paralelamente ao *controller*, foi desenvolvido um módulo *android* multifuncional para o uso de fiscais durante atividades de campo. O aplicativo conta com uma janela de mapa, onde o fiscal tem acesso a sua localização geográfica. Por meio de um *chat*, o fiscal pode enviar e receber mensagens da central de monitoramento e de outros fiscais. Além disso, o aplicativo conta com uma *funcionalidade* que permite a execução de diversos tipos de inspeções e preenchimento de formulários. As informações preenchidas nos diferentes campos dos formulários são enviadas automaticamente para a central de monitoramento, o que permite o acompanhamento da inspeção durante o seu andamento. A Figura 3 mostra alguns formulários do módulo *android*.

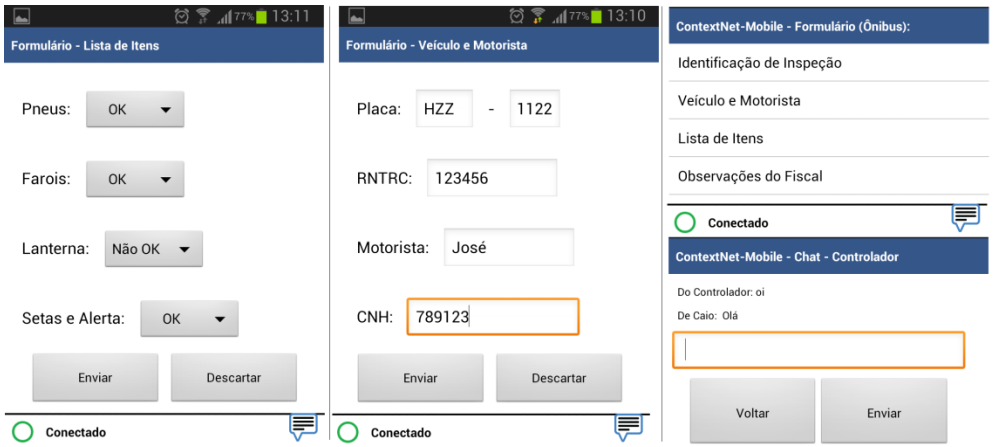


Figura 3 - Formulários do Android

Para que seja possível realizar essa troca de informações entre a aplicação web e os nós móveis, foi implementado dentro do módulo *ARFF-Library* uma classe (*HelperSerializableMessage*) responsável pelo envio de mensagens no ARRF. A Figura 4 mostra como enviar uma mensagem (i.e. a instância de um tópico) para o domínio SDDL. O parâmetro *serializableMessage* é a mensagem que será enviada, já a variável *privateMessageTopic* conterá as informações de endereçamento, ou seja, se a mensagem será endereçada a um nó móvel específico, a um grupo de nós ou para todos os nós.

```
private void WriteTopic(Serializable serializableMessage,
                        PrivateMessageTopic privateMessageTopic) throws IOException {
  ApplicationMessage applicationMessage = new ApplicationMessage();
  applicationMessage.setContentObject(serializableMessage);
  privateMessageTopic.message = Serialization.getObjectByteStream(applicationMessage);

  ddsNode.writeTopic(PrivateMessageTopic.class.getSimpleName(), privateMessageTopic);
}
```

Figura 4 - Enviando uma mensagem para um Tópico do Domínio

Para o envio de uma mensagem em *broadcast* é necessário informar ao SDDL que não há um Gateway, nó móvel e grupo específico de envio. Para tal é necessário atribuir o valor de *broadcast*, como visto na Figura 5. O passo seguinte é usar o método da Figura 4 para enviar a mensagem para o domínio.

```
PrivateMessageTopic privateMessageTopic = new PrivateMessageTopic();

privateMessageTopic.leastSignificantBitsGatewayId = UniversalDDSLayerFactory.BROADCAST_FLAG;
privateMessageTopic.mostSignificantBitsGatewayId = UniversalDDSLayerFactory.BROADCAST_FLAG;
privateMessageTopic.leastSignificantBitsVehicleId = UniversalDDSLayerFactory.BROADCAST_FLAG;
privateMessageTopic.mostSignificantBitsVehicleId = UniversalDDSLayerFactory.BROADCAST_FLAG;
privateMessageTopic.groupId = UniversalDDSLayerFactory.BROADCAST_FLAG;
privateMessageTopic.groupType = UniversalDDSLayerFactory.BROADCAST_FLAG;
```

Figura 5 - Mensagem em broacast

Os passos para enviar mensagens para grupos é semelhante, a diferença reside na propriedade *groupId* e *groupType* que receberão seus respectivos valores. Por questões de implementação, no ARFF foi convencionado como padrão para o *groupId* o valor um para o grupo de fiscais e dois para o grupo de ônibus. Já para o envio de mensagens para um nó móvel específico, as únicas informações necessárias são o identificador do *Gateway* e do nó móvel (ônibus). O *groupType* é utilizado na lógica de processamento de grupos separando os nós móveis por tipo. Poderiam ter sido definidos dois tipos de grupos no ARFF - um tipo para os ônibus e outro para os fiscais. Para o tipo de grupo ônibus, poderiam haver vários grupos - dos ônibus já fiscalizados ou não fiscalizados, por exemplo.

4.2. Group Process Logic

O *GroupDefiner* é responsável por avaliar as adesões de grupo de todos os nós móveis. Este componente tem por objetivo definir os grupos em que cada nó móvel deve participar. Para desenvolver o serviço de processamento de grupos, deve-se implementar a interface *GroupSelector* ou *GroupSelectorSupportingProhibitedGroups* caso no sistema a ser desenvolvido haja a necessidade de restrições de regiões, ou seja, se existem regiões no mapa que os ônibus não podem circular. Ambas as interfaces possuem o método *processGroups,* onde ficará a lógica de definição dos grupos.

Para a codificação dos grupos proibidos, o primeiro passo é carregar a lista de regiões cujos ônibus não podem entrar (mais detalhes sobre as regiões podem ser vistas na seção 4.3). Em seguida deve-se testar se o objeto é um ônibus e se está dentro de uma região proibida. A Figura 6 exibe como processar os grupos da aplicação.

```java
@Override
public Set<Integer> processGroups(Message nodeMessage) {
  Object applicationObject =
      Serialization.getObjectFromBytes(nodeMessage.content);
  Set<Integer> groupList = new HashSet<Integer>();

  if (applicationObject instanceof Inspector) {
    groupList.add(1);
  }
  else if (applicationObject instanceof Vehicle) {
    groupList.add(2);
  }
  return groupList;
}
```

```java
@Override
public Set<Integer> processProhibitedGroups(Message nodeMessage) {
  Object applicationObject =
    Serialization.getObjectFromBytes(nodeMessage.content);
  // code for load regions
  Set<Integer> groups = new HashSet<Integer>();
  if (applicationObject instanceof Vehicle) {
    Vehicle vehicle = (Vehicle) applicationObject;
    try {
      for (MacroRegion region : listRegion) {
        if (region.isVehicleInMacroRegion(vehicle)) {
          groups.add(region.getId());
          prohibitedGroups.add(region.getId());
        }
      }
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
  return groups;
}
```

Figura 6 - Processamento de Grupos

Percebe-se também que o identificador da região foi adicionado a uma lista de grupos proibidos (variável de classe: *prohibitedGroups*). Isso é necessário, pois a interface *GroupSelectorSupportingProhibitedGroups* possui o método *isProhibitedGroup,* o qual é responsável por repassar para o SDDL uma lista dos grupos proibidos. Com os exemplos mostrados, nota-se que o SDDL fornece um mecanismo simples e ágil para implementar a lógica de processamento de grupo para qualquer tipo de aplicação.

4.3. Macro Region

Em algumas situações é importante que nenhum veículo da frota adentre nesta região, ou caso seja necessária à entrada, esse veículo pode ser acompanhado de perto pela central. A seguir será mostrado como foi feito esse tipo de monitoramento no ARFF utilizando a API do SDDL.

Na prática, para o SDDL, cada região dita como proibida é na verdade um novo grupo, e assim está ligada ao *GroupDefiner*. No ARFF, foi criado um serviço que identifica o evento de entrada/saída de um nó móvel (ônibus) em uma região proibida.

```java
@Override
public void onNewData(Object topicSample) {
  if (topicSample instanceof GroupAdvertisementTopic) {
    String msg = "";
    GroupAdvertisementTopic region = ((GroupAdvertisementTopic) topicSample);
    boolean showAlert = false;
    for (Integer groupId : region.groupOperationCollection) {
      /* Code-If groupId is greater than 2, object has entered,
      else left a region - Set Message and show alert */
      UUID gatewayId = new UUID(region.mostSignificantBitsGatewayId,
                                region.leastSignificantBitsGatewayId);
      UUID vehicleId = new UUID(region.mostSignificantBitsVehicleId,
                                region.leastSignificantBitsVehicleId);
      ChangedSpecialRegion changedSpecialRegion =
        new ChangedSpecialRegion(msg, region.groupType, gatewayId,
                                 vehicleId, groupId, showAlert);
      SendMessage(changedSpecialRegion);
    }
  }
}
```

Figura 7 - Evento de Entrada/Saída de Região Proibida

Para que o serviço mostrado na Figura 7 funcione, é necessário implementar a interface *UDIDataReaderListener*. Assim, o serviço é notificado, assincronamente, sobre novos dados de entrada/saída de grupos recebidos no domínio DDS. Por questão de padronização do SDDL, sempre que há uma entrada em um grupo, o é utilizado um identificador positivo, caso contrário é utilizado um identificador negativo. Portanto, qualquer identificador de grupo maior que dois (os valores um e dois são reservados

para fiscais e ônibus respectivamente) ou menor que menos dois implica na entrada/saída de um ônibus em uma região. Para que a *web application* (*controller*) seja notificada deste evento, é criado uma instância do objeto *ChangedSpecialRegion* com uma mensagem de alerta destinada ao motorista do ônibus, tipo de grupo, os identificadores do *gateway* e do ônibus, e *boolean* (*showAlert*) informando se algum alerta deverá ser emitido no mapa. Esse objeto é enviado para o domínio DDS.

4.4. Simulator

O módulo de simulação do ARFF foi desenvolvido com o intuito de emular vários nós móveis e consequentemente o envio/recebimento de informações de contexto (latitude, longitude, velocidade e a hora de ocorrência do evento), uma vez que seria inviável realizar a simulação com nós reais. Para a implementação foi utilizada a *Client-Lib* do SDDL, que é um componente de software usado na implementação dos aplicativos nos clientes móveis. Ela esconde a maior parte dos detalhes dos protocolos de comunicação e trata problemas de conectividade com os *Gateways*. Os testes de performance do SDDL, bem como de simulação, podem ser vistos com mais detalhes em [David, Lincoln et al. 2012]. No início da simulação cada nó móvel associa-se a um *Gateway*, no entanto, pode haver troca de *Gateway* ao longo de sua movimentação emulada.

Cada nó móvel emulado está associado a uma *thread* que periodicamente envia sua nova localização para o *Gateway* ao qual está conectado (via MR-UDP). As trajetórias percorridas por cada nó móvel (ônibus e fiscais) são definidas em um arquivo de configuração do módulo simulador.

Para desenvolver o simulador, é preciso implementar a interface *NodeConnectionListener* do MR-UDP, pois ela representa os eventos (ex.: conexão, desconexão, chegada de novas mensagens) entre o nó móvel e o *Gateway*. Toda abstração de envio da mensagem fica encapsulada no *ApplicationMessage*, e basta que o objeto com suas novas informações de contexto e seu respectivo identificador sejam informados. De maneira simples, consegue-se emular um grande número de nós móveis sem sobrecarregar a capacidade de processamento do servidor. A Figura 8 mostra a utilização da *Client-Lib* do SDDL.

```
@Override
public void run() {
  // Some code ...
  Point2D.Double newPos = new Point2D.Double();
  // code for calculate new positions
  long time = System.currentTimeMillis();

  InspectorTrackingInformation tracking =
    new InspectorTrackingInformation(newPos.getX(), newPos.getY(),
                                     10.1, time - (30 * 1000));
  inspectorLocalInfo.getTrackingInformation().add(tracking);
  lastPos = newPos;

  applicationMessage = new ApplicationMessage();
  applicationMessage.setContentObject(inspectorLocalInfo);
  applicationMessage.setTagList(tags);
  applicationMessage.setSenderID(inspectorLocalInfo.getId());

  try {
    myConnection.sendMessage(applicationMessage);
  }
  catch (IOException e) {
    e.printStackTrace();
  }
}
```

Figura 8 - Envio de dados de contexto

4.5. Demonstração no SBRC

A demonstração pode ser feita com o uso dois notebooks (com acesso a internet), um para emular os nós móveis e outro para emular o núcleo SDDL e rodar a aplicação web, além de um *tablet* com o Sistema Operacional *Android*. A demonstração inclui a

emulação de 10 ônibus e 6 fiscais, sendo que um dos fiscais executa o cliente móvel ARFF (em um *tablet*), permitindo interagir com o usuário operando a central de monitoramento do ARFF (*controller*). Através desse, pode-se trocar mensagens instantâneas com o cliente, ordenar fiscalizações de um ônibus, e acompanhar o preenchimento do formulário de fiscalização em tempo real. Os nós móveis (fiscais e ônibus) possuirão localização arbitrária e atualizações a cada 5 segundos. Em paralelo, uma demonstração com 5000 nós móveis pode ser realizada para mostrar a escalabilidade e desempenho, apesar do grande volume de dados.

5. Conclusão

Este artigo apresenta o ARFF, desenvolvido com o uso do SDDL (*Scalable Data Distribution Layer*), um *middleware* de comunicação distribuída. Com o ARFF, foi possível mostrar como implementar o envio de mensagens e de dados de contexto, assim como mostrar como é realizada a definição de grupos e como foi definido o serviço de descoberta de entrada/saída de regiões proibidas. A versão corrente, bem como sua documentação, estão disponíveis em: http://www.lac.inf.puc-rio.br/arff/.

Como trabalhos futuros, serão desenvolvidas APIs de *QoS*, segurança e balanceamento de carga. Outras evoluções poderiam ser realizadas através do uso de informações contexto e prover mecanismos que permitissem a associação de regras de contexto com outras funcionalidades de colaboração. Por exemplo, caso esteja chovendo e um veículo transportando carga perigosa, uma mensagem automática seria enviada informando que uma determinada velocidade máxima não deverá ser ultrapassada.

Referências

Caporuscio, M., Inverardi, P. and Pelliccione, P. (2002). Formal Analysis of Clients Mobility in the Siena Publish / Subscribe Middleware. Technical report, Department of Computer Science, University of L'Aquila.

Corradi, A., Foschini, L. and Nardelli, L. (jun 2010). A DDS-compliant infrastructure for fault-tolerant and scalable data dissemination. In *The IEEE symposium on Computers and Communications*. . IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5546756, [accessed on Sep 12].

David, L, Vasconcelos, R, Alves, L, et al. (2012). A Large-scale Communication Middleware for Fleet Tracking and Management. In *Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos (SBRC 2012), Salao de Ferramentas*.

David, Lincoln, Vasconcelos, Rafael, Alves, Lucas, et al. (2012). A Communication Middleware for Scalable Real-time Mobile Collaboration. In *IEEE 21st International WETICE, Track on Adaptive and Reconfigurable Service-oriented and component-based Applications and Architectures (AROSA)*.

OMG (2010). The Real-time Publish-Subscribe (RTPS) Wire Protocol DDS Interoperability Wire Protocol Specification (DDS-RTPS). http://www.omg.org/spec/DDS-RTPS/2.1/, [accessed on Feb 25].

Rowstron, A. and Druschel, P. (2001). Pastry : Scalable , decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*.

Sistema de Coleta e Disseminação de Dados de Trânsito

Sérgio de Oliveira[1], Fernando A. Teixeira[1, 2], Daniel F. Macedo[2], André L. L. de Aquino[3], David H. S. Lima[3], Cristiano M. da Silva[1, 2], Rone I. da Silva[1], Pedro M. Shiroma[1]

[1]Campus Alto Paraopeba – Universidade Federal de São João Del Rei
MG 443, Km 7 – 36420-000 – Ouro Branco – MG – Brazil

[2]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627 – Belo Horizonte – MG – Brazil

[3]Instituto de Computação - Universidade Federal de Alagoas –
Av. Lourival Melo Mota, s/n - Maceió – AL - Brazil

{sergiool, teixeira, rone, cristiano, pshiroma}@ufsj.edu.br,
damacedo@dcc.ufmg.br, alla@ic.ufal.br, dhs.lima@gmail.com

Abstract. *This paper describes a smart traffic system employs wireless communication to collect and disseminate data. The system collects information from several sources and presents them to users. This work includes the use of the data sources, like cameras, RFID, and car tracking equipment. Further, the proposed architecture allows the direct communication between cars, or the dissemination of data by a metropolitan network.*

Resumo. *Este artigo apresenta um sistema de trânsito inteligente que emprega comunicação sem fio para a coleta e disseminação de dados. O sistema obtêm informações originadas em fontes diversas e as apresenta ao usuário. O trabalho conta com o uso de fontes de coleta de dados, como câmeras, RFID e equipamentos de rastreio de veículos. A arquitetura proposta possibilita que os eventos cheguem ao usuário em comunicação direta entre os carros, ou ainda sejam enviados por rede metropolitana.*

URL do sistema (manuais, documentação e aplicação):
http://www.winet.dcc.ufmg.br/doku.php?id=cia2:home

1. Descrição e Motivação do Problema Tratado pela Ferramenta

O aumento do número de veículos em circulação e a carência de bons serviços de transporte público têm gerado problemas para a manutenção do trânsito nos grandes centros urbanos. Embora o poder público realize investimentos na ampliação das vias de circulação, os efeitos desta ação podem ser potencializados com a otimização do tráfego nas vias já existentes. Neste sentido, uma alternativa viável é o monitoramento das vias de trânsito conjugada com a disponibilização dos dados obtidos para que os condutores sejam capazes de realizar escolhas sobre o seu trajeto. Assim, motoristas podem optar por vias com melhor fluxo, ou mesmo esperar até que o fluxo das vias melhore antes de seguir para o seu destino.

Este trabalho apresenta um sistema de monitoramento do tráfego que dissemina suas informações, de forma inteligente, aos motoristas. O sistema é composto por

módulos interligados que permitem o fluxo de dados em redes heterogêneas, bem como um aplicativo Android para acesso e consulta aos dados de trânsito.Para atingir este objetivo, diversas tecnologias de comunicação são integradas, dentre elas: i) dispositivos RFID para rastrear os veículos e identificar o fluxo em vias; ii) redes de sensores para identificar eventos diversos como inundações ou danos nas via; iii) sistemas de navegação para alertar aos motoristas sobre as condições das vias; iv) câmeras para a identificação de eventos e sua disponibilização através da rede metropolitana; v) métodos para a disseminação de informações entre os veículos e dos veículos para a infraestrutura de monitoramento (V2I e V2V).

Esse sistema foi desenvolvido no escopo da meta CDT, Coleta e Disseminação de Dados de Tráfego, parte do projeto (CIA)^2, Construindo Cidades Inteligentes, da Instrumentação dos Ambientes ao Desenvolvimento de Aplicações, financiado pelo CTIC-RNP, Centro de Pesquisa e Desenvolvimento em Tecnologias Digitais para Informação e Comunicação da Rede Nacional de Pesquisa.

Este documento encontra-se organizado da seguinte forma: A seção 2 apresenta a arquitetura do sistema. A seção 3 apresenta os módulos de coleta de dados. A seção 4 apresenta o módulo de persistência de dados, enquanto a seção 5 apresenta o módulo de interação com os condutores. A seção 6apresenta a demonstração a ser realizada no SBRC. A seção 7 apresenta as conclusões e os trabalhos futuros previstos.

2. Arquitetura do Sistema e Funcionalidades

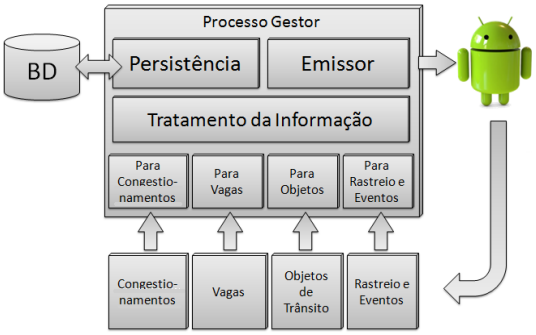


Figura 1 - Arquitetura do sistema

A fim de atender a demanda proposta, foi projetado um sistema com arquitetura em camadas, que permite realizar a comunicação entre uma infraestrutura e veículos ou diretamente entre veículos (Figura 1). A infraestrutura é composta por coletores de informações e um processo gestor que gerencia as informações recebidas e as envia para os motoristas, além de armazená-las para futuras análises. Em caso de contato entre veículos as informações podem ser repassadas de forma *ad hoc*, evitando eventuais atrasos provocados pela comunicação veículo-gestor-veículo e reduzindo a sobrecarga da rede.

Os módulos de coleta obtêm dados de diversas fontes (câmeras, etiquetas RFID, etc), e enviam dados para a camada superior. Os dados enviados contêm informações sobre o contexto (local, hora e coordenadas geográficas do dispositivo). É possível também receber informações de usuários, que informam pela interface móvel a sua posição, velocidade, que indica o fluxo na sua via, a conclusão de trajetos, e eventos sobre as condições da via como congestionamento, acidentes, animais na pista, etc.

As informações recebidas são pré-processadas por uma camada composta por adaptadores, que garantem o isolamento entre a tecnologia de coleta e a camada de tratamento da informação. A camada de tratamento da informação encapsula os dados pré-processados pelos adaptadores e repassa para a camada de negócio os objetos com informações sobre o trânsito, independentes de tecnologia de redes ou de *hardware*. A camada de negócio analisa o contexto das informações e envia as informações,

acionando o processo emissor. O emissor envia os dados para os motoristas interessados através da infraestrutura de rede metropolitana ou através de rede celular.

A ferramenta inclui, também, a comunicação veículo para veículo. Esse tipo de comunicação é importante, porque a informação demandada pode estar disponível a poucos metros e não disponível, ainda, na rede metropolitana. Ou ainda, é possível que a rede metropolitana não esteja disponível, por falta de recursos de comunicação do equipamento ou ausência de sinal. Por exemplo, um motorista pode informar sobre um acidente ou um animal na pista em uma rodovia e, nesse caso, a melhor forma de fazê-lo é enviar diretamente aos demais veículos.

Inicialmente, realizamos testes com o protocolo 802.11 padronizado para as redes veiculares, com placas adquiridas exatamente para essa finalidade. Os testes indicaram bom alcance e baixo tempo de conexão, facilitando a comunicação *ad hoc* entre os veículos. No entanto, como a ferramenta foi desenvolvida em ambiente Android, focada em *smartphones* e *tablets*, a comunicação *ad-hoc* foi implementada usando o protocolo *WifiDirect*. Esse protocolo tem a vantagem de não exigir conhecimento prévio de endereçamento ou qualquer outra informação sobre o destinatário da mensagem.

3. Módulos de Coleta

Foram desenvolvidos três módulos de coleta de dados, apresentados a seguir. Os módulos são complementares, pois os dados produzidos por eles servem para enriquecer as informações coletadas sobre as vias e sobre o trânsito. Por exemplo, os dados de uma câmera para coleta de congestionamentos podem ser agregados aos dados de contagem de dados fornecidos por uma leitora RFID, de forma a prover uma visão mais precisa da situação do trânsito no momento.

3.1 Módulo de Coleta de Congestionamentos

Em geral os engenheiros de tráfego são responsáveis por definir os intervalos de cada semáforo, o sentido das vias, as restrições de fluxo, dentro outros parâmetros a fim de obter a melhor fluidez do sistema evitando os congestionamentos. Entretanto, tal tarefa só é possível se ele dispuser de dados a respeito do estado das vias para que possa tomar as decisões corretas. Um dos principais dados necessários é o fluxo de veículos. O módulo de coleta de congestionamentos em vias urbanas desenvolvido permite estimar o estado de ocupação de uma via de rolamento.

A estimação do dado utiliza imagens da cena de interesse. Inicialmente, uma imagem é pré-processada utilizando o filtro de *floodfill* de *watershed*(Figura 2a) a fim de obtermos uma classificação de quais pontos pertencem à via e quais não pertencem (Figura 2b). Em seguida, a máscara gerada anteriormente recebe pesos que serão utilizados na contabilidade do estado da via, estes pesos variam inversamente proporcionais à distância da via em relação à câmera (Figura2c). Este processo é realizado apenas uma vez e a imagem gerada é salva para posterior processamento.

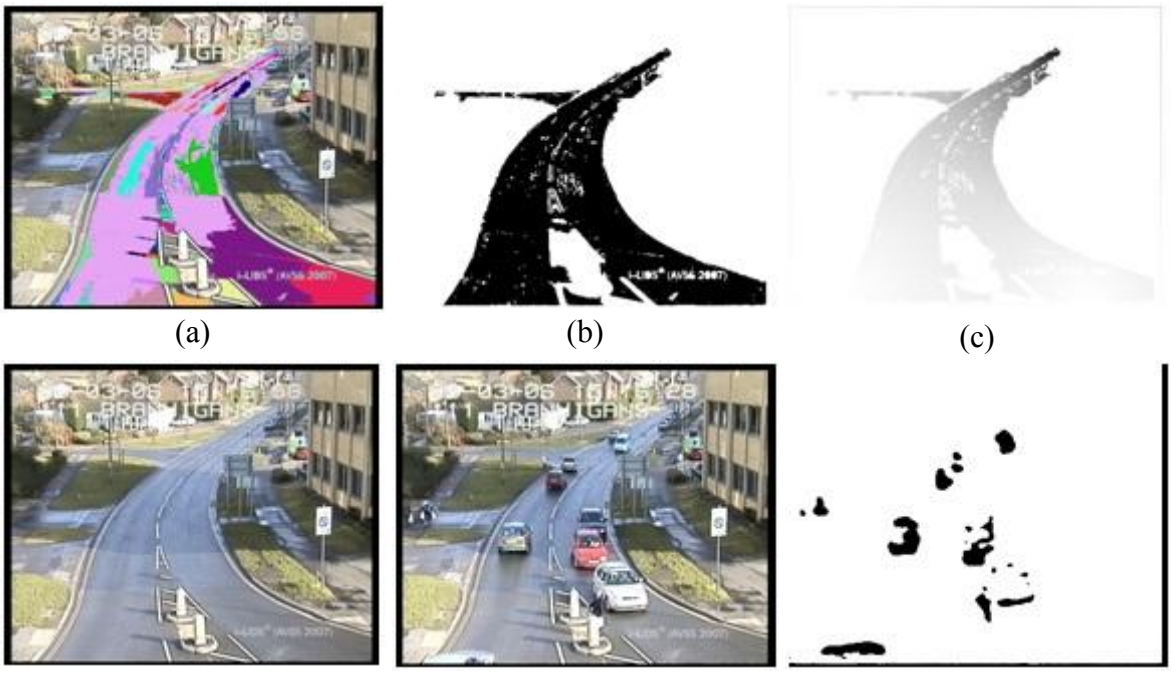


Figura 2 - Imagens de fluxo de carros

Em seguida, utilizamos a subtração de fundo para determinar quais objetos não pertencem à cena (Figura 2f), e retorna uma imagem binarizada onde 0 significa ponto pertencente ao fundo e 1 significa ponto pertencente a um objeto. Cada ponto desta imagem é multiplicado pela máscara com pesos (Figura2c) e somado, gerando um valor normalizado que reflete o grau de ocupação da via.

3.2. Módulo de Coleta de Vagas de Estacionamento

A localização de vagas utiliza técnicas de visão computacional que são usadas também em diversos tipos de coletas de dados de trânsito. O módulo de coleta de vagas para estacionamentorealiza a comunicação com as câmeras e possui como função principal o processamento das imagens obtidas das câmeras para detectar a existência ou não de possíveis vagas de estacionamento.

O fluxo de execução do módulo pode ser visualizado na Figura 3. Quando o usuário desejar localizar uma vaga de estacionamento, ele acessará a interface que enviará uma requisição para módulo de coleta informando quais os estacionamentos foram selecionados pelo usuário.

Ao receber a requisição, o módulo identifica as câmeras localizadas em cada estacionamento e envia a requisição de uma imagem do momento (*snapshot*), cada câmera capta o *snapshot* do estacionamento e o envia para o módulo que processa as imagens e decide se existem vagas disponíveis nos estacionamentos selecionados.

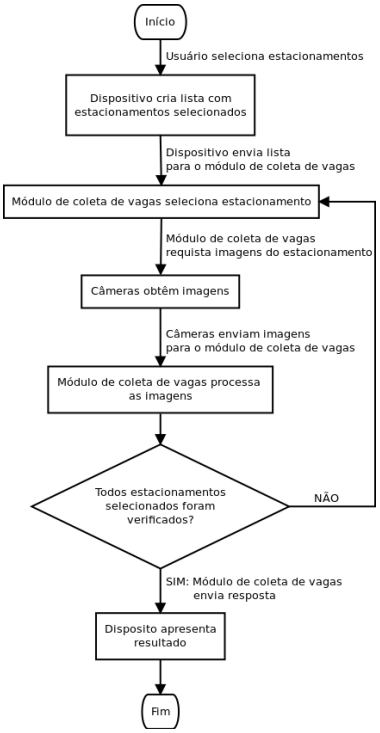


Figura 3 - Fluxo de execução

Foram utilizadas três abordagens na etapa de processamento das imagens. Os resultados obtidos em cada etapa podem ser visualizados na Figura 4. Na Figura, todas as técnicas são aplicadas ao mesmo cenário que possui vaga disponível e em todos os casos nosso sistema infere que possui vaga disponível. A abordagem inicial utiliza apenas duas etapas para o processamento das imagens, a primeira etapa é a subtração da imagem obtida pela requisição do módulo para as câmeras pela imagem do estacionamento vazio (imagem base), a próxima etapa é a binarização da imagem resultante da subtração. A binarização consiste em transformar os valores dos pontos para 0 ou 1 de acordo com um *threshold* escolhido.

A segunda abordagem consiste em realizar uma equalização nas imagens. A equalização tem como objetivo melhorar o contraste da imagem para uma melhor definição das cores primárias, as demais operações são iguais à primeira abordagem.

A terceira abordagem é formada por três etapas e tem a vantagem de não necessitar da imagem base, algo que as abordagens anteriores necessitavam. A primeira etapa é a utilização do operador de *Prewitt* (Prewitt 1970), que tem como objetivo detectar o contorno dos objetos contidos na imagem, o segundo passo é realizar a binarização da imagem obtida do passo anterior. Com a imagem binarizada, podemos observar que existem alguns buracos brancos no interior do objeto, para reduzir este efeito é aplicada uma técnica de dilatação dos pontos na imagem.

Para obtenção do resultado da aplicação das técnicas, foram utilizadas imagens geradas através de um experimento em um estacionamento de pequeno porte. Obtivemos como resultado em ordem crescente de eficácia a primeira, a segunda e a terceira abordagem. A variação da luminosidade no ambiente é um problema encontrado comum a todas as abordagens apresentadas.

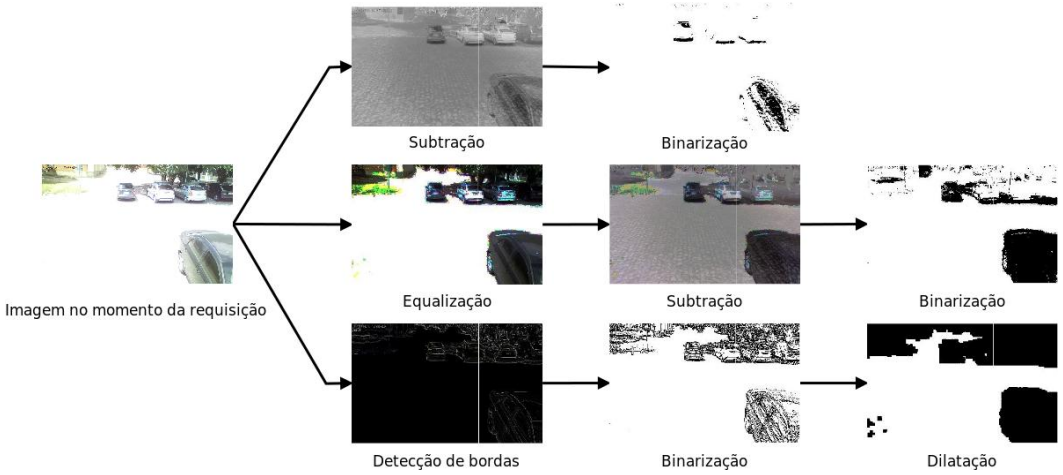


Figura 4: Abordagens de processamento de imagens

3.3. Módulo de Coleta de Objetos de Trânsito

Uma forma barata e escalável de identificar objetos de trânsito é o uso de etiquetas RFID (Sarma et al, 2003). Algumas das etiquetas RFIDs atuais custam poucos centavos de dólares, e podem ser lidas de distâncias que variam de centímetros a metros. Além disso, as etiquetas RFID não necessitam de uma fonte de energia, sendo carregadas por indução. No projeto CIA^2 empregamos etiquetas para implementar um código de barra eletrônico, de forma a identificar os objetos a serem monitorados. A legislação Brasileira já prevê a incorporação de etiquetas RFID aos carros (SINIAV, 2013), facilitando a implementação em larga escala de tal sistema. Uma vez que os carros possuem etiquetas RFID, realizamos o rastreamento e contagem de veículos.

A contagem de veículos permite aplicações tais como identificação de congestionamentos, controle de vagas em estacionamentos, mensuração de público em eventos, identificação de fluxo de veículos em vias, dentre outras. O rastreamento, por sua vez, permite a identificação de rotas, que serão utilizadas para conhecer os hábitos dos motoristas e assim melhorar o fluxo de veículos. Além disso, o rastreamento permite a identificação de congestionamentos e até mesmo a identificação de infrações por excesso de velocidade (via tempo de passagem entre dois pontos).

Empregamos leitores RFID em sinais e em cancelas, para o controle de vagas em um estacionamento e também para a contagem de veículos nas vias. O leitor RFID identifica a passagem de um veículo pela leitura da etiqueta RFID, e envia esta informação para um banco de dados.

4. Módulo de Persistência

O Módulo de Persistência tem a finalidade de armazenar e disponibilizar para o usuário as principais informações manipuladas pela ferramenta. Cada modulo de coleta por gerar eventos, os quais correspondem a acontecimentos importantes dentro do contexto da aplicação. Este módulo fornece uma visão global sobre o ambiente monitorado e possibilita a integração entre diversas aplicações diferentes.

As informações de cada evento armazenado são compostas por um conjunto de dados específicos do evento, o horário e o local onde ele ocorreu. O local de ocorrência de eventos pode ser um ponto (como a posição de um sensor que coletou uma temperatura), uma linha (como uma via de onde se tem a média de velocidade) ou um polígono (como uma região de onde se tem o número de vagas de estacionamento). Devido às características geográficas das informações a serem armazenadas, optou-se pela utilização de um banco de dados geográfico, o Postgis (Ramsey, 2011). O banco de dados criado segue o modelo apresentado na Figura 5.

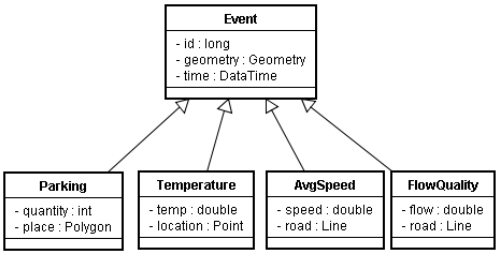


Figura 5: Modelo do banco de dados do Módulo de Integração.

No modelo, *Event* é a superclasse com os atributos comuns a todos os eventos, a qual possui o identificador do evento (*id*), o instante que o evento foi percebido (*time*) e um campo geométrico abstrato (*geometry*). O modelo apresentado contém apenas 4 subclasses (*Parking*, *Temperature*, *AvgSpeed* e *FlowQuality*), as quais representam informações de número de vagas disponíveis

em um estacionamento, temperatura, média de velocidade e qualidade do fluxo em uma via, respectivamente. Cada uma dessas classes possui um campo geométrico que implementa o campo abstrato *geometry* da superclasse *Event*. Como exemplo, temos o campo *place* da classe *Parking*, o qual é um polígono que representa a região geográfica de um estacionamento.

Além do banco de dados, o Módulo de Persistência define uma interface de comunicação com as demais aplicações do projeto. Para isso, utiliza-se objetos JSON (*JavaScript Object Notation*) (JSON, 2013).

5. Módulo de Interação com Condutores

Visando portabilidade e disponibilidade para um maior número de veículos, a interface do sistema foi desenvolvida para o sistema operacional Android 4.0. O módulo de Interação com os Condutores tem diversas telas, para dar acesso às funcionalidades da ferramenta, incluindo: i. Identificar eventos de trânsito relevantes em sua região ou em uma região de interesse, Figura 6; ii. Incluir um evento de trânsito observado em sua localização; e iii. Localizar vagas de estacionamento, Figura 7.

Figura 6 - Interface de eventos

É possível apresentar os dados de estacionamento, bem como de eventos, integrados ao aplicativo Google Maps, através de API específica. A Figura 6 apresenta essa possibilidade, indicando vagas de estacionamento disponíveis. Poderia apresentar também os eventos de trânsito observados pelo sistema, facilitando, assim, a tomada de decisões sobre qual rota deve ser escolhida.

A tela de inserção de eventos considera a inclusão de um endereço IP de destino, que não é necessário para a comunicação *WiFiDirect*. Essa tela possui botões simples para informar eventos, como animal na pista, acidente ou congestionamento. Esses botões podem ser ajustados conforme o trajeto que está sendo realizado e podem também ser inseridos na tela de navegação. O importante é que essas informações possam ser inseridas sem muito esforço e atenção do motorista.

Figura 7. Telas do cliente

6. Demonstração Planejada para o SBRC

Iremos demonstrar no SBRC o módulo de interação com os condutores, apresentado na seção anterior. Este módulo encontra-se disponível para download no URL do sistema, bem como os manuais de instalação da ferramenta em dispositivos Android.

A primeira demonstração será o controle de um estacionamento. A fim de demonstrar o funcionamento, nove etiquetas RFID serão lidas com o objetivo de

simular a entrada de carros em um estacionamento. Assim que as etiquetas forem sendo lidas pelo sensor, a quantidade de vagas disponíveis no estacionamento deverá diminuir de acordo com a quantidade de etiquetas lidas. Será mostrada na tela do dispositivo Android uma lista contendo o registro dos veículos que atualmente estão estacionados.

Outra demonstração será a comunicação *ad hoc* envolvendo dois dispositivos Android. Iremos demonstrar como dois veículos poderão se comunicar em uma estrada quando não houver cobertura 3G ou Wi-Fi. Demonstrar-se-á essa situação enviando uma mensagem de acidente diretamente para outro Android via *Wi-Fi Direct*.

A visualização das funcionalidades do sistema e a interação das partes do todo serão mostradas via cliente Android, o qual abarcará a demonstração de todo o processamento ocorrido no *middleware* e a comunicação ad hoc.

7. Conclusões e Trabalhos Futuros

Este artigo apresentou o sistema de coleta e disseminação de dados de tráfego, desenvolvido na meta CDT do projeto $(CIA)^2$, financiado pelo CTIC-RNP. A ferramenta tem por objetivo auxiliar os motoristas na solução de problemas de trânsito, identificando elementos de trânsito, como eventos, congestionamentos ou vagas para estacionamento. Sua integração com um sistema de navegação com GPS pode ser interessante para criar um novo modelo para esse tipo de sistema.

Os módulos integrantes dessa ferramenta foram desenvolvidos e integrados para o sistema Android, visando sua utilização em *tablets* e *smartphones*. Outros módulos podem ser desenvolvidos e integrados à aplicação, identificando aspectos importantes do trânsito. A arquitetura usada para a integração permite a disseminação dos eventos pela rede metropolitana, bem como *ad hoc*, em modelo de rede veicular. É possível, ainda, usar pontos de disseminação de dados para as redes veiculares, em pontos estratégicos como semáforos, postos de combustíveis, etc.

Referencias Bibliográficas

JSON, Introducing - disponível em http://www.json.org/, acesso 18 de fevereiro de 2013

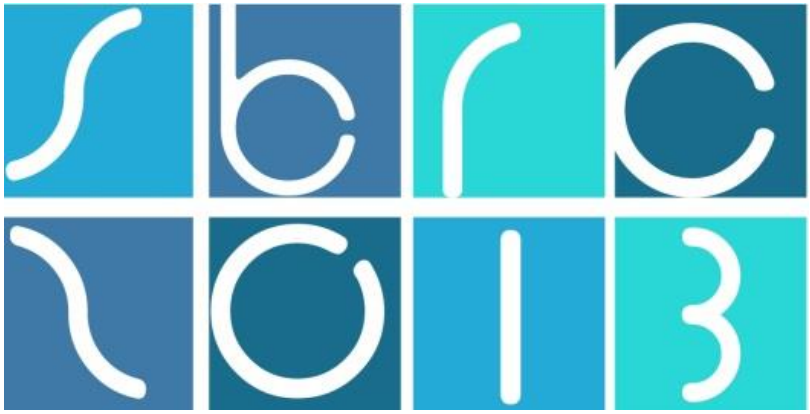
PREWITT, J. M. S. (1970) "**Object Enhancement and Extraction**" in Picture Processing and Psychopictorics.

RAMSEY, P. (2011), **Postgis manual 1.5.2**, Reflection Research Corporation,disponível em http://postgis.net/docs/manual-2.0/, acesso em 18 de fevereiro de 2013

SARMAAND, S. E.; Weisand, S. A.; Engels, D. W. - **RFID Systems Security and Privacy Implications** - 4th International Workshop onCryptographic Hardware andEmbedded Systems, 2003

SINIAV – **Sistema Nacional de Identificação Automática de Veículos**, disponível em http://siniav.net, acesso em 17 de fevereiro de 2013

TRULLOLS O., Barcelo-Ordinas J. M., Fiore M., Casetti C., Chiasserini C.-F., **"A Max Coverage Formulation for Information Dissemination in Vehicular Networks"**, IEEE WiMob, pp.154-160, 2009.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Salão de Ferramentas

Sessão Técnica 2

Computação em Nuvem e Sistemas Peer-to-Peer

Uma nuvem privada oportunista para execução de aplicações Bag-of-Tasks

Patricia Alanis[1], Abmar Barros[1], Francisco Brasileiro[1], Marcos Nóbrega[1]

[1]Laboratório de Sistemas Distribuídos (LSD)
Universidade Federal de Campina Grande (UFCG)
Caixa Postal 58.429-900 – Campina Grande – PB – Brazil

{patriciaam,abmar,marcosancj}@lsd.ufcg.edu.br, fubica@dsc.ufcg.edu.br

Abstract. *Cloud computing has emerged as a technology capable of providing elastic and flexible access to computational resources, thus meeting the needs of specific applications in a large-scale environment. However, in the typical deployment scenario of a private cloud – unlike the public cloud context, in which there is the Spot instance concept – there is no paradigm matching the needs of BoT applications, which do not require dedicated resources or QoS guarantees. In this paper we propose an opportunistic approach for the private cloud that enables the usage of idle resources in an existing computing infrastructure as a service, allowing the execution of BoT applications in a friendly and elastic fashion.*

Resumo. *A computação em nuvem surgiu como uma tecnologia capaz de prover acesso elástico e flexível a recursos computacionais, assim atendendo às necessidades específicas de aplicações em ambientes de grande escala. No entanto, no cenário típico de implantação de nuvens privadas – diferente do contexto de nuvens públicas, no qual existe o conceito de instâncias Spot – não há um paradigma que atenda à classe de aplicações BoT, que não requerem recursos dedicados ou garantias de QoS. Neste trabalho propomos uma abordagem de nuvem privada oportunista que permite o uso dos recursos ociosos de uma infraestrutura computacional preexistente como um serviço, possibilitando a execução de aplicações BoT de forma amigável e elástica.*

1. Introdução

Atualmente a Computação na Nuvem (Cloud Computing) é um dos principais paradigmas da computação, no qual recursos computacionais são oferecidos como serviços. O termo faz referência ao hardware, software e aplicações entregues como serviços através da Internet [Antonopoulos and Gillam 2010], ou de uma rede local, no caso de nuvens privadas. Neste modelo o principal objetivo é a entrega personalizada e flexível de infraestrutura computacional, software e aplicações como serviços. No modelo de nuvem pública, os usuários pagam apenas pelos recursos consumidos (o uso de espaço em disco, tempo de CPU, a transferência de dados). No modelo privado, busca-se uma melhor utilização dos recursos através da consolidação de servidores.

Os recursos na nuvem são oferecidos através de abstrações de serviços, que atualmente são divididos em três categorias principais: *Software como Serviço (SaaS), Plataforma como Serviço (PaaS) e Infraestrutura como Serviço (IaaS)*. Considerando o modelo de IaaS, motivações distintas levam o usuário a utilizar nuvens públicas ou implantar uma nuvem privada. Na perspectiva de utilização de nuvem pública, o

usuário, ao invés de ter *datacenters* próprios, solicita máquinas, redes e armazenamento sob demanda a um provedor que fornece os recursos via instâncias virtuais, geralmente utilizando protocolos de comunicação REST ou SOAP [Bhardwaj et al. 2010].

Já a implantação de um IaaS privado típico, apesar de requerer a obtenção de *hardware* dedicado, possibilita: a consolidação de servidores, que faz com que diferentes serviços de diferentes demandas de QoS executem sobre uma abstração elástica e flexível de infraestrutura que condiz com suas necessidades, diminuindo desperdício ao aumentar a utilização; e o controle e a segurança providos por uma rede interna [Hudic and Weippl 2012].

O cenário de *deployment* típico de IaaS privado faz sentido para uma série de aplicações corporativas que têm demandas altas de QoS, como serviços que devem manter uma alta taxa de disponibilidade, ou aplicações com prazo de finalização. Porém existem classes de aplicações que ainda se beneficiariam de uma infraestrutura não dedicada de nuvem, que consequentemente entregaria uma QoS mais baixo, que é o caso das aplicações BoT (*Bag-of-Tasks*).

Aplicações BoT são aplicações paralelas cujas tarefas são independentes entre si. Apesar de sua simplicidade, aplicações BoT são usadas em uma variedade de cenários, como mineração de dados, varredura de parâmetros, simulações, processamento de imagens, entre outras [Cirne et al. 2003]. Por serem independentes, a falha de uma das tarefas não causa a falha da execução como um todo, assim essas aplicações são passíveis de serem executadas em uma infraestrutura com baixas garantias de QoS, como por exemplo, nos períodos ociosos dos recursos de uma infraestrutura [Litzkow et al. 1988].

No contexto de nuvens públicas, a Amazon propõe o conceito de instâncias Spot [Amazon 2013], que, entre as soluções de IaaS, é a que provê menores garantias de QoS. Neste paradigma, o usuário dá um lance no preço de instância por hora e utiliza a instância adquirida até que seu lance seja ultrapassado. Este paradigma de baixa QoS não encontra analogia no cenário típico de implantação de nuvem privada.

Além disso, os usuários finais, de uma forma geral, não estão acostumados com a interface da nuvem, que, a priori, se resume a uma coleção de instâncias com acesso SSH. Assim, para ter suas aplicações BoT rodando na nuvem, eles precisam de um *middleware* que distribua suas tarefas na nuvem de forma paralela e elástica.

Assim, este trabalho apresenta uma abordagem de nuvem privada oportunista, que explora os recursos computacionais ociosos disponíveis em uma infraestrutura, com o propósito de construir uma plataforma tecnológica capaz de fornecer suporte a aplicações BoT, que são mais adequadas a ambientes oportunistas e heterogêneos, de uma forma amigável, dinâmica e flexível a um baixo custo.

O restante do documento está organizado como segue. A Seção 2 apresenta uma abordagem de nuvem privada oportunista e um *broker* de nuvem como soluções para os problemas apresentados nesta seção. A Seção 3 descreve os passos necessários para a demonstração desta solução, enquanto a Seção 4 contém as nossas considerações finais.

2. Arquitetura da solução

Em face ao problemas expostos na seção anterior, este trabalho apresenta uma abordagem baseada em uma nuvem privada utilizando o software Eucalyptus, porém de

uma forma oportunista. Assim, é possível descobrir os recursos computacionais ociosos que pertencem a uma infraestrutura física local. Para os laboratórios, isto significa que é possível agregar o poder computacional de *desktops* – normalmente utilizados pelos pesquisadores, mas ociosos fora do horário de trabalho – em uma nuvem privada que independe de recursos dedicados.

Além disso, este trabalho propõe um *broker* de nuvem, que recebe como entrada aplicações BoT (*Bag-of-Tasks*) no formato JSON, e é responsável por iniciar instâncias na nuvem, escalonar as tarefas para as instâncias, transferir arquivos de entrada e saída e executar as tarefas remotamente.

Nesta seção são descritas a arquitetura do Eucalyptus, as mudanças necessárias para torná-lo uma plataforma oportunista, e o projeto de um *broker* de nuvem.

2.1. Arquitetura do Eucalyptus

Eucalyptus é uma plataforma de nuvem baseada em Linux capaz de criar nuvens privadas e híbridas dentro de uma infraestrutura de TI. A arquitetura do Eucalyptus é altamente modular, com componentes internos que consistem em serviços Web tornando-os fáceis de substituir e expandir. A flexibilidade do Eucalyptus lhe permite exportar uma variedade de APIs para usuários através de ferramentas de cliente (Euca2ools). Atualmente, exporta uma interface que é compatível com o EC2 da Amazon e serviços S3, o que permite aos usuários do Eucalyptus agrupar recursos de nuvem privadas e nuvens públicas para formar uma nuvem híbrida.

2.2. Funcionalidades do Eucalyptus

Eucalyptus foi desenvolvido para suportar a computação de alto desempenho (HPC) e pode ser instalado sem modificação em todas as principais distribuições do Linux, incluindo Ubuntu, RHEL, CentOS, openSUSE e Debian. Para a implementação, gestão e manutenção das máquinas virtuais, rede e armazenamento, o Eucalyptus disponibiliza uma variedade de recursos, tais como: administração de chaves SSH, gestão de imagens, gerenciamento baseado em Linux de maquinas virtuais, gerenciamento de endereços IP, gestão de grupos de segurança, de volumes e *snapshots*.

2.3. Componentes do Eucalyptus

A nuvem Eucalyptus contém cinco tipos de componentes. Segue uma descrição detalhada de cada um deles:

- Node Controller (NC): É responsável por executar ações sobre os recursos físicos que hospedam as VMs. Ele roda em cada um dos nós e gerencia o ciclo de vida das instâncias. Interage com o hipervisor e o *cluster controller*. Tem como principais funções controlar atividades da VM, incluindo a execução, monitoração e finalização das instâncias, gerência do ponto da rede virtual, coleta de dados relacionados à disponibilidade dos recursos e entrega de relatórios para o *cluster controller*.
- Cluster Controller (CC): É responsável pela gestão do conjunto de NCs de uma infra estrutura de nuvem. Coordena o fluxo de entrada de pedidos recebidos, monitorando as informações do estado de todas as instâncias dos NCs e decide para qual NC enviar as requisições de instâncias baseado na quantidade de recursos livres disponíveis e na descrição da requisição.

- Cloud Controller (CLC): É responsável pelo processamento de solicitações recebidas de usuários finais ou administradores. Ele pode tomar decisões de programação, processos de autenticação, monitora a disponibilidade de recursos em diversos componentes da infraestrutura, incluindo nós e controladores de cluster, realiza arbitragem de recursos e decide quais *clusters* são usados para fornecer recursos.
- Storage Controller (SC): Fornece um nível de armazenamento em rede que pode ser dinamicamente anexado pelas VMs.
- Walrus fornece um mecanismo para armazenamento persistente e controle de acesso de imagens de máquinas virtuais e dados do usuário.

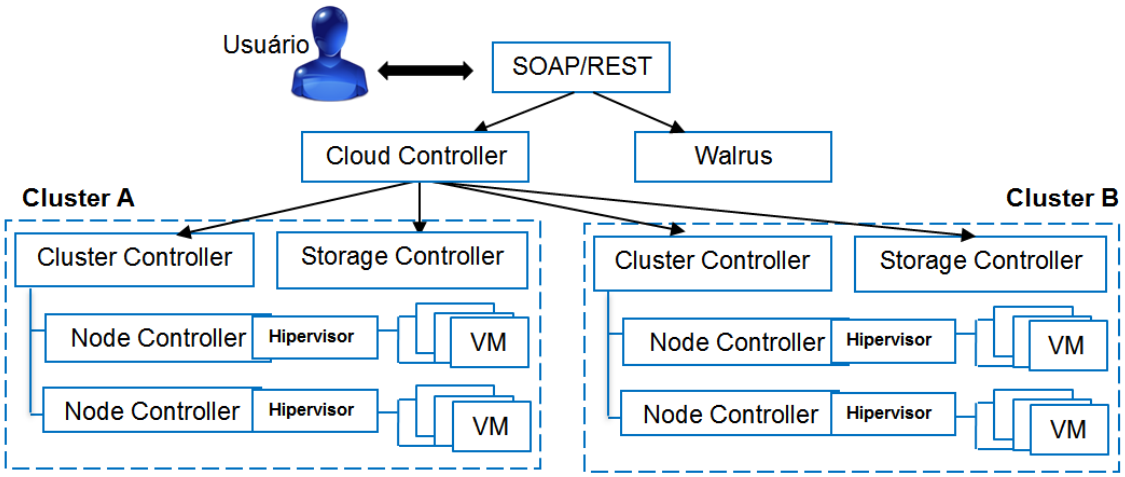


Figura 1. Arquitetura do Eucalyptus

2.4. Um *node controller* oportunista

Para adotar uma abordagem oportunista junto ao Eucalyptus, implementamos um novo *node controller*. A decisão de reimplementar este componente foi motivada pela possível heterogeneidade dos ambientes em que o NC será instalado.

Por utilizar XEN ou KVM como hipervisores, a implementação original do NC impõe uma série de requisitos de sistema para sua implantação. Esses requisitos podem ser de *software*, como versões específicas de kernel de Linux; ou de *hardware*, como extensões de virtualização (Intel VT, AMD-V). Numa infra-estrutura heterogênea, como o conjunto de *desktops* de um laboratório, esses requisitos nem sempre são alcançados., o que inviabiliza a implantação de uma nuvem privada usando a distribuição atual do Eucalyptus.

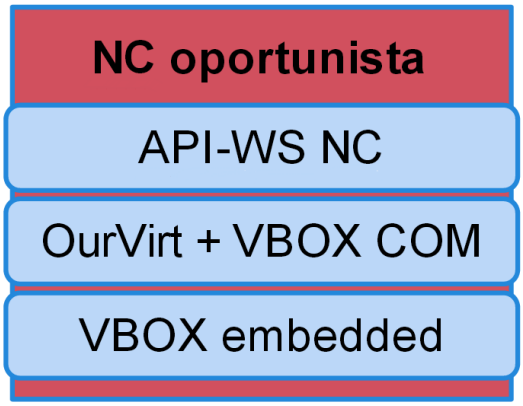


Figura 2. Arquitetura do *node controller* oportunista

Decidiu-se então reimplementar o NC em Java, uma vez que este dependeria somente da existência de uma JVM para ser executado, e utilizar VirtualBox como hipervisor, já que seus executáveis estão disponíveis para uma série de sistemas operacionais – incluindo Linux, Windows e MacOS – e não têm requisitos de *hardware* além da compatibilidade com x86.

A Figura 2 representa a arquitetura do NC oportunista apresentado neste trabalho. A camada de comunicação (API-WS NC), é implementada por meio de um *web-service*, que é forma padrão de comunicação do Eucalyptus. Para se comunicar com o hipervisor, usou-se a API OurVirt (github.com/OurGrid/OurVirt), que por sua vez utiliza a interface XPCOM do VirtualBox para gerenciar as máquinas virtuais.

Como também pode-se notar na figura 2, foi utilizada um versão *embedded* do VirtualBox em detrimento à versão distribuída no site oficial. Isso se deu pelo fato do hipervisor, e principalmente seu SDK, serem extremamentes sensíveis a mudanças inseridas entre versões. Dessa forma, era impraticável manter o NC funcional sem ter a garantia que ele se comunicaria com uma versão específica do VirtualBox.

Para determinar ociosidade, foram utilizadas estratégias diferentes para sistemas operacionais distintos. No Windows e MacOS é possível fazer chamadas de sistema que retornam quando aconteceu a última interação do usuário com o sistema. Para sistemas baseados em Linux, foi criado um script de inicialização do gerente de janelas X, que recupera periodicamente informações de atividade do próprio X e as publicam no espaço de arquivos do node controller.

2.5. Um *broker* de nuvem

A interface de nuvem, em nível de IaaS, não é amigável para o usuário final. Para ter uma aplicação executando na nuvem, o usuário precisa estimar a quantidade necessária de instâncias, criar a requisição de instâncias (que envolve escolher uma imagem, o tipo da instância, configuração de firewall, etc), monitorar os estados das instâncias, transferir arquivos e executar comandos usando conexões SSH, e repetir tarefas no caso de falha de conexão ou execução.

Para viabilizar a utilização da nuvem oportunista por pesquisadores de diferentes áreas, este trabalho apresenta um *broker* de nuvem, que é capaz de executar aplicações BoT na nuvem de forma transparente ao usuário.

A arquitetura do *broker* é constituída por dois componentes: o *broker daemon* e o *broker client*. O *broker daemon* é uma aplicação REST que executa em plano de fundo e é responsável por escalonar tarefas, requisitar instâncias, transferir arquivos e executar comandos. Já o *broker client* é a interface com usuário. Por meio deste o usuário pode submeter tarefas ao *broker* e recuperar a informação de seus estados. A Figura 3 mostra o exemplo de uma aplicação BoT que pode ser submetida ao *broker*.

```
{"name": "primes", "tasks": [
  {
    "init": [{"local": "primes.py", "remote": "primes.py"}],
    "remote": "python primes.py 2 1000 > output",
    "final": [{"local": "output", "remote": "output"}]
  },
  {
    "init": [{"local": "primes.py", "remote": "primes.py"}],
    "remote": "python primes.py 1001 2000 > output",
    "final": [{"local": "output", "remote": "output2"}]
  }
]}
```

Figura 3. Exemplo de aplicação BoT como entrada para o *broker* de nuvem

Ao receber um novo conjunto de tarefas, o *broker daemon* a armazena em memória e dispara uma execução de seu escalonador – um *workqueue* com reexecução no caso de falhas – para determinar quais instâncias alocar e quantas requisições o *broker* vai precisar fazer para executar as tarefas.

Para isso, o *broker* se comunica com o Eucalyptus através de um *webservice* compatível com o padrão Amazon, utilizando as credenciais do usuário, que estão armazenadas no arquivo de configurações do *broker*. Quando as instâncias estão disponíveis, o *broker* verifica se as portas SSH das instâncias estão abertas, transfere os arquivos de entrada via SCP, executa os comandos remotos via SSH e por fim recupera as saídas também via SCP.

Quando todas as saídas de um conjunto de tarefas são recuperadas, o *broker daemon* marca a execução como terminada, estado que pode ser constatado pelo usuário através do *broker client*.

3. Proposta de demonstração

A demonstração proposta assume a instalação e configuração dos componentes e dependências de Eucalyptus, e do *node controller* em máquinas *desktop*, para isso utilizaremos a infraestrutura do LSD (Laboratório de Sistemas Distribuídos) e notebooks *onsite*. Já o *broker* de nuvem estará executando na máquina dedicada à demonstração.

A proposta é de executar, por meio do *broker* de nuvem, uma aplicação BoT que utilize entrada em tempo real – como processamento de fotos capturadas durante o simpósio – usando a infraestrutura de nuvem oportunista instalada no LSD. Os resultados das execuções também serão exibidos em tempo real, por meio de um *script* que monitora os estados das execuções.

O código do *node controller* oportunista está disponível em https://github.com/OurGrid/opportunistic-nc, enquanto sua documentação pode ser encontrada em https://github.com/OurGrid/opportunistic-nc/wiki.

Já o *broker* de nuvem está disponível em https://github.com/OurGrid/cloudbroker, enquanto sua documentação se encontra em https://github.com/OurGrid/cloudbroker/wiki.

4. Conclusão

Empresas, instituições acadêmicas e científicas estão migrando para soluções em nuvem, por motivos que envolvem custo, flexibilidade e elasticidade. Porém nem todas as organizações conseguem custear o uso de nuvens públicas ou privadas. Este trabalho apresenta a abordagem de nuvem privada oportunista, que se apresenta como uma solução factível e útil para a classe de aplicações BoT.

As principais contribuições alcançadas neste trabalho foram o desenvolvimento de uma cloud privada oportunista capaz de monitorar os recursos de um ambiente computacional com o proposito de realocar esses recursos para aplicações Bag-of-Tasks; e a implementação de um *broker* de nuvem, que facilita o uso dessa infraestrutura por usuários de diversas áreas da ciência ou do mercado.

Com a realização deste trabalho, concluiu-se que o monitoramento e gerenciamento oportunista de uma infraestrutrua possibiliat um melhor aproveitamento dos recursos, maximizando e otimizando seu uso e, simultaneamente, amortizando os custos relacionados a sua aquisição. Além disso, contribui para a difusão do uso de ambientes oportunistas na nuvem que funcionem como suporte para o desenvolvimento de projetos corporativos, acadêmicos ou de investigação.

Referências

Antonopoulos, N. and Gillam, L. (2010). *Cloud Computing*. Springer London.

Bhardwaj, S., Jain, L. and Jain, S. (2010). Cloud computing: A study of infrastructure as a service (IAAS). *International Journal of engineering and information Technology*, v. 2, n. 1, p. 60–63.

Hudic A. and Weippl E. (2012). Private Cloud Computing: Consolidation, Virtualization, and Service-Oriented Infrastructure. Computers & Security 31(4).

Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D., Santos-neto E. and Medeiros R. (2003) Grid computing for bag of tasks applications. *Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and E-Government.*

Amazon Web Services. Amazon EC2 Spot Instances. (2013). Disponível em: <http://aws.amazon.com/pt/ec2/spot-instances/>. Acesso em: 01 mar. 2013.

M. Litzkow, M. Livny, and M. Mutka (1988). Condor: A Hunter of Idle Workstations. *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104-111, June 1988.

Just-in-Time Clouds
Uma abordagem para Federação de Clouds Privadas

Edigley Fraga[1], Jonathan Brilhante[1], Rostand Costa[1], Francisco Brasileiro[1], Pedro Bignatto[2], Diego Desani[2], Hermes Senger[2], Airton Pereira[3], Vinícius Garcia[3], Rodrigo Assad[3], Fernando Trinta[7], Ana Cristina Oliveira[5], Henryson Chagas[5], Aleciano Ferreira[5], Marco Spohn[1], Reinaldo Gomes[1], Philippe Navaux[4], Eduardo Roloff[4], Otávio Carvalho[4], Marcos Barreto[7], Raimundo Macêdo[7], Alírio Sá[7]

[1]Laboratório de Sistemas Distribuídos – Departamento de Sistemas e Computação –
Universidade Federal de Campina Grande (UFCG)
Av. Aprígio Veloso, s/n, Bloco CO, Bodocongó,
58.429-900, Campina Grande, PB

[2]Universidade Federal de São Carlos (UFSCar)
São Carlos – SP

[3]Universidade Federal de Pernambuco (UFPE)
Recife – PE

[4]Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS

[5]Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB)
Campina Grande – PB

[6]Universidade Federal do Ceará
Fortaleza – CE

[7]Universidade Federal da Bahia
Salvador – BA

{edigley,jonathan,rostand.costa,fubica}@lsd.ufcg.edu.br,
ana.oliveira@ifpb.edu.br, {reinaldo,maspohn}@dsc.ufcg.edu.br,
{senger.hermes,junaobignatto,ddesani,aleciano,balrou}@gmail.com,
{faps,vcg}@cin.ufpe.br, fernando.trinta@lia.ufc.br,
{navaux,eroloff,omcarvalho}@inf.ufrgs.br,
{marcoseb,macedo,aliriosa}@ufba.br

Abstract. *This paper describes the Just-in-Time (JiT) Cloud approach for private clouds federation. A JiT Cloud aggregates several geographically distributed JiT Data Centres (JiT DCs), providing on-demand Infrastructure-as-a-Service. The system is comprised of three distributed software components (client, cloud and dc) for the management of the federated infrastructure, the cloud users and admins, the virtual machine images (templates), and the life-cycle of virtual machines. Currently there is a JiT Cloud deployment comprised of six JiT DCs distributed among three Brazilian states (Paraíba, São Paulo e Rio Grande do Sul).*

Resumo. *Este artigo descreve a solução Just-in-Time (JiT) Cloud para a federação de clouds privadas. Uma JiT Cloud consiste na agregação de diversos JiT Data Centres (JiT DCs) geograficamente dispersos e provê infraestrutura computacional como serviço (IaaS) por meio da alocação sob demanda de máquinas virtuais. A solução é composta por três módulos de software que permitem a gerência da infraestrutura por parte de um administrador e de máquinas virtuais por parte de usuários clientes. Atualmente existe uma JiT Cloud em produção composta por seis JiT DCs distribuídos em três estados brasileiros (Paraíba, São Paulo e Rio G. do Sul).*

1. Introdução

O paradigma de computação na nuvem (*cloud computing*) permite a provisão de infraestrutura de Tecnologia da Informação sob a forma de um serviço (*infrastructure-as-a-service ou IaaS*) adquirido sob demanda. Aplicações científicas que processam grandes cargas de trabalho podem potencialmente obter benefícios a partir da elasticidade de *IaaS* para chegar a seus resultados mais rapidamente. Infelizmente, os provedores públicos atuais de *IaaS* precisam limitar a quantidade de máquinas virtuais simultaneamente alocadas para um único usuário [Costa 2011]. Para lidar com esta limitação, Costa propôs o conceito de *Just in Time (JIT) Clouds* [Costa 2013], cujos provedores incorrem em custos de propriedade somente quando os recursos são demandados pelos clientes. Este artigo apresenta um *middleware* para suporte ao conceito de *JiT Clouds*.

O restante do artigo está organizado como segue. A Seção 2 dá uma visão geral do sistema. Na Seção 3, são apresentadas as principais funcionalidades providas, enquanto que a Seção 4 contém uma descrição da arquitetura da solução. Por sua vez, a Seção 5 inclui um roteiro de demonstração das principais funcionalidades. A Seção 6 apresenta as considerações finais e os trabalhos futuros a serem realizados.

2. Visão Geral

Na abordagem de *JiT Clouds*, que objetiva a federação de recursos computacionais amortizados pertencentes a diferentes *data centres*, um *provedor* permite aos *clientes* a alocação sob demanda e de forma transparente de *máquinas virtuais* (*VMs*, do inglês *virtual machines*) que serão iniciadas em *máquinas físicas* pertencentes aos fornecedores dos recursos. Pelo uso dos recursos, o *provedor* tarifa o *usuário*, que paga pelos recursos utilizados. Por sua vez, o *provedor* realiza um repasse de parte do valor pago para o *fornecedor* dos recursos amortizados. A automação dessas etapas é realizada pelos diversos componentes que integram uma *JiT Cloud*.

Os componentes mínimos necessários são os seguintes:

- ***JiT Cloud CLI:*** Módulo cliente padrão para acesso à *JiT Cloud*. Consiste em um conjunto de comandos que permite a interação com a interface do *JiT Cloud Middleware* para utilizar os serviços providos pela *JiT Cloud*.
- ***JiT Cloud Middleware:*** O *JiT Cloud Middleware* é a implementação de um coordenador de uso e oferta de infraestrutura-como-um-serviço (IaaS, do inglês *Infrastructure-as-a-Service*) baseada na federação de recursos computacionais

amortizados. O *JiT Cloud Middleware* interage com cada agrupamento de recursos através de uma abstração chamada *JiT Data Centre (DC)*.

- ***JiT DC Middleware:*** O *JiT DC Middleware* é o responsável por gerenciar os recursos amortizados que estão sendo federados por uma *JiT Cloud*. Um *JiT DC Middleware* é vinculado a um *JiT Cloud Middleware*, que por sua vez interage com vários *JiT DCs Middleware*.

Ao emitir os comandos do módulo cliente, cada usuário tem acesso apenas às máquinas virtuais que lhe pertencem e sua identificação na *JiT Clouds* é assegurada com base em credenciais. Estas credenciais lhe são fornecidas no momento de seu registro.

De forma semelhante, o administrador da infraestrutura se utiliza do mesmo módulo. No entanto, além do acesso à API cliente, também lhe é disponibilizado o acesso à API administrativa. Por meio desta, pode-se registrar usuários, estabelecer políticas de uso e registrar *JiT DCs*. O registro de *data centres* permite a ampliação da infraestrutura resultante, sendo vital para uma *JiT Cloud*.

Todas as operações realizadas via módulo cliente são fornecidas pelas APIs básicas da *JiT Cloud*, disponibilizadas via *web services* (*WS*) e acessadas programaticamente pelo cliente padrão destas APIs, implementado no módulo *JiT Cloud CLI*. Outras formas de interação com a *JiT Cloud* que não sejam via linha de comando podem ser implementadas, a exemplo de um portal *web* (atualmente em fase de desenvolvimento) que também funcione como cliente das APIs fornecidas.

O *JiT DC Middleware* é o módulo responsável por lidar com os recursos físicos nos quais as máquinas virtuais são instanciadas, por meio de uma abstração para a *cloud* privada do fornecedor. A comunicação com a *cloud* privada é feita via API *Amazon AWS EC2* [AWS 2013], padrão *de facto* para provisão de *IaaS*. Por sua vez, a *cloud* privada é implantada com base no *middleware* de *cloud Eucalyptus* [Eucalyptus 2013], utilizado como um gerente de recursos dentro de um *JiT DC*. É de sua responsabilidade a abstração de recursos físicos em máquinas virtuais. As soluções de virtualização utilizadas são KVM (*Kernel-based Virtual Machine*) [Kivity 2007] e XEN [Barham 2003]. Embora estes monitores de *VMs* suportem máquinas com sistema operacional Windows, apenas sistemas *Unix-like* são considerados na *JiT Cloud* atualmente.

De forma similar ao *JiT Cloud Middleware*, o *JiT DC Middleware* também expõe duas APIs, uma cliente e outra com funções administrativas. A *JiT DC Client API* é utilizada para as funções básicas de provisionamento de máquinas virtuais. A comunicação entre *JiT Cloud* e *JiT DC* se dá via chamada remota de procedimentos e a autenticidade da troca de mensagens é assegurada via troca de certificados de segurança.

3. Principais Funcionalidades

Nesta seção, são apresentados os casos de uso mais significativos para a operação de uma *JiT Cloud*. Os casos de uso são agrupados em *gerência de infraestrutura, registro de usuários*, *registro de imagens de máquinas virtuais e gerência do ciclo de vida de máquinas virtuais*. É importante frisar que, no caso da *JiT Cloud*, todas as operações são voltadas para um ambiente federado, envolvendo diferentes domínios administrativos.

3.1. Gerência de infraestrutura

Para gerência da infraestrutura, é vital poder realizar o registro de novos *JiT DCs*, situação necessária para a expansão da nuvem federada. Deve ser possível também realizar a remoção de um *JiT DC*, fazendo com que o *data centre* não seja mais considerado para escalonamentos de instâncias. Para a remoção, são observadas diversas pré-condições, a exemplo de não ter *VMs* em execução.

```
# Para listar os JiT Data Centers que fazem parte da JiT Cloud:
sh bin/jit-describe-dcs

# Para adicionar um novo JiT Data Center à JiT Cloud:
sh bin/jit-register-dc -n <jit_dc_name> \
                       -e http://<ip>:<port>/services/JiTDCUserAPIService \
                       -a http://<ip>:<port>/services/JiTDCAdminAPIService

# Para remover um JiT DC da JiT Cloud:
sh bin/jit-remove-dc –n <jit_dc_name>
```

Figura 1 Comandos para gerência da infraestrutura

As operações de registro e remoção são realizadas apenas pelo administrador da *JiT Cloud*, no entanto, a listagem dos *JiT DCs* pode ser realizada por qualquer usuário, pois ele poderá utilizar a informação resultante para especificar em qual *JiT DC* deseja que suas máquinas virtuais sejam iniciadas.

3.2. Registro de usuários

De forma similar à gerência de infraestrutura, as operações envolvendo usuários são tipicamente administrativas. Do ponto de vista funcional, para o registro de um novo usuário basta um identificador (nome) e um *e-mail*. Já do ponto de vista de negócio, outros dados seriam necessários, a exemplo de número de cartão de crédito para posterior tarifação de recursos consumidos. Como resultado, é gerado um par de credenciais para o usuário (*accessKey* e *secretAccessKey*) a serem utilizadas na configuração do módulo cliente.

```
# Para listar todos os usuários da JiT Cloud:
sh bin/jit-describe-clients

# Para registrar um usuário na JiT Cloud:
sh bin/jit-register-client -n <user_name> -e <user_e-mail> [-ssh]

# Para deregistrar um usuário na JiT Cloud:
sh bin/jit-remove-client -n <user_name> -e <user_e-mail>
```

Figura 2 Comandos para gerência de usuários

Também é possível a inativação de usuário (caso em que os registros de uso e de faturação são mantidos, mas o usuário não pode mais iniciar novas *VMs*) e a listagem dos usuários cadastrados. É importante notar que a inativação pode ser um procedimento realizado automaticamente, por exemplo, na adoção de um modelo de tarifação pré-pago.

3.3. Registro de imagens de máquinas virtuais

Antes de se iniciar uma máquina virtual é preciso haver um arquivo *template*, que servirá como o sistema de arquivos básico para a *VM*. Assim como em outras soluções de *IaaS*, numa *JiT Cloud* esses *templates* são denominados *imagens de VMs*. Por exemplo, é o *template* que define se a *VM* será *Debian*, *Ubuntu* ou *CentOS*. Além do sistema de arquivos, outros arquivos são necessários, a exemplo do *kernel* que será utilizado.

```
# Para listar as imagens de máquinas virtuais registradas:
sh bin/jit-describe-images

# Para realizar o upload de uma nova imagem para o JiT Cloud Storage
sh bin/jit-upload-image -i <vm_img> -k <kvm_kernel> -r <kvm-initrd>
                        -a <vm_image_architecture>

# Para registrar na JiT Cloud uma imagem presente no JiT Cloud Storage
sh bin/jit-register-image -e <vm_image_id> -s <vm_image_size> -o <vm_image_os> \
                          -d <vm_image_description> -a <vm_image_architecture>

# Para remover uma imagem da JiT Cloud:
sh bin/jit-remove-image -i <vm_image_id>
```

Figura 3 Comandos para gerência de imagens de *vms*

Assim sendo, é possível realizar *upload* e registro de imagens de máquinas virtuais. Embora seja uma operação comumente administrativa, nada impede que o próprio usuário realize *upload* e registro de uma imagem para a *JiT Cloud*. Posteriores transferências dessa imagem para os *DCs* participantes, quando necessárias, são realizadas automaticamente pelos componentes *JiT Clouds*. Qualquer usuário pode realizar a listagem das imagens públicas presentes no catálogo de imagens e, no caso de suas próprias, realizar a remoção dessas.

3.4. Gerência do ciclo de vida de máquinas virtuais

Dado que existe infraestrutura disponível, usuário e imagens de *VMs*, pode-se partir para a provisão de recursos computacionais. Para tanto, é possível a instanciação de máquinas virtuais por usuários ativos e devidamente autenticados na *JiT Cloud*. Na requisição, é informada a imagem base, o tipo de *VM* (de acordo com a configuração de *memória, disco e cpu*) e a quantidade de instâncias a serem provisionadas. Opcionalmente, pode ser informado o *JiT DC* alvo das alocações ou a estratégia de escalonamento. As instâncias são alocadas de acordo com as restrições estabelecidas, podendo ocorrer alocações em diferentes *JiT DCs*. Ao final, são retornados os identificadores de todas as instâncias alocadas.

```
# Para listar instâncias do usuário que estejam executando na JiT Cloud:
sh bin/jit-describe-instances

# Para iniciar novas instâncias na JiT Cloud:
sh bin/jit-run-instances -t <type> -i <vm_image_id> -n <number_of_instances>

# Para terminar instâncias que estejam executando na JiT Cloud:
sh bin/jit-terminate-instances -i <instance_ids>

# Para criar um snapshot de uma máquina virtual:
sh bin/jit-create-snapshot -v <vm_id> -i <instance_ip>
```

Figura 4 Comandos para gerência do ciclo de vida de máquinas virtuais

De posse dos identificadores, que também podem ser obtidos pela listagem de instâncias do usuário, pode-se emitir um comando para encerrá-las ou, ainda, para a criação de um *snapshot*. O *snapshot* (cópia do sistema de arquivo principal) poderá ser utilizado para iniciar uma nova instância com o estado idêntico ao da instância que lhe deu origem.

4. Arquitetura da Solução

A Figura 5 apresenta os componentes e agentes de uma *JiT Cloud*. A arquitetura é composta por três macros componentes: *JiT Cloud Client (Broker e Toolkit), JiT Cloud (Middleware)* e *JiT DC (Middleware)*. O primeiro é responsável pelo acesso do cliente e do administrador à infraestrutura. O macro componente *JiT Cloud* permite a federação de recursos computacionais amortizados acessados por meio de implantações do componente *JiT DC*.

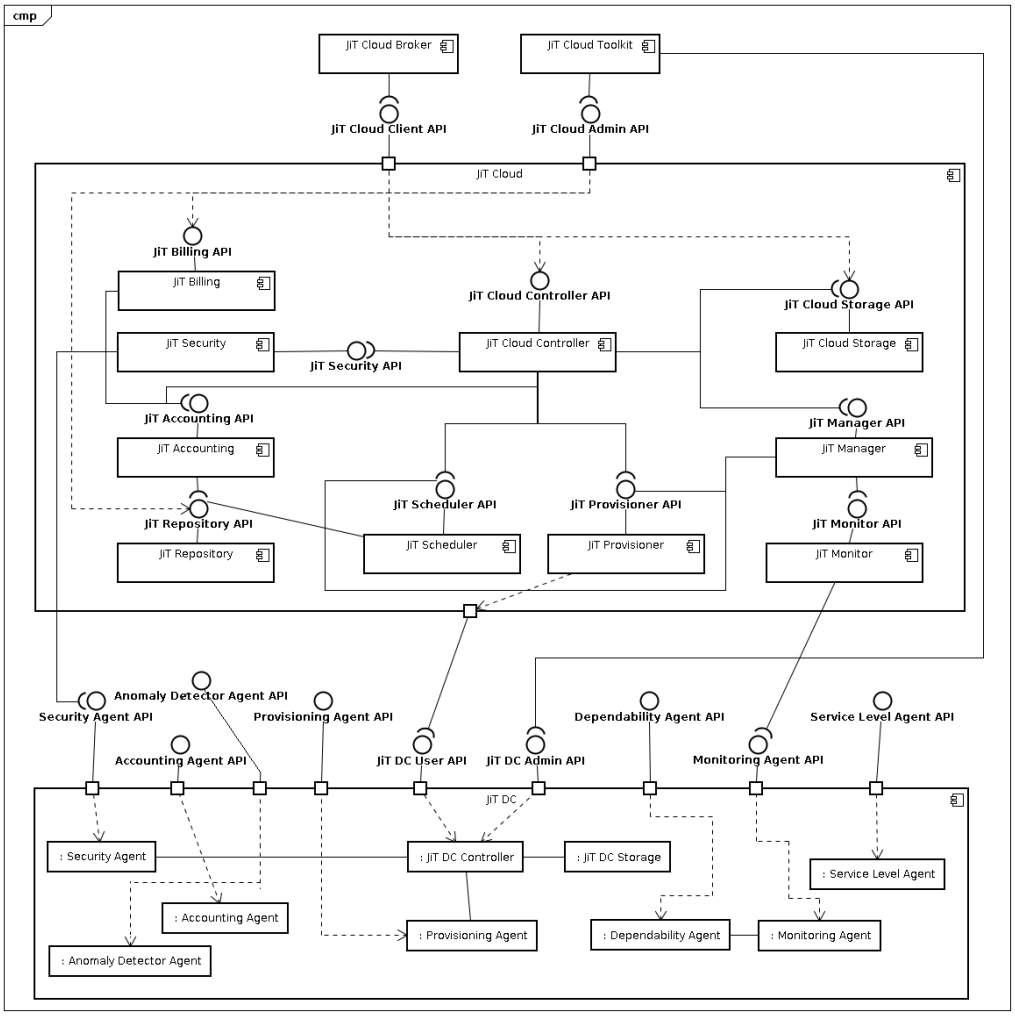


Figura 5 Componentes e Agentes da Arquitetura JiT Clouds

Na *JiT Cloud*, o componente *JiT Cloud Controller* realiza a intermediação entre as solicitações dos usuários e o efetivo cumprimento dessas solicitações, realizado nos recursos físicos existentes nos *JiT DCs*. Para tanto, este controlador se utiliza da funcionalidade especializada provida por componentes plugáveis, a exemplo de componentes de escalonamento, de segurança, tarifação e provisionamento. Por sua vez, em um *JiT DC*, o componente *JiT DC Controller* lida com os recursos físicos e se utiliza da funcionalidade de agentes especializados, tais como agentes de monitoramento, de provisionamento e de qualidade de serviço. Na arquitetura, APIs administrativas lidam com os recursos físicos (servidores), enquanto a API do cliente da *JiT Cloud* trata de entidades lógicas (máquinas virtuais e imagens).

A implementação foi realizada em Java e, para possibilitar uma arquitetura extensível, optou-se por utilizar padrões de integração (EIP - *Enterprise Integration Patterns*), com base no uso da plataforma *opensource Mule ESB* [Mule ESB 2013]. Ainda, para permitir comunicação assíncrona via troca de mensagens entre componentes e agentes, foi utilizado o padrão *AMQP* [AMQP 2013, RabbitMQ 2013]. Embora foque em (*IaaS*), em cima das APIs externas podem ser construídos serviços de alto nível, caracterizados como *Plataforma como Serviço* (*PaaS*) e *Software como Serviço* (*SaaS*).

5. Roteiro de Demonstração das Principais Funcionalidades

O *Cloud Middleware* foi implantado na infraestrutura montada na Universidade Federal de Campina Grande (UFCG), enquanto as implantações do *DC Middleware* foram realizadas na própria UFCG, do Instituto Federal da Paraíba (IFPB), e nas Universidades Federais de São Carlos (UFSCAR) e do Rio Grande do Sul (UFRGS), conforme ilustrado na Figura 6.

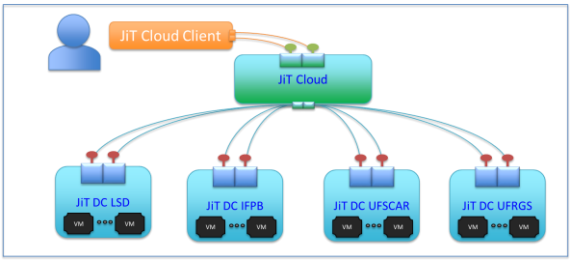


Figura 6 Ilustração da JiT Cloud em Produção

```
wget jitclouds.lsd.ufcg.edu.br/downloads/preconfigured/cloudclient-1.0.zip
unzip cloudclient-1.0-testbed.zip
cd cloudclient-1.0
export JITCLIENT_HOME=`pwd`
cd $JITCLIENT_HOME

sh bin/jit-describe-dcs
   jit_dc_lsd          http://150.165.15.131:65082/services/JiTDCUserAPIService
   ...
   jit_dc_ufrgs        http://143.54.85.52:65082/services/JiTDCUserAPIService

sh bin/jit-describe-images
   jit-b4420d5c      x86_64      Debian_6          2      OurGrid Worker
   ...
   jit-b4420dfc       i386       Debian_5          2      Snapshot Enabled

sh bin/jit-describe-vm-types
   m1.small       1       2       256
   ...
   m1.xlarge      2      20      2048

sh bin/jit-describe-instances
   i-33AC41A4      150.165.15.150         jit_dc_lsd        RUNNING         m1.small

sh bin/jit-run-instances -i jit-b4420d5c -t m1.small -n 1 -d jit_dc_ifpb
   i-48EE094F

sh bin/jit-describe-instances
   i-33AC41A4      150.165.15.150         jit_dc_lsd        RUNNING         m1.small
   i-48EE094F      192.168.20.9           jit_dc_ifpb       PENDING         m1.small

sh bin/jit-terminate-instances -i i-33AC41A4

sh bin/jit-describe-instances
   i-33AC41A4      150.165.15.150         jit_dc_lsd        SHUTTING_DOWN   m1.small
   i-48EE094F      192.168.20.9           jit_dc_ifpb       PENDING         m1.small

sh bin/jit-run-instances -i jit-b4420d5c -t m1.small -n 4
   i-1CE93B47    i-D6DF3E8A    i-34770623    i-30730629

sh bin/jit-describe-instances
   i-33AC41A4      150.165.15.150         jit_dc_lsd        TERMINATED      m1.small
   i-48EE094F      192.168.20.9           jit_dc_ifpb       RUNNING         m1.small
   i-1CE93B47      150.165.15.148         jit_dc_lsd        PENDING         m1.small
   i-D6DF3E8A      150.165.15.145         jit_dc_lsd        PENDING         m1.small
   i-34770623      192.168.20.10          jit_dc_ifpb       PENDING         m1.small
   i-30730629      192.168.20.12          jit_dc_ifpb       PENDING         m1.small
```

Figura 7 Exemplo de acesso à JiT Cloud

Será disponibilizado um *JiT Cloud Client* já apontando para a infraestrutura da *JiT Cloud* em produção e com credenciais de acesso já configuradas. Para testar o acesso, é necessária uma máquina com sistema operacional *Linux* (ou *MAC OS*) com suporte à plataforma *Java*. A Figura 7 ilustra como o *JiT Cloud Client* pode ser obtido e como pode ser usado para o caso básico do ciclo *run/describe/terminate VMs*.

6. Conclusão e Trabalhos Futuros

Para o desenvolvimento do *JiT Cloud* foram utilizadas tecnologias consolidadas e padrões de projetos de integração, permitindo flexibilidade e extensibilidade por meio de componentes plugáveis em tempo de configuração. Como forma de aumentar a elasticidade da solução, pretende-se evoluí-la para dar suporte a *JiT Clouds* entre-pares (*p2p*), combinando *clouds* federadas e grades entre-pares [OurGrid 2004].

Os produtos de *software* e manuais que fazem parte da Solução *JiT Clouds* podem ser acessados na página http://jitclouds.lsd.ufcg.edu.br.

Agradecimentos

Os autores gostariam de agradecer o apoio financeiro do Centro de Pesquisa e *Desenvolvimento em Tecnologias Digitais para Informação e Comunicação* (CTIC). Hermes Senger agradece à *Fapesp* e ao *CNPQ* pelo apoio recebido.

Referências

Costa, R., Brasileiro, F., Lemos Filho, G., Sousa, D. (2011) "Sobre a Amplitude da Elasticidade dos Atuais Provedores de Computação na Nuvem", In: Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC2011). Campo Grande, MS, Brasil, Maio 2011. Sociedade Brasileira de Computação (SBC).

Costa, R. (2013) "Just in Time Clouds: Uma Abordagem Baseada em Recursos Terceirizados para a Ampliação da Elasticidade de Provedores de Computação na Nuvem", Tese de Doutorado, Universidade Federal de Campina Grande, Março 2013.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt,I., and Warfield, A. (2003). Xen and the art of virtualization. In Proceedings of the nineteenth ACM symposium on Operating systems principles, SOSP '03, pages 164–177, New York, NY, USA. ACM.

AWS (2013). Amazon Web Services (AWS), http://aws.amazon.com.

Eucalyptus (2013). Eucalyptus Home Page, http://www.eucalyptus.com/.

Kivity, A., Kamay, Y., Laor, D., Lublin, U., and Liguori, A. (2007). KVM: The Linux Virtual Machine Monitor. In Ottawa Linux Symposium, pages 225–230.

OurGrid (2004). OurGrid Community, http://www.ourgrid.org/, 2004.

Mule ESB (2013). Mule Enterprise Service Bus http://www.mulesoft.org/, 2013

RabbitMQ (2013). Rabbit Message Queue, http://www.rabbitmq.com/, 2013

AMQP (2013). Advanced Message Queuing Protocol, http:// www.amqp.org/, 2013

OpenBox: Ferramenta de Alta Disponibilidade e de Gerenciamento de Cópias de Segurança de Dados

Victor Dias de Oliveira[1,2], **Matheus Bandini**[1], **Fábio Licht**[2], **Bruno Schulze**[1], **Antonio Mury**[1]

[1]Laboratório Nacional de Computação Científica (LNCC)
Av. Getúlio Vargas, 333 – Quitandinha – 25660-004 – Petrópolis – RJ – Brasil

[2]Universidade Estácio de Sá (UNESA)
Rua Bingen, 45 – Bingen – 25651-075 – Petrópolis – RJ – Brasil

Abstract. *This paper presents a tool for distributed virtualized environments, aiming to ensure the high availability of information in Database Management Systems (DBMS) in the context of cloud computing. The tool is capable of monitoring high availability systems, assigning functions for the better management of backups files, with the ability of restoring them to different DBMS available for use.*

Resumo. *Este artigo apresenta uma ferramenta para ambientes distribuídos virtualizados, com o objetivo de garantir a alta disponibilidade de informações por parte dos Sistemas de Gerenciamento de Banco de Dados (SGBD) no contexto de computação em nuvem. A ferramenta é capaz de monitorar um sistema de alta disponibilidade, atribuindo funções para gerenciar a criação de cópias de segurança, com a capacidade de restauração a diversos SGBDS existentes no mercado.*

1. Introdução

O avanço tecnológico das últimas décadas proporcionou diversas abordagens fundamentadas em computação para a solução de sistemas complexos, desde a ampla utilização de supercomputadores até a difusão da Internet com conexão de alta velocidade, permitindo que ambientes de computação paralela e distribuída de alto desempenho pudessem ser idealizados e implantados, garantindo a alta disponibilidade dos sistemas computacionais.

Aliado a isto, o avanço das tecnologias de virtualização de recursos computacionais e de redes vieram a contribuir para o surgimento do conceito de computação em nuvem, possibilitando a replicação de infraestruturas (IaaS – *Infrastructure-as-a-Service*), plataformas (PaaS – *Platform-as-a-Service*) e aplicações (SaaS – *Software-as-a-Service*), por meio da virtualização de ambientes dedicados específicos, atendendo às demandas mais pontuais em função das necessidades dos usuários [Vecchiola et al. 2009].

Devido à demanda crescente no uso de ambientes computacionais de alto desempenho e, consequentemente, à grande massa de dados gerada pelas aplicações nesses ambientes, surge a necessidade de gerenciar essas informações de forma a garantir a sua segurança e integridade [Wiesmann et al. 2000], uma vez que a perda ou inconsistência de um banco de dados pode acarretar em perdas significativas de informações importantes e até mesmo perdas monetárias, no caso do mundo comercial [Ekanayake and Fox 2009].

No caso da alta disponibilidade, ela pode ser alcançada em diversos níveis, incluindo os de aplicação, infraestrutura, *datacenter* e até o de redundância geográfica [Filho 2004]. Na sua forma mais básica, configurações de alta disponibilidade em infraestrutura são constituídas por: balanceadores de carga, servidores *web* e servidores de banco de dados.

Dentro deste contexto, este trabalho tem como objetivos principais: 1) a implementação de um sistema robusto de banco de dados, capaz de garantir o armazenamento seguro das informações para ambientes de computação distribuída, de modo que uma falha qualquer não ocasione a interrupção de seu funcionamento; 2) garantir esse funcionamento através de *middlewares* de replicação de bancos de dados, de modo a ser transparente tanto para o usuário quanto para a aplicação e; 3) apresentar a ferramenta desenvolvida para gerenciar a geração de cópias de segurança das informações, compactando-as e permitindo a restauração para mais de um SGBD.

Este trabalho encontra-se organizado da seguinte forma: a seção 2 apresenta trabalhos relacionados com ferramentas de gerenciamento de cópias de seguranças e restauração; a seção 3 descreve as funcionalidades e a arquitetura da ferramenta apresentada neste trabalho e a seção 4 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Apresentaremos a seguir alguns dos trabalhos existentes relacionados ao conceito de *backup*, restauração e gerenciamento de banco de dados, utilizados na proposta apresentada.

2.1. Ferramentas de Apoio e Gerenciamento de Banco de Dados

O PgDump é um utilitário capaz de criar e gerenciar cópias de segurança de um banco de dados PostgreSQL. É possível realizar cópias de segurança consistentes, mesmo que o banco de dados esteja sendo utilizado. Os *backups* podem ser feitos no formato de *script*, personalizado ou tar. As cópias de segurança no formato de *script* são arquivos de texto puro, contendo os comandos SQL necessários para reconstruir o banco de dados no estado em que este se encontrava quando foi salvo [Momjian 2001].

O S.O.S Backup EasyDB é um software que realiza *backups online* dos bancos de dados MySQL e PostgreSQL. Tem como características principais: 1) *Backup online* para as bases de dados MySQL e PostgreSQL e; 2) Gerenciamento remoto pelo S.O.S Backup Manager; 3) Compressão de dados no formato ZIP. Entretanto, algumas das suas limitações consistem em ser um *software* que funcione apenas em sistemas operacionais Windows, e por apenas apresentar uma versão paga do sistema [Virtos, SOSBackup 2010].

Bacula é um conjunto de programas que permite administrar *backup*, restauração e verificação dos dados de computadores em uma rede de sistemas mistos. Tem como características os seguintes itens: 1)Módulos para diferentes sistemas operacionais; 2)Infinidade de recursos para a customização de *backups*; 3) Ferramenta de *backup* multi-banco-de-dados; 4) Suporte para MySQL, PostgreSQL e SQLite. Entretanto, essa ferramenta realiza suas cópias de segurança através da cópia do diretório aonde os SGBDs armazenam toda a estrutura do banco de dados. Isso não permite que os dados de um SGBD sejam migrados para outro. [Di Francesco and Sweden 2012]

As ferramentas citadas demonstram ser bastante maduras em função de seu tempo de uso e também pela quantidade de investidores e desenvolvedores envolvidos. Estas ferramentas são complexas e demandam um estudo aprofundado por parte dos interessados. Além disso, não oferecem suporte à alta disponibilidade.

Procurou-se neste trabalho disponibilizar uma ferramenta simples, capaz de garantir a alta disponibilidade das informações e, ao mesmo tempo, facilitar o gerenciamento das cópias de segurança, por meio da transposição de dados entre diferentes SGBDs. Mais detalhes serão apresentados na Seção 3.

3. Funcionalidades e Arquitetura da Ferramenta

Nesta seção serão descritos detalhes da arquitetura, do funcionamento e benefícios do sistema proposto, bem como a ferramenta de gerenciamento de rotinas de criação de cópias de segurança - OpenBox (**disponível para acesso através da URL: http://comcidisv01.lncc.br/openbox**).

3.1. Ferramenta OpenBox

O OpenBox foi desenvolvido integrando as ferramentas apresentadas na Seção 2 com o objetivo de oferecer um ambiente funcional, robusto e de fácil uso, no âmbito de gerenciamento de banco de dados, podendo ser utilizado para o apoio de aplicações no contexto de Computação em Nuvem. O sistema é subdividido em dois módulos: No primeiro é proposto, no contexto de Computação em Nuvem, um serviço de *IaaS* com o objetivo de garantir a alta disponibilidade de dados em ambiente que utilize o SGBD PostgreSQL. O segundo tem por objetivo gerenciar remotamente os recursos disponibilizados, capaz de realizar a cópia de segurança das informações, a restauração e o agendamento de tarefas de forma agregada, simplificando a operação destas tarefas por parte do usuário. A Figura 1 demostra de maneira simplificada a arquitetura do sistema OpenBox.

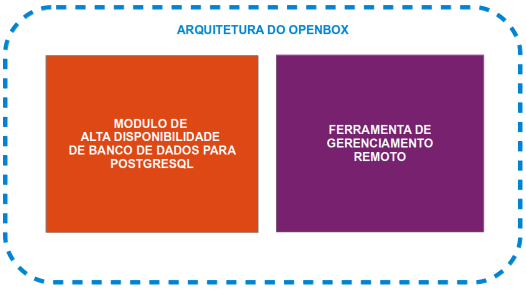


Figura 1. Arquitetura do Sistema

O primeiro módulo, denominado ambiente de alta disponibilidade, tem por objetivo tornar o armazenamento de dados robusto, consistente e seguro, utilizando-se de um *middleware* cuja função é se comportar como uma camada de abstração entre os usuários e o *cluster* de banco de dados que compõem o sistema. Para a obter essa abstração foi utilizada a ferramenta PgPool-II [Perkov et al. 2011].

O segundo módulo é responsável pelo gerenciamento do ambiente de alta disponibilidade provendo informações sobre cada máquina do sistema, como por exemplo, a quantidade de memória consumida e a disponível, a capacidade em disco, os dados trafegados pela interface de rede, entre outras funcionalidades. Essas informações são obtidas

pelo uso da ferramenta ICAL (*Intelligent Collector Algorithm*) [Alvarenga et al. 2012]. Entretanto, dada a necessidade de cópias de segurança automatizadas, a ferramenta também é capaz de gerar e agendar cópias padronizadas e restaurá-las para SGBDs diferentes (no momento para PostgreSQL e MySQL), tornando-se assim, uma ponte para a troca de informações. Todos esses recursos são acessadas a partir de uma interface *web* com funcionalidades específicas de acordo com o nível de permissão do usuário.

3.2. Módulo I - Ambiente de Alta Disponibilidade

Para garantir a segurança das informações, optou-se por utilizar o *middleware* de replicação de bases de dados denominado PgPool-II. Suas características principais, que influenciaram o seu uso neste trabalho, são: o modo de replicação síncrona, o balanceamento de carga, o *pool* de conexões, a capacidade de detecção de falhas, a remoção automática do nó defeituoso e a execução de *scripts* personalizados na realocação daquele nó que apresentava falhas.

De forma prática, a replicação do banco de dados com PgPool-II é composto por 3 máquinas virtuais. A primeira é denominada nó principal, e as outras duas são as máquinas que mantém o servidor de banco de dados. Todos os nós representados pelas máquinas virtuais fazem parte da infraestrutura virtualizada do ambiente (*IaaS*).

O nó principal, o servidor que gerencia o ambiente através do PgPool, é a máquina intermediária entre a aplicação e os servidores PostgreSQL. Este age de forma transparente e suporta até 128 nós de replicação. Quando uma requisição é realizada por meio de uma aplicação, essa ação será retransmitida do nó principal até os servidores PostgreSQL. Embora o PgPool-II seja visto pela aplicação como um servidor PostgreSQL real, os servidores de *backend* o veem como um de seus clientes. Isso acontece devido à característica de transparência do *middleware*.

Se um banco de dados é replicado (configurado no modo de replicação), quando realizada uma consulta em qualquer um dos servidores, o resultado é sempre o mesmo, pois as bases de dados são mantidas sincronizadas. Dada essa característica, foi possível configurar o PgPool-II para aproveitar a funcionalidade de replicação, a fim de reduzir a carga em cada servidor PostgreSQL. Ele faz isso através da distribuição de consultas de leitura (SELECT) entre os servidores disponíveis, melhorando o rendimento global do sistema. Em um cenário ideal, o desempenho de leitura melhora a medida em que número de servidores de banco de dados aumenta. Esse método, denominado de Balanceamento de Carga, é indicado para um ambiente onde há um grande número de usuários que executam muitas transações de leitura ao mesmo tempo. A Figura 2 apresenta a estrutura de alta disponibilidade do sistema.

3.3. Módulo II - Interface do Gerenciamento de Cópia de Segurança

O módulo II foi desenvolvido com o objetivo de gerar cópias de segurança, gravando os dados selecionados pelo administrador em um formato padrão de arquivo definido pelo sistema de forma segura. Além disso, permite a restauração desses dados não apenas aos SGBDs de origem como também para outros, tais como o MySQL, SQLlite, SQLServer, dentre outros, tornando o OpenBox uma ponte capaz de interligar diversos sistemas gerenciadores de banco de dados.

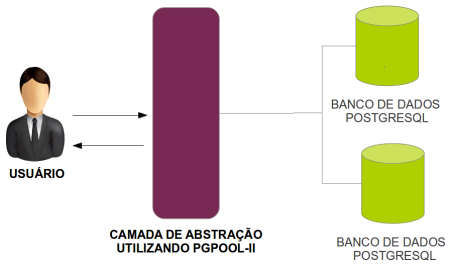


Figura 2. Ambiente de alta disponibilidade configurado pelo OpenBox

Este módulo é uma aplicação *SaaS* (*software* como serviço) no contexto de Computacional em Nuvem, permite um melhor gerenciamento das cópias de segurança, possibilitando a sua restauração para diferentes SGBDs, reduzindo o tamanho dos arquivos através da compactação e permitindo *backups* agendados. Os *backups* são armazenados na nuvem pelo OpenBox ficando a disposição do administrador sempre que necessário. O OpenBox foi desenvolvido utilizando a linguagem de programação JAVA em conjunto com a biblioteca JQuery (biblioteca javascript para realizar chamadas ao sistema de maneira assíncrona).

3.3.1. Núcleo do Sistema

O núcleo do OpenBox é o sistema encarregado de descobrir todas as características de um banco de dados. Ele tem como função varrer a estrutura de um SGBD, determinando as tabelas existentes, as estruturas de cada uma, suas chaves e metadados.

Para executar estas funções em vários SGBDs, foi desenvolvida uma ferramenta para criar uma camada de abstração entre os bancos de dados, denominadas de dialetos. Como cada sistema de gerenciamento de banco de dados possui suas funções e modo de interagir com seus respectivos banco de dados, o OpenBox, através da criação do dialetos para cada SGBD, permite que os diversos bancos de dados troquem informações entre si, funcionando como um canal de troca de mensagens entre eles.

O gerenciamento de arquivos, o gerenciamento de tarefas, a compactação das cópias de segurança, a funcionalidade de *backup* e a restauração são algumas das funcionalidades executadas pelo o núcleo desenvolvido nesse projeto.

3.3.2. Dialetos

Um dialeto é a maneira pela qual uma língua é falada em uma região específica. Logo, trata-se de uma variante linguística sobre um mesmo elemento, constituída de características próprias.

Os SGBDs fazem uso da linguagem estruturada denominada SQL. Esta é a linguagem padrão para interagir com bancos de dados relacionais. Entretanto, cada SGBD possui maneiras próprias de interagir com seu respectivo sistema. Nesse momento, o OpenBox (por meio da camada de dialetos) tem a capacidade da tradução dessas características próprias dos SGBDs, tornando possível a troca de dados entre o PostgreSQL e

o MySQL (podendo ser estendido para outros SGBDs).

3.3.3. *Pool* de Conexões

O *pool* de conexões é a parte encarregada de gerenciar todas as conexões criadas no OpenBox. Com essa funcionalidade é possível estabelecer conexões com PostgreSQL e o MySQL. Abrir conexões de modo excessivo em banco de dados é um processo custoso, podendo causar a diminuição da capacidade de atender a solicitações dos usuário. Por esse motivo, o OpenBox limita para até 5 o número máximo de conexões por usuário.

3.3.4. Cópia de Segurança

A cópia de segurança utiliza-se da estrutura padrão gerada em XML, formato para a criação de documentos com dados organizados de forma hierárquica. Através do uso de arquivos XML, o OpenBox é capaz de criar interfaces com outros bancos distintos permitindo então conter os mesmos dados. A hierarquia da estrutura XML é composta primeiramente da seguinte ordem: nome da base de origem, nome da tabela e finalizando, cada linha da tabela é separada pelo nome da coluna e seus respectivos valores.

3.3.5. Gerenciador de Tarefas

O gerenciador de tarefas, é o modulo capaz de identificar ações realizadas no ambiente OpenBox. Ao realizar um *backup* ou uma restauração, o gerenciador tem como função verificar o status das tarefas procurando saber se a mesma foi concluída com êxito, se está em andamento ou se em algum momento houve uma falha. Além de apresentar informações tais como: o nome do banco de dados, a tabela envolvida, o SGBD originado e o endereço IP.

3.3.6. Gerenciador de Arquivos

O gerenciador de arquivos é um programa *web* que serve para gerenciar e organizar os arquivos armazenados na nuvem do OpenBox. Tem como funções em sua interface: 1) Excluir as cópias de segurança; 2) Obter informações extras dos arquivos; 3) Compactar os backups; 4) Visualizar os arquivos no formato XML na web e; 5) Selecionar um cópia de segurança para restauração.

3.3.7. Restauração da Cópia de Segurança

Para utilizar-se da restauração de dados entre SGBDs diferentes oferecida pelo OpenBox, primeiramente o usuário deverá estabelecer uma conexão principal da qual fará uso. Após esta configuração, o usuário deve selecionar qual cópia de segurança deseja restaurar.

A restauração para uma nova tabela é realizada através da verificação da existência do nome digitado com as tabelas existentes no banco de dados. Entretanto, caso o nome já se encontra utilizado, o usuário é informado e a restauração é desabilitada. Caso

contrário, o núcleo do OpenBox se encarrega de gerar a tabela automaticamente e inserir as informações. Com o uso dos dialetos, é possível criar e executar comandos SQL automaticamente em tempo de execução conforme as características de cada SGBD.

3.3.8. Agendamento de Backup

O agendamento de *backup* é uma funcionalidade da ferramenta OpenBox na qual é especificada a tabela cuja cópia de segurança deseja-se criar, bem como a data e a hora para execução da tarefa. O OpenBox faz uso do Quartz, biblioteca desenvolvida em java para agendamentos de tarefas.

4. Conclusões e Trabalhos Futuros

Esta ferramenta foi desenvolvida com o objetivo de garantir a alta disponibilidade através de *clusters* de banco de dados, utilizando os recursos virtualizados disponíveis na nuvem.

O OpenBox é uma ferramenta que possibilita a recuperação de bases inconsistentes, utiliza-se do balanceamento de carga entre os servidores com o objetivo de obter ganho de desempenho, além de ser capaz de gerenciar a criação de cópias de segurança, de modo a permitir sua restauração para diferentes tipos de SGBDs, de forma simples, robusta, com isso apresenta-se como uma ferramenta de assessoramento aos seus usuários contribuindo para tomadas de decisões relativas à geração de *backup*.

Para monitorar e gerenciar os recursos disponíveis no ambiente, utiliza-se a ferramenta de monitoramento denominada ICAL (*Intelligent Collector Algorithm*), que devido a sua flexibilidade, pode de ser configurado de acordo com a situação e o nível de detalhamento das informações desejadas, reduzindo a sobrecarga dos servidores.

Como propostas para trabalhos futuros, surgem como possibilidades: a integração de mais SGBDs à ferramenta de gerenciamento de cópias de segurança, através do desenvolvendo de novos dialetos para tradução das características específicas de cada banco de dados. Atualmente, os esforços estão concentrados em desenvolver dialetos para os gerenciadores de banco de dados SQLlite e o SQLServer. Além disso, a integração também, da ferramenta Linux Heartbeats, cuja função é monitorar o status do servidor e permitir que uma nova instância do nó principal assuma as funções em caso de falhas. Pretende-se também avaliar quantitativamente o ganho de desempenho resultante do balanceamento de carga obtido (item 2 parágrafo seguinte).

Dentre as principais contribuições alcançadas neste trabalho, destacam-se: 1) o desenvolvimento de uma ferramenta segura e de simples utilização, capaz de utilizar um *cluster* de banco de dados para garantir a alta disponibilidade das informações, por meio do uso de ferramentas de replicação de dados e balanceamento de carga. 2) com a realização de testes envolvendo o balanceamento de carga proveniente das aplicações, foi possível observar um ganho de desempenho dos servidores, devido à distribuição das requisições concorrentes; 3) a gerência de cópias de segurança do OpenBox possibilita, através de uma interface de fácil uso, o seu gerenciamento e a criação de uma ponte de troca de dados entre diferentes SGBDs; 3) o monitoramento dos recursos do *cluster* de alta disponibilidade, feito com o auxílio da ferramenta ICAL, oferece informações a respeito do consumo de CPU, memória e disco por parte dos serviços de banco de dados;

4) o desenvolvimento de Dialetos permite a integração de diversos sistemas gerenciadores de banco de dados e; 5) a incorporação de um módulo capaz de realizar *backups* compactados torna possível a redução do espaço em disco necessário para armazenar as informações.

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 - E-26/112.221/2008).

Referências

Alvarenga, V., Yokoyama, D., Barbosa, J., Bandini, M., Schulze, B., and Mury, A. (2012). Ferramenta para monitoramento e gerência de ambientes virtuais de computação de alto desempenho. In *Salão de Ferramentas / XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 888–895.

Di Francesco, P. and Sweden, V. (2012). Design and implementation of a mlfq scheduler for the bacula backup software.

Ekanayake, J. and Fox, G. (2009). High performance parallel computing with clouds and cloud technologies. In *Cloud Computing - 1st Intl. Conference, CloudComp 2009, Munique, Alemanha, Out. 19-21, 2009 Revised Selected Papers*, volume 34 of *LNICST, Social Informatics and Telecommunications Engineering*, pages 20–38. Springer.

Filho, N. A. P. (2004). Serviços de pertinência para clusters de alta disponibilidade. Technical report, Universidade de São Paulo, Disponível em: `http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04102004-104700/publico/dissertacao.pdf`.

Momjian, B. (2001). *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley.

Perkov, L., Pavkovic, N., and Petrovic, J. (2011). High-availability using open source software. In *MIPRO, 2011 Proceedings of the 34th International Convention*, pages 167–170. IEEE.

Vecchiola, C., P, S., and Buyya, R. (2009). High-performance cloud computing: A view of scientific applications. In *ISPAN*, pages 4–16. IEEE CS.

Virtos, SOSBackup (2010). Virtos: Backing Up Data at Lightning Speed. Disponivel em `http://software.intel.com/en-us/articles/virtos-backing-up-data-at-lightning-speed`.

Wiesmann, M., Pedone, F., Schiper, A., Kemme, B., and Alonso, G. (2000). Understanding replication in databases and distributed systems. *In Proceedings of the 20th International Conference on Distributed Computing Systems, 2000*, 1:464–474.

TVPP: A Research Oriented P2P Live Streaming System

João F. A. e Oliveira[1], Rodrigo S. M. Viana[1], Alex B. Vieira[2], Marcus V. M. Rocha[3], Sérgio V. A. Campos[1]

[1]Departamento de Ciência da Computação – UFMG

[2]Departamento de Ciência da Computação – UFJF

[3]Assembléia Legislativa de Minas Gerais

Abstract. *In the past five to six years, peer to peer live streaming has grown to support millions of users but the behavior of such systems is still not fully understood. Researchers experiment and try to understand popular/commercial platforms with proprietary code by crawling the network with one or more nodes, through network traffic analysis, reverse engineering and by recreating partial network graphs. However, most existing systems do not provide tools for analyzing the behavior in a comprehensive way, and these experiments cannot fully explain how this technology works. We introduce a tool called TVPP that supports peer-to-peer live streaming, and is being developed with the purpose of answering questions about how the underlying technology behaves and how it can be improved by allowing fine tuning of system parameters such as neighborhood size and bandwidth limits, and tests over alternative algorithms implementation. Further, it allows experimental data acquisition without additional network traffic analyzers which can impact experimental results.*

1. Introduction

Nowadays, P2P live streaming systems are widely available and extensively used by millions of users across many different architectures. An increase in the number of publications, in the past five years that discuss performance issues, load behavior, cross-platform analysis and many more topics related to P2P live streaming systems, shows that researchers are still trying to better understand the details behind these systems.

A major issue remains which makes it hard to study real life systems: most popular live systems, such as TvAnts[1], UUSee[2], SopCast[3], PPLive[4] and PPStream[5] are commercial applications, with no publicly available source code, which makes it harder for researchers to gather useful data or log files. Most studies targeting these systems deal with a black box and must rely on educated guesses with regard to system architecture, protocols and internals. Crawling the network, analyzing traffic, recreating partial network graphs are examples of methods frequently used to infer system structure and behavior [Vieira et al. 2009, Horvath et al. 2008, Silverston et al. 2009, Ali et al. 2006, Tang et al. 2009]. Moreover, in such a scenario where systems do not facilitate third party testing or reverse engineering, control of key protocol parameters, such as partnership or bandwidth limits, media buffer size, chunk scheduling and partner maintenance strategies, which would allow deeper experimentation and analysis, are absent.

[1]tvants.en.softronic.com;[2]www.uusee.com;[3]www.sopcast.org;[4]www.synacat.com;[5]www.ppstream.com

In this paper we present a new P2P live streaming system: TVPP. TVPP is designed to provide a similar service to popular proprietary commercial systems, such as SopCast, but with a few advantages for researchers.

- easy data acquisition: no need for additional network traffic analyzers
- configurability: several key parameters to experiment with
- modularity: easier testing of new algorithms
- collection of arbitrary data: one can change code to output what he needs
- exact analysis: one can look at the code to understand results

Through those advantages TVPP allows answering key questions, such as:

- what if there is only a limited number of partners for each peer?
- what if *x*% of peers are not willing or able to contribute with upstream bandwidth?
- what if downstream rate is reduced for some time?
- what if *x*% of some peers are receiving contents A and sending contents B (e.g. contents B could be contents A with added user comments)?

TVPP focuses on addressing the questions mentioned before and many other key design aspects of P2P live streaming systems. Also, TVPP source code is available for experimentation and modification. Through this paper we present the TVPP as a tool describing its design choices, structure, and what can be done with it. We have already conducted tests configuring TVPP with parameters to mimic SopCast behavior. The results show that, in this case, our system has performance comparable to that of SopCast.

The remainder of this paper is organized as follows. Several design choices for TVPP are mentioned in Section 2. After that we describe a quick "how to use" TVPP, detailing a few parameters available. In the next section we discuss related work. Section 5 expresses our conclusions, remarks and future work. And, finally, Section 6 details the proposed demonstration that will show our tool features.

2. The TVPP Design

TVPP is a mesh-pull P2P live streaming system, using the structures similar to those of the most popular platforms [Sentinelli et al. 2007]. An interesting feature is that TVPP has been designed to be expandable, as described below, since it aims to experiment with a wide range of questions, and therefore must be able to include additional monitoring mechanisms. Some of the key mechanisms that have been developed to maintain this functionality are particularly interesting to explore in further detail, such as chunk scheduling, overlay maintenance and logging. In Section 3 we explores a full list of configurable parameters present on TVPP.

2.1. Overlay Maintenance

In mesh-pull applications, peers are organized in a mesh like network, without any hierarchy, where each peer only requests or sends data to its partners. This design has the advantage of being scalable and fault resilient [Fodor and Dan 2007, Hei et al. 2008].

In TVPP, as in SopCast, mesh construction and maintenance relies on a centralized bootstrap server. For each channel available in the system, the bootstrap stores a list of nodes connected to that channel, the channel's source peer and its most recent produced chunk ID. In order to keep each channel's peer list up to date, peers send a *ping* packet to

the bootstrap periodically. The channel source, or server peer, also sends ping messages, which also updates the last generated chunk ID value for that channel. This value is used in a further moment to initialize new peers. Peers are removed from their channels' peer lists if they fail to report their existence to the bootstrap for a configurable period, typically a few seconds.

A channel's source peer (one that creates media and starts its distribution) announces its intent to create a channel to the bootstrap. Other peers join the network sending a peer request message to the bootstrap asking for peers watching an existing channel. After receiving a peer request message, the bootstrap selects a subset of peers from that channel's peers list and sends it back to the requester. This message also contains the last generated chunk ID so new peers know from which chunk to start asking.

Upon receiving the requested subset from the bootstrap, peers store them in a candidate peer list. Until it reaches neighborhood size limit, they will periodically try to establish a partnership with some candidates, turning that candidate into a partner. The candidate selection strategy is configurable and new strategies can be implemented. Partners may face connection issues or leave the network. In these cases they can be turned back into candidates or be dropped. Also, from time to time, peers will send new peer request messages to the bootstrap in order to refresh their candidates list. Another feature is that, periodically, partnerships can be undone. The selection strategy of partnerships to be undone is also configurable.

2.2. Chunk Scheduling

Another technique implemented in TVPP is described in [Zhang et al. 2005], where authors propose a model – known as the data-driven model. According to this model, all peers would only request data to neighbors that actually have the data. This is possible by making each peer broadcast periodically which chunks of data they have. This design generates more control overhead, but it prevents request/transmission redundancy.

The broadcasted message contains the buffer map which has an integer indicating the newest chunk of media present on the peers' buffer and a bit map where each position determines the presence or not of a previous chunk. The nth position of the map represents the id of the newest chunk minus n. Thus each buffer map covers a dynamic range of chunk IDs. Buffer map messages are also used as an "I am alive" message. If a peer stays more than a configurable period of time without receiving it from a neighbor, the peer removes that neighbor from its list.

The media server peer will naturally be the first node to have any chunk available. Through buffer map messages its partners will know that it has some chunks that they do not have. This behavior is reproduced over the entire network, once any node announces the presence of a chunk its partners might try to request, if they do not have it. Every few milliseconds each peer compares its own buffer map with its partners' maps looking for the newest chunk to request. Using a configurable selection strategy, it then selects a partner to serve that chunk and adds it to a finite request list signaling its intent to ask that partner for that particular chunk. The oldest chunk request is dropped if a new request arrives and the list is full. This request scheduling rule follows the earliest deadline first (EDF) rule, in which chunks closer to meet the deadline are requested first. After receiving a chunk, the respective request is removed from the request list, the chunk is

stored and the buffer map sets it as present.

2.3. Logging

Ping messages that are sent to the bootstrap also carry performance data. Using the bootstrap as the logging server increases the traffic at the bootstrap, but it has the advantage that there is no need for a special server to store log data.

These special ping – or *log* messages – include the following performance data: the number of packets generated, sent and received per second, packets missed since the last log message (a discontinuity measure), average hop count for each chunk received, and neighborhood size. The list can be easily extended, as new measures are envisioned and required. A more detailed chunk performance data is sent also in each message. This chunk data relates to one sample chunk sent during this period and is composed by its chunk ID, hop count and timestamps indicating when it was generated or consumed. All that data are used to get useful insights about latency, deadline miss rate, and distance that a chunk travels before being delivered. These are especially interesting because they are very hard (in some cases impossible) to be captured from commercial systems.

Another feature is that each peer periodically sends to the bootstrap a list of what other peers they are connected to. This is done through a different message with a list of peer addresses and a timestamp upon bootstrap receival. With those reports, one may track the evolution of the overlay over time.

2.4. New Modules/Algorithms

TVPP's object-oriented design includes generic interfaces for many modules. These interfaces provide flexibility to implement new features or algorithms. For instance, it is easy to create a new kind of message that will be exchanged between nodes since the send/receive methods receive a Message object as parameter and all messages inherit from Message. So, to create a new kind of message, one must only create a class that extends from Message, add a new entry at the group of message kinds, describe the structure of the new kind of message through an abstract method inherit from Message and introduce a method to handle that message.

Two other mechanisms which are easy to alter are the scheduler policy and the bootstrap peer selection algorithms. Both have been implemented as a strategy pattern. To extend these mechanisms, one must develop his algorithm as a extension of a strategy interface and patch the new strategy in with a few lines of code on headers and at the parameter handler.

Implementing a new peer performance metric and logging it is also simple. Since the log message is basically a wrapper, one must add the new metric at the log message, to the chain that constructs it and make sure that it will be unwrapped and written at the log file on the other side, in this case the bootstrap.

3. Usage

TVPP contains two applications, the bootstrap and the client. Both programs are Linux compatible, but can be compiled through Cygwin[6] to create a Windows executable. The programs can be run through command line.

[6]http://www.cygwin.com/

Clients can either be the channel source or a viewer. In order to start an experiment one must start a bootstrap, a channel source and any number of viewers. We recall that each program must have its own address, an ip-port pair. Usually our own experiments are conducted on PlanetLab[7] and we only attach one host to each node. More than one logical viewers can be located at the same physical host.Peers and bootstrap can be configurable to use specific ports.

As previously mentioned, the bootstrap stores data about each channel. Another parameter is channel id. The bootstrap can hold several channels, thus TVPP can perform parallel experiments. Since each channel creates a different overlay, results do not interfere with each other.

Finally, the most interesting parameters are those that can actually alter experimental results. Currently, one can fix the following set of parameters:

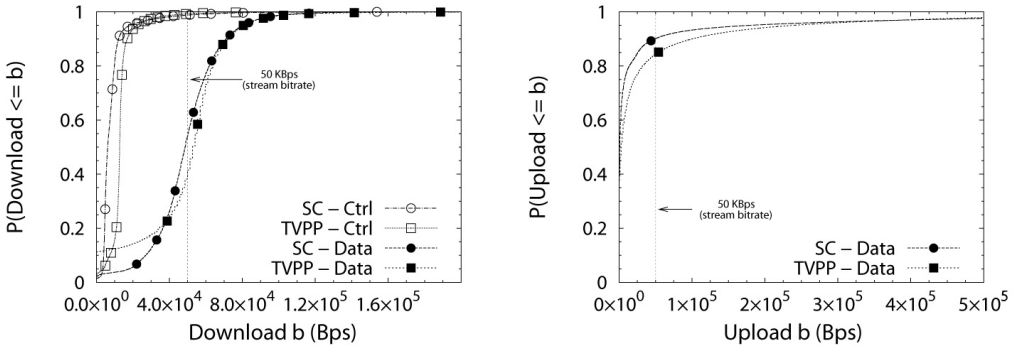
- mode, initially thought as a way to define if a peer will be a source or a viewer, this parameter can be used to create special kinds of peers, e.g. a free rider;
- buffer size, that affects buffer map message sizes and the peer capacity to hold chunks;
- maximum number of neighbors, relates with the overlay, more partners means more chunk sources, a more connected network, but it also increases buffer map messages throughput since they are periodically broadcasted to all neighbors;
- request limit, defines the amount of chunks that can be simultaneously requested;
- unresponsive partnership timeout, this timeout defines how many seconds a peer must wait before removing an unresponsive neighbor from its partners list;
- upload and download limits, these restrict TVPP to send or receive a maximum number of bytes using a leaky bucket;
- bootstrap's peer subset selection algorithm, the subset of peers returned by the bootstrap can be random, oriented by IP distance or RTT (round-trip time) between peers, but one can create his own selection strategy;
- partnership candidate selection algorithms, for connecting or disconnecting a neighbor, these algorithms may be the same as above since they are also peer selection strategies;
- chunk scheduler algorithm, another peer selection algorithm but with chunk transmission purposes.

Using this tool, we have already conducted some experiments. In order to illustrate its use, we have compared TVPP with SopCast(SC). We have setup over 500 PlanetLab nodes, have streamed a 100 minutes 400Kbps average bitrate video which looped continuously during our experiments and have captured data for 60 minutes in both SopCast and TVPP systems. TVPP allowed us to set neighborhood limits for server and clients at 10 and 50 peers, respectively, following characteristics observed on SopCast. Figure 1 presents a comparison between peer upload and download rates for both systems that shows one of many similarities between the systems. A point in those curves shows peers that have at least the indicated upload/download rate in a given second in time.

4. Related Work

In 2005, CoolStreaming [Zhang et al. 2005] gained academic notoriety as the first P2P live streaming system to publish information about its internal scheme. Its data-driven

[7]http://www.planet-lab.org/



(a) Download grouped by control and data packets

(b) Upload restricted to data packets

Figure 1. Cdf of download and upload rates of each peer in each second for both systems. Stream bitrate is ≈50KBps.

model has been extensively cited over the few years. Nevertheless, the system was never deployed as a testing platform and few measurements were done over it [Li et al. 2008].

Several measurement papers have been published that try to understand popular P2P live streaming systems such as PPLive and SopCast [Vieira et al. 2009, Horvath et al. 2008, Silverston et al. 2009, Ali et al. 2006, Tang et al. 2009] though few papers had proprietary data to work on, such as UUSee [Wu et al. 2007]. Most of the research published on popular systems rely on crawling the network with a subset of nodes and capturing network traffic, but that approach is inefficient while trying to answer some questions like those stated in Section 1. Especially, *what if* questions that propose abnormal situations.

The NAPA-WINE project [Abeni et al. 2010] is the closest effort toward building a P2P live streaming system with academic purposes in the last few years. Their work is focused at performance and system optimization, especially through network awareness, creating partnerships using insights from ALTO [Seedorf and Burger 2009]. They have designed a development toolkit and a set of libraries to facilitate the implementation and integration of experimental algorithms into their project. Their design also includes monitoring the latency and the available bandwidth between two peers, or the presence of Network Address Translation (NAT). Contrary to NAPA-WINE, which concentrates on measuring network behavior, TVPP focuses on plain experimentation and therefore can easily provide raw logs, receive new algorithms and extensions, and adapt to several different kinds of experiments, such as freeriding, or polluting clients.

5. Conclusions

In this paper we present TVPP, a P2P live streaming system that has been designed to simplify experiments and to assist researchers in better understanding the behavior of P2P-TV systems. TVPP works similarly to popular existing systems such as SopCast, but enables more efficient experimentation and data acquisition by eliminating the need of external network traffic analyzers, by allowing configuration of system parameters, and by providing a set of performance and overlay data. Moreover, TVPP can also provide additional information about system behavior, such as latency and chunk miss rate, that impact user experience but cannot be easily obtained in existing systems.

Figure 2. Screenshot of an experiment showing the bootstrap (top left), a channel source (bottom left) and two viewers with their players (middle and right).

This tool is already operational, supporting experiments and providing results in some of our studies. We expect to continue extending it to include new features. Meanwhile, we already envision a few tweaks that would make it even better and easier for researchers. Some of those are: embedding a configurable churn generator, using a xml-like file to configure experiments, altering parameters in execution time, and real-time graphical monitoring.

We believe that TVPP can provide an efficient alternative to P2P-TV experimentation, providing a more direct and straightforward way of evaluating new algorithms. In addition to that, we believe that TVPP can also be used as an efficient P2P live streaming system in real applications.

6. Demo Proposal

The demo presentation can be held with one or multiple computers. We expect to show a live capture with bootstrap, media source and other peers, deploying a real experiment on PlanetLab. Compile and run sequence on participants computers will also be stimulated. Figure 2 shows a sample experiment using 400+ nodes. In this experiment the source has sent media to the top viewer that redistributed it to the network. The bottom viewer has received the stream through the nodes in PlanetLab and therefore with a near 3 seconds latency. On each screen it can be observed output that show message exchange, expose peer lists and report about chunk IDs (what is the newer chunk, what is being requested, what have been received, what is being played). By default, we plan on starting an empty overlay with the chronometer video. Afterwards we will insert a single peer, then we shall remotely start hundreds of PlanetLab peers, and, finally, start another peer. This will show peers in two different layers of the network, one closer to the server and other farthest.

A TVPP stable version and documentation can be found at: *http://vod.dcc.ufmg.br/tvpp/*. Yet, newer versions are available through request.

References

Abeni, L., Bakay, A., Biazzini, M., Birke, R., Leonardi, E., Cigno, L., Kiraly, C., Mellia, M., Niccolini, S., Seedorf, J., et al. (2010). Network Friendly P2P-TV: The NapaWine Approach. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–2. IEEE.

Ali, S., Mathur, A., and Zhang, H. (2006). Measurement of commercial peer-to-peer live video streaming. In *Proc. of Workshop in Recent Advances in Peer-to-Peer Streaming*. Citeseer.

Fodor, V. and Dan, G. (2007). Resilience in live peer-to-peer streaming [peer-to-peer multimedia streaming]. *Communications Magazine, IEEE*, 45(6):116–123.

Hei, X., Liu, Y., and Ross, K. (2008). IPTV over P2P streaming networks: the mesh-pull approach. *Communications Magazine, IEEE*, 46(2):86–92.

Horvath, A., Telek, M., Rossi, D., Veglia, P., Ciullo, D., Garcia, M., Leonardi, E., and Mellia, M. (2008). Dissecting PPLive, SopCast, TVAnts. *submitted to ACM Conext*.

Li, B., Xie, S., Qu, Y., Keung, G., Lin, C., Liu, J., and Zhang, X. (2008). Inside the new coolstreaming: Principles, measurements and performance implications. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1031–1039. Ieee.

Seedorf, J. and Burger, E. (2009). Application-layer traffic optimization (ALTO) problem statement. *draft-marocco-alto-problem-statement-04 (work in progress)*.

Sentinelli, A., Marfia, G., Gerla, M., Kleinrock, L., and Tewari, S. (2007). Will IPTV ride the peer-to-peer stream?[Peer-to-Peer Multimedia Streaming]. *Communications Magazine, IEEE*, 45(6):86–92.

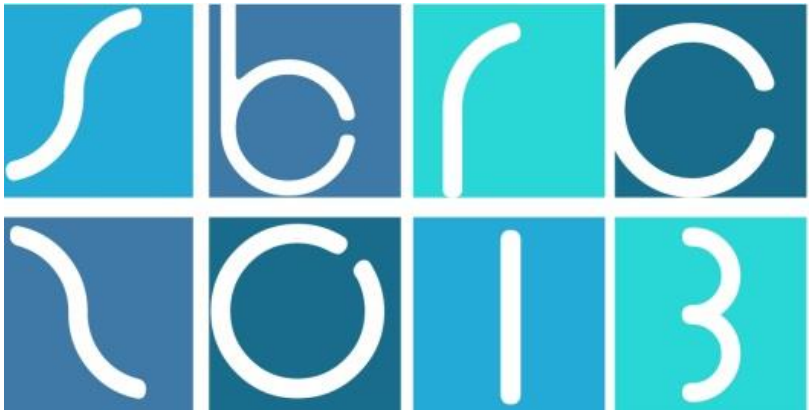
Silverston, T., Fourmaux, O., Botta, A., Dainotti, A., Pescapé, A., Ventre, G., and Salamatian, K. (2009). Traffic analysis of peer-to-peer IPTV communities. *Computer Networks*, 53(4):470–484.

Tang, S., Lu, Y., Hernández, J., Kuipers, F., and Van Mieghem, P. (2009). Topology dynamics in a P2PTV network. *NETWORKING 2009*, pages 326–337.

Vieira, A., Gomes, P., Rocha, M., Almeida, J., and Campos, S. (2009). A behaviour model of the SopCast users. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, pages 1–8. ACM.

Wu, C., Li, B., and Zhao, S. (2007). Magellan: Charting large-scale peer-to-peer live streaming topologies.

Zhang, X., Liu, J., Li, B., and Yum, T. (2005). CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming. In *proceedings of IEEE Infocom*, volume 3, pages 13–17. Citeseer.



31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
Brasília-DF

Salão de Ferramentas

Sessão Técnica 3

Simulação e Emulação de Redes

Uma ferramenta livre para geração de topologias físicas de redes de telecomunicações

Marina Girolimetto[1], Rafael Augusto Galuppo[1], Claunir Pavan[1]

[1]Universidade Federal da Fronteira Sul (UFFS) – Ciência da Computação
Campus Chapecó – SC – Brasil

marina.gtto@yahoo.com.br, galuppo.rafael@hotmail.com,
claunir.pavan@uffs.edu.br

Abstract. *We present a software to generate realistic optical transport network topologies, named NTT Gen (Network Transport Topology Generator). Topologies generated from this software resemble characteristics of real transport networks and suitable for routing algorithms, techno-economic and network services analysis.*

Resumo. *Neste artigo, apresentamos um software gerador de topologias realísticas de redes ópticas de transporte de telecomunicações, chamado de NTT Gen (Network Transport Topology Generator). As topologias geradas refletem vários aspectos das redes reais, incluindo a distribuição do grau nodal, número médio de saltos, conectividade par a par de ligações disjuntas e conectividade par a par de nós disjuntos. As topologias podem ser utilizadas para simular diferentes cenários, em trabalhos como análise de algoritmos de roteamento, tecno-econômica e comportamento de serviços de rede.*

1. Introdução

Topologias de redes geradas por computador são amplamente empregadas em tarefas tais como análise de desempenho de algoritmos, estratégias de segurança, engenharia de tráfego, análise tecno-econômica, entre outras. A falta de informações sobre topologias de redes reais em número suficiente para conduzir simulações extensivas motivou o desenvolvimento de softwares para a geração de topologias artificiais [Gunes and Sarac 2007]. Contudo, no âmbito das redes de telecomunicações, a utilização de topologias aleatórias ou topologias baseadas na Internet não são aplicáveis [Pavan et al. 2010], [Naldi 2005].

Um software gerador de topologias de redes que represente as características de redes ópticas de transporte de telecomunicações reais pode evitar decisões incorretas, como a subestimação do impacto de falhas nas ligações ou nos nós e erros de análise de performance em algoritmos de engenharia de tráfego.

A literatura dispõe de um conjunto de geradores de topologias para propósitos diversos: [Waxman 1988], [Doar 1996], [Zegura et al. 1996], [Cheng et al. 2008], [Faloutsos et al. 1999], [Jin et al. 2000], [Magoni 2002], [Palmer and Steffan 2000], [Medina et al. 2001]. Por exemplo no artigo [Waxman 1988], o autor apresenta um modelo para gerar grafos aleatórios onde os vértices (que representam os nós da rede) são distribuídos sobre um plano e arestas (que representam as ligações da rede) são adicionadas ao grafo usando uma função de probabilidade baseada na distância Euclidiana entre os vértices. Em [Doar 1996], [Zegura et al. 1996] a hierarquia multinível encontrada na Internet é utilizada para gerar topologias que refletem a

Internet. Nos geradores apresentados em [Cheng et al. 2008], [Jin et al. 2000], [Magoni 2002], [Palmer and Steffan 2000], [Medina et al. 2001] a distribuição do grau nodal das redes geradas segue uma lei de potência (power-law). Entretanto, esses esforços visaram reproduzir características da Internet que é uma rede livre de escala (scale-free network) [Faloutsos et al. 1999] e, redes de transporte de telecomunicações não são da categoria livre-de-escala.

Neste contexto, o artigo [Pavan et al. 2010] propõe um método para a geração de topologias físicas para redes ópticas de telecomunicações. Nele, os autores também identificaram algumas características de redes reais, como a distribuição do grau nodal, o número médio de saltos necessários para interconexão dos nós, a conectividade par a par de ligações disjuntas, a conectividade par a par de nós disjuntos e o coeficiente de agrupamento. Neste trabalho implementamos um software seguindo o método proposto por [Pavan et al. 2010] e o disponibilizamos gratuitamente para a comunidade científica. Em adição o software calcula uma nova medida, a centralidade de intermediação (betweenness centrality), que indica o número de caminhos mínimos que passam por um nó.

2. Gerador de topologias: NTT Gen

O *NTT Gen (Network Transport Topology Generator)* é uma ferramenta para geração de topologias físicas para redes ópticas de transporte de telecomunicações. O software foi implementado em linguagem *Java*, utilizando API *Graphstream* [Balev et al. 2010-2012] e requer a Java Virtual Machine 7.0 e está disponível para download no endereço: https://github.com/nttgen/nttgen.

Em [Pavan et al. 2010] é proposto um método que tem como base o modelo de Waxman [Waxman 1988]. Topologias geradas através do modelo de Waxman apresentam uma distribuição do grau nodal que tende a Poisson, a mesma distribuição identificada para redes reais de telecomunicações [Pavan et al. 2010].

As topologias de redes de transporte de telecomunicações são, por natureza, sobreviventes, o que significa que cada par de nós deve conter, pelo menos, dois caminhos disjuntos por ligação. A fim de obter esta e outras características, algumas restrições foram incluídas ao modelo de Waxman incluindo a distribuição (posicionamento) dos nós do plano, como pode ser visto no fluxograma da Figura 1.

A probabilidade de uma ligação existir entre um par de nós (i, j), segundo o modelo de Waxman, é dada por:

$$P(i,j) = \beta\, e^{-d(i,j)/v\alpha} \qquad (1)$$

em que d é a distância Euclidiana entre os nós origem e destino (i, j), v é a distância máxima entre dois nós e α e β são parâmetros para calibração que tem intervalo entre 0 e 1. Na sequência descrevemos o software com mais detalhes.

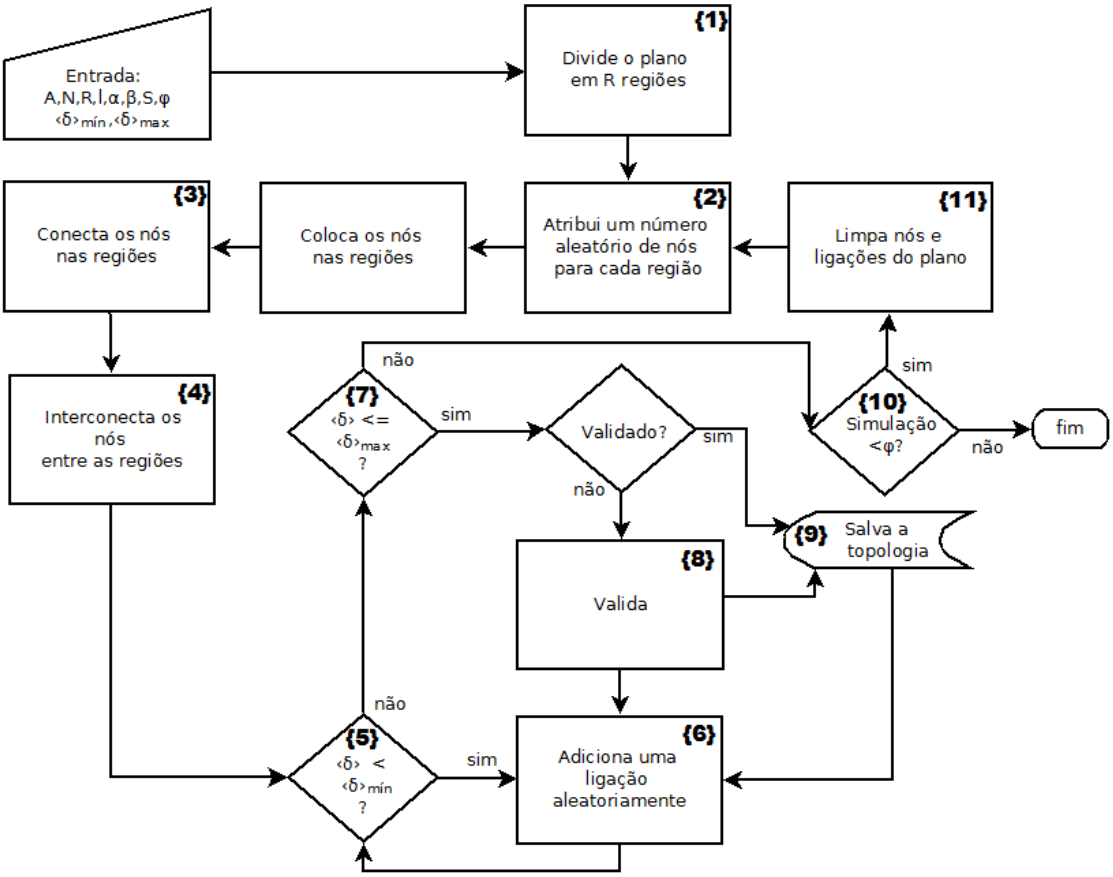


Figura 1. Diagrama de fluxo do algoritmo

Dado um conjunto de entradas (ver Tabela 1), o software simula um plano de lado A, onde as topologias serão instaladas. Este plano será dividido em R regiões de igual tamanho (indicação {1} na Figura 1).

Tabela 1. Variáveis de entrada do NTT Gen

Variável	Descrição
A	Raiz quadrada do plano
N	Número de nós
R	Número de regiões
l	Distância mínima entre nós
α	Parâmetro de probabilidade de ligação Waxman
β	Parâmetro de probabilidade de ligação Waxman
$\langle\delta\rangle_{mín}$	Grau mínimo
$\langle\delta\rangle_{max}$	Grau máximo
S	Posição dos nós (variável ou uniforme)
φ	Número de simulações

Após a divisão do plano em regiões, um número de nós é atribuído a cada região {2}. Este número de nós é aleatório, contudo, considera as restrições de distância mínima entre os nós e espaço dentro de cada região. A atribuição pode ser realizada de modo uniforme ou de modo variável. No modo uniforme o número de nós N é dividido pelo número de regiões e, portanto, cada região recebe o mesmo número de nós (exceto quando N for ímpar), e no modo variável cada região pode receber uma quantidade diferente de nós. Para distribuir os nós neste modo, o algoritmo sorteia inicialmente um número entre 0 e o número de nós que caberiam na primeira região (canto superior esquerdo da área), ou o total de nós N (considera o que for menor). Se ainda restarem nós para serem distribuídos nas regiões vizinhas, o algoritmo executa o mesmo passo até inserir todos os nós. Caso os nós não couberem na área, o programa é finalizado e uma mensagem é dirigida ao usuário.

Depois da inserção dos nós no plano, podem existir regiões sem nós, com um nó, com dois ou mais de dois nós. Na sequência são feitas as ligações entre os nós de uma mesma região {3}. Caso a região tiver somente dois nós, eles são ligados diretamente. Se existirem mais de dois nós, será formado um ciclo seguindo a probabilidade de Waxman.

Uma vez que os nós no interior de cada região estão interconectados, é possível interligar os nós entre as diferentes regiões {4}. Portanto, para cada região, pelo menos dois nós serão interligados com outros nós de regiões vizinhas.

A partir de então, é verificado se a topologia alcançou o grau médio mínimo esperado. Se o grau médio mínimo estipulado nos parâmetros de entrada for maior que o atual {5}, então ligações são adicionadas {6} até o grau médio mínimo ser atingido. Estas novas ligações também obedecem a probabilidade de Waxman.

Enquanto o grau médio estiver no intervalo estipulado na entrada do programa, cada nova topologia gerada é armazenada {9}, desde que seja sobrevivente. Uma topologia é sobrevivente a uma falha em uma ligação (um link) se cada par de nós estiver conectado por pelo menos dois caminhos disjuntos por ligações. A verificação de sobrevivência é feita usando o algoritmo de Suurballe [Sousa and Oliveira 2010] {8}. Esta validação é executada repetidamente até obter uma topologia sobrevivente. Novas ligações quando adicionadas em uma topologia sobrevivente {6} não alteram a característica de sobrevivência, portanto, não será preciso nova validação.

Neste ponto, caso o número de simulações não tenha chegado ao máximo escolhido {10}, as variáveis serão zeradas {11} e novos posicionamentos para os nós no plano serão definidos para criar novas topologias. A Figura 2 apresenta a interface do programa *NTT Gen.*

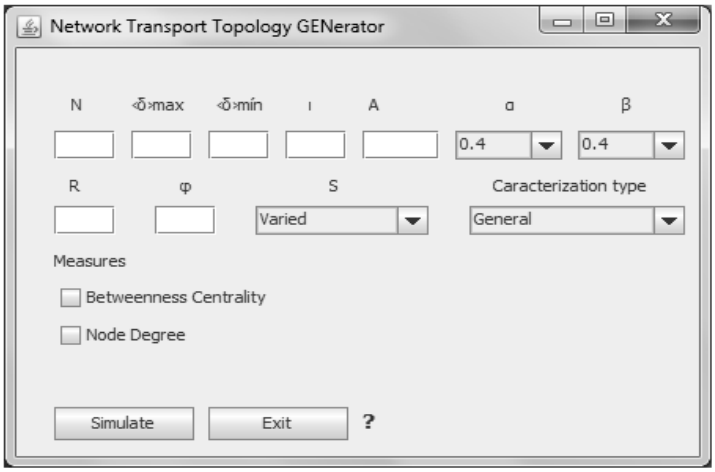


Figura 2. Interface do programa NTT Gen.

O programa cria dois arquivos de saída. Em um dos arquivos são registradas as posições dos nós no plano em coordenadas (x, y), junto com a informação das ligações das topologias e o comprimento de cada ligação. No outro arquivo são armazenadas medidas como o número médio de saltos do caminho de trabalho ‹h›, número médio de saltos do caminho de backup ‹h'›, o grau do nó mínimo ($\langle\delta\rangle_{mín}$), médio ($\langle\delta\rangle_{avg}$) e máximo ($\langle\delta\rangle_{max}$) e o betweenness centrality minímo (bcMin), médio (bcMed) e máximo (bcMax) de cada topologia. Outras medidas serão implementadas em novas versões do programa. A validação das topologias geradas por este método foi estudada no artigo [Pavan et al. 2010], onde foi observado através de testes estatísticos paramétricos e não paramétricos de Kolmogorov-Smirnov, que as medidas das variáveis das topologias resultantes deste método seguem as mesmas distribuições das variáveis das redes reais, ao nível de significância de 5%. A Figura 3 exibe um exemplo de saída para uma rede com seis nós.

a)

Id	nNodes	nEdges	<h>	<h'>	<δ>mín	<δ>max	<δ>avg	bcMin	bcMed	bcMax
Topology [1:1]	6	9	1,4	2,27	2	4	2,67	0,00	2,00	6,00

b) Topology [1:1] (Links)

From	To	Lenght
0	1	2,83
0	2	2,24
1	2	3,00
1	3	3,00
2	4	3,61
2	5	5,10
3	4	6,08
3	5	2,83
4	5	5,00

c) Topology 1

X	Y
0	5
2	7
2	4
5	7
4	1
7	5

d)

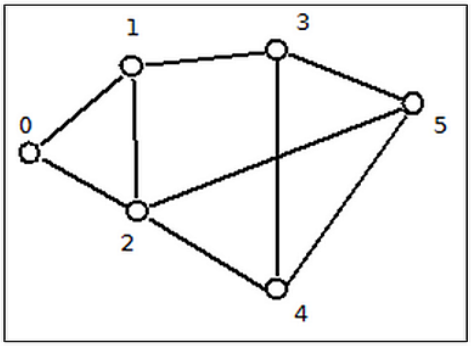


Figura 3. Exemplo de saída do NTT Gen: a) medidas calculadas para cada topologia gerada; b) e c) ligações diretas entre os nós e posição dos nós no plano, respectivamente; d) visualização gráfica de uma topologia gerada.

2.1. Desempenho

A fim de observarmos o desempenho da aplicação em função dos parâmetros de entrada, realizamos um conjunto de simulações considerando diferentes tamanhos de rede (em número de nós) e calculamos o tempo de execução do NTT Gen. Em relação ao número de nós, consideramos valores entre N = 10 e N = 80; para o grau médio mínimo e máximo usamos $\langle\delta\rangle_{mín} = \langle\delta\rangle_{max} = 3$; espaço entre os nós $l = 2$; raíz quadrada da área A = 100; parâmetros de probabilidade das ligações Waxman $\alpha = \beta = 0.4$; o número de regiões $R = \lfloor N/4 \rfloor$; número de simulações com cada N foi de $\varphi = 1$; para a posição dos nós usamos a opção "variável/varied" e o tipo de caracterização "geral/general". Os resultados foram obtidos em um PC Pentium(R) Dual-Core CPU, 2.20GHz com 4GB de RAM.

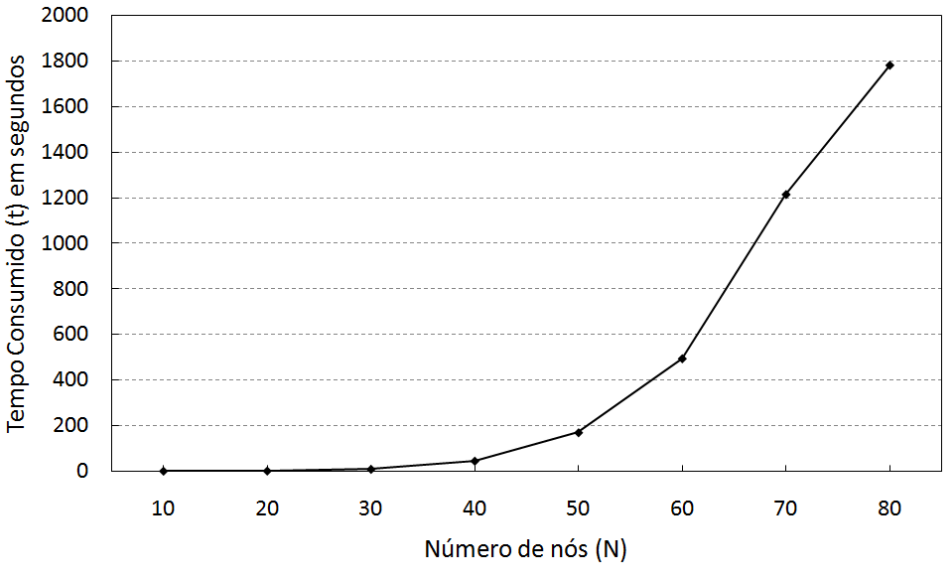


Figura 4. Gráfico demonstrativo do desempenho do NTT Gen

O gráfico da Figura 4 mostra o tempo consumido para gerar uma topologia de rede validada (i.e., sobrevivente) considerando os parâmetros indicados anteriormente. Os tempos variam entre 0.347 segundos e 29.6 minutos para a geração de redes com número de nós entre 10 e 80 e grau médio igual a 3. Contudo, observamos que estes tempos podem variar dependendo da distribuição dos nós nas regiões (que é realizado de maneira aleatória pelo algoritmo). Este é um aspecto que pretendemos melhorar para as próximas versões do programa. O tempo consumido para a geração das topologias inclui o cálculo do número médio de saltos para o caminho de trabalho e número médio de saltos para o caminho de backup. Note que as redes de transporte de telecomunicações, foco do trabalho, têm em geral menos de 100 nós, como pode ser constatado através da lista de redes reais apresentadas em [Pavan et al. 2010].

3. Conclusão

Nós estudamos um método proposto na literatura para a geração de topologias físicas de redes de telecomunicações e implementamos um software que chamamos de *NTT Gen (Network Transport Topology Generator)*. O software produz topologias com as características identificadas como relevantes para as redes de telecomunicações.

Adicionalmente, implementamos novas medidas de grafos que podem ser úteis no processo de simulação e dimensionamento de redes.

Agradecimentos

Trabalho realizado com o auxílio financeiro da UFFS – Universidade Federal da Fronteira Sul, através dos projetos "Desenvolvimento de uma ferramenta computacional para a geração de topologias de redes de telecomunicações" e "Caracterização e geração de topologias realísticas para simulação de redes ópticas de transporte de telecomunicações" referente aos editais 168/UFFS/2011 e 160/UFFS/2012, respectivamente.

Referências

Balev, S., Baudry, J. Dutot, A., Pigné, Y. and Savin, G. (2010-2012) GraphStream – A Dynamic Graph Library, http://graphstream-project.org/, LITIS computer science lab.

Cheng, L., Hutchinson, N. and Ito, M. (2008) "Realnet: a topology generator based on real Internet topology," in *Proc. of the 22nd Int. Conf. on Advanced Information Networking and Applications, AINAW '08*, p. 526–532.

Doar, M. (1996) "A better model for generating test networks," in *Proc. of the IEEE Global Telecommunications Conf., GLOBECOM '96*, pp. 86–93.

Faloutsos, M., Faloutsos, P. and Faloutsos, C. (1999) "On power-law relationships of the Internet topology," in *Proc. of the Conf. On Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM '99*, New York, NY, USA: ACM, p. 251–262.

Gunes, M.H. and Sarac, K. (2007) "Inferring subnets in router-level topology collection studies," in *IMC '07: Proc. of the 7th ACM SIGCOMM Conf. on Internet Measurement*, New York, NY,USA: ACM, p. 203–208.

Jin, C., Qian, C. J. and Jamin, S. (2000) "Inet: Internet topology generator," Technical Report CSE-TR-433-00, EECS Department, University of Michigan.

Magoni, D. (2002) "Nem: a software for network topology analysis and modeling," in *Proc. of the 10th IEEE Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, MASCOTS '02*, p. 364–371.

Medina, A., Lakhina, A., Matta, I. and Byers, J. (2001) "Brite: an approach to universal topology generation," in *Proc. of the 9th Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS '01*, p. 346–353.

Naldi, M. (2005) "Connectivity of Waxman topology models," in *Comput.Commun.*, vol. 29, no. 1, p. 24–31.

Palmer, C. and Steffan, J. (2000) "Generating network topologies that obey power laws," in *Proc. of the IEEE Global Telecommunications Conf., GLOBECOM '00*, vol. 1, p. 434–438.

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

Sousa, A. F. de and Oliveira, J. M. S. dos S. (2010) "Protecção máxima de redes de telecomunicações", http://ria.ua.pt/handle/10773/7557

Waxman, B. (1988) "Routing of multipoint connections," in *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–162.

Zegura, E., Calvert, K. and Bhattacharjee, S. (1996) "How to model na internetwork," in *Proc. of the 15th Annu. Joint Conf. of the IEEE Computer Societies. Networking the Next Generation. INFOCOM'96*, vol. 2, p. 594–602.

Mini-CCNx: prototipagem rápida para Redes Orientadas a Conteúdo baseadas em CCN

Carlos Manuel Silvestre Cabral[1], **Christian Esteve Rothenberg**[2], **Maurício Ferreira Magalhães**[1]

[1]Faculdade de Engenharia Elétrica e Computação
Universidade Estadual de Campinas
Campinas, SP – Brasil (UNICAMP)

[2]Fundação CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações
Campinas, SP – Brasil

{cabral,mauricio}@dca.fee.unicamp.br, esteve@cpqd.com.br

Abstract. *Experimentally driven research in Information-Centric Networking (ICN) is crucial to the evaluation of new proposals that bring named pieces of content as the main element of networks. This paper presents a new fast prototyping tool for ICN based on the CCN (Content-Centric Networking) model, Mini-CCNx, that aims at filling an existing gap in generally available experimental platforms. Using container-based emulation and resource isolation techniques, Mini-CCNx appears as a flexible, scalable, high-fidelity, and low-cost tool that enables experiments on emulated networks with hundreds of nodes in a commodity laptop.*

Resumo. *A pesquisa experimental em Redes Orientadas a Conteúdo (ROCs) é crucial para a validação das novas propostas que trazem o conteúdo como elemento central das redes. Este artigo apresenta uma nova ferramenta de prototipagem rápida para as ROCs baseadas no modelo CCN (Content-Centric Networking), o Mini-CCNx, o qual busca preencher uma lacuna existente entre as plataformas de experimentação da área. Utilizando emulação baseada em contêineres e técnicas de isolamento de recursos, o Mini-CCNx se apresenta como uma ferramenta flexível, escalável, de baixo custo, e com comportamento e fidelidade de desempenho compatíveis ao de um sistema real, podendo criar experimentos em redes emuladas com centenas de nós em um simples laptop.*

1. Introdução

Os princípios que guiaram o desenvolvimento da Internet nas décadas de 1960 e 1970 não são mais tão relevantes atualmente como foram no passado. Se originalmente o intuito principal era o compartilhamento remoto de recursos e interação majoritariamente textual entre estações de trabalho e servidores, atualmente o foco está muito mais centrado na obtenção e acesso a conteúdos e serviços multimídia [Plagemann et al. 2006]. Com a tecnologia de redes atual, quando se deseja obter um conteúdo, é necessário fazer um mapeamento entre *o que* um usuário quer e *onde* este conteúdo está, trazendo inúmeros desafios como mobilidade, segurança, fornecimento de dados de maneira *multicast* e outros, desafios esses que não foram previstos originalmente pelos desenvolvedores da Internet.

Com essa motivação, as Redes Orientadas a Conteúdo (ROCs) trazem propostas para endereçar essa mudança de paradigma, colocando o conteúdo como item central das redes [de Brito et al. 2012]. Propostas como NetInf [Dannewitz et al. 2013], PSIRP [PSIRP] e CCN [Jacobson et al. 2012], dentre outras, introduzem inúmeros conceitos e estratégias nesse intuito. Como em toda nova proposta, a pesquisa experimental para as ROCs é crucial para a avaliação da viabilidade de novas idéias.

Ao tentar avaliar estratégias de encaminhamento e novas técnicas de redução de estado nos roteadores de núcleo das ROCs, identificamos uma lacuna existente entre as ferramentas disponíveis nessa área de pesquisa. Não foi encontrada uma ferramenta de baixo custo, com escalabilidade, flexível e com comportamento e fidelidade de desempenho compatíveis aos de um sistema real para a avaliação de cenários e propostas de ROCs. Essa dificuldade para experimentação motivou o desenvolvimento do Mini-CCNx, apresentado nesse artigo. Inspirado nas práticas bem-sucedidas de prototipagem rápida para Redes Definidas por Software (SDNs - *Software-Defined Networks*) [Lantz et al. 2010], o Mini-CCNx é uma ferramenta capaz de construir uma ROC completa, com centenas de nós, em um simples *laptop*, com altíssima flexibilidade, agilidade e configurabilidade. Além disso, ela provê um ambiente de prototipagem com resultados de grande fidelidade, o que permite que aplicações e conceitos testados utilizando o Mini-CCNx migrem de forma suave para ambientes e redes reais. O Mini-CCNx é baseado no modelo CCN (*Content-Centric Networking*) e utiliza a implementação oficial desse modelo, denominada CCNx[CCNx] e usada no Projeto NDN[NDN Testbed], o que traz maior relevância para a ferramenta e para a comunidade atual de usuários do projeto.

O resto do artigo está organizado da seguinte forma. A seção 2 faz uma análise das ferramentas existentes e detalha os objetivos do Mini-CCNx. A seção 3 descreve em detalhes a arquitetura da ferramenta. A seção 4 apresenta uma análise de performance e fidelidade do Mini-CCNx. A seção 5 explica as duas demonstrações a serem feitas no Salão de Ferramentas do SBRC 2013 e mostra onde encontrar a documentação e o código. A seção 6 finaliza o artigo com a conclusão.

2. Objetivos e Motivação

Existem várias plataformas e ferramentas que podem ser utilizadas na pesquisa e desenvolvimento das ROCs, cada uma com seus prós e contras. Idealmente, seria interessante possuir uma ferramenta que reunisse as seguintes características: *(i)* flexibilidade (capacidade de criação ágil de diversos cenários), *(ii)* escalabilidade (criação de topologias com um número suficientemente alto de nós), *(iii)* baixo custo (possibilidade de ser executada em um *laptop* ou *desktop* comum) e *(iv)* realismo (o comportamento apresentado deve ser compatível com o de um ambiente real e o código utilizado na ferramenta deve ser o mesmo utilizado posteriormente na rede real).

2.1. Análise das ferramentas existentes

Podemos dividir tais plataformas e ferramentas em, ao menos, três categorias:

Simuladores. São flexíveis, escaláveis, e em geral têm baixo custo. Como exemplos de simuladores específicos para ROCs, pode-se citar o *ndnSIM*[ndnSIM] e o ccnSim[ccnSim]. Porém, ambas as ferramentas apresentam as desvantagens de qualquer simulador: *(i)* não apresentam realismo e *(ii)* os modelos de hardware, pilhas de

protocolos e geração de tráfego de um simulador, por melhor que sejam implementados, não representam com total fidelidade todas as características reais dos mesmos.

Testbeds. São infraestruturas experimentais que oferencem recursos reais ou virtualizados para testes em ambientes reais. Podem ser compartilhados entre vários pesquisadores, como o *testbed* oficial do NDN[NDN Testbed] ou especificamente criados para um certo projeto, como por exemplo a criação de um ambiente local com máquinas virtuais representando os *hosts*, *switches* e roteadores. *Testbeds* apresentam realismo, ou seja, executam código real em ambientes com pilha de protocolos e tráfego real. Porém, em geral, apresentam *(i)* alto custo de criação e manutenção e *(ii)* reduzida flexibilidade e escalabilidade na definição de cenários e topologias personalizadas.

Emuladores. Como os simuladores, possuem flexibilidade na definição de topologias, além de um baixo custo de hardware. Como os *testbeds*, apresentam realismo, pois executam código real (aplicações, kernel de SO, etc). O Mini-CCNx, descrito nesse artigo, se encaixa nessa categoria e é especificamente focado nas ROCs.

A Tabela 1 sumariza as características analisadas e inclui duas outras: facilidade de configuração da plataforma e a possibilidade de configuração de parâmetros de *links*, como atraso e perda. Dada a inexistência de uma solução que atenda a todos os requisitos acima, optou-se pelo desenvolvimento de uma nova ferramenta, o Mini-CCNx.

Tabela 1. Resumo das características das plataformas de teste para ROCs

	Simuladores Ex. ndnSIM e ccnSim	***Testbeds*** Ex. *Testbed* NDN	**Emuladores** Ex. Mini-CCNx
Flexibilidade	Alta	Baixa	Alta
Escalabilidade	Alta	Baixa	Alta
Custo	Baixo	Alto	Baixo
Realismo	Não	Sim	Sim
Facilidade de Configuração	Média	Baixa	Alta
Configuração de Parâmetros de *Links*	Sim	Com Restrições	Sim

3. Arquitetura

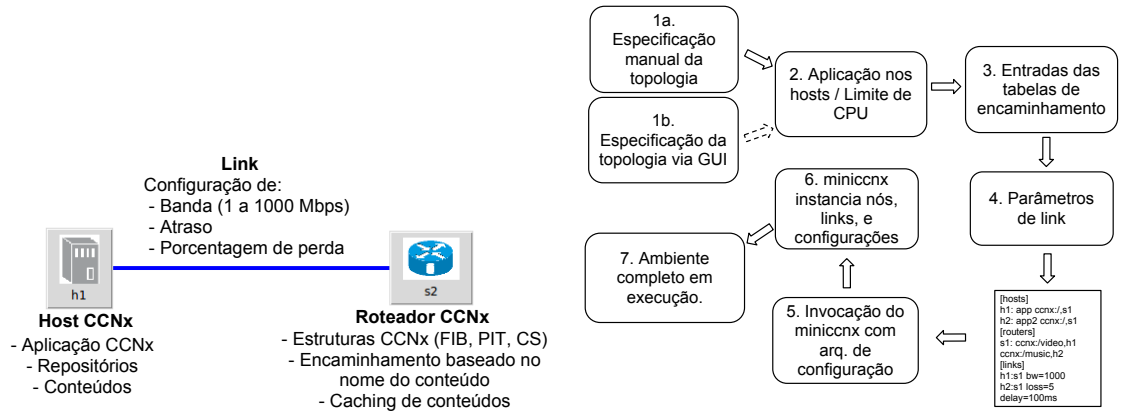
O Mini-CCNx reúne todas as características analisadas na seção anterior. Através da edição de um arquivo texto é possível criar as mais diferentes topologias e cenários CCN de maneira flexível e rápida. O custo para executá-lo é baixo, sendo possível levantar centenas de nós em um simples *laptop*. O mesmo código executado na ferramenta poderá ser posteriormente utilizado em uma rede real. Além disso, é possível configurar rapidamente, com um arquivo de configuração, parâmetros como atraso, banda e perda dos *links*.

O Mini-CCNx é um *fork* do Mininet-HiFi [Handigol et al. 2012] (originalmente proposto para redes OpenFlow) que adiciona uma série de mecanismos e novas classes para instanciar ROCs baseadas no modelo CCN utilizando o código oficial do Projeto CCNx. Estruturas de dados específicas do modelo como tabelas de encaminhamento e repositórios foram implementadas. Também foi implementado todo um mecanismo automático de criação dinâmica de conectividade IP sobre a qual as conexões CCNx operam, tornando isso totalmente transparente ao usuário. Além disso, várias ferramentas de suporte, configuração e construção de topologias também foram desenvolvidas com o intuito de auxiliar o pesquisador no desenvolvimento de seus cenários de teste.

O Mini-CCNx utiliza emulação baseada em contêineres no nível de Sistema Operacional [Soltesz et al. 2007]. Isso permite que grupos de processos tenham visões independentes dos recursos do sistema (*ID*s de processos, sistemas de arquivos e interfaces de rede) apesar de compartilharem um único kernel, alcançando assim uma escalabilidade maior do que um sistema baseado em máquinas virtuais em um único *hypervisor*. O Mini-CCNx também incorpora o conceito de isolamento de desempenho e de provisão de recursos utilizando os *cgroups*[1] do kernel Linux. Com isso, é possível configurar, para cada nó, a porcentagem máxima de uso de CPU. Todo esse isolamento de recursos traz ao Mini-CCNx a garantia de uma grande fidelidade nos experimentos.

3.1. Visão Geral

O Mini-CCNx conta com três componentes principais: *hosts* CCNx, roteadores CCNx e *links*. A Figura 1(a) mostra um resumo das funcionalidades de cada componente.



(a) Componentes Mini-CCNx (b) Fluxo de trabalho de experimentação

Figura 1. Visão geral e fluxo de trabalho

***Host* CCNx:** Responsável por executar as aplicações CCNx. Pode ser implementado pela classe *CCNHost* (sem limitação de recursos) ou pela classe *CPULimitedCCNHost* (com limitação de recursos). Uma aplicação executando em um *host* pode se comportar como cliente quando expressa o interesse por um conteúdo através de uma mensagem *Interest*, ou como servidor, quando retorna um *Content Object* em resposta a um *Interest* recebido. Os *hosts* também são responsáveis por armazenar conteúdos em seus repositórios.

Roteador CCNx: Implementado pela classe *CCNRouter*. Responsável por fazer o encaminhamento dos pacotes *Interest* e *Content Object* e também o *caching* de dados.

***Links*:** Responsáveis por ligar, ponto-a-ponto, os nós CCNx (*hosts* e roteadores). A classe *Link* foi adaptada para suportar a conexão entre nós CCN. Pode-se inserir, para cada *link*, os seguintes parâmetros: *(i) bw=<1-1000>* (banda, em Mbps, do *link*), *(ii) delay=<0-10000>ms* (atraso, em ms, do *link*) e *(iii) loss=<0-100>* (porcentagem de perda de pacotes no *link*).

Após configurar e iniciar o Mini-CCNx, é possível obter um terminal para cada *host* ou roteador, examinar e alterar suas tabelas de encaminhamento, adicionar conteúdos, executar aplicações personalizadas baseadas em conteúdo e coletar dados e

[1] http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt

métricas desejados. É importante notar que *hosts* e roteadores utilizam a implementação oficial mais recente do Projeto CCNx, executando o *daemon ccnd*. Caso o usuário atualize a versão desse código, o Mini-CCNx transparentemente utilizará essa nova versão, sem necessidade de atualização ou recompilação.

3.2. Fluxo de trabalho utilizando o Mini-CCNx

A Figura 1(b) exemplifica o típico fluxo de utilização do Mini-CCNx.

1. A especificação da topologia é feita editando o arquivo de configuração do Mini-CCNx especificando *hosts*, roteadores e os *links* entre eles (etapa *1a* na Figura 1(b)). Opcionalmente, pode-se utilizar a ferramenta gráfica `miniccnxedit` fornecida para gerar um template desse arquivo (etapa *1b* na Figura 1(b)).

2. No próprio arquivo de configuração, é possível especificar qual a aplicação será executada automaticamente em cada *host*. Opcionalmente, pode-se definir o limite de uso de CPU, em porcentagem, para cada nó.

3. Ainda no arquivo de configuração, pode-se inserir entradas nas tabelas de encaminhamento dos *hosts* e roteadores.

4. Pode-se especificar ainda parâmetros para cada *link* (banda, perda ou atraso). Ao fim dessa etapa, o arquivo de configuração estará concluído.

5. A ferramenta `miniccnx` deve ser invocada, tendo como argumento o arquivo de configuração criado.

6. A ferramenta faz o *parsing* das configurações, instancia os nós, configura as entradas nas tabelas de encaminhamento, aplica os parâmetros especificados para cada *link* e executa as aplicações nos *hosts*.

7. Agora o ambiente está criado e executando, permitindo ao usuário a interação dinâmica com as aplicações baseadas em conteúdo. Opcionalmente, ele pode coletar quaisquer métricas e *logs* que sejam necessários para a avaliação de seus cenários.

4. Análise de Desempenho e Fidelidade

Todos os testes foram feitos utilizando um *laptop* mediano, representante típico de um hardware de uso pessoal (processador Intel Core I5 2410M e com 4GB de memória RAM). A idéia é justamente analisar o desempenho e a fidelidade do Mini-CCNx ao se utilizar recursos de baixo custo. Serão analisadas as seguintes dimensões de desempenho da ferramenta: *(i)* escalabilidade, *(ii)* coerência e *(iii)* fidelidade ante experimentos reais[2].

4.1. Escalabilidade

Primeiramente, será analisada a escalabilidade da ferramenta em termos de quantos nós e enlaces ela é capaz de instanciar simultaneamente. Para isso, foram escolhidas duas topologias representativas, *full mesh* e linear. Na topologia *full mesh* todos os nós possuem conexões para todos os outros nós. Já na topologia linear (Tabela 2(b)), os nós são ligados linearmente entre si. Tais configurações foram escolhidas justamente por representarem os dois extremos de conectividade entre nós - qualquer outra topologia apresentará um nível de conectividade que estará entre esses dois extremos.

As Tabelas 2(a) e 2(b) mostram o número de nós instanciados (*CCN Routers*), a memória utilizada especificamente pelo Mini-CCNx, a memória utilizada especificamente

[2]Análise de isolamento em https://www.dropbox.com/s/zeyk0q25jotn512/miniccnx.pdf

Tabela 2. Escalabilidade

(a) Topologia *full mesh*

Nº de Nós	Mem. Mini-CCNx (MB)	Mem. *ccnd* (MB)	Mem. Total (MB)	Nº de *Links*	Tempo de inic. (s)
4	15.1	7.2	22.3	6	<1
8	15.3	13.7	29.0	28	3
16	15.7	28.9	44.6	120	11
32	18.2	57.1	75.3	496	48
64	26.0	114.5	140.6	2016	118
128	62.0	229.8	291.8	8128	753

(b) Topologia linear

Nº de Nós	Mem. Mini-CCNx (MB)	Mem. *ccnd* (MB)	Mem. Total (MB)	Nº de *Links*	Tempo de inic. (s)
4	15.0	7.2	22.2	3	<1
16	15.1	28.9	44.0	15	1
64	15.7	113.8	129.5	63	6
256	18.1	458.6	476.7	255	36
512	21.6	918.5	940.1	511	95
1024	27.8	1835.4	1863.2	1023	228
1536	35.3	2754.2	2789.6	1535	320

pelas instâncias do *daemon ccnd* que executam em cada nó, a memória total utilizada (Mini-CCNx + *daemons ccnd*), o número de *links* instanciados e o tempo de iniciação de cada cenário. Pode-se notar que a principal parcela do uso de memória se refere ao *daemon ccnd*, cuja memória cresce linearmente com o número de nós. O tempo de iniciação está fortemente relacionado ao número de *links* instanciados. Por isso, nota-se um tempo maior na topologia *full mesh* quando comparada com a topologia linear.

4.2. Coerência

A coerência ante a variação de parâmetros de experimento (atraso, banda dos *links*, número de *hops*) será analisada. A ferramenta pode ser considerada coerente se, por exemplo, o atraso total aumentar linearmente com o número de *hops*, como o esperado.

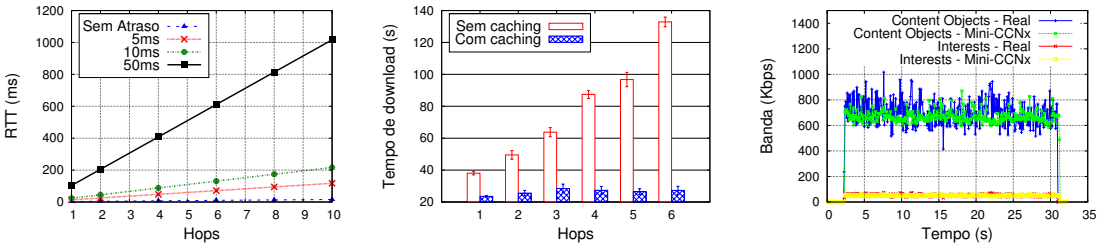
Dois cenários simples utilizando nós CCN foram utilizados. No primeiro (Figura 2(a)), o RTT (*round-trip time*) foi medido para diferentes números de *hops* e diferentes valores de atrasos configurados nos *links* da topologia. No segundo (Figura 2(b)), foi medido o tempo médio de *download* de um arquivo de 100 MB para vários números de *hops*, ou seja, fazendo com que o produtor do conteúdo ficasse cada vez mais distante do cliente. Ainda neste cenário, foram analisados os casos sem *caching* (no qual as requisições pelo arquivo precisam passar por todos os *hops* até chegar ao produtor) e com *caching* (no qual partes do arquivo poderão estar localizadas no *cache* dos nós CCN mais próximos ao requisitante). Ambos os gráficos apresentam valores com intervalo de confiança de 95%.

Na Figura 2(a), pode-se notar o crescimento linear do RTT com o aumento do número de *hops*, como esperado. Além disso, para os diferentes valores de atraso, o comportamento do RTT é consistente. Por exemplo, se todos os *links* têm atraso de 10ms e o número de *hops* é 2, o *ping* deve levar 20 ms na ida, mais 20 ms na volta, mais o tempo de processamento em cada nó, o que é justamente o observado. No cenário da Figura 2(b), nota-se o aumento do tempo de *download* conforme aumenta a distância entre produtor e cliente do conteúdo para o caso sem *caching*, como esperado. Já no caso com *caching*, percebe-se que o tempo médio de *download* é menor do que no caso anterior e pouco relacionado ao número de *hops*. Nesse caso, o tempo de *download* é mais influenciado pela quantidade de conteúdo em *cache* nos nós mais próximos ao requisitante.

4.3. Fidelidade ante Experimentos Reais

A ferramenta possuirá fidelidade se reproduzir o comportamento de experimentos reais. Para a análise de fidelidade, foi criada uma simples topologia com dois *desktops* reais com placas de 100Mbps ligados diretamente entre si, ambos com instalações nativas do código oficial do Projeto CCNx. Em seguida, utilizando o gerador de tráfego

`ccntraffic`[3], o primeiro *desktop* gera constantemente mensagens *Interest* requisitando diferentes conteúdos enquanto o segundo responde com *Content Objects* de 1024 *bytes*. O mesmo cenário foi reproduzido utilizando o Mini-CCNx, com 100Mbps e 200 μs de atraso como parâmetros de *link*. A Figura 2(c) mostra o tráfego de *Interests* e *Content Objects* para ambos os cenários. Pode-se notar que o comportamento da banda utilizando o Mini-CCNx é muito próximo ao comportamento do ambiente real.



(a) Coerência: RTT x nº de hops (b) Coerência: Download (c) Fidelidade: Real x Mini-CCNx

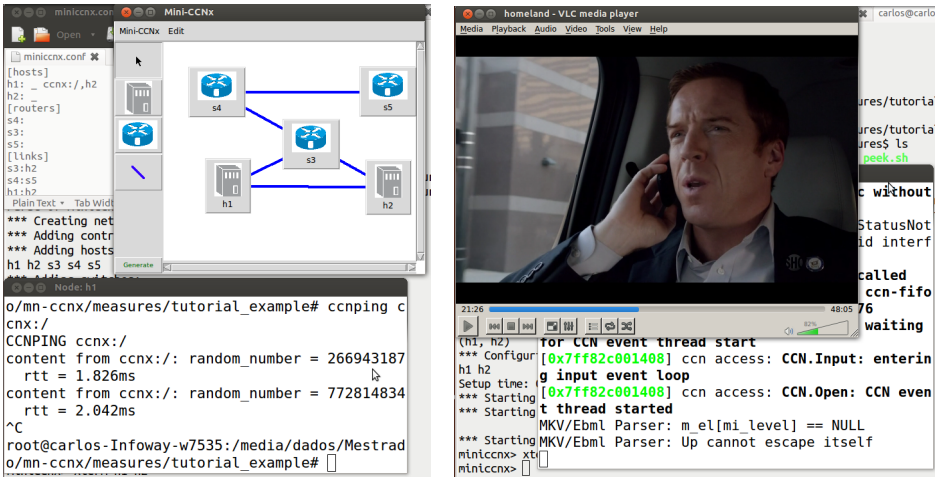
Figura 2. Análise de coerência e fidelidade

5. Documentação, código e demonstração

O Mini-CCNx é um projeto de código livre. Ele está atualmente disponível em *https://github.com/carlosmscabral/mn-ccnx*. Como tal, os autores encorajam o *download*, teste e contribuições para a ferramenta. Documentação, tutorial e exemplos podem ser encontrados em *https://github.com/carlosmscabral/mn-ccnx/wiki*.

A demonstração será feita em duas partes. A primeira (Figura 3(a)) mostrará um passo-a-passo de como utilizar a ferramenta para testar uma aplicação orientada a conteúdo: criação de topologia, definição de parâmetros de *links*, inserção de entradas nas tabelas de encaminhamento e execução de aplicações serão explicados e demonstrados. A segunda parte (Figura 3(b)) mostrará um exemplo prático de como uma aplicação

[3]http://wiki.arl.wustl.edu/onl/index.php/CCNx:_traffic_generation



(a) Fluxo de trabalho (b) Aplicação de vídeo

Figura 3. Demonstrações para o Salão de Ferramentas

de *streaming* de vídeo totalmente orientada a conteúdo se comporta ante a alteração de parâmetros de *link*, como atraso e perda, utilizando as facilidades do Mini-CCNx.

6. Conclusão

O Mini-CCNx, tendo como base a experiência de sucesso das ferramentas de prototipagem rápida para Redes Definidas por Software, apresenta-se como uma ferramenta inovadora, que vem preencher uma lacuna atualmente existente na experimentação de ROCs. A ferramenta apresenta realismo, é flexível e tem baixo custo: toda uma rede orientada a conteúdo com centenas de *hosts* é capaz de ser executada em um simples *laptop* utilizando o Mini-CCNx, com alta configurabilidade e fidelidade dos resultados. Assim, o Mini-CCNx pode ser útil para o desenvolvimento e teste de aplicações orientadas a conteúdo, estratégias de encaminhamento, protocolos de roteamento e técnicas de *caching* dentre outros desafios de pesquisa em ROCs.

Referências

[ccnSim] ccnSim. Scalable chunk-level CCN simulator. http://perso.telecom-paristech.fr/ drossi/index.php?n=Software.ccnSim.

[CCNx] CCNx. Official implementation of the CCN model. https://www.ccnx.org/.

[Dannewitz et al. 2013] Dannewitz, C., Kutscher, D., Ohlman, B., Farrell, S., Ahlgren, B., and Karl, H. (2013). Network of Information (NetInf) - An Information-Centric Networking Architecture. *Computer Communications*.

[de Brito et al. 2012] de Brito, G. M., Velloso, P. B., and Moraes, I. M. (2012). Redes Orientadas a Conteúdo: Um Novo Paradigma para a Internet. *SBRC 2012*.

[Handigol et al. 2012] Handigol, N., Heller, B., Jeyakumar, V., Lantz, B., and McKeown, N. (2012). Reproducible network experiments using container-based emulation. *CoNEXT '12*, page 253.

[Jacobson et al. 2012] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M., Briggs, N., and Braynard, R. (2012). Networking named content. *Communications of the ACM*, 55(1):117.

[Lantz et al. 2010] Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop. In *Hotnets '10*, pages 1–6, New York, New York, USA. ACM Press.

[NDN Testbed] NDN Testbed. NDN Routing Topology. http://netlab.cs.memphis.edu/script/htm/topology.html.

[ndnSIM] ndnSIM. NS-3 based NDN simulator. http://ndnsim.net/.

[Plagemann et al. 2006] Plagemann, T., Goebel, V., Mauthe, A., Mathy, L., Turletti, T., and Urvoy-Keller, G. (2006). From content distribution networks to content networks - issues and challenges. *Computer Communications*, 29(5):551–562.

[PSIRP] PSIRP. Publish-Subscribe Internet Routing Paradigm. http://www.psirp.org/.

[Soltesz et al. 2007] Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. (2007). Container-based operating system virtualization. *ACM SIGOPS Operating Systems Review*, 41(3):275.

JSensor: Um simulador paralelo para redes de sensores em larga escala

Matheus L. Silva[1], **Danniel H. Ribeiro**[1], **Joubert C. Lima**[1]
Andre L. L. Aquino[2], **Ricardo A. R. Oliveira**[1]

[1]Departamento de computação – Universidade Federal de Ouro Preto (UFOP)
Ouro Preto – MG – Brasil

[2]Instituto de Computação – Universidade Federal de Alagoas (UFAL)
Maceió – AL – Brasil.

matnetmg@yahoo.com.br, {dannielhugo, joubertlima}@gmail.com

{alla.lins, rrabelo}@gmail.com

Abstract. *This paper presents a large scale sensor network simulator called JSensor which executes parallel simulations for shared memory or multi-core architectures. JSensor allows asynchronous large scale sensor networks simulations. The application code level of concurrence on the tested shared memory is close to* $90 - 95\%$*, with super linear speed-up as the number of computer cores increases, and near-linear runtime function of the number of nodes up to one million and six hundred thousand nodes flooding data in our tested scenario. JSensor consumes about 4 GB of RAM to simulate 1.6 million flooding sensors.*

Resumo. *Este artigo apresenta o JSensor, uma ferramenta para simulação de rede de sensores sem fio em larga escala que realiza simulações de forma paralela, em arquiteturas de memória compartilhada ou multicores. JSensor se mostrou eficiente, com simulações assíncronas com tempo de execução próximo ao linear e um speedup super linear para um nível de concorrência de aproximadamente* $90 - 95\%$*. Estes resultados foram obtidos considerando um cenário com um milhão e seiscentos mil nós enviando dados do tipo inundação. O JSensor consome cerca de 4 GB de RAM para simular 1,6 milhões de sensores.*

1. Descrição e motivação do problema tratado pelo JSensor

Redes de sensores sem fio (RSSFs) com milhares de nós sempre esteve presente nas discussões a respeito da viabilidade da utilização de tais redes em ambientes reais. Com o advento da combinação desses sensores inteligentes com aplicações de monitoramento ambiental, redes veiculares, cidades inteligentes etc, tal discussão tornou-se mais presente. Por essa razão, a simulação de aplicações de RSSFs em larga escala tem se tornado uma desafiadora área [Khan et al. 2011]. Tais aplicações estão demandando novas ferramentas de simulação que permitam e suportem o desenvolvimento de modelos de comunicação e distribuição de nós, roteamento e processamento de dados em redes com milhares de nós.

Recentemente, algumas aplicações reais para o monitoramento ambiental com RSSFs propõem o uso de centenas de nós. Segundo Oliveira et

al. [Oliveira and Rodrigues 2011], uma RSSFs deve acomodar milhares de nós, logo torna-se necessário se mostrar que as soluções teóricas disponíveis são adequadas para essas redes. Para isso, ferramentas que possam simular tais ambientes tornam-se necessárias. A título de exemplo, se estivermos interessados em efetuar um micro monitoramento ambiental na floresta amazônica, seria necessário a utilização de milhares de nós para cobrir uma grande área. Outro exemplo, se quisermos realizar um micro monitoramento da qualidade do ar no estado de São Paulo, mais uma vez, seria necessário a utilização de milhares de nós. Vale destacar que tanto a floresta amazônica quanto o estado de São Paulo possuem estacões de sensoriamento que efetuam um monitoramento macro.

A simulação de um grande número de nós pode causar um grande impacto nas métricas de desempenho das ferramentas de simulação caso elas não sejam projetadas para tal. O usuário ou projetista almeja construir aplicações RSSFs cada vez maiores e mais rápidas. Esta demanda traz desafios ao simulador como, por exemplo, como garantir robustez e segurança no mecanismo de comunicação entre os sensores, qual a localidade dos sensores, qual o nível de concorrência dos processos do simulador, como balancear a carga para garantir maior equidade na alocação de processos do Sistema Operacional, como garantir dependências causais entre sensores sendo executados por diferentes processos e como oferecer tolerância a falhas de comunicação ou processamento dos sensores. Construir simuladores que combinam tais funcionalidades e ainda permitam a modelagem de sensores cada vez mais reais, isto é, com elevado número de instruções, complexos e custosos computacionalmente, é um enorme desafio.

Neste artigo apresentamos um simulador em paralelo de RSSFs em larga escala, chamada JSensor (`http://www.joubertlima.com.br/JSensor/`). Esse simulador é uma extensão do Sinalgo (`http://disco.ethz.ch/projects/sinalgo`) que é um arcabouço para teste e validação de algoritmos distribuídos e que foi descontinuado em 2008. O JSensor foi executado em máquinas com múltiplos núcleos de processamento e mostrou-se eficiente, com um tempo de execução próximo ao linear e com um *speedup* super linear, refletindo um nível de concorrência no simulador de aproximadamente $90 - 95\%$. O consumo de memória não ultrapassou os 4 GB de RAM. Os resultados foram obtidos a partir de experimentos com até 1.6 milhão de sensores enviando dados do tipo inundação.

Para mostrar a eficiência e as limitações do JSensor o comparamos com o OMNeT++ (`http://www.omnetpp.org`) que é um simulador orientado a objeto para simulação discreta, escrito em linguagem C++ e com interface gráfica 2D. De forma geral, tanto o simulador quanto as interfaces são altamente portáveis, podendo ser executados em qualquer Sistema Operacional. Possui uma extensão chamada *Horizon* que permite executar simulações em arquiteturas multiprocessadas e as extensões *Castalia* e *MiXiM* que permitem simular RSSFs. No entanto, tanto o *Castalia* como o *MiXiM* executam sobre o núcleo OMNeT++ e não sobre o *Horizon*, com isto propriedades específicas de RSSFs não podem ser executadas em paralelo. Portanto, diferente do JSensor, o OMNet++ não permite o paralelismo direto de cenários baseados em RSSFs, implementados no *Castalia*. Assim, caso seja necessário a simulação de cenários com requisitos específicos de RSSFs o projetista deverá reimplementá-los e revalida-los diretamente no *Horizon*.

Além do OMNet++, identificamos alguns simuladores com características específicas, como a escalabilidade e capazes de executar em ambientes paralelos [Lee et al. 2009, Lewis et al. 2002]. Ao contrário do JSensor, tais simuladores não apresentam características específicas de RSSFs ou não simulam um grande número de nós.

2. Arquitetura do JSensor e descrição das principais funcionalidades

O JSensor é totalmente escrito em Java, o que o torna portável para qualquer plataforma. Ele permite a execução de simulações assíncrona que são puramente baseada em eventos. O núcleo possui um conjunto de eventos de envio/recepção de mensagens e de temporizadores, os quais são ordenados pelo tempo em que devem ocorrer. Repetidamente, o simulador retira o evento mais recente da lista e o executa.

Toda a interface gráfica do JSensor é baseada na interface do Sinalgo. Dessa forma, as simulações podem ser visualizadas e manipuladas graficamente via GUI ou podem ser executadas em blocos via linha de comando.

Para implementar cenários de RSSFs em paralelo, o JSensor provê um conjunto de classes que, em conjunto com as classes já existentes no Sinalgo, podem ser utilizadas pelos projetistas para implementar suas simulações. A Figura 1 ilustra o conjunto de classes base do JSensor, bem como suas relações. O JSensor possui uma fachada representada pela classe `Runtime` que é responsável pelo gerenciamento das múltiplas *threads* que simulam os sensores em paralelo.

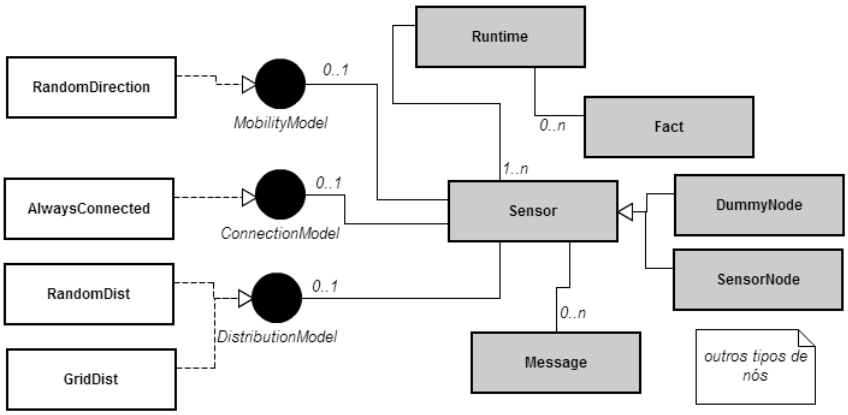


Figura 1. Principais classes do JSensor.

Além da classe `Runtime`, que será detalhada mais adiante, temos a classe abstrata `Node` que representa os nós sensores da rede. Todo novo tipo de nó criado deve estender a classe `Node`. As seguintes classes concretas derivadas de `Node` estão disponíveis no JSensor: (i) `DummyNode` que é o nó básico que não executa nada específico; (ii) `SensorNode` que é responsável por detectar eventos e coletar dados; (iii) `RobotNode` que é responsável por coletar os dados sensoriados pelo nó `SensorNode`; (iv) `RouteNode` que é responsável por rotear mensagens em tráfego; e (v) `SinkNode` que é o nó destino para todos os dados coletados. De uma forma geral, cada nó possui uma serie de métodos que lhe conferem calculo de vizinhança, estado inicial, comunicação e ações específica do projetista da rede.

A classe abstrata `Message` é usada para implementar as mensagens enviadas pelo nó durante a simulação. A classe `Message` define o tipo de mensagem e a forma como elas são criadas. O único método da classe `Message` é o `clone()` que retorna uma cópia idêntica da mensagem anterior. Note que o usuário poderá descrever qualquer mensagem no JSensor, apenas herdando a classe `Message`. Como padrão o JSensor permite o envio de dois tipos de mensagens: *flooding* ou *unicast*.

A classe abstrata `Fact` representa os eventos que ocorrem no ambiente monitorado, chuva, fogo, e tantos outros fenômenos podem ser modelados como fatos no JSensor. Se um fato expira ele é removido automaticamente da simulação. Cada fato é associado a um ou mais `AbstractFactEvent` que é uma classe abstrata responsável por atualizar os valores dos fatos.

Além das classes descritas acima, o JSensor permite a especificação de modelos para distribuição e conectividade dos nós. O modelo de *distribuição do nó* garante posição geográfica inicial em uma área monitorada. O modelo de distribuição requer que o usuário implemente apenas o método `getNextPosition()` que retorna uma posição inicial para um dado nó. Como padrão o JSensor possui implementado os modelos de distribuição: (i) `RandomDistribution` que distribui aleatoriamente os nós na área monitorada; e (ii) `GridDistribution` que distribui os nós numa grade.

O modelo de *conectividade dos nós* permite a manipulação da conectividade entre os nós. Para isso, os usuários possuem os seguintes métodos: `updateConnection(Node` n`, ...)`, para um dado nó n, este método atualiza sua lista de vizinhos e retorna `true` se ocorreu alguma mudança nos vizinhos e `false` caso contrário; `isNear(Node` n_1`, Node` n_2`)` verifica se dois nós estão próximos um do outro, considerando o raio de comunicação; e `isConnected(Node` $from$`, Node` to`)` define quando dois nós estão conectados. Apenas este método deve ser implementado pelo usuário. Como padrão o JSensor disponibiliza o modelo de conectividade `AlwaysConnected` que permite a conexão de um nó com todos os que estão no seu raio de comunicação.

A Figura 2 ilustra a estratégia JSensor para paralelizar as simulações. O núcleo do JSensor possui uma fachada representada pela classe `Runtime` que é responsável pelo gerenciamento de múltiplas *threads* (*Worker Threads*) simulando sensores em paralelo. O número de *threads* criadas é sempre igual ao número de processadores na máquina. Cada *Worker Thread* gerencia um conjunto de sensores proporcional ao número de processadores na máquina. Não há qualquer particionamento baseado na posição do sensor. Com isto, evitamos desbalanceamento de carga em cenários onde aglomerados de sensores se formam na grade de simulação.

Cabe a classe `Runtime` manter ordenado os tempos de todos os eventos produzidos pelos sensores. O JSensor cria uma barreira de sincronização entre suas *threads* somente nos tempos especificados pelos sensores ao criarem seus eventos futuros. Desta forma, garantimos a simulação paralela de todos os sensores que possuem eventos no tempo corrente do simulador. O JSensor compartilha uma grade de células entre as *threads* e, consequentemente, entre os sensores. Cada célula possui as coordenadas `lat`, `long`, `alt` e quais sensores estão nesta posição do globo. Desta forma, o JSensor representa um espaço compartilhado e seguro onde os sensores residem, montado sob

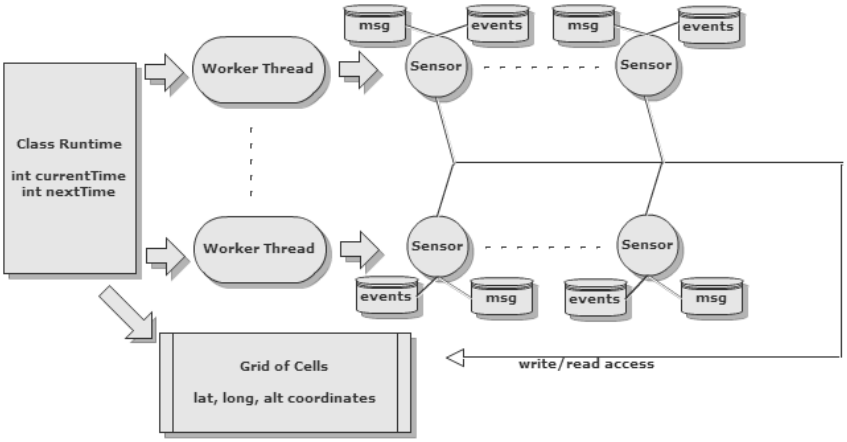


Figura 2. Execução do JSensor.

demanda, pois uma célula só existe se há pelo menos um sensor residindo nela.

O Algoritmo 1 ilustra a execução da classe `Runtime`. Como pode ser observado, só é necessário que o usuário especifique o conjunto de sensores $S_{k,n/k}$, onde n/k são os números de sensores que executam no processador k, e o tempo de simulação $time$. Ademais, os possíveis estados que um nó pode assumir em um determinado evento é representado por σ que possui valores pré-definidos: σ_1 processando uma mensagem; σ_2 atualizando lista de vizinhos; e σ_3 atualizando lista de eventos.

Algorithm 1 Execução da classe `Runtime`

Entrada: $time$, S
1: **enquanto** $t \leq time$ **faça**
2: $event \leftarrow$ evento da lista de eventos a ser executado no instante t
3: **para** $i \leftarrow 0 \ldots |\sigma|$ **faça**
4: **para** $j \leftarrow 0 \ldots k$ **faça**
5: Executa o estado σ_i de todos os nós $S_{j,*}$ que fazem parte do evento $event$
6: **fim para**
7: Sincroniza todos S que executaram σ_i
8: **fim para**
9: **fim enquanto**

De forma detalhada temos os seguinte passos no Algoritmo 1: Linhas 1 - 9 temos a execução da simulação controlada pelo tempo especificado pelo usuário; Linha 2 consome o evento a ser executado no instante t. Tal evento será representado e executado por um conjunto específico de sensores; Linhas 3 - 8 temos a verificação de todos os estados que cada sensor deve assumir durante a execução de cada evento. Para cada transição de estado todas as *threads* varrerem todos os nós para tentarem simula-los (linhas 4 - 6). Assim, haverão execuções concorrentes dos eventos de um tipo pré-definido e anteriores ao instante t apenas. De forma mais específica, implementamos um *pool de threads* pra evitar a criação e remoção de *threads* de forma recorrente. Com o *pool de threads* mantemos algumas *threads* ativas enquanto a simulação durar. A linha 7 é a barreira de sincronização das *threads*. A cada grupo iniciado de *threads* para cada estado existe a necessidade de aguardar que todas as demais *threads* terminem antes que haja a mudança de estado. Desta forma, o JSensor garante que o sensores podem se mover, processar, enviar/receber mensagens e criar novos eventos em paralelo, porém um evento por vez.

3. Avaliação do desempenho

Nesta seção avaliamos o JSensor vs. OMNeT++ considerando até 1.600.000 (um milhão e seiscentos mil) sensores. As mensagens enviadas são do tipo inundação. De forma aleatória cada sensor manda uma mensagem para o sorvedouro seguindo a estratégia de inundação, ou seja, cada sensor envia a mensagem para o seu conjunto de vizinhos que reenvia para os seus vizinhos até que a mensagem chegue ao sorvedouro. Sob o ponto de vista prático essa estratégia é pouco utilizada. Consideramos a inundação na avaliação de desempenho do JSensor, pois essa estratégia de propagação de dados gera um alto tráfego na rede, exigindo bastante dos recursos do simulador, onde 100% dos nós, se alcançáveis, participam da simulação.

Como o JSensor permite a concorrência entre sensores com eventos do mesmo tipo e num mesmo tempo, durante a simulação há apenas um tipo de evento e portanto os sensores enviam mensagens a seus vizinhos concorrentemente quando seus eventos ocorrem simultaneamente. Vale salientar que não estamos tratando colisão, ou seja, todas as mensagens enviadas são recebidas e propagadas. Após os sensores serem criados a aplicação inicia o envio de mensagens por intermédio de 10 mil nós.

Desta forma, em pouco tempo todos os sensores possuem mensagens para consumir e enviar. Nos cenários simulados cada sensor não possui mais do que $d = 8$ sensores como seus vizinhos, onde $d = \pi\, R\, N/L^2$, $R = 10$ é o raio de comunicação, $N = \{100k, 200k, 400k, 800k, 1.600k\}$ o número de sensores e L^2 é a área de sensoriamento e varia de acordo com o número de nós [Aquino et al. 2013].

Os experimentos foram executados em uma máquina com as seguintes configurações: dois núcleos Intel Xeon E5405 quad-core com 2 GHz; 16 GB de RAM compartilhada; quatro drives SATAII com 7200 rpm; Sistema Operacional Linux Ubuntu 12.01 LTS 64-bit; e todos os algoritmos foram codificados com o java 64-bit (JRE 1.7.0_07). Cada simulação foi executada quinze vezes. Analisamos o JSensor quanto a seu tempo de execução, consumo de memória e aceleração.

O JSensor consome cerca de 60% da memória que OMNeT++ consome quando o número de sensores ultrapassa 1M (um milhão), porém para baixo número de sensores, ou seja, menos que 300k (trezentos mil) sensores, o JSensor necessita de um pouco mais de memória que o OMNeT++. A Figura 3 apresenta o comportamento descrito acima. Isso ocorre pois até certo ponto a complexidade introduzida pelo JSensor para garantir paralelismo o afeta no consumo de memória, porém este custo não aumenta muito a medida que o número de sensores se eleva. Em contrapartida, a complexidade dos sensores no OMNeT++ o afeta à medida que o número de sensores aumenta. Este detalhamento obrigatório no OMNeT++ afeta seu consumo de memória, mesmo quando muitas funcionalidades nativas do sensor não estão sendo avaliadas na simulação.

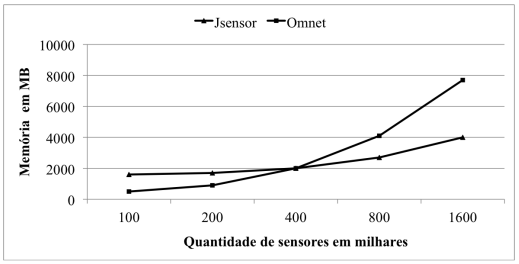
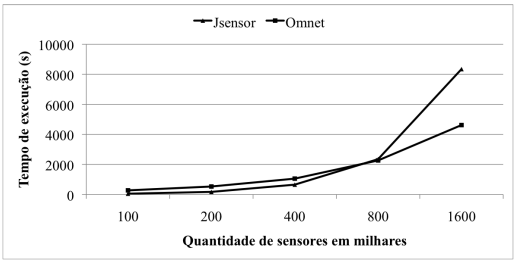


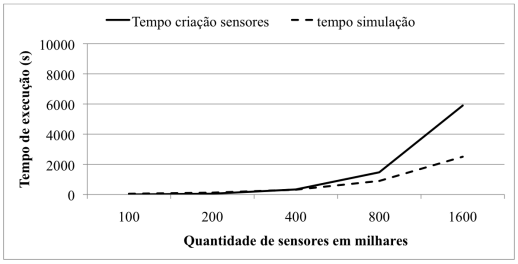
Figura 3. Utilização de memória.

A Figura 4 ilustra as medidas para o tempo de execução quando simulam 100-

1.600k sensores do tipo inundação. Na Figura 4(a) para poucas centenas de milhares de sensores, o JSensor se mostra eficiente. Porém, a partir de 800k sensores o JSensor é penalizado no seu tempo de execução em relação ao OMNeT++. Isso se deve ao tempo de criação dos sensores que ainda se apresenta bastante custoso no JSensor, conforme pode ser observado na Figura 4(b) que mostra os tempos de simulação e criação de sensores no JSensor para 100-1600k sensores. O custo de simular a execução de um sensor passa a ser muito menor que o custo de criá-lo. Note que se desconsiderarmos o tempo necessário para a criação dos sensores, o JSensor ficaria com aproximadamente a metade do tempo de execução do OMNeT++.



(a) Comparação dos tempos de execução.



(b) Tempos de execução do JSensor.

Figura 4. Avaliação dos tempos de execução.

Realizamos também testes para medir a aceleração do JSensor. Nos testes variamos de 1-8 o número de *threads* JSensor e com 100k e 1600k o número de sensores. Podemos aferir que o JSensor é escalável, possuindo aceleração superlinear, com nível de paralelismo na ordem de 90-95% para 100k sensores, segundo a lei de Amdahl. A Figura 5 mostra que a aceleração melhora à medida que o número de sensores aumenta. Este comportamento foi formalmente explicado pela lei de Gustafson [Gustafson 1988], que detectou que o nível de paralelismo varia conforme o tamanho da entrada, pois a porção sequencial sempre fica menos significativa à medida que o custo computacional total aumenta.

O comportamento superlinear pode ser explicado pela exaustiva execução de criação de sensores em paralelo. O nível de repetição de instruções é altíssimo e favorável a *cache-hit*. Tais instruções já podem ser armazenadas e decodificadas em *cache*, o que acelera ainda mais sua execução. Outra vantagem é a utilização de paralelismo no nível da instrução, uma vez que são instruções independentes. Os núcleos de processamento ociosos podem ser utilizados para a executar instruções de outras *threads*.

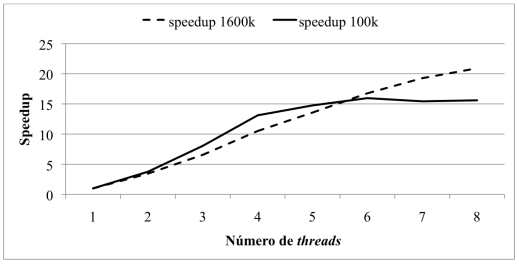


Figura 5. Speedup do JSensor.

No caso do JSensor, ao inicializar duas *threads* quatro núcleos podem ser facilmente utilizados, conferindo aceleração de quatro e não de dois a aplicação. Note que à medida que o numero de *threads* JSensor aumenta o nível de paralelismo implícito ou no nível da instrução diminui, pois os núcleos ociosos também diminuem. De qualquer forma, a politica de *cache* ainda continua beneficiando o JSensor, portanto para a

entrada com 1.600k sensores a aceleração aumenta até oito *threads*. Já com 100k sensores a aceleração estabiliza com cinco *threads* JSensor. Neste ponto, tanto o benefício da política de *cache* quanto o paralelismo explicito e implícito chegam a um limite. A aceleração JSensor tende a diminuir a medida que a criação de sensores fica mais eficiente, portanto ter aceleração super linear não basta.

Para avaliar a corretude no JSensor, armazenamos o número de mensagens e seus conteúdos de forma a conseguir compará-los ao final de um experimento. Após as simulações, verificamos não somente o número de mensagens, mas também os tempos de envio, recebimento e a ordem de propagação. Observamos que o JSensor e OMNeT++ geraram os mesmos logs. Isso evidencia a corretude das simulações executadas pelo JSensor, uma vez que o OMNeT++ já é um simulador testado e validado.

Alguns pontos importantes que estão em fase de melhoramento é o desenvolvimento de uma versão com memória distribuída, o que nos permitirá executar simulação em diversas máquinas e utilizando diversos núcleos. Além disso, pretendemos incluir diferentes modelos de rede e de aplicação para permitir a maior utilização do JSensor pela comunidade da área de RSSFs. Por fim, com o objetivo de validar os modelos de aplicações em RSSFs pretendemos comparar o JSensor com o TOSSIM, que é um simulador que integra as funcionalidades do TinyOS, um Sistema Operacional proprietário de alguns sensores comercias.

URL do JSensor e Demonstração planejada para o SBRC

Sítio: `http://www.joubertlima.com.br/JSensor/`.

Iremos demonstrar uma execução completa do JSensor, com definição e explicação dos cenários possíveis. Sendo eles a área de simulação, tamanho do raio de conexão, criação dos nós e escolha dos modelos do usuário. Estabelecidas tais definições, execução da simulação e posterior verificação dos resultados obtidos, tanto os logs como a verificação dos tempos de execução, utilização de memória e *speedup*.

Referências

Aquino, A. L., Junior, O. S., Frery, A. C., de Albuquerque, E. L., and Mini, R. A. (2013). Musa: Multivariate sampling algorithm for wireless sensor networks. *IEEE Transactions on Computers*, (in press).

Gustafson, J. L. (1988). Reevaluating amdahl's law. *Communications of the ACM*, 31(5).

Khan, M., Askwith, B., Bouhafs, F., and Asim, M. (2011). Limitations of simulation tools for large-scale wireless sensor networks. In *IEEE Workshops of International Conference on Advanced Information Networking and Applications (AINA'11)*.

Lee, H., Manshadi, V., and Cox, D. (2009). High-fidelity and time-driven simulation of large wireless networks with parallel processing. *IEEE Communications Magazine*, 47(3).

Lewis, P., Lee, N., , Welsh, M., and Culler, D. (2002). Tossim: Accurate and sacalable simulation of entire tinyos applications. In *5th Symposium on Operating Systems Design and Implementation*.

Oliveira, L. M. L. and Rodrigues, J. J. P. C. (2011). Wireless sensor networks: a survey on environmental monitoring. *Journal of Communications*, 6(2).

Índice por Autor