XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
24 a 28 de maio de 2010
Gramado, RS

VI Workshop de Redes Dinâmicas e Sistemas Peer-to-Peer (WP2P)

Editora
Sociedade Brasileira de Computação (SBC)

Organizadores
Luis C. E. De Bona (UFPR)
Antônio Jorge Gomes Abelém (UFPA)
Luciano Paschoal Gaspary (UFRGS)
Marinho Pilla Barcellos (UFRGS)

Realização
Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)

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

Copyright © 2010 da Sociedade Brasileira de Computação
Todos os direitos reservados

Capa: Josué Klafke Sperb
Produção Editorial: Flávio Roberto Santos, Roben Castagna Lunardi, Matheus Lehmann, Rafael Santos Bezerra, Luciano Paschoal Gaspary e Marinho Pilla Barcellos.

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

Dados Internacionais de Catalogação na Publicação (CIP)

Workshop de Redes Dinâmicas e Sistemas Peer-to-Peer (6. : 2010 : Gramado, RS).

Anais / VI Workshop de Redes Dinâmicas e Sistemas Peer-to-Peer; organizadores Luis C. E. De Bona et al. − Porto Alegre : SBC, c2010.
113 p.

ISSN 2177-496X

1. Redes de computadores. 2. Sistemas distribuídos. I. De Bona, Luis C. E.. II. Título.

Promoção

Sociedade Brasileira de Computação (SBC)

Diretoria

Presidente
José Carlos Maldonado (USP)

Vice-Presidente
Marcelo Walter (UFRGS)

Diretor Administrativo
Luciano Paschoal Gaspary (UFRGS)

Diretor de Finanças
Paulo Cesar Masiero (USP)

Diretor de Eventos e Comissões Especiais
Lisandro Zambenedetti Granville (UFRGS)

Diretora de Educação
Mirella Moura Moro (UFMG)

Diretora de Publicações
Karin Breitman (PUC-Rio)

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

Diretora de Secretarias Regionais
Thais Vasconcelos Batista (UFRN)

Diretor de Divulgação e Marketing
Altigran Soares da Silva (UFAM)

Diretor de Regulamentação da Profissão
Ricardo de Oliveira Anido (UNICAMP)

Diretor de Eventos Especiais
Carlos Eduardo Ferreira (USP)

Diretor de Cooperação com Sociedades Científicas
Marcelo Walter (UFRGS)

Promoção

Conselho

Mandato 2009-2013
Virgílio Almeida (UFMG)
Flávio Rech Wagner (UFRGS)
Silvio Romero de Lemos Meira (UFPE)
Itana Maria de Souza Gimenes (UEM)
Jacques Wainer (UNICAMP)

Mandato 2007-2011
Cláudia Maria Bauzer Medeiros (UNICAMP)
Roberto da Silva Bigonha (UFMG)
Cláudio Leonardo Lucchesi (UNICAMP)
Daltro José Nunes (UFRGS)
André Ponce de Leon F. de Carvalho (USP)

Suplentes - Mandato 2009-2011
Geraldo B. Xexeo (UFRJ)
Taisy Silva Weber (UFRGS)
Marta Lima de Queiroz Mattoso (UFRJ)
Raul Sidnei Wazlawick (UFSC)
Renata Vieira (PUCRS)

Laboratório Nacional de Redes de Computadores (LARC)

Diretoria

Diretor do Conselho Técnico-Científico
Artur Ziviani (LNCC)

Diretor Executivo
Célio Vinicius Neves de Albuquerque (UFF)

Vice-Diretora do Conselho Técnico-Científico
Flávia Coimbra Delicato (UFRN)

Vice-Diretor Executivo
Luciano Paschoal Gaspary (UFRGS)

Membros Institucionais
CEFET-CE, CEFET-PR, IME, INPE/MCT, LNCC, PUCPR, PUC-RIO, SESU/MEC, UECE, UERJ, UFAM, UFBA, UFC, UFCG, UFES, UFF, UFMG, UFPA, UFPB, UFPE, UFPR, UFRGS, UFRJ, UFRN, UFSC, UFSCAR, UNICAMP, UNIFACS, USP.

Realização

Comitê de Organização

Coordenação Geral
Luciano Paschoal Gaspary (UFRGS)
Marinho Pilla Barcellos (UFRGS)

Coordenação do Comitê de Programa
Luci Pirmez (UFRJ)
Thaís Vasconcelos Batista (UFRN)

Coordenação de Palestras e Tutoriais
Lisandro Zambenedetti Granville (UFRGS)

Coordenação de Painéis e Debates
José Marcos Silva Nogueira (UFMG)

Coordenação de Minicursos
Carlos Alberto Kamienski (UFABC)

Coordenação de Workshops
Antônio Jorge Gomes Abelém (UFPA)

Coordenação do Salão de Ferramentas
Nazareno Andrade (UFCG)

Comitê Consultivo
Artur Ziviani (LNCC)
Carlos André Guimarães Ferraz (UFPE)
Célio Vinicius Neves de Albuquerque (UFF)
Francisco Vilar Brasileiro (UFCG)
Lisandro Zambenedetti Granville (UFRGS)
Luís Henrique Maciel Kosmalski Costa (UFRJ)
Marcelo Gonçalves Rubinstein (UERJ)
Nelson Luis Saldanha da Fonseca (UNICAMP)
Paulo André da Silva Gonçalves (UFPE)

Realização

Organização Local
Adler Hoff Schmidt (UFRGS)
Alan Mezzomo (UFRGS)
Alessandro Huber dos Santos (UFRGS)
Bruno Lopes Dalmazo (UFRGS)
Carlos Alberto da Silveira Junior (UFRGS)
Carlos Raniery Paula dos Santos (UFRGS)
Cristiano Bonato Both (UFRGS)
Flávio Roberto Santos (UFRGS)
Jair Santanna (UFRGS)
Jéferson Campos Nobre (UFRGS)
Juliano Wickboldt (UFRGS)
Leonardo Richter Bays (UFRGS)
Lourdes Tassinari (UFRGS)
Luís Armando Bianchin (UFRGS)
Luis Otávio Luz Soares (UFRGS)
Marcos Ennes Barreto (UFRGS)
Matheus Brenner Lehmann (UFRGS)
Pedro Arthur Pinheiro Rosa Duarte (UFRGS)
Pietro Biasuz (UFRGS)
Rafael Pereira Esteves (UFRGS)
Rafael Kunst (UFRGS)
Rafael Santos Bezerra (UFRGS)
Ricardo Luis dos Santos (UFRGS)
Roben Castagna Lunardi (UFRGS)
Rodolfo Stoffel Antunes (UFRGS)
Rodrigo Mansilha (UFRGS)
Weverton Luis da Costa Cordeiro (UFRGS)

Mensagem dos Coordenadores Gerais

Bem-vindo(a) ao XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2010)! Esta edição do simpósio está sendo realizada de 24 a 28 de maio de 2010 na pitoresca cidade de Gramado, RS. Promovido pela Sociedade Brasileira de Computação (SBC) e pelo Laboratório Nacional de Redes de Computadores (LARC) desde 1983, o SBRC 2010 almeja não menos que honrar com uma tradição de quase 30 anos: ser reconhecido como o mais importante evento científico em redes de computadores e sistemas distribuídos do país, e um dos mais concorridos em Informática. Mais do que isso, pretende estimular intercâmbio de idéias e discussões qualificadas, aproximá-lo(a) de temas de pesquisa efervescentes e fomentar saudável aproximação entre estudantes, pesquisadores, professores e profissionais.

Para atingir os objetivos supracitados, reunimos um grupo muito especial de professores atuantes em nossa comunidade que, com o nosso apoio, executou com êxito a tarefa de construir um **Programa Técnico** de altíssima qualidade. O SBRC 2010 abrange as seguintes atividades: 20 sessões técnicas de artigos completos, cobrindo uma grande gama de problemas em redes de computadores e sistemas distribuídos; 2 sessões técnicas para apresentações de ferramentas; 5 minicursos ministrados de forma didática, por professores da área, sobre temas atuais; 3 palestras e 3 tutoriais sobre tópicos de pesquisa avançados, apresentados por especialistas nacionais e estrangeiros; e 3 painéis versando sobre assuntos de relevância no momento. Completa a programação técnica a realização de 8 *workshops* satélites em temas específicos: WRNP, WGRS, WTR, WSE, WTF, WCGA, WP2P e WPEIF. Não podemos deixar de ressaltar o **Programa Social**, organizado em torno da temática "*vinho*", simbolizando uma comunidade de pesquisa madura e que, com o passar dos anos, se aprimora e refina cada vez mais.

Além da ênfase na qualidade do programa técnico e social, o SBRC 2010 ambiciona deixar, como marca registrada, seu esforço na busca por excelência organizacional. Tal tem sido perseguido há mais de dois anos e exigido muita determinação, dedicação e esforço de uma equipe afinada de organização local, composta por estudantes, técnicos administrativos e professores. O efeito desse esforço pode ser percebido em elementos simples, mas diferenciais, tais como uniformização de datas de submissão de trabalhos, portal *sempre* atualizado com as últimas informações, comunicação sistemática com potenciais participantes e pronto atendimento a qualquer dúvida. O nosso principal objetivo com essa iniciativa foi e continua sendo oferecer uma elevada *qualidade de experiência* a você, colega participante!

Gostaríamos de agradecer aos membros do Comitê de Organização Geral e Local que, por conta de seu trabalho voluntário e incansável, ajudaram a construir um evento que julgamos de ótimo nível. Gostaríamos de agradecer, também, à SBC, pelo apoio prestado ao longo das muitas etapas da organização, e aos patrocinadores, pelo incentivo à divulgação de atividades de pesquisa conduzidas no País e pela confiança depositada neste fórum. Por fim, nossos agradecimentos ao Instituto de Informática da UFRGS, por viabilizar a realização, pela quarta vez, de um evento do porte do SBRC.

Sejam bem-vindos à Serra Gaúcha para o "SBRC do Vinho"! Desejamos que desfrutem de uma semana agradável e proveitosa!

Luciano Paschoal Gaspary
Marinho Pilla Barcellos
Coordenadores Gerais do SBRC 2010

Mensagem do Coordenador do WP2P

Redes e sistemas distribuídos dinâmicos se tornaram um foco de pesquisa importante e presentes no dia-a-dia de indivíduos e organizações. Neste contexto estão incluídas as Redes Móveis e Ad Hoc e os sistemas P2P (Par-a-Par ou Peer-to-Peer). De forma geral, são sistemas tipicamente descentralizados, construídos a partir da interconexão de um grande número de nodos capazes de se auto-organizar, mantendo desempenho aceitável, mesmo frente a padrões de comportamento altamente dinâmicos. Este paradigma tem se mostrado promissor para uma grande gama de aplicações e serviços, como compartilhamento de conteúdo e streaming de áudio e vídeo, apresentando casos de usos de sucessos envolvendo milhares de nodos.

O Workshop de Redes Dinâmicas e Sistemas P2P (WP2P) tem por objetivo constituir um fórum para discussão do estado da arte em tecnologias, aplicações, sistemas e redes dinâmicas e identificar os desafios de pesquisa nesta área. Em 2010, em sua sexta edição, o WP2P conta com 8 artigo selecionados em diferentes linhas de pesquisa. Todos os artigo receberam três avaliações independentes dos membros do Comitê de Programa. O workshop foi realizado em 28 de maio de 2010, em Porto Alegre, Rio Grande do Sul, fazendo parte da vasta grade de atividades do XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.

Agradecemos a todos que participaram e contribuiram para a realização deste workshop. Também agradecemos ao Comitê de Programa, composto de pesquisadores conceituados e atuantes na área, que doaram seu tempo para avaliar de forma criteriosa os trabalhos oferecendo contribuições substânciais para estes. Por fim agradecemos a Coordenação Geral do SBRC, que foi facilitadora e apoiadora em toda o processo.

Luis C. E. De Bona
Coordenador do WP2P 2010

Comitê de Programa do WP2P

Alexandre Sztajnberg, Universidade do Estado do Rio de Janeiro (UFRJ)
Artur Ziviani, Laboratório Nacional de Computação Científica (LNCC)
Carlos Kamienski, Universidade Federal do ABC (UFABC)
Daniel Figueiredo, Universidade Federal do Rio de Janeiro (UFRJ)
Djamel Sadok, Universidade Federal do Pernambuco (UFPE)
Eduardo de Almeida, Universidade Federal do Paraná (UFPR)
Elias P. Duarte Jr., Universidade Federal do Paraná (UFPR)
Fabíola Greve, Universidade Federal da Bahia (UFBA)
Francisco Brasileiro, Universidade Federal de Campina Grande (UFCG)
Joni da Silva Fraga, Universidade Federal de Santa Catarina (UFSC)
José Neuman, Universidade Federal do Ceará (UFC)
Jussara Almeida, Universidade Federal de Minas Gerais (UFMG)
Lisandro Granville, Universidade Federal do Rio Grande do Sul (UFRGS)
Marinho Barcellos, Universidade Federal do Rio Grande do Sul (UFRGS)
Marta Mattoso, Universidade Federal do Rio de Janeiro (UFRJ)
Ronaldo Ferreira, Universidade Federal de Mato Grosso do Sul (UFMS)
Rossana Andrade, Universidade Federal do Ceará (UFC)
Thais Vasconcelos Batista, Universidade Federal do Rio Grande do Norte (UFPR)

Revisores do WP2P

Alexandre Sztajnberg, Universidade do Estado do Rio de Janeiro (UFRJ)
André Lange, Institut de Recherche en Informatique et Systèmes Aléatoires
Artur Ziviani, Laboratório Nacional de Computação Científica (LNCC)
Carlos Kamienski, Universidade Federal do ABC (UFABC)
Daniel Figueiredo, Universidade Federal do Rio de Janeiro (UFRJ)
Davi Boger, Universidade Federal de Santa Catarina (UFSC)
Djamel Sadok, Universidade Federal do Pernambuco (UFPE)
Eduardo de Almeida, Universidade Federal do Paraná (UFPR)
Elias P. Duarte Jr., Universidade Federal do Paraná (UFPR)
Erich Lunardel Silvestre, Universidade Federal de Santa Catarina (UFSC)
Fabíola Greve, Universidade Federal da Bahia (UFBA)
Francisco Brasileiro, Universidade Federal de Campina Grande (UFCG)
Guilher Galante, Universidade Federal do Rio Grande do Norte (UFPR)
Jéferson Nobre, Universidade Federal do Rio Grande do Sul (UFRGS)
Joni da Silva Fraga, Universidade Federal de Santa Catarina (UFSC)
José Neuman, Universidade Federal do Ceará (UFC)
Jussara Almeida, Universidade Federal de Minas Gerais (UFMG)
Lisandro Granville, Universidade Federal do Rio Grande do Sul (UFRGS)
Marinho Barcellos, Universidade Federal do Rio Grande do Sul (UFRGS)
Marta Mattoso, Universidade Federal do Rio de Janeiro (UFRJ)
Rafael Esteves, Universidade Federal do Rio Grande do Sul (UFRGS)
Ronaldo Ferreira, Universidade Federal de Mato Grosso do Sul (UFMS)
Rossana Andrade, Universidade Federal do Ceará (UFC)
Thais Vasconcelos Batista, Universidade Federal do Rio Grande do Norte (UFPR)

Sumário

Sessão Técnica 3 – Redes P2P em Redes Móveis

VI Workshop de Redes Dinamicas e Sistemas Peer-to-Peer

♦

Sessão Técnica 1

Distribuição de Conteúdo

Centralidade em Redes P2P de Transmissão ao Vivo

João F. A. e Oliveira, Pedro de C. Gomes, Alex B. Vieira, Sérgio V. A. Campos

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Minas Gerais, Brasil 30332-0250

{holiver,pcgomes,borges,scampos}@dcc.ufmg.br

Abstract. *In peer-to-peer live streaming systems a few nodes have a special role on content transmission as they have much higher upload rate. Understanding how this pattern emerge is important in several ways, such as to defend these networks against attacks like data pollution or denial of service. This work addresses the problem of identifying these nodes through the use of centrality-like metrics on a real, SopCast-based, p2p live streaming environment. Results show that degree and closeness centrality gives the best correlation with upload rate from all the metrics tested.*

Resumo. *Em redes par-a-par de transmissão ao vivo existem alguns poucos pares que desempenham um papel especial, eles apresentam uma taxa de transmissão significativamente maior que a dos demais. Entender como pares especiais se formam é importante de várias maneiras, como na defesa da rede contra ataques de poluição de conteúdo ou negação de serviço. Este trabalho trata o problema de identificação dos nós especiais através do uso de métricas de centralidade. Os experimentos foram conduzidos em canais criados na rede SopCast, uma das aplicações p2p mais populares para transmissão ao vivo. Os resultados indicam que a centralidade de grau e closeness, entre todas as métricas utilizadas, tem as melhores correlações com a taxa de transmissão de um participante.*

1. Introdução

Ao longo dos últimos anos, redes par-a-par, ou p2p, tem sido usadas para a distribuição de diferentes tipos de mídias através de compartilhamento de arquivo ou na forma de conteúdo sob demanda. Muitos desses sistemas, como o Gnutella [Ripeanu et al. 2002], Kazaa [Leibowitz et al. 2003], eDonkey [Tutschku 2004] e BitTorrent [Cohen 2003], são hoje famosos, consolidados. Seus respectivos sucessos, enquanto sistemas distribuídos, são oriundos de importantes características atribuídas a seus projetos em p2p como tolerância a falha, escalabilidade e otimização do uso de recursos. Todavia, a principal característica de redes p2p é o fato do modelo de transmissão ser cooperativo, sendo que durante a entrega do conteúdo os pares assumam comportamento tanto de clientes quanto de servidores [Schollmeier 2001].

Esse trabalho estuda um tipo de rede p2p que é utilizada para a distribuição ao vivo de conteúdo. Esta tem todas as vantagens e desvantagens de um projeto p2p, com algumas restrições adicionais tais como as necessidades de baixa latência de exibição, características de tempo real, baixa resiliência a poluição e um ponto central de falha associado à fonte da mídia. As redes p2p de distribuição de mídia estudadas nesse artigo são as do tipo *mesh-pull*, cuja descrição será dada nos parágrafos a seguir.

A maioria dos sistemas par-a-par de transmissão ao vivo são baseadas no sistema do Coolstreaming, conhecido também como rede sobreposta orientada a dados (*data-driven overlay*), onde um nó sempre encaminha dados para outros que os esperam, sem papéis prescritos como pai e filho. Em outras palavras, em vez de determinar uma estrutura específica que restrinja a transmissão, a disponibilidade do dado é o que guia a direção do fluxo [Zhang et al. 2005].

O gerenciamento de parcerias, que dá o tipo de organização da rede, indica que os pares devem ao entrar na rede requisitar parceiros de um *bootstrap* que responde com um subconjunto aleatório de pares já presentes na rede. Apesar desse método de inicialização da rede parecer gerar grafos aleatórios foi observado em [Wu et al. 2008] que de fato as redes formadas são do tipo ”mundo pequeno”[Watts 2004]. Este nome ilustra características desse tipo de rede, que apresenta uma distância pequena entre dois vértices quaisquer e participantes que se agrupam em pequenas comunidades.

Sobre o fluxo contínuo de mídia, segundo o protocolo, deve ser dividido em pequenos pedaços, chamados *chunks*, que são as unidades trocadas pelos pares. A fonte da mídia gera continuamente esses *chunks*, cada par da rede (incluindo a fonte) anuncia para seus vizinhos quais *chunks* eles tem e, finalmente, todos escalonam quais pedaços irão pegar de quais parceiros embasados nas suas necessidades individuais e no mapa de *chunks* dos parceiros.

É importante observar que esse esquema de transmissão implica uma característica: para cada *chunk* disseminado é formado um grafo de distribuição *tree-like* com raiz no servidor de mídia. Assim, os dados são injetados na rede através da fonte de conteúdo, os vizinhos da fonte podem requisitar tais dados assim que souberem que eles estão disponíveis, depois os vizinhos dos vizinhos e assim por diante.

Enfim, é possível observar um grafo de propagação de dados em forma de árvore, uma *overlay* com modelo mundo pequeno e, se a rede estiver sobre a Internet, provavelmente uma *underlay* com modelo livre de escala [Faloutsos et al. 1999]. O interessante é que nenhum modelo isoladamente consegue explicar uma característica observada em experimentos reais: mais da metade de todos os dados é transmitida por praticamente cinco por cento dos nós da rede, suas taxas de *upload* são comumente maiores que a da fonte e tais pares serão chamados de super nós de agora em diante. Esse comportamento foi observado em diversas medições usando SopCast[1] sobre o PlanetLab[2] sem nenhuma tendência para que um par específico pertencesse ao conjunto dos super, o que traz a questão: Por que e como surgem os super nós?

Na verdade, essa questão apareceu primeiramente enquanto estudando a propagação de poluição e o papel essencial dos nós com alta taxa de *upload* no ataque. Um super poluidor pode deteriorar mais de 30% de todo *download* da rede e alcançar mais de 50% dos nós [Oliveira et al. 2009]. Responder a essa questão é importante porque se for conhecido como os super nós se formam será possível identificá-los e, então, construir mecanismos de defesa, de incentivo e de prioridade na entrega de dados mais eficientes e conscientes dessa classificação.

Finalmente, o objetivo e contribuição desse artigo é o levantamento de quais

[1]http://www.sopcast.com/
[2]http://www.planet-lab.org/

métricas topológicas de centralidade podem ser melhores na determinação dos super nós, sendo este resultado obtido através da correlação de *ranks* de taxa de *upload* e de tais métricas calculados sobre dados reais e simulados. Na próxima seção serão apresentados os trabalhos relacionados. Na seção 3, é exposto o ambiente de estudo, um pouco sobre o sistema real estudado, o PlanetLab e sobre a simulação. A seção 4 trata diversas definições sobre redes complexas e procedimentos adotados nesta pesquisa. Os resultados sobre correlação são abordados na seção 5. E, finalmente, a seção 6 exibe as conclusões do trabalho.

2. Trabalhos Relacionados

Sobre a aplicação de métricas de centralidade em sistemas de distribuição ao vivo P2P, destaca-se [Wu et al. 2008], que durante uma caracterização do sistema UUSee[3], apresenta o uma correlação de taxa de *upload* e distribuição de graus entre todos os nós de um sistema. Este tipo de mensuração é amplamente aplicada em redes sociais tanto na Internet (Flicker[4], Last.FM[5], Facebook[6] e outras como YouTube[7] [Santos et al. 2009, Kumar et al. 2006]), quanto fora dela em cenários de resolução de problemas em grupo, política, desenvolvimento urbano e projetos de organizações como descrito no trabalho de [Freeman 1979].

Relativo ao tipo de rede estudada, alguns trabalhos, como [Ali et al. 2006, Hei et al. 2007, Silverston and Fourmaux 2007, Tran et al. 2004], realizam medições de parâmetros do comportamento dos pares. Entretanto, nesses trabalhos os dados capturados ficaram basicamente restritos à experiência dos coletores, ou participantes, e pouco abordam sobre visões mais abstratas da rede como um todo.

A proposta deste artigo foi motivada por pesquisas anteriores encontradas em [Oliveira et al. 2009], que aborda um ataque de poluição à rede na perspectiva de um nó que tem alta taxa de *upload* média. A noção de super nó implica um forte ponto de ataque para esse tipo de sistema.

3. Ambiente

As próximas subseções explicarão um pouco sobre as ferramentas usadas nesse trabalho, o SopCast, o PlanetLab e o simulador criado sobre a plataforma Oversim(OMNet++).

3.1. SopCast

Parte do conjunto de dados reais usado nesse experimento é o mesmo de [Oliveira et al. 2009]. Todos os dados foram gerados a partir da aplicação SopCast, um dos mais populares sistemas de distribuição de vídeo ao vivo. Um canal privado foi criado a fim de transmitir um fluxo de 120 *kbps*. Somente pares que sabiam o *id* do canal poderiam ingressar na *overlay* específica e assistir seu conteúdo, ou seja, supõe-se um ambiente isolado de agentes externos (não haviam usuários reais, somente *crawlers* ou *bots*). Além disso, o *churn* da rede é garantido na forma de parcerias promíscuas, ao invés dos nós entrando e saindo, dado que trocas de parceiros são muito frequentes.

[3]http://www.uusee.com/
[4]http://www.flickr.com
[5]http://last.fm
[6]http://www.facebook.com
[7]http://www.youtube.com

Para cada rodada do experimento, esta metodologia de captura compreendeu, inicializar uma versão Linux do cliente SopCast, em um conjunto de 400 nós do PlanetLab. Ao final de cinco minutos todos os participantes deixavam a rede ao mesmo tempo. Apesar da aplicação não ter código-fonte aberto, a coleta é representada pelos *logs* do tráfego na camada de rede obtidas através da execução simultânea do Wireshark[8].

3.2. PlanetLab

PlanetLab é um consórcio mundial de instituições de pesquisa que mantém um ambiente global para o desenvolvimento e testes de aplicações distribuídas. Cada instituição mantém um ou mais nós na Internet que são servidores de máquinas virtuais. Uma conta de acesso ao PlanetLab é chamada de *slice*. Se uma instituição mantiver pelo menos um nó em operação lhe é concedido o direito de criar *slices* e cada *slice* tem o poder de controlar um conjunto de máquinas virtuais em outros nós da rede.

O uso do PlanetLab trouxe muitas vantagens no que se refere a aquisição de dados. A mais óbvia foi a disponibilidade de centenas de nós, o que ajuda a recuperar dados mais representativos e confiáveis. Este experimento, por exemplo, se baseia em dados de cerca de 400 nós. Outra vantagem é que os nós do PlanetLab são dispersos tanto em redes distintas quanto geograficamente, isso evita que aspectos de localização mascarem o comportamento da rede. Por fim, o fato dos nós do PlanetLab possuírem IPs reais e não sofrerem filtragem de pacote evita a necessidade de tratar o problema do NAT (*Network Address Translation*) [Bellovin 2002], onde participantes da transmissão que estão atrás do mesmo *firewall* são identificados como um único participante.

3.3. OverSim

A simulação foi criada com base no Coolstreaming [Zhang et al. 2005] que descreve diversos algoritmos para o projeto de um sistema de distribuição de conteúdo p2p como gerenciamento de participantes, representação e troca do mapa de *bits* do *buffer* e escalonamento de *chunks*. Uma rodada da simulação usa 400 nós, simulando cinco minutos de interações da transmissão de um fluxo de 120*kbps*. O simulador foi desenvolvido com o suporte dos *frameworks* OMNet++[9] e OverSim[10] [Baumgart et al. 2007], eles oferecem suporte para desenvolvimento de simuladores de redes sobrepostas com abstrações de *churn*, canais de transmissão, largura de banda, *bootstrapping* e, inclusive, roteadores e *backbones* para um *underlay*, se necessário.

4. Características de Redes Complexas

Este trabalho fundamenta-se em várias métricas de teoria dos grafos e no estudo de propriedades topológicas das redes complexas. Nesta seção são apresentados os conceitos destas áreas que são determinantes para compreensão do texto.

A rede *overlay* que dissemina um canal do SopCast pode ser modelada por um grafo, onde os vértices representam os participantes da transmissão, e as arestas as parcerias entre eles. Ao interpretar que uma parceria abre uma comunicação bilateral considera-se o grafo não direcionado. Dois vértices são chamados vizinhos se há uma aresta que os

[8]http://www.wireshark.org
[9]http://www.omnetpp.org/
[10]http://www.oversim.org/

conecte, ou no caso da rede p2p, parceiros. O grau de um vértice é o número de arestas que incidem sobre ele, ou neste caso, o seu número de parceiros. O grau médio é a média aritmética do grau de todos os participantes na rede. Note que este valor é igual a duas vezes o número de arestas, divido pelo número de vértices, uma vez que cada aresta contribui para o grau de dois vértices. O valor do grau médio é um dos indicativos do nível de conectividade do grafo.

4.1. Métricas de Centralidade

A teoria dos grafos e análise de rede definem várias medidas de centralidade de um vértice em um grafo que permitem determinar sua importância relativa. Três dessas métricas, que são usadas nesse trabalho, são a centralidade de grau, o *closeness* e o *betweenness*. Elas foram criadas para analisar qualquer tipo de grafo, contudo, para se adaptar ao cenário distinto de distribuição foram criados algumas variações das mesmas. Por exemplo, todos os dados se originam de um único nó, a fonte ou servidor de mídia, e, por isso, foram calculadas duas variações das métricas de *betweenness* e *closeness*, onde somente consideraram-se os caminhos mínimos de todos os vértices para o nó fonte.

A primeira métrica, centralidade de grau, é uma medida relativa ao número de arestas que cada vértice tem. Uma alta centralidade de grau está normalmente associado a uma maior chance de qualquer dado que trafega na rede passar pelo dado vértice. Além disso, a centralidade de grau pode ser calculada para grafos não direcionados e direcionados, onde nesse último caso são considerados o grau de entrada (arestas que chegam no vértice) e grau de saída (arestas que saem do vértice). Já para os experimentos desta pesquisa, onde a rede forma um grafo não direcionado, temos, segundo [Newman 2003], que a métrica pode ser calculada como o grau de cada vértice sobre a quantidade de vértices do grafo menos um:

$$C_{Grau}(v) = \frac{grau(v)}{n-1}$$

A métrica de *closeness* [Sabidussi 1966], é uma medida topológica de proximidade espacial. Trazendo essa medida para a teoria de grafos ela define quão perto um vértice está de todos os outros através das conexões estabelecidos no grafo e passa a ser descrita como a distância geodésica (caminho mínimo) média entre o vértice v e todos os outros vértices alcançáveis a partir dele ($t \in V \setminus v$).

$$C_{Closeness}(v) = \frac{1}{\sum_{t \in V \setminus v} d_G(v,t)}$$

Outra métrica, baseada no *closeness*, levou em consideração que o único vértice alcançável que importa é a fonte, logo, desconsiderando a variação de t. Enfim, essa medida se tornou somente o inverso do caminho mínimo até o servidor de mídia, portanto, foi considerado somente como a distância.

$$C_{Distancia}(v, fonte) = \frac{1}{d_G(v, fonte)}$$

O *betweenness* [Brandes 2001] é uma medida descritiva de quão interno um vértice é no grafo. O valor dele para um vértice v é a quantidade de caminhos mínimos entre os nós s e t que passam por v. Na fórmula a seguir σ_{st} é o número de caminhos mínimos de s a t, e $\sigma_{st}(v)$ é o número de caminhos mínimos de s a t que passam pelo vértice v:

$$C_{Betweenness}(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Por fim, essa fórmula sofreu uma alteração para formar uma nova métrica e t passou a ser percebido como um único vértice específico, a fonte de mídia. Assim, foi calculado um *betweenness* especial, partindo de todos os nós somente até o servidor que passem por v, como:

$$C_{B.Fonte}(v, fonte) = \sum_{s \neq v \neq fonte \in V} \frac{\sigma_{sfonte}(v)}{\sigma_{sfonte}}$$

5. Correlações de Upload e Centralidade

O objetivo principal deste trabalho é a identificação de super nós na *overlay* de um sistema p2p de transmissão de mídia ao vivo. Através de métricas de centralidade [Koschützki et al. 2005], como visto na seção 4 e com base na forma com a qual o conteúdo é disseminado através da rede, supõe-se que será possível revelar os super nós dado que a função de tais métricas é determinar a importância relativa de um vértice num grafo (no caso, um nó numa rede).

De cada experimento foram calculados ou observados, para cada nó, as seguintes métricas: taxa de *upload*, grau, *closeness*, distância ao servidor de conteúdo, *betweenness* normal e dos nós à fonte. Essas medidas foram ordenadas para formar *ranks*, ou seja, é associado o número um ao nó mais importante para uma dada métrica, ao segundo mais importante, dois, e assim por diante. Para a taxa de *upload*, grau, *closeness* e ambos os *betweenness*, os nós mais próximos da primeira posição são aqueles com maiores valores para a medida, enquanto que para distância, são os que tem os menores valores. Por exemplo, para distância, o nó mais perto da fonte é possivelmente o mais valioso por que receberá a mídia primeiro, portanto, sua posição no *rank* terá um valor mais próximo de um.

A criação dos *ranks* foi feita de forma a não ter um critério de desempate, se mais de um par tiver o mesmo valor para uma dada métrica a posição deles no *rank* será igual e a posição do próximo nó com valor de métrica diferente será incrementado do número de nós presentes na posição anterior do *rank*. Isso funciona para quase todas as medidas, contudo, distância dos participantes à fonte é uma medida que variou muito pouco, o que implica em muitos nós empatados numa mesma posição do *rank*. Por isso, a fim de melhorar o resultado da medida, foi criado uma métrica baseada na ordenação do par de métricas (distância, grau) tornando o grau o critério de desempate.

Os experimentos se encaixam em três cenários, com cinco repetições de cada: o real, sobre a rede SopCast, e dois simulados, no Oversim, representando redes onde os

	Grau	Close.	Distância	(Dist.,Grau)	Betw.	B.Fonte
Real SopCast	0.8893	0.8869	0.7994	0.8345	0.8319	0.7562
Sim. Heterogêneo	0.6864	0.6925	0.2960	0.5725	0.7204	0.6991
Sim. Homogêneo	0.1683	0.2316	0.7378	0.7213	0.1236	0.9446

Tabela 1. Coeficientes de Correlação de Pearson.

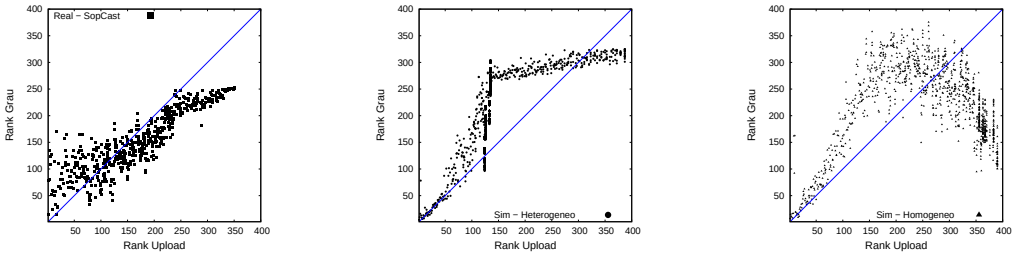
pares tem uma distribuição de largura de banda heterogênea e homogênea. A opção por dois cenários de simulação foi embasada em dois fatos: no desconhecimento da largura de banda dos nós da rede real e nas diferenças comportamentais do protocolo no que diz respeito ao escalonador de *chunks*. Dado que o algoritmo de escalonamento de *chunks* atua de forma a solicitá-los do parceiro mais rápido, se todo nó tem a mesma velocidade, o escalonamento realiza uma seleção aleatória entre as possíveis origens para cada *chunk*, caso contrário é muito provável que o algoritmo selecione sempre os mesmos vizinhos durante as requisições. O objetivo de trazer cenários simulados foi tentar comparar e validar o que foi observado no cenário real.

Foram remontados os grafos da rede a partir dos *logs* de troca de fluxo de mídia, ignorando a parte inicial da coleta a fim de eliminar instabilidades típicas da inicialização da rede. Aqui, as métricas descritas anteriormente foram calculadas a cada 5 segundos sobre um grafo temporalmente dinâmico com base no intervalo entre 100 e 300 segundos dos *logs*. Depois, os *ranks* são obtidos sobre o valor médio de cada métrica em cada nó. Finalmente, para apresentação dos resultados foram calculadas as médias entre os *ranks* das métricas de cada experimento.

A tabela de coeficientes de Pearson (tabela 1) sumariza os resultados e mostra que grau e *closeness* dão *ranks* sistematicamente mais correlatos que as demais métricas. Essa resposta é encontrada também nos experimentos individuais onde não existe uma dominância clara sobre qual das duas é a melhor. Os resultados da simulação heterogênea validam esse resultado, indicando ainda que o *betweenness* é também uma boa métrica, apesar de não ter sido tão interessante para esse conjunto de dados reais. É importante observar que o cenário real e simulado heterogêneo são os que se equivalem e que geram conclusões mais correlatas entre si. Finalmente, o *betweenness* dos nós somente para a fonte emergiu como uma excelente métrica num cenário levemente diferente onde a rede tem uma característica homogênea de pares.

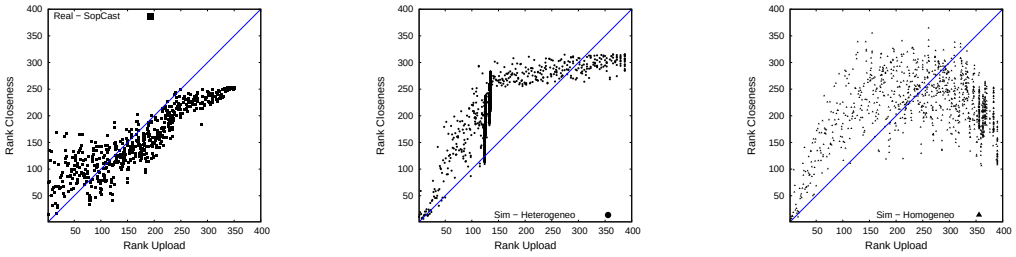
A primeira medida analisada foi centralidade de grau, representada pela figura 1. A imagem 1(a) é uma das que apresentam melhores correlações visuais entre as métricas para os dados reais. A tabela 1 também confirma esse dado numericamente através do coeficiente de correlação de Pearson entre taxa de *upload* e grau. Além disso, os experimentos reais indicam individualmente correlações moderadas no aspecto analisado enquanto a média destaca uma correlação bem forte. É possível observar que a dispersão dos pontos dos experimentos reais no início do *rank* (até 100) tem menos precisão do que o resto dos pontos e baixa exatidão comparado com o resultado esperado, o que implica uma deficiência da métrica em identificar super nós. Apesar disso, em 80% dos experimentos o nó de maior taxa de *upload* teve também o maior grau. Os dados simulados da rede heterogênea aparentam ter dois crescimentos diferentes por que alguns experimentos tiveram pouca variação da taxa de *upload* e muitos nós (+ de 50%) não fizeram *upload*

portanto saturando o *rank*. No entanto, a correlação no início da curva (até 50) é visualmente linear. Este comportamento é melhor explicado pelo modelo de disseminação em árvore, um nó que possui mais parceiros tem grandes chances de emergir como um super nó, além disso, pode ser inferido que a forma de disseminação supera os modelos de organização da *overlay* e da *underlay*. Finalmente, na distribuição de *ranks* de graus da simulação homogênea da figura 1(c) observamos uma forma de sino indicando uma correlação quase inexistente entre as métricas para esse caso.



(a) Real (b) Simulado Heterogêneo (c) Simulado Homogêneo

Figura 1. Correlação entre ranks de Grau e Upload.

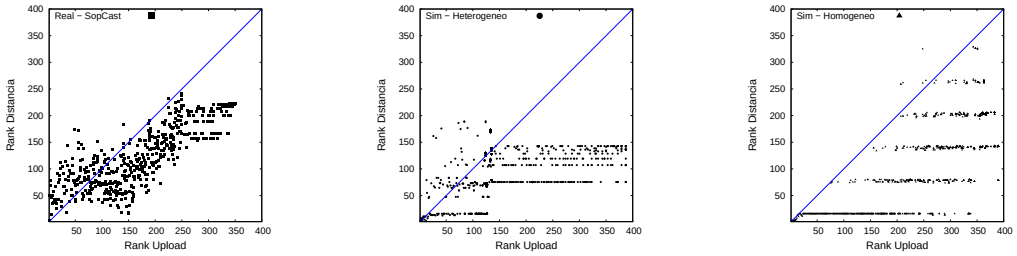


(a) Real (b) Simulado Heterogêneo (c) Simulado Homogêneo

Figura 2. Correlação entre ranks de Closeness e Upload.

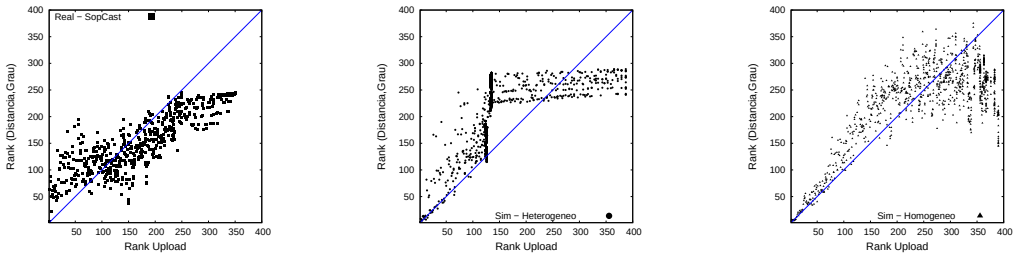
O estudo de *closeness* (figura 2) obteve resultados similares ao de grau inclusive no que diz respeito a análises mais finas como: o par de maior *closeness* ser o de maior taxa de *upload*. Numericamente é possível observar na tabela de coeficientes que essa métrica é tão boa quanto a da centralidade de grau.

A terceira métrica, distância, presente na figura 3, apresentou a pior representação visual apesar de posar numericamente como uma correlação pouco mais fraca que as últimas para o caso real. Individualmente nos experimentos reais essa métrica é sistematicamente ruim comparada com as melhores métricas, sendo isso validado através da simulação heterogênea. No gráfico 3(a) a distribuição real teve poucos patamares por conta da média dos experimentos, contudo, geralmente esta métrica teve pouca variação, assim, aglomerando a análise em níveis. De toda forma, isso permitiu deduzir que a árvore de disseminação tenta se manter curta mas com carga baixa na raiz, uma vez que, para distância do nó-fonte > 1, *ranks* baixos (menores distâncias) concentram mais pontos. Ademais, distância não aparenta ser uma boa métrica neste cenário onde a rede não permanece estática com o tempo e, apesar de numericamente a simulação homogênea encontrar um bom resultado, é possível observar graficamente que é impraticável identificar super nós através dessa métrica.



(a) Real (b) Simulado Heterogêneo (c) Simulado Homogêneo

Figura 3. Correlação entre ranks de Distância e Upload.

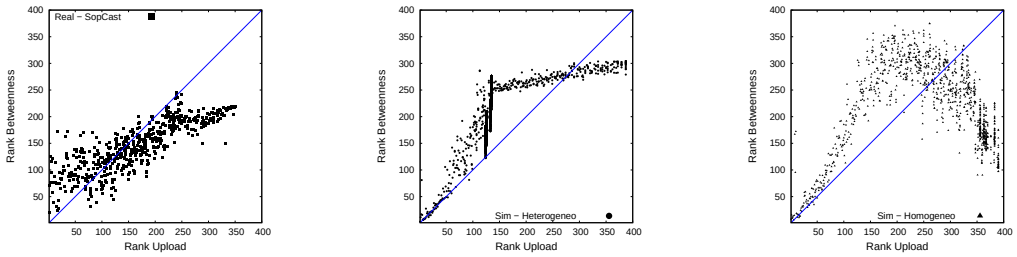


(a) Real (b) Simulado Heterogêneo (c) Simulado Homogêneo

Figura 4. Correlação entre ranks de (Distância,Grau) e Upload.

Na figura 4, é mostrado a métrica que usa o par (distância, grau) ordenado para formar o *rank*. Essa medida foi capaz de melhorar a variação de valores do eixo Y, especialmente sobre a média dos experimentos, mas o resultado geral não melhorou muito. Visualmente alguns dos níveis observados na figura 3 são ainda um pouco presentes na figura 4 e muito presente nos experimentos individuais. Numericamente a métrica composta alcançou um resultado intermediário comparado com seus componentes.

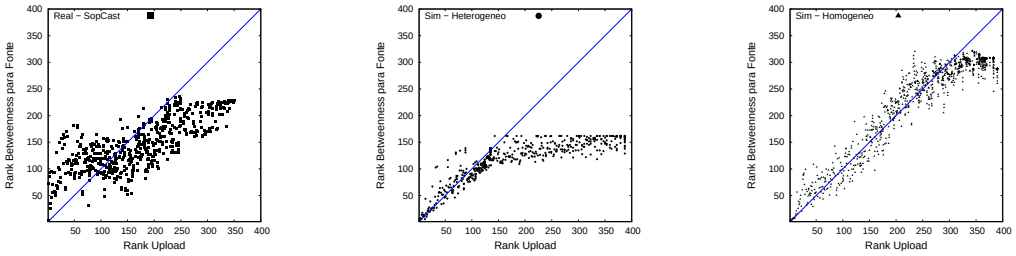
O gráfico do *rank* de *betweenness* pode ser visto na figura 5. A análise é novamente similar à das métricas de *closeness* e grau, trazendo poucas diferenças praticas. Todavia, de acordo com a tabela 1 essa métrica é menos efetiva que as citadas para o caso real apesar de ser até melhor que as demais de acordo com a simulação heterogênea.



(a) Real (b) Simulado Heterogêneo (c) Simulado Homogêneo

Figura 5. Correlação entre ranks de Betweenness e Upload.

Finalmente, a última métrica analisada foi o *betweenness* de todos os nós somente até a fonte que deu resultados bem diferentes para os três cenários, como mostra a figura 6 e a tabela 1. Essa exceção era esperada justamente por conta da diferença comportamental



(a) Real (b) Simulado Heterogêneo (c) Simulado Homogêneo

Figura 6. Correlação entre ranks de Betweenness para Fonte e Upload.

do escalonador. Nos cenários heterogêneos é mais provável que os nós mais distantes do servidor façam nenhum ou pouco envio e que sejam aqueles que tem os menores valores para essa métrica. Além disso, os caminhos de distribuição de dados são mais fixos que no homogêneo. A rede analisada é remontada a partir dos *logs* de distribuição, e nos cenários heterogêneos a métrica tem menor variabilidade já que ao longo do tempo o grafo sobre o qual se faz as análises muda pouco. Ou seja, como podemos verificar através da figura 6(c), essa métrica criada é muito boa para identificar o *rank* de *upload* dos nós num cenário de seleção de *chunks* de pares aleatório, mas não para o cenário real que é heterogêneo e o escalonamento leva banda em consideração.

6. Conclusões e Trabalhos Futuros

O trabalho conclui que tanto a centralidade de grau quanto o *closeness* oferecem as melhores correlações com taxa de *upload* entre todas as métricas de centralidade testadas (grau, *closeness*, distância, o par (distância, grau), *betweenness* e *betweenness* dos nós para a fonte). No cenário real, apesar do coeficiente de correlação dessas métricas serem altos para a média dos experimentos, individualmente eles são moderados, próximos de 0.50. Através do conjunto de dados da simulação com pares heterogêneos foi possível validar os resultados obtidos e induzir ao entendimento de que o modelo de disseminação supera tanto o modelo de *overlay* quanto o de *underlay*, ou seja, o algoritmo de escalonamento de *chunks* influencia mais na correlação do que a organização das redes sobreposta e física.

Através do *betweenness* dos nós para a fonte, uma modificação da métrica de *betweenness* original, foi possível encontrar a melhor correlação num ambiente de pares homogêneos, ou no qual o algoritmo de escalonamento escolha parceiros aleatórios que, minimamente, tenham o *chunk* desejado. Esse resultado, aliado às outras diferenças entre os coeficientes de correlação de Pearson para os cenários simulados, indica que a largura de banda dos pares influencia significativamente os resultados. Isso motiva novos experimentos e testes, incluindo o *rank* de larguras de banda nas medidas e outras variações como o par (largura de banda, grau) ou (largura de banda, *closeness*).

Para os trabalhos futuros planeja-se uma identificação mais sistemática de super nós apoiada por uma combinação destes resultados. Além disso, espera-se adquirir informação sobre a distribuição da largura de banda entre os nós do Planetlab. Com isso será possível comparar cenários reais e simulados através da nova métrica proposta anteriormente de forma que leve a uma conclusão definitiva sobre o comportamento emergente de super nós.

Referências

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*.

Baumgart, I., Heep, B., and Krause, S. (2007). OverSim: A Flexible Overlay Network Simulation Framework. In *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, pages 79–84.

Bellovin, S. M. (2002). A technique for counting natted hosts. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 267–272, New York, NY, USA. ACM.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177.

Cohen, B. (2003). Incentives build robustness in bittorrent.

Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, New York, NY, USA. ACM.

Freeman, L. (1979). Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

Hei, X., Liang, C., Liang, J., Liu, Y., and Ross, K. (2007). A Measurement Study of a Large-Scale P2P IPTV System. *Multimedia, IEEE Transactions on*, 9(8):1672–1687.

Koschützki, D., Lehmann, K. A., Peeters, L., Richter, S., Podehl, D. T., and Zlotowski, O. (2005). *Centrality Indices*, volume 3418/2005 of *Lecture Notes in Computer Science: Network Analysis*, chapter Part I: Elements, pages 16–61. Springer Berlin / Heidelberg.

Kumar, R., Novak, J., and Tomkins, A. (2006). Structure and evolution of online social networks. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 611–617, New York, NY, USA. ACM.

Leibowitz, N., Ripeanu, M., and Wierzbicki, A. (2003). Deconstructing the kazaa network. In *WIAPP '03: Proceedings of the The Third IEEE Workshop on Internet Applications*, page 112, Washington, DC, USA. IEEE Computer Society.

Newman, M. (2003). The structure and function of complex networks. *Arxiv preprint cond-mat/0303516*.

Oliveira, J., Vieira, A., and Campos, S. (2009). Poluição de conteúdo em sistemas p2p live streaming. In *Simpósio Brasileiro de Sistemas Multimídia e Web - Webmedia*.

Ripeanu, M., Iamnitchi, A., and Foster, I. (2002). Mapping the gnutella network. *IEEE Internet Computing*, 6(1):50–57.

Sabidussi, G. (1966). The centrality index of a graph. *Psychometrika*, 31(4):581–603.

Santos, R., Rocha, B., Rezende, R., and Loureiro, A. (2009). Characterizing the YouTube video-sharing community. *http://security1.win.tue.nl/ bpontes/pdf/yt.pdf*, 12.

Schollmeier, R. (2001). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. *Peer-to-Peer Computing, IEEE International Conference on*, 0:0101.

Silverston, T. and Fourmaux, O. (2007). Measuring P2P IPTV Systems. In *Proc. of ACM NOSSDAV*.

Tran, D., Hua, K., and Do, T. (2004). A peer-to-peer architecture for media streaming. *Selected Areas in Communications, IEEE Journal on*, 22(1):121–133.

Tutschku, K. (2004). A measurement-based traffic profile of the eDonkey filesharing service. *Lecture notes in computer science*, pages 12–21.

Watts, D. J. (2004). *Six Degrees: The Science of a Connected Age (Open Market Edition)*. W.W. Norton & Co.

Wu, C., Li, B., and Zhao, S. (2008). Exploring large-scale peer-to-peer live streaming topologies. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 4(3):1–23.

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.

Reduzindo o Tráfego Gerado por Aplicações Par-a-Par para Distribuição de Vídeo sob-Demanda na Internet

Josilene Moreira[1], Petrônio Gomes[1], Victor Souza[2], Djamel Sadok[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE – Brazil

[2]Ericsson Research - Stockholm, Sweden

{josilene,petronio,jamel}@cin.ufpe.br, victor.souza@ericsson.com

Abstract. *New strategies for Video on-Demand distribution (VoD) have been proposed, including the use of peer-to-peer (P2P) networks. However, this strategy transfers the cost of distribution from content providers to service providers (ISPs). One solution to minimize this cost is the use of caches that store the P2P traffic and try to keep it local to the ISP. This paper evaluates the use of caches for P2P VoD traffic, proposing an optimization for a partial cache algorithm that improves the performance of the original algorithm up to 12.7%. The major contribution is to demonstrate that the use of cooperation between ISPs improves the efficiency of the cache in more than 100%, significantly reducing transit traffic (12.6%).*

Resumo. *Novas estratégias de distribuição de Vídeo sob-demanda (VoD) têm sido propostas, entre elas o uso de redes par-a-par (P2P). Entretanto esta estratégia transfere o custo de distribuição dos provedores de conteúdo para os provedores de serviço de Internet (ISPs). Uma das soluções para minimizar este custo é o uso de caches que armazenam o tráfego P2P e procuram mantê-lo local ao ISP. Este artigo avalia o uso de caches para tráfego P2P VoD, propondo uma otimização de um algoritmo de cache parcial que melhora o desempenho do algoritmo original em até 12,7%. A maior contribuição é demonstrar que o uso de cooperação entre ISPs melhora a eficiência da cache em mais de 100%, reduzindo significativamente o tráfego de trânsito (12,6%).*

Palavras-chave*: estratégias de cache, caches cooperativas, VoD*

1. Introdução

Nos dias de hoje, a popularidade de sistemas Par-a-Par (P2P) para distribuição de conteúdo tem aumentado rapidamente. Esta classe de aplicações é responsável pela maior parte do tráfego gerado entre ISPs (Provedores de Serviço Internet) e o crescente uso desse tipo de rede de compartilhamento sugere uma tendência de aumento ainda maior no futuro [Hefeeda and Saleh 2008].

Recentemente, além de ser usada para compartilhamento de arquivos, a tecnologia P2P também tem sido largamente utilizada em aplicações de distribuição de vídeo na Internet, tanto ao vivo (*Live Stream*) como sob-demanda (*Video on-Demand*). Diversas aplicações de origem chinesa como *PPLive*, *TVU*, *SOPCast* e *PPStream*, entre

outras, utilizam esta tecnologia. Adicionalmente, algumas redes de distribuição de conteúdo (*CDNs*) também têm usado a tecnologia P2P para alavancar seus negócios e diminuir os custos de distribuição. Entre elas estão a *Velocix*[1] , a *GridNetworks* (recentemente incorporada pela *Global Media Services*[2] e a *Akamai*[3], que adquiriu a empresa especialista em P2P *Red Swosh* em 2007 e começou a oferecer serviços baseados nesta tecnologia em 2009[4].

Entretanto, os Provedores de Serviço de Internet (ISPs) não estão satisfeitos com o uso da tecnologia P2P. A principal razão é que os custos de distribuição estão sendo transferidos dos provedores de conteúdo para os ISPs [Blond et al. 2008] [Hefeeda and Saleh 2008] [Karagiannis et al. 2005] . Como a maioria dos protocolos P2P não está preocupada em selecionar nós dentro do mesmo ISP, o uso destas aplicações tem gerado um custo de tráfego de trânsito bastante significativo para os ISPs. Embora existam acordos de troca de tráfego entre ISPs (enlaces de *peering*), os quais geralmente não incorrem em custos, é impossível que um ISP tenha acordos com todos os ISPs existentes. Normalmente, um ISP paga por enlaces de trânsito que o conectam indiretamente aos outros ISPs com os quais ele não possui conexão direta. Pesquisas recentes mostram que apenas cerca de 8% do tráfego gerado por aplicações P2P permanece interno ao ISP; 92% trafega por enlaces de trânsito e de *peering* [Shen et al. 2007]. Ressalta-se ainda que a maior parte dos *bytes* transferidos em aplicações P2P decorre de grandes objetos [Gummadi et al. 2003] [Leibowitz et al. 2002] .

Com o objetivo de reduzir este tráfego, os ISPs têm tomado iniciativas como o bloqueio de aplicações P2P. Esta solução não é tão simples, visto que pode depender de técnicas sofisticadas tanto de *hardware* como de *software* e, mesmo assim, as aplicações sempre estão tentando alguma forma de esquivar-se à detecção dos seus protocolos [Feitosa et al. 2008]. Uma segunda alternativa é o uso de *caches* a fim de melhorar a localidade de tráfego, as quais atuam interceptando as requisições dos nós clientes [Dán 2009] [Karagiannis et al. 2005]. As *caches* geralmente usam o mesmo protocolo do sistema P2P e funcionam como um nó de grande capacidade que "atrai" as requisições dos usuários por ter grande capacidade e largura de banda. O conteúdo primeiro é buscado internamente ao ISP e só será acessado externamente se não for encontrado na *cache* ou se a sua capacidade for excedida.

Este trabalho estuda especificamente a contribuição das *caches* para a redução do tráfego P2P em sistemas de distribuição de Video sob-Demanda (VoD). Um sistema de vídeo geralmente distribui objetos bem maiores do que um sistema convencional de compartilhamento de arquivos como *Kazaa* e *BitTorrent*, e portanto seu uso intenso impacta fortemente no perfil de tráfego de um ISP. Apresenta-se um algoritmo de *cache* parcial modificado que melhora a eficiência da *cache* em 12,7% sobre o algoritmo original, através do armazenamento dos segmentos mais populares dos objetos [Hefeeda

---

[1] Velocix. http://www.velocix.com/. Acessado em 10/12/2009.

[2] BusinessWire.http://www.businesswire.com/portal/site/home/permalink/?ndmViewId=news_view&newsId=20090417005446&newsLang=en. Acessado em 05/04/2010

[3] Akamai. http://www.akamai.com/.Acessado em 15/04/2010

[4] Ligth Reading. http://www.lightreading.com/document.asp?doc_id=121710. Acessado em 12/12/2009.

and Saleh 2008] [Yu et al. 2006]. Entretanto, a principal contribuição deste trabalho é a avaliação da estratégia de cooperação entre *caches* de diferentes ISPs sobre a redução do tráfego de trânsito. O uso de *caches* cooperativas chega a melhorar o desempenho do algoritmo de *cache* em mais de 100%, aumenta o tráfego local em 128% e reduz o tráfego de trânsito em até 12,6%.

O artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados, a Seção 3 descreve a metodologia utilizada nas avaliações, a Seção 4 apresenta os resultados e a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos relacionados

Algumas pesquisas recentes tratam das estratégias de redução de tráfego entre ISPs. Em [Karagiannis et al. 2005], os autores apresentam um dos primeiros trabalhos a avaliar o impacto do tráfego P2P sobre os ISPs, mostrando que os custos de distribuição estão sendo transferidos dos provedores de conteúdo para os provedores de serviço de Internet. As idéias do uso de *caches* e do conceito de localidade de tráfego são avaliadas a partir de coletas de tráfego BitTorrent[5] , concluindo que essas estratégias podem beneficiar significativamente os ISPs, reduzindo seus custos.

Em [Choffnes and Bustamante 2008], é proposta uma abordagem para reduzir o custo do tráfego inter-ISP sem que o desempenho do sistema P2P seja sacrificado. Na seleção, que preza por localidade, os pares se comunicam com vizinhos que teoricamente estão próximos utilizando as informações coletadas pelo mecanismo de redirecionamento de uma CDN (*Content Distribution Network*). Isso implica que não é necessária uma nova infra-estrutura e nem cooperação entre provedores de serviços de Internet para colocar em prática essa abordagem. Para avaliar a solução proposta foi realizada uma implementação de um cliente BitTorrent que tentam encontrar 'caminhos entre nós' que possibilitem reduzir o tráfego inter-ISP. No entanto, esta solução depende do uso de uma espécie de oráculo, que é a CDN, para encontrar os nós mais próximos.

Devido à tensão existente entre aplicações P2P e ISPs, os autores em [Shen et al. 2007] propuseram uma nova estratégia chamada de HPTP, *HTTP-based Peer-to-Peer*. Essa técnica propõe a utilização de *caches* já existentes nos ISPs usadas para o armazenamento de tráfego *web* para alternativamente armazenar tráfego P2P. Para isso, os autores descreveram um processo denominado '*HTTPifying*', que consiste na segmentação dos arquivos P2P em pedaços menores para serem encapsulados, transportados e tratados pelas *caches* como tráfego HTTP. Nesse trabalho, por meio de simulações, ganhos significativos foram identificados, como a redução da carga de tráfego entre ISPs e no *backbone* da Internet sem comprometer a aplicação P2P envolvida.

Em [Hefeeda and Saleh 2008], um estudo sobre características relevantes do tráfego P2P foi realizado, avaliando o benefício do uso de *caches*. O tráfego P2P foi coletado e caracterizado, analisando-se as distribuições de popularidade dos objetos. Posteriormente, um algoritmo de *cache* parcial foi proposto, baseado na modelagem realizada anteriormente. Seguindo a idéia de *cache* para objetos *web*, o algoritmo visa minimizar os principais problemas provenientes do tráfego P2P. As simulações mostram

[5] BitTorrent. http://www.bittorrent.com. Acessado em 15/10/2009.

altos índices de desempenho, sendo possível constatar a importância do algoritmo de *cache* parcial para esse tipo de tráfego. Este algoritmo serviu como base para as alterações e otimizações propostas pelo nosso trabalho.

Apesar de ter um grande potencial, esquemas de caches cooperativas envolvendo tráfego P2P não têm sido muito abordados na literatura. Em [Hefeeda and Noorizadeh 2008], é defendida a idéia de que o esquema de caches cooperativas é mais útil ao tráfego P2P que ao tráfego mais comum da *web*, pois os objetos P2P são repetitivos [Leibowitz et al. 2002] e apresentam pouquíssimas mudanças, sendo considerados praticamente imutáveis por [Gummadi et al. 2003]. Em [Dán 2009], o autor propõe um esquema de *caches* cooperativas compatível com as relações de negócio existentes entre ISPs, onde o problema é modelado usando a teoria dos grafos. Os resultados apresentados demonstram matematicamente a capacidade do esquema de *caches* cooperativas de minimizar os custos oriundos do tráfego P2P para os ISPs. O estudo é realizado para vídeo ao vivo e mostra uma análise matemática, não utilizando simulação das requisições para os objetos armazenados nas *caches* como faz o nosso trabalho.

Em [Hefeeda and Noorizadeh 2008], também é analisado o potencial de *caches* cooperativas para a redução do tráfego de trânsito causado por aplicações P2P. São propostos dois modelos de cooperação entre *caches* de diferentes ASs e *cache*s de um mesmo AS. Um *trace* com oito meses de duração foi coletado para clientes *Gnutella*[6] e diversas simulações foram executadas, alternando entre os dois modelos propostos. Por fim, foram apresentados os resultados que destacam a relevância da cooperação entre *caches* e o *overhead* gerado. Nosso trabalho é complementar a este, pois explora especificamente o potencial de *caches* para tráfego de VoD, além de usar simulações específicas que reproduzem as requisições de clientes para objetos de vídeo em *caches* cooperativas.

3. Metodologia

Para avaliação do desempenho das *caches* foram criadas carga de dados (*workloads*) sintéticas através do gerador *ProWGen* [Busari and Wiliamson 2002]. Este gerador permite modelar diversas características importantes de uma carga de dados tais como popularidade dos objetos, correlação temporal entre as requisições e tamanho médio dos objetos, proporcionando a reprodução de um conjunto de dados muito próximos da realidade.

A fim de reproduzir os cenários da maneira mais realista possível para as avaliações de desempenho, as cargas sintéticas foram construídas a partir das características de coletas de tráfego P2P reais (*traces*) descritas em [Hefeeda and Saleh 2008]. A construção das cargas sintéticas reproduz inicialmente as características de distribuição de popularidade, tráfego *cacheable* (passível de ser armazenado na *cache)* e requisições dos dois sistemas autônomos (ASs) mais representativos das coletas efetuadas. A partir dos cenários básicos destes dois ASs, estas características são então variadas, a fim de obter uma avaliação mais abrangente. As requisições dos usuários foram criadas para objetos de VoD e processadas de encontro às *cache*s usando os algoritmos propostos.

[6] Gnutella. http://www.gnutella.com. Acessado em 12/12/2009.

3.1.Cenários

Utilizando o *ProWGen*, foram simulados inicialmente dois sistemas autônomos (ASs) onde o tráfego e os objetos requisitados apresentam diferentes características de popularidade e de percentual de tráfego *cacheable* (Tabela 1).

Tabela 1. Característica dos cenários

	AS397	**AS95**
Número de usuários	9315	9316
Número de objetos	3000	3000
Número de blocos por vídeo	20	20
Número de requisições	186300	186320
Tráfego *cacheable*	48%	54%
Distribuição de popularidade	MZipf (0.62, 8)	MZipf (0.6, 50)
Tamanho dos vídeos	1 GB	1 GB

Estes ASs foram escolhidos por apresentarem características distintas e serem bastante representativos dos perfis de tráfego observados. Para o AS397 aproximadamente 4,5 TB (48%) do tráfego pode ser potencialmente armazenado pela *cache* (tráfego *cacheable*). Já no AS95, este tráfego é de 54%, o que representa cerca de 5 TB. Além disso, a distribuição de popularidade é diferente para os dois ASs. Nas Seções 4.2 e 4.3 analisamos como estas duas características podem impactar no desempenho do algoritmo.

3.2. Requisições dos usuários

As requisições de objetos de VoD são distintas daquelas de compartilhamento de arquivos em sistemas P2P. Nos principais algoritmos de VoD-P2P como PPLive [Huang et al. 2008], o controle de quais segmentos do vídeo serão requisitados é de responsabilidade dos nós usuários. Nestes sistemas, os usuários assistem ao vídeo enquanto este está sendo baixado, com um certo atraso inicial (*startup delay*) e certa falha de continuidade (*playback continuity*). A aplicação é encarregada de escalonar quais segmentos o usuário precisa baixar a fim de minimizar atrasos. Neste trabalho as requisições dos objetos foram geradas para corresponder àquelas realizadas em sistemas de VoD.

Objetos de vídeo geralmente são maiores do que objetos *web* [Leibowitz et al. 2002]. Um objeto é composto por um determinado número de segmentos, que é a menor unidade manipulada pela *cache*. As requisições dos objetos são feitas em blocos compostos por *n* segmentos. Nos cenários dos experimentos, cada bloco tem um tamanho fixo de 50 segmentos, cada segmento com 1MB. Dessa forma, um objeto de 1GB é composto por mil segmentos de 1MB, e é requisitado em 20 blocos de 50 segmentos. As requisições são geradas seqüencialmente para os blocos do mesmo objeto, considerando que o usuário assiste ao vídeo do início ao fim (não são consideradas operações de adiantar ou retroceder a reprodução).

Apesar das requisições serem geradas seqüencialmente para os blocos de um mesmo vídeo, na Seção 4.4 isola-se o efeito da seqüencialidade, analisando-se a correlação temporal sobre o desempenho do algoritmo, gerando requisições com correlação temporal fraca, média e forte.

3.3. Algoritmo de *cache*

Através da análise de um algoritmo de armazenamento parcial apresentado na Figura 1, o qual propõe a escolha dos segmentos a serem armazenados de acordo com a popularidade e o tamanho dos objetos, verificou-se que o algoritmo continha algumas falhas graves e que o desempenho da *cache* poderia ser otimizado. O algoritmo original determina a quantidade e quais segmentos serão armazenados na *cache*. A escolha é realizada dinamicamente e atualizada a cada vez que o objeto é requisitado.

A função de utilidade do objeto *i*, baseada em sua popularidade, é denotada por *Yi* enquanto que a função de utilidade do objeto mais popular é armazenada em *Y*. A filosofia do algoritmo é manter na *cache* um número de segmentos diferente para cada objeto, proporcionalmente à sua popularidade.

```
SE objeto i ∉ cache
        insira um segmento do objeto i na cache, removendo caso necessário
SENÃO
        hit = segmentos de i na cache ∩ segmentos requisitados
        Yi = Yi + (hit/segmentos de i na cache)
        cache miss = (segmentos requisitados – hit)/(tamanho do segmento)
        x = (Yi/Y)*(tamanho médio dos objetos da rede)
        k = min(cache miss, max(x, 1))
        SE espaço ocupado da cache + k segmentos > tamanho da cache
                remova k segmentos do objeto menos popular
        adicione k segmentos do objeto i a cache
FIM
```

Figura 1. Algoritmo básico

No entanto este algoritmo apresenta uma falha grave, pois um objeto *i* que foi muito popular em um determinado período não tem sua função de utilidade decrementada. Este fato pode gerar duas conseqüências; a primeira é que o objeto *i* tardará muito a sair da *cache.* A segunda é que um objeto *j* que esteja se tornando muito popular terá que ultrapassar o valor da função de utilidade do objeto *i* que já foi mais popular um dia para que seja considerado o mais 'valioso' (com a maior função de utilidade) da *cache*. Isto pode não acontecer nunca, e assim o objeto será mantido por um período muito longo na *cache*, diminuindo o seu desempenho.

O algoritmo também pode ser melhorado com relação ao seu desempenho geral, através da alteração de sua política de remoção de objetos da *cache* e do cálculo dos segmentos a serem armazenados. Na verdade, com a aplicação destas modificações, podemos considerar que um novo algoritmo foi construído. As modificações realizadas para melhorar seu desempenho e corrigir a falha na temporalidade dos objetos mais populares são:

(i) Remoção de objetos *(eviction)*: quando é necessário remover algum item da *cache,* o objeto menos popular é identificado e os segmentos são retirados do fim para o início. O algoritmo original removia quaisquer segmentos do objeto menos popular.

(ii) Modificação do total de segmentos a ser armazenado: a variável *hit* deixa de ser proporcional ao número de segmentos do objeto na *cache* e é modificada para ser proporcional ao número de acertos. Isto faz com que o algoritmo responda mais rapidamente às alterações de popularidade dos objetos.

(iii) Introdução do GDS (*Greedy Dual Size*): no algoritmo original, um vídeo que foi um dia muito popular não tem a sua função de utilidade decrementada no decorrer do tempo. Desse modo, para que este vídeo possa sair um dia da *cache,* a cada vez que um vídeo é removido, o valor da sua função de utilidade é decrementado de todos os valores das funções de utilidade dos outros objetos que ainda estão na *cache*.

Percebeu-se que, com essas modificações, o algoritmo tornou-se mais "agressivo", adaptando-se mais rapidamente às mudanças de popularidade dos objetos e assim alcançando melhor desempenho, além de corrigir o problema da longa permanência de um objeto muito popular na *cache*.

3.4. Modelos de cooperação

Este estudo analisa a cooperação entre *cache*s de ISPs diferentes, conforme a Figura 2.

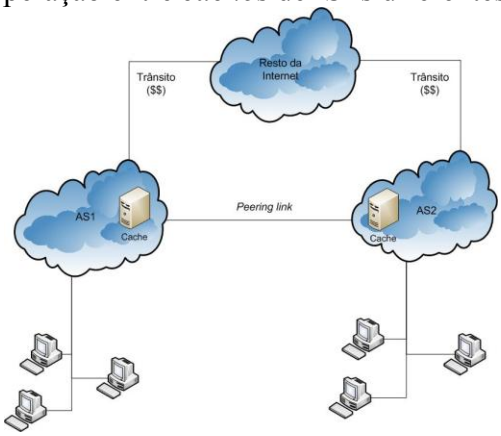


Figura 2. Modelo de cooperação entre *caches* de diferentes ASes

Cada ISP é representado por um AS distinto, AS1 e AS2. Normalmente os ISPs possuem acordos que permitem que uma quantidade de tráfego semelhante seja trocada entre eles sem incorrer em custo, conhecido como acordos de troca de tráfego ou *peering*. A comunicação com outros ISPs com os quais não existem acordos de troca (representado na figura como "resto da Internet") geralmente acontece através de enlaces de trânsito e incorre em custo para o ISP que origina o tráfego.

Para as análises da cooperação deste trabalho, cada ISP possui uma *cache* com o objetivo de armazenar os objetos mais populares e assim reduzir a quantidade de tráfego de trânsito, reduzindo o custo total de tráfego para o ISP. Quando uma determinada *cache* de um ISP recebe uma requisição de um de seus usuários, uma das seguintes situações pode acontecer:

(1) Todos os segmentos são encontrados e a requisição é completamente atendida localmente pela *cache*; o algoritmo de *cache* descrito é aplicado para decidir que segmentos do objeto devem ser armazenados;

(2) Apenas parte dos segmentos é encontrada e a requisição é parcialmente atendida pela *cache* local; o algoritmo de *cache* decide que segmentos devem ser armazenados, ou

(3) Nenhuma parte do objeto requisitado é encontrada na *cache* local.

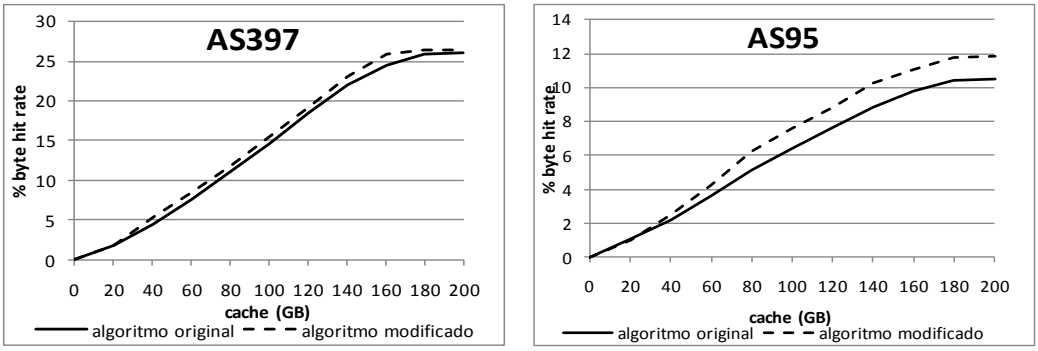
Nos casos (2) e (3) a requisição que não foi atendida (completa ou parcialmente) será redirecionada para a *cache* de um ISP vizinho, acontecendo assim a cooperação. Dessa forma espera-se que a maioria do tráfego gerado pelos usuários P2P possa ser

atendida pelo próprio ISP que originou a requisição ou por um ISP com o qual haja um acordo de troca de tráfego, minimizando o tráfego de trânsito.

4. Resultados

4.1.O Impacto do Algoritmo sobre a Eficiência da *Cache*

A fim de avaliar o desempenho do algoritmo modificado em condições reais de tráfego P2P, foram consideradas as características originais dos ASs 397 e 95 (Figura 3). Observa-se que em ambos os cenários o desempenho do algoritmo modificado é melhor do que o desempenho do algoritmo original. Entretanto, verificam-se duas situações distintas. O algoritmo modificado apresenta um melhor desempenho para o AS95, sendo 12,7% melhor que o algoritmo original, o que representa 1,18 TB de tráfego que é mantido internamente ao AS95. Já no AS397 o algoritmo modificado obtém uma diferença positiva de desempenho de 1,44%; como o tráfego total do AS95 é de 9,3 *terabytes*, mesmo este pequeno aumento na eficiência do algoritmo de *cache* representa quase 120 GBytes de tráfego que será mantido internamente ao AS.



(a) (b)

Figura 3. Desempenho do algoritmo modificado

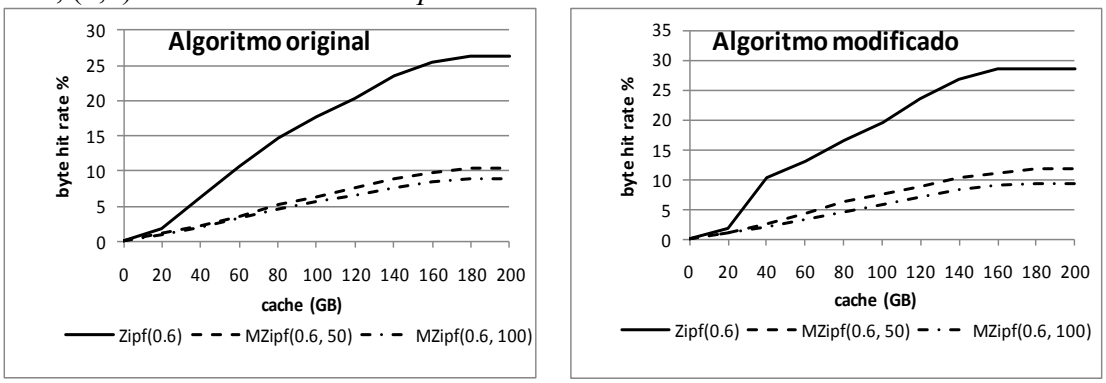
Mesmo no AS397, onde a diferença de desempenho é pouco melhor para o algoritmo modificado, a economia em termos de tráfego é bastante representativa. Ressalta-se que estamos tratando de uma simulação com um conjunto de dados que, embora representativo, é ainda bastante reduzido se comparado com dados reais. Em situações reais, onde o tráfego P2P é muito maior, a contribuição também será proporcionalmente mais alta, diminuindo ainda mais o custo de tráfego para o ISP.

Também é possível perceber que existe uma diferença no desempenho máximo alcançado em cada um dos ASs. No AS397 o desempenho máximo é de 25,83%, enquanto que no AS95 o desempenho máximo é de 11,83%. Dois fatores podem contribuir para a diferença no desempenho do algoritmo nos diferentes cenários, o percentual de tráfego passível de ser armazenado na *cache* (*cacheable*) isto é, os objetos que são requisitados mais de uma vez, e a distribuição de popularidade dos objetos, os quais investiga-se detalhadamente a seguir.

4.2.O Impacto da Distribuição de Popularidade

Embora nossos cenários sejam representativos de dois ASs reais, na prática as características dos ASs podem variar. Deste modo, nesta seção analisamos como a distribuição de popularidade afeta o desempenho do algoritmo de *cache* (Figura 4).

A distribuição MZipf possui dois parâmetros, sendo o primeiro a inclinação da curva e o segundo o fator de *plateau*. O fator de *plateau* indica a concentração das requisições para os objetos mais populares; quanto menor este fator significa que os objetos mais populares são requisitados mais freqüentemente. Quando o fator de *plateau* é zero, a distribuição iguala-se a uma Zipf. Para a avaliação do impacto da distribuição, o primeiro parâmetro foi fixado de acordo com o valor aproximado apresentado nos dois ASs, (0,6) e variou-se o fator de *plateau.*



(a) (b)

Figura 4. Impacto da distribuição de popularidade

Em primeiro lugar, observa-se que ambos os algoritmos apresentam melhor desempenho quando a distribuição de popularidade segue a Zipf(0.6). Este resultado é esperado, pois o algoritmo procura otimizar exatamente o armazenamento dos objetos mais populares. Assim, quando a distribuição é Zipf (fator de *plateau* = 0), existe um maior número de requisições para os objetos mais populares, e então o algoritmo é mais eficiente. Por outro lado, quanto maior o fator de *plateau*, maior será o 'achatamento' da curva da distribuição e, portanto, haverá menos requisições para os objetos mais populares diminuindo a eficiência do algoritmo. É o que acontece para MZipf(0.6,50) e MZipf(0.6,100), como mostram as figuras 4(a) e 4(b).

Adicionalmente se verifica que o algoritmo modificado atinge um melhor desempenho para todas as variações de distribuição de popularidade. Assim, é possível justificar o desempenho diferente para os dois cenários apresentados na seção 4.1. Esta variação decorre das diferentes distribuições de popularidade encontradas nos ASs. Enquanto que no AS397 a distribuição é MZipf (0.62, 8) e portanto há um maior número de requisições para os objetos mais populares, no AS95 a distribuição é MZipf (0.6, 50), indicando que as requisições acontecem em menor número para os objetos mais populares.

4.3. O Impacto do Tráfego Passível de Ser Armazenado na *Cache* (*cacheable*)

Na prática, nem todo objeto é requisitado mais de uma vez pelos usuários de um ISP. Se um objeto é requisitado apenas uma vez, não será vantajoso colocá-lo na *cache,* pois ocupará o espaço de outros objetos mais populares. A fim de avaliar como o volume dos objetos *cacheable* afeta o desempenho do algoritmo variou-se este parâmetro para o AS397 e AS95 (Figura 5).

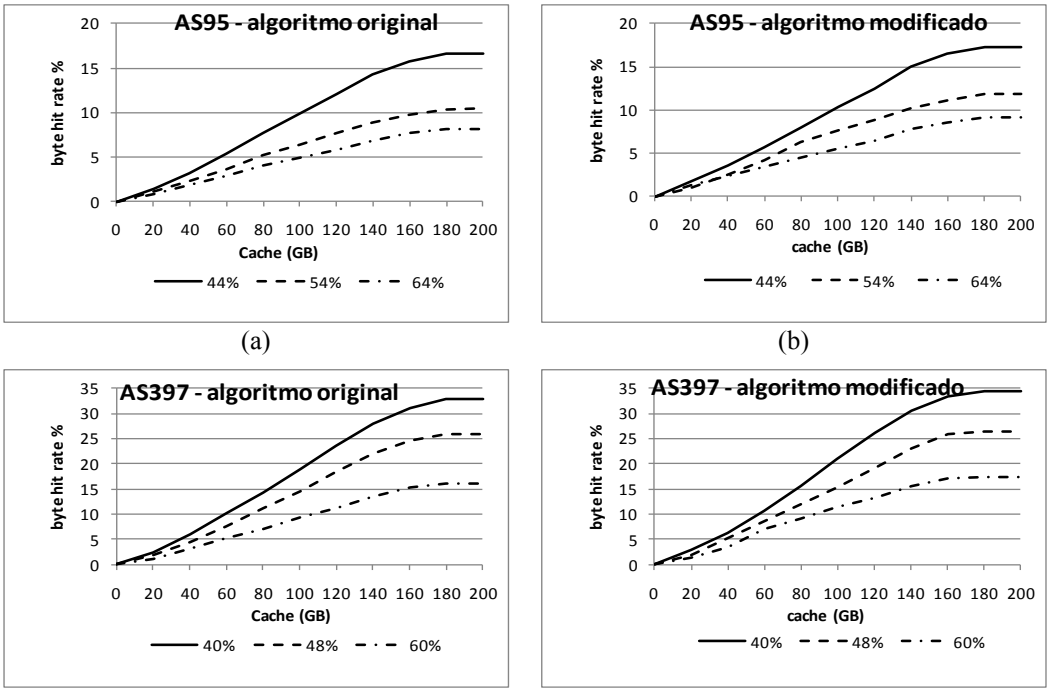


Figura 5. Impacto do tráfego *cacheable*

Primeiramente observa-se que o algoritmo modificado sempre obtém um melhor desempenho que o algoritmo original. Em todos os casos, menor tráfego *cacheable* implica em uma maior porcentagem de acerto. Isso acontece porque quanto menor a porcentagem de objetos requisitados mais de uma vez, o número de objetos mais populares tende a ser menor. Se poucos objetos são mais populares e o algoritmo guarda esses objetos, sua eficiência será maior.

4.4. O Impacto da Correlação Temporal

Como descrito na Seção 3.2, as requisições foram geradas de forma seqüencial para blocos de segmentos de um objeto. A fim de avaliar como a correlação temporal entre as requisições dos segmentos pode afetar o desempenho do algoritmo, variou-se a intensidade da correlação. A variação da correlação temporal foi obtida a através da geração de requisições pelo *ProWGen* com parâmetros de correlação fraca, média e alta para o AS397 (Figura 6).

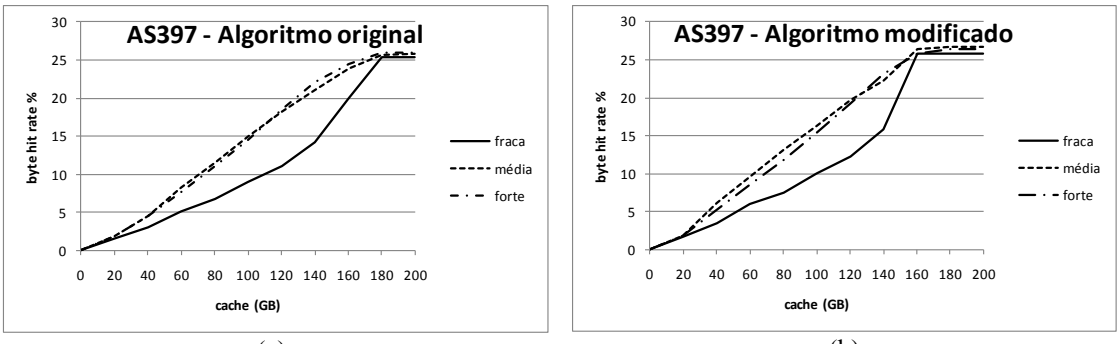


Figura 6. Impacto da correlação temporal

Observa-se que o desempenho do algoritmo praticamente não é afetado pela correlação temporal das requisições e a taxa de acertos da *cache* estabiliza com valores

máximos muito próximos. O algoritmo modificado obtém o desempenho máximo para uma *cache* menor, de 160 GB, enquanto que para o algoritmo original o desempenho máximo é obtido com uma *cache* de 180 GB.

4.5.O Impacto do Tamanho do Segmento

Nesta seção analisamos como o tamanho do segmento, unidade de inserção e remoção de dados na *cache,* pode afetar o seu desempenho. As requisições são feitas em blocos com 50 segmentos de tamanho variável de 1, 2, 5 e 10 MB. Utiliza-se o AS397 (Figura 7).

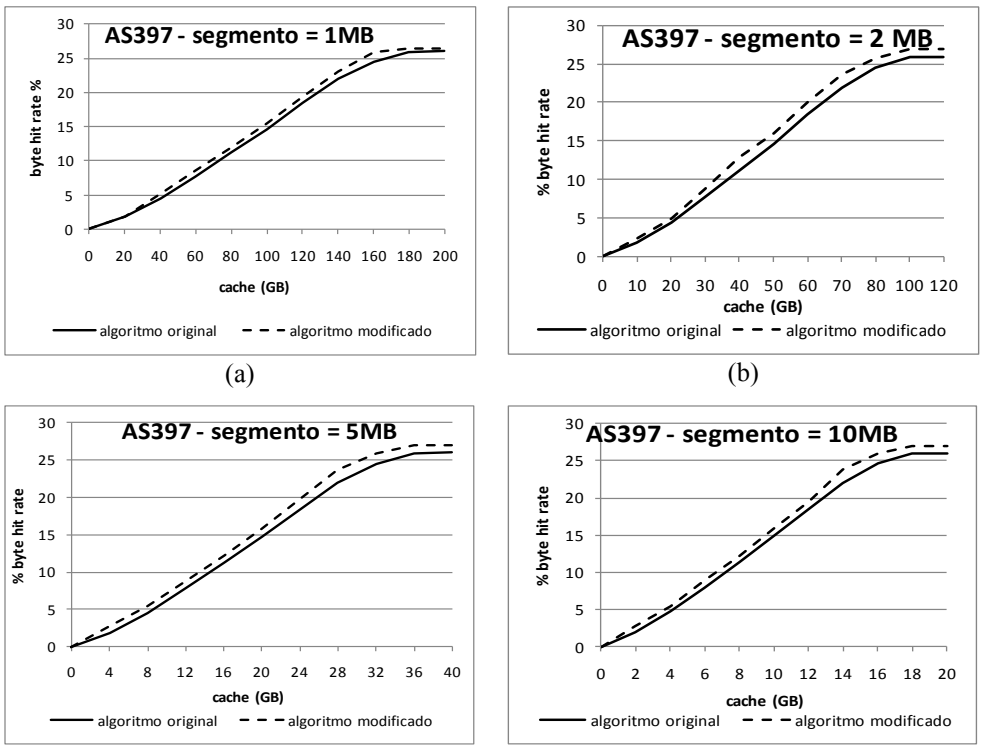


Figura 7. Impacto do tamanho do segmento

Primeiramente verifica-se que o desempenho máximo apresenta apenas uma pequena variação quando o tamanho do segmento é modificado. Ainda assim, o algoritmo modificado alcança uma melhor taxa de acertos em todos os casos. Uma diferença de 1,32% na taxa de acertos, como no caso do segmento de tamanho 5 MB, representa uma melhora no desempenho de 5%, o que pode corresponder a uma fatia considerável de tráfego. Observa-se ainda que o tamanho da *cache* necessário para atingir o desempenho máximo é inversamente proporcional ao tamanho do segmento. Para um tamanho de segmento dez vezes maior, verifica-se que o tamanho de *cache* necessário para atingir o desempenho máximo é reduzido para 10%, como visto nas Figuras 7(a) e 7(d). Quando o tamanho do segmento é muito pequeno, apenas pequenos pedaços do objeto são inseridos na *cache* a cada acerto, o que faz com que seja necessário um tamanho de *cache* maior para que o algoritmo mostre a sua eficiência.

4.6.*Cache*s cooperativas

Nessa seção são apresentados os resultados dos experimentos que envolvem a cooperação entre as *caches* dos sistemas envolvidos, conforme o modelo descrito na

Seção 3.4. Os ISPs que participam da cooperação são modelados como dois sistemas autônomos com as características do AS397 e do AS95 (Tabela 1). De acordo com a literatura, a cooperação faz sentido para ASs de tamanhos similares [Dán 2009] [Hefeeda and Noorizadeh 2008] (Figura 8).

Para avaliar a taxa de acertos nesse cenário, inicialmente foram gerados dois conjuntos de requisições distintos, mas para os mesmos objetos de vídeo. O *ranking* de popularidade dos objetos foi o mesmo nos dois ISPs, sendo mantidas as demais características dos ASs. As requisições foram processadas simultaneamente.

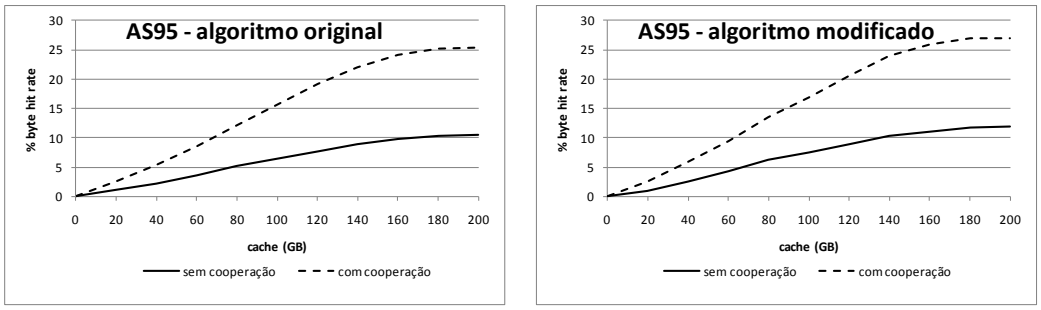


Figura 8. Comparações para os cenários com cooperação e sem cooperação

Percebe-se que, tanto para o algoritmo original como para o modificado, a taxa de acertos servidos pela cooperação no AS95 aumenta em mais de 100% (lembrando que parte do tráfego provém de requisições que chegam através do *link* de cooperação).

Entretanto, embora dois ISPs vizinhos possam tender a apresentar as mesmas características de popularidade, essa popularidade pode não ser exatamente a mesma. Para avaliar a influência deste aspecto sobre a cooperação, a correlação entre a popularidade dos objetos foi variada através de três cenários. No cenário 1, o *ranking* de popularidades é o mesmo para os dois ISPs. No cenário 2, o objeto mais popular do AS397 é o quarto mais popular do AS95 e, no cenário 3, o objeto mais popular do AS397 é o décimo primeiro mais popular do AS95 (Figura 9).

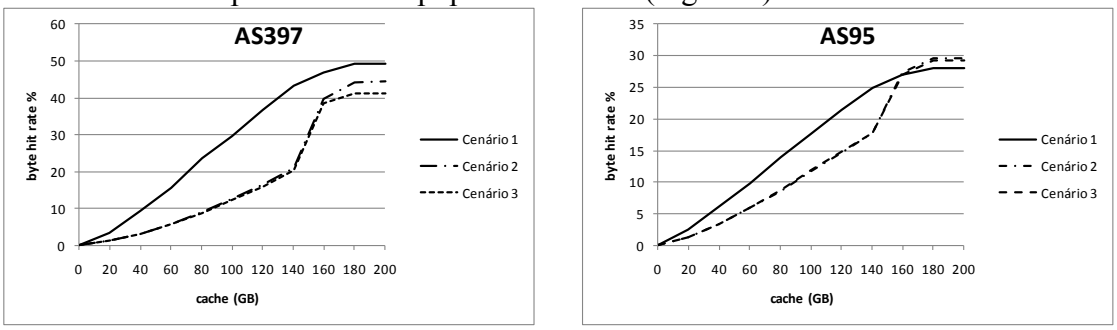


Figura 9. Impacto da cooperação para diferentes graus de correlação

Primeiramente, analisemos o desempenho da cooperação tomando como referência o AS397. O AS397 mantém na *cache* local os objetos mais populares e atende tanto as suas requisições como aquelas recebidas do AS95. Todas as requisições para objetos feitas a partir do AS95 e não encontrados na sua *cache* são encaminhadas para o AS397. À medida que os dois ASs vão emitindo as suas requisições, o AS397 começa a receber as requisições locais para os objetos mais populares e os armazena na *cache.* Após alguns instantes, o AS397 começa a receber as requisições do AS95, que foram redirecionadas, para estes mesmos objetos que provavelmente já estão na sua

cache, o que faz com que as requisições do AS95 sejam atendidas através da cooperação, aumentando sua taxa de acerto. Dessa forma, a cooperação faz com que o AS95 obtenha um desempenho ainda maior do que sem cooperação.

Observando o AS95, uma menor correlação significa que as requisições redirecionadas do AS397 nem sempre serão atendidas, uma vez que os objetos mais populares do AS397 não são tão populares no AS95. Entretanto, a cooperação faz com que a *cache* do AS95 sirva as requisições encaminhadas do AS vizinho, embora a contribuição fornecida seja menor que a recebida. Ao serem encaminhadas, as requisições não afetam a popularidade dos objetos na *cache* local, e provavelmente os objetos mais populares do sistema vizinho não estarão na *cache* do AS95. Isso acontece porque os objetos mais populares no AS397 não serão tão requisitados no AS95, causando uma menor cooperação do segundo em relação ao primeiro.

Quando o esquema de cooperação entre *caches* é utilizado, o principal efeito é o aumento da quantidade de tráfego servida pelo enlace de *peering*. A Tabela 2 mostra o ganho em termos de localidade de tráfego obtido pelos dois ISPs (AS397, AS95).

Tabela 2. Redução de tráfego através da cooperação (GB)

AS397	***Cache* local**	**Enlace de *peering***	**Enlace de trânsito**	**Total**
Sem cooperação	1180	0	8135	9315
Com cooperação	1180	1023	7112	9315
AS95	***Cache* local**	**Enlace de *peering***	**Enlace de trânsito**	**Total**
Sem cooperação	595	0	8721	9316
Com cooperação	595	762	7959	9316

Percebe-se que o esquema de cooperação reduz significativamente o volume de tráfego que usa o enlace de trânsito, permitindo que o tráfego seja servido pelo enlace de *peering*. Para o AS397, o tráfego servido através do enlace de *peering* é de 86,7% do tráfego local, enquanto que para o AS95 é de 128%. Para o AS95, o tráfego que transita pelo enlace de *peering* chega a ser maior que o tráfego servido pela *cache* local. Para o AS397 há uma redução de 1023 GB no uso do enlace de trânsito, representando 12,6% a menos de tráfego. A redução para o AS95 é de 762 GB (8,74%).

5. Conclusões e Trabalhos Futuros

Os ISPs consideram o tráfego P2P como indesejado, pois geralmente a falta de localidade na escolha dos nós para troca de dados provoca um aumento do tráfego de trânsito e, conseqüentemente, do custo de tráfego para os ISPs. Uma das soluções para este problema é o uso de *caches* com o objetivo de manter o tráfego P2P local ao ISP.

Este trabalho analisa o impacto do uso de *caches* para o armazenamento de tráfego decorrente de aplicações P2P de Vídeo sob-Demanda, onde geralmente os objetos são grandes e impactam fortemente no perfil de tráfego do ISP. Através da otimização de um algoritmo de *cache* parcial que armazena os segmentos dos objetos mais populares, verifica-se que o algoritmo de *cache* é capaz de obter um desempenho bem melhor que o algoritmo original em alguns cenários. Observa-se ainda que a distribuição de popularidade dos objetos, o percentual de tráfego *cacheable* e o tamanho dos segmentos impactam na eficiência do algoritmo, enquanto que se observa muito pequeno impacto da correlação temporal.

A maior contribuição deste trabalho é o estudo da cooperação entre as *caches* de

diferentes ISPs. Através da modelagem de requisições entre *caches* de ISPs vizinhos é possível verificar que a cooperação pode proporcionar uma redução de tráfego de trânsito bastante significativa, de até 12,6%. Ao mesmo tempo, o tráfego servido pelo enlace de *peering* chega a ser até 128% a mais do que o tráfego servido pela *cache* local.

Como trabalho futuro pretende-se usar a Teoria dos Jogos para modelar um maior número de caches cooperativas, tanto dentro do mesmo ISP como em ISPs vizinhos, a fim de analisar os ganhos e buscar um equilíbrio na cooperação.

Referências

Blond, S. L., Legout, A. and Dabbous, W. (2008) "Pushing BitTorrent Locality to the Limit", INRIA, Tech. Rep. 0034382.

Busari, M. and Wiliamson, C. (2002) "ProWGen: A Synthetic Workload Generation Tool for Simulation evaluation of Web Proxy *Cache*s", Journal of Comp. Networks.

Choffnes, D. R. and Bustamante, F. E. (2008) "Taming the Torrent - A practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems", In SIGCOMM'08.

Dán, G. (2009) "Cooperative Caching and Relaying Strategies for Peer-to-Peer Content Delivery" at *7th International Workshop on P2P Systems (IPTPS '08)*.

Feitosa, E., Souto, E., Sadok, D. (2008) "Tráfego Internet Não Desejado: Conceitos, Caracterização e Soluções". Minicurso VIII SBSeg, Gramado, RS, Brasil.

Gummadi, K. P., Dunn, R. J., Saroiu, S., D.Gribble, S., Levy, H. M. and Zahorjan, J. (2003) "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload" in Proc. SOSP'03, p. 314-329.

Hefeeda, M. and Noorizadeh, B. (2008) "Cooperative Caching: The Case for P2P Traffic", In *Local Computer Networks, 33rd IEEE Conference,* p. 12-19.

Hefeeda, M. and Saleh, O. (2008) "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems", In *IEEE/ACM Transactions on Networking.*

Huang, Y., Fu, T. Z., Chiu, D., Lui, J. C. and Huang, C. (2008) "Challenges, Design and Analysis of a Large-Scale P2P-VoD System", In *SIGCOMM Comp. Com. Rev.* 38.

Karagiannis, T., Rodriguez, P. and Papagiannaki, K. (2005) "Should Internet Service Providers Fear Peer-Assisted Content Distribution?", In *Internet Measurement Conference 2005*.

Leibowitz, N., Bergman, A., Ben-Shaul, R.and Shavit, A. (2002) "Are File Swapping Networks *Cacheable*? Characterizing P2P Traffic", Proc. of 7th Int. WWW Caching Workshop.

Shen, G., Wang, Y., Xiong, Y., Zhao, B. Y. and Zhang, Z. (2007) "HPTP: Relieving the Tension between ISPs and P2P", In *IPTPS*.

Yu, J.; Chou, C. T.; Yang, Z.; Du, X. and Wang, T., "A dynamic caching algorithm based on internal popularity distribution on streaming media" in *Multimedia Systems,* 2006.

PALMS: Um Protocolo ALM Simples para Distribuição de Conteúdo

Daniel Tetsuo Huzioka, Elias Procópio Duarte Jr.

Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brazil

{danielh,elias}@c3sl.ufpr.br

Abstract. *ALM Protocols are used to deploy multicast communication at the application layer, using end-host processing and upload capabilities. Several ALM protocols have been proposed, each one presenting varying features for varying applications. This paper proposes PALMS, a simple ALM protocol for content-distribution over the Internet. The protocol architecture consists of a server, responsible for content generation, a tracker, responsible for aiding nodes to join the system, and peers, which are clients that are also responsible for the core multicast transmissions. PALMS organizes peers in groups. Each group presents a tree topology and dictates which peer should send the content to which other peers. The server is the root of all groups, and sends one copy of the content for each group allowing its peers to disseminate the content within the group. PALMS was implemented using the PeerSIM simulator and we present results comparing PALMS with Narada, a popular application-layer multicast protocol. Results shows that our protocol achieves an acceptable server-to-peers delay and allows content distribution using fewer messages than Narada.*

Resumo. *Os protocolos ALM (*Application Layer Multicast*) são usados para implementar a difusão seletiva na camada de aplicação, utilizando a capacidade de processamento e comunicação dos nós da rede para o repasse de mensagens. Diversos protocolos ALM já foram propostos, cada um contendo características específicas diferentes para aplicações distintas. Este trabalho propõe o PALMS, um protocolo ALM Simples para a distribuição de conteúdo na Internet. A arquitetura do PALMS consiste de um* servidor *responsável pela geração de conteúdo, um* tracker *responsável pelo auxílio à entrada de nós na rede e pelos* peers*, os clientes da rede que também são responsáveis por grande parte das transmissões* multicast*. No PALMS, os* peers *são organizados em* grupos*, que agem como árvores de compartilhamento determinando a topologia de transmissão de conteúdo entre* peers*. Todos os grupos têm como raiz o servidor, que dissemina uma cópia do conteúdo para cada grupo. Um protótipo funcional do PALMS foi implementado no simulador PeerSIM e comparado com outro protocolo ALM bastante difundido, o Narada. Resultados mostram que nosso protocolo cumpre sua função de distribuição de conteúdo com um atraso aceitável entre* peers *e servidor utilizando uma quantidade menor de mensagens de controle que o Narada.*

1. Introdução

A transmissão por multicast permite que uma mensagem seja transmitida de uma só vez para um grupo de destinatários. Em redes de distribuição de conteúdo, a transmissão por multicast é a mais adequada quando há apenas uma fonte de conteúdo na rede. Diversas soluções multicast em nível de rede já foram propostas [Holbrook et al. 2006, Eriksson 1994]. Em especial, destaca-se o IP Multicast [Holbrook et al. 2006], que adiciona suporte multicast ao IPv4 (protocolo mais utilizado pelos roteadores da Internet). Devido a problemas de implantação, analisados detalhadamente em [Diot et al. 2000], o IP Multicast não é suportado por todos os roteadores da Internet, não podendo ser utilizado em sistemas cujo objetivo seja atender a usuários gerais da Internet.

Para contornar esses problemas de implantação, foram propostos diversos protocolos ALM (*Application Layer Multicast*, ou Difusão Seletiva em Nível de Aplicação, em português), que criam uma rede sobreposta para simular a distribuição multicast na camada de aplicação, sem a necessidade de modificação na camada de rede. Um protocolo ALM permite a comunicação multicast na Internet em sua estrutura atual. Em [Hosseini et al. 2007], os autores categorizam diversos protocolos ALM de acordo com algumas propriedades. Essas propriedades incluem, entre outras: *domínio da aplicação*, *organização dos peers* e *tipo de roteamento* de mensagens, descritas a seguir.

O *domínio da aplicação* é o objetivo final do sistema, descrevendo também o tipo de conteúdo distribuído por ele. Exemplos de *domínios de aplicação* de protocolos ALM são a distribuição de arquivos, distribuição de áudio e vídeo sob demanda, transmissão ao vivo em tempo real e videoconferência com múltiplos participantes. A *organização dos peers* inclui a topologia da rede, os métodos de inserção de novos nós, métodos de distribuição de conteúdo, entre outros. Exemplos de *organização dos peers* compreendem a forma como a topologia é criada, a existência ou não de níveis hierárquicos e a existência ou não de pontos de referência para a entrada dos *peers* na rede. O *roteamento de mensagens* define como as mensagens são transmitidas, dada a topologia da rede.

O protocolo ALM proposto neste trabalho pode ser categorizado de acordo com as propriedades mencionadas acima. Como domínio da aplicação, o protocolo inclui-se na categoria de transmissão de conteúdo proveniente de uma única fonte. Quanto ao tipo de conteúdo, este pode se apresentar como um conteúdo estático ou contínuo [Moraes et al. 2008]. Na distribuição de conteúdo estático, o conteúdo completo já se encontra disponível *a priori*, permitindo aos clientes receberem o conteúdo em qualquer ordem, podendo receber o final do conteúdo antes do início. Por outro lado, na distribuição de conteúdo contínuo este não se encontra totalmente disponível no início da transmissão, sendo gerado à medida que é transmitido [Li 2006]. Apesar da distribuição de conteúdo contínuo apresentar maiores restrições de banda e de tempo de recebimento, ele possibilita pouca interação do usuário sobre a execução do conteúdo (não permitindo, por exemplo, a execução de comandos de *retroceder* e *avançar*), ao contrário da distribuição de conteúdo estático. O protocolo proposto foi projetado para suportar as restrições da distribuição de conteúdo contínuo, embora seja possível a distribuição de conteúdo estático realizando pequenas modificações no sistema.

Em relação à organização da rede, o protocolo proposto utiliza-se de *grupos* de *peers*, que efetivamente formam árvores de compartilhamento de conteúdo entre os nós. Como mecanismo de roteamento, o protocolo apresenta a técnica de roteamento a partir

de uma árvore geradora com raiz no servidor, que determina quais *peers* devem receber e transmitir conteúdo para outros *peers*. Protocolos ALM são intrinsecamente P2P (*par-a-par*, do inglês *peer-to-peer*), pois neles os *peers* desempenham funções tanto de receptores como transmissores dos dados. Protocolos ALM voltados à distribuição de conteúdo proveniente de uma única fonte, como é o caso deste protocolo, podem ser ainda categorizados como protocolos híbridos, uma vez que possuem a rede P2P mas também possuem um servidor, entidade que não recebe nem consome dados.

Quanto à arquitetura, o PALMS é composto por um *servidor*, um *tracker* e pelos *peers*. O *servidor*, também chamado de *fonte*, é o responsável pela geração de conteúdo e por ditar a periodicidade de disponibilização do fluxo. O *tracker* é o responsável por auxiliar os *peers* a entrarem na rede, além de manter uma lista contendo informações dos *peers* ativos. Os *peers* desempenham dois papéis importantes na rede. O papel de cliente, que recebe e consome o fluxo recebido, e o papel de retransmissor, que irá propagar o conteúdo recebido a outros *peers*. No PALMS, os *peers* são organizados em *grupos*, que agem como árvores de compartilhamento determinando a topologia da rede de transmissão de conteúdo entre *peers*. Todos os grupos têm como raiz o servidor, que transmite uma cópia do conteúdo por grupo. Um protótipo funcional do PALMS foi implementado no simulador PeerSIM e comparado com outro protocolo ALM bastante difundido, o Narada. Resultados mostram que nosso protocolo cumpre sua função de distribuição de conteúdo utilizando multicast em nível de aplicação, utilizando uma quantidade significativamente menor de mensagens de controle que o Narada e mantendo um nível aceitável de atraso entre o *servidor* e os *peers*.

O restante do artigo está organizado da seguinte maneira. A Seção 2 apresenta trabalhos relacionados, descrevendo outros protocolos ALM. A Seção 3 descreve a proposta do trabalho, o Protocolo ALM Simples. A Seção 4 apresenta resultados experimentais de uma implementação do PALMS no simulador PeerSIM e comparações com o protocolo NARADA [Chu et al. 2000]. A Seção 5 conclui o artigo.

2. Trabalhos Relacionados

Os trabalhos relacionados incluem outros protocolos ALM e sistemas de distribuição de conteúdo, voltados às mais diversas funções. Dentre os diversos trabalhos existentes, são descritos o protocolo Narada [Chu et al. 2000] e o protocolo Nice [Banerjee et al. 2002], além de menções a outros trabalhos importantes na área. O protocolo Narada foi um dos primeiros a utilizar usuários finais para a comunicação *multicast*. O protocolo Nice utiliza-se de uma rede hierárquica em *clusters*. Nesta seção, esses protocolos são apresentados, bem como uma visão geral de outros trabalhos relacionados recentes.

Em [Chu et al. 2000], é apresentado o sistema Narada para distribuição de conteúdo. Este sistema organiza os *peers* em uma rede sobreposta na camada de aplicação. Tendo um *rendezvous point* como uma entidade para auxiliar a entrada de nós no sistema, o Narada mantém sua estrutura de rede de maneira descentralizada através de duas redes sobrepostas. Uma árvore geradora mínima (*minimum spanning tree*), específica para cada nó, é utilizada para o roteamento das mensagens, e uma rede em malha (do inglês *mesh*) é utilizada para o controle dos membros. Quando um nó é inserido na rede, ele se conecta a alguns nós arbitrários, que propagam a informação de inserção aos demais nós. O Narada utiliza mensagens de *refresh* (ou atualização) com *número*

sequencial para manter a composição da rede atualizada, testando probabilisticamente uma conexão com um nó cujo *refresh* não foi recebido. O Narada utiliza também um mecanismo semelhante ao BGP [Bellovin and Zinin 2006] para a atualização de tabelas de roteamento, e utiliza um mecanismo de retransmissão por caminho reverso (do inglês *reverse-path-forwarding*) para propagação dos dados. Como mecanismos de otimização da rede, o Narada utiliza duas funções, o *grau de utilidade* e o *custo de concenso* de enlace. O *grau de utilidade* mede o ganho obtido por um nó na inserção de uma ligação direta entre um outro nó na rede sobreposta (tornando os dois vizinhos), enquanto o *custo de concenso* mede a importância de uma ligação existente entre dois nós. Cada nó mede periodicamente o *custo de concenso* com seus vizinhos e o *grau de utilidade* com nós arbitrários não-vizinhos, adicionando ou removendo ligações quando necessário.

O NICE é um protocolo ALM voltado à distribuição de *streams* a partir de uma única fonte. Tendo como topologia uma árvore, o NICE abstrai cada nó da árvore como um conjunto de *peers* (chamados *clusters*), que formarão o grupo multicast. Partindo dos *clusters* que na estrutura da rede são folhas, cada *cluster* possui um *peer* especial, chamado *cluster leader*, que recebe informações de outros níveis da árvore. Hierarquicamente, a árvore é montada criando-se um grupo acima da folha, que possui todos os *cluster leaders* dos *clusters* da folha. Estes grupos de *cluster leaders* também serão *clusters*, tendo seus próprios *cluster leaders* que formarão um *cluster* hierarquicamente superior. Desta forma, têm-se uma árvore cuja raiz é a origem do fluxo, que propaga o fluxo aos *cluster leaders* do primeiro nível, que por sua vez possuem *cluster leaders* pertencentes ao segundo nível e assim por diante. A inserção na rede se dá pela inserção do *peer* em um *cluster* da folha. Iniciando o processo pelos clusters de hierarquia mais alta (aqueles que recebem o fluxo diretamente da fonte), o *peer* verifica qual *cluster leader* mais se assemelha a ele (métricas como distância e latência podem ser utilizadas), verificando então qual dos *clusters leader* dos *clusters* inferiores a aquele *cluster* mais se assemelha a ele, seguindo até sua inserção em um *cluster* da folha. Em casos emergenciais, por exemplo quando um *peer* entra na rede e seu processo de inserção é lento, a fonte pode enviar o fluxo diretamente ao novo *peer*, até que um *cluster* seja encontrado para ele. A manutenção dos *clusters* é dada através de mensagens de *heartbeat* dentro do cluster, podendo haver demoções de líderes ou junções/separações de *clusters*, quando necessário.

Diversos outros protocolos relacionados foram apresentados recentemente, sendo muitos voltados à melhoria da organização da rede utilizando conhecimentos de localidade entre os *peers*. Em [Zhang et al. 2008], um espaço n-dimensional é utilizado para representar a distância dos nós entre si, onde n é o número de pontos de referência utilizados. Utilizando medidas como o *ping* entre os nós para determinar a distância entre eles, o *taonet* procura construir uma topologia onde nós dimensionalmente próximos estejam próximos também na rede sobreposta. Em [Zhang et al. 2009], *peers* são agrupados em *clusters* hierárquicos e utilizam membros do *cluster* como pontos de referência para determinar a proximidade. Em [Dai et al. 2009], o número de saltos é utilizado para criar um modelo de rede, permitindo diversos métodos de escolha de *peers* vizinhos. Outro protocolo, que não utiliza conceitos explícitos de localidade, é o sistema de *streaming* multimídia híbrido cliente-servidor assistido por *peers* não confiáveis, apresentado em [Mello 2009], onde *peers* realizam acordos de compartilhamento (de envio e recebimento) temporários par ao recebimento do conteúdo.

3. PALMS: Um Protocolo ALM Simples

O PALMS (Protocolo ALM Simples) é um protocolo ALM voltado à distribuição de conteúdo contínuo proveniente de uma única fonte pela rede, e permite a transmissão de dados entre processos de usuários finais utilizando multicast em nível de aplicação. Nesta seção, são descritas a arquitetura do sistema, a organização dos *peers* em grupos e o modelo de organização do conteúdo.

3.1. Arquitetura do Sistema

O sistema proposto é composto por três entidades, *servidor*, *tracker* e *peer*. Ao *servidor*, cabe a tarefa de gerar e transmitir os dados que serão distribuídos no sistema. Ao *tracker*, cabe a tarefa de permitir a entrada organizada de novos *peers* na rede. Aos *peers*, cabe a tarefa de recepção, execução e propagação do conteúdo a outros *peers*. A Figura 1 mostra a estrutura geral da arquitetura do protocolo. A seguir, cada um dos componentes da arquitetura é descrito.

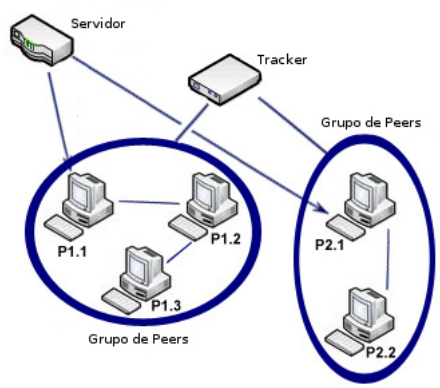


Figura 1. Arquitetura do protocolo PALMS.

3.1.1. O Servidor de Fluxo

O *servidor*, também chamado de fonte, é a entidade responsável pela produção do conteúdo e pela periodicidade em que eles são disponibilizados na rede. O conteúdo é gerado como um fluxo contínuo (em inglês, *stream*), e é descrito em detalhes na Seção 3.3.

É função do servidor determinar o intervalo de tempo em que cada novo elemento do fluxo será disponibilizado. Este intervalo é fixado no início da transmissão, mantendo-se constante até seu término.

Em relação à distribuição de conteúdo, o servidor desempenha a função de *seeder* das redes *torrent* [Cohen 2003], que diferencia-se de um *peer* comum tanto por não precisar receber conteúdo de nenhum outro *peer* como por não consumi-lo. O servidor pode enviar o conteúdo produzido a qualquer *peer* requisitante, respeitando seu limite de banda. As requisições de conteúdo são descritas na Seção 3.1.3.

Em relação à topologia da rede, o servidor não possui conhecimento de todos os nós presentes na rede, como também não participa da organização dos *peers*. Apenas *peers* que recebem conteúdo diretamente do servidor são conhecidos por ele.

3.1.2. O Tracker

O *tracker* é a entidade organizadora do sistema, e desempenha funções semelhantes às encontradas nos *trackers* das redes *torrent* [Cohen 2003] e aos *Rendezvous Point* de alguns protocolos ALM [Banerjee et al. 2002, Chu et al. 2000]. No PALMS, o *tracker* desempenha duas funções: (i) auxiliar na inserção de *peers* na rede e (ii) auxiliar na organização da rede.

É função do *tracker* disponibilizar o endereço do sistema na rede, possibilitando a entrada de novos *peers* a partir deste endereço. Ou seja, um novo *peer* conecta-se ao sistema através do *tracker*, e não através do servidor, como é o caso dos sistemas baseados no modelo cliente-servidor. Essa mudança retira a carga de inserção de novos *peers* do servidor, permitindo que ele execute apenas a função de distribuição de conteúdo.

O *tracker* mantém uma lista de *peers* ativos na rede, com informações em relação à disponibilidade de retransmissão de cada *peer*. Informações sobre disponibilidade de retransmissão podem ser enviadas pelo próprio *peer* ou por outros *peers* que detectem essa condição. Um *peer* é adicionado à lista quando ele é inserido com sucesso na rede (descrito em detalhes na Seção 3.1.3), sendo responsabilidade do *peer* reportar o sucesso de sua inserção. Um *peer* é removido da lista quando informa sua saída ao *tracker*.

Para auxiliar a inserção de novos *peers*, o *tracker* informa a cada novo *peer* duas informações iniciais: o endereço do servidor e um conjunto de *peers*. O endereço do servidor faz-se necessário pois na ocorrência de falhas na transmissão entre os *peers*, é feita uma requisição do conteúdo diretamente ao servidor, visando evitar interrupções significantes na execução do conteúdo. O conjunto de *peers* é um subconjunto da lista de *peers* mantida pelo *tracker*. De tamanho fixo, esse subconjunto é composto apenas por *peers* que se encontram disponíveis na rede, e seus integrantes são escolhidos aleatoriamente dentre todos os *peers* da lista. A escolha por um subconjunto aleatório baseia-se na premissa otimista que haverá pelo menos um *peer* disponível que consiga propagar o conteúdo ao novo *peer* e, como a cada requisição um subconjunto aleatório diferente é escolhido, a carga de propagação fica distribuída entre um grande número de *peers* na rede.

Para auxiliar na organização da rede, o *tracker* mantém um contador de *número de grupos* (*groupID*), que é incrementado sempre que um novo grupo é formado (os grupos são descritos na Seção 3.2). Quando um *peer* inicia sua inserção na rede, ele pode não conseguir se conectar a nenhum *peer* do conjunto de *peers* enviado pelo *tracker* (ou o conjunto pode estar vazio). Neste caso, o *peer* requisita ao *tracker* a criação de um novo grupo. Esse processo assegura a distinção entre os grupos, sem a necessidade de atualizações constantes pelo *tracker*.

3.1.3. Os Peers

Os *peers* são os processos dos usuários finais do sistema, e são responsáveis não somente por consumir o conteúdo, mas também auxiliar na propagação do mesmo pela rede. Há duas maneiras de um *peer* receber o conteúdo, diretamente do servidor ou de outro *peer*. Em ambas as situações, caso ele tenha disponibilidade de banda, esse *peer* poderá também

enviar o conteúdo recebido à outro *peer*, formando assim a rede P2P.

Quando um *peer* requisita sua entrada na rede, ele inicialmente recebe do *tracker* as informações sobre o servidor e uma lista de *peers*, descrita na Seção 3.1.2. O *peer* então dispara simultaneamente mensagens de requisição de inserção a todos os *peers* da lista. A aceitação ou não da requisição de inserção depende da disponibilidade de banda para retransmissão do conteúdo por parte dos *peers* receptores da mensagem. Apenas uma única resposta positiva é necessária para completar a inserção do *peer* na rede, sendo as demais descartadas. O disparo simultâneo paralelo se justifica pois, o *peer* que aceitar mais rapidamente à requisição será o *peer* mais indicado para a propagação do conteúdo ao novo *peer*, pois apresenta o menor RTT (*round-trip-time*) dentre os *peers* da lista recebida.

Quando o *peer* p_i dispara requisições a todos os *peers* da lista recebida pelo *tracker*, aqueles que se encontram disponíveis enviam seus respectivos identificadores de grupos a p_i. A primeira resposta recebida por p_i, proveniente do *peer* p_j pertencente ao grupo g_x, informa a p_i que há um *peer* p_j disponível no grupo g_x. P_i então requisita sua inserção ao *peer* p_j, que informa a p_i o sucesso de sua inserção no grupo g_x e passa a retransmitir o conteúdo ao *peer* p_i.

Quando um *peer* p_k pretende deixar a rede, ele informa sua saída a todos os *peers* que recebem seu conteúdo diretamente. Cada um destes *peers* requisita uma nova criação de grupo ao *tracker*. O peer p_k deve ainda informar ao *tracker* de sua saída, para sua remoção da lista de *peers* do tracker. Caso receba conteúdo diretamente do servidor, p_k deve informar também ao servidor, que encerra a transmissão de conteúdo a p_k. Um parâmetro do sistema informa aos *peers* o tempo máximo de atraso entre o recebimento das mensagens de conteúdo dos *peers*. Caso este tempo seja excedido, o *peer* receptor assume que seu *transmissor* tenha deixado a rede, tornando-se um *peer* órfão.

Caso um *peer* encontre-se em estado órfão e não esteja retransmitindo conteúdo a nenhum outro *peer* (ele era um nó folha na árvore de compartilhamento), ele requisita ao *tracker* uma nova lista de *peers* para realizar acordos. Caso contrário, o *peer* órfão cria um novo grupo.

A criação de novos grupos pelos *peers* agora órfãos faz com que eles recebam conteúdo diretamente do *servidor* (ver Seção 3.2 para detalhes). A atualização para um novo grupo ocorre do *peer* orfão até todos os seus filhos, que propagam a informações a todos os filhos até os nós folhas. Essa técnica permite a rápida recuperação do sistema a uma saída súbita de um *peer*, e leva em consideração a natureza não-confiável dos *peers*.

3.2. Organização dos Peers em Grupos

Os grupos são utilizados para separarem os *peers* em árvores de compartilhamento distintas. A topologia interna de um *grupo* é representada por uma árvore, onde a relação pai-filho corresponde a uma transmissão do nó pai ao nó filho. Cada grupo possui um *peer* que recebe os dados diretamente do servidor (o primeiro *peer* do grupo), que fica responsável por propagá-los aos seus *peers* filhos, que por sua vez propagam para seus próprios *peers* filhos.

Quando um *peer* p_x deseja criar um novo grupo, por exemplo o primeiro *peer* de todo o sistema, ele primeiro envia uma requisição ao *tracker*. O *tracker* então incrementa seu contador de *id de grupos*, anteriormente gid_x, para gid_{x+1} e envia esse novo valor a

p_x. Ao receber a resposta, p_x muda seu próprio id de grupo para gid_{x+1}. Qualquer outro *peer* que se conectar à p_x será pertencente ao grupo gid_{x+1}.

Quando um novo grupo é criado pelo *tracker*, o primeiro *peer* inserido no grupo deverá requisitar o conteúdo diretamente ao servidor. Caso o primeiro *peer* inserido no grupo tenha nós conectados a ele (a criação foi proveniente de um rompimento de um grupo anterior), o id do novo grupo é propagado a estes *peers*, que propagam aos *peers* conectados a eles e assim sucessivamente.

Quando um *peer* conecta-se a um grupo já existente, ele se conecta a apenas um único *peer* do grupo (ao qual ele enviou a requisição), passando a receber conteúdo diretamente desse *peer*. Este *peer* recém inserido pode apresentar recursos para outras retransmissões de conteúdo, informando ao *tracker* de sua disponibilidade e possibilitando futuras inserções de novos *peers*. A topologia da rede resultante desse processo forma uma árvore interna aos *grupos* para o compartilhamento de conteúdo. A topologia de compartilhamento em árvore é apresentada na Figura 2. As setas incidentes nos grupos indicam quais *peers* recebem fluxo diretamente do servidor, sendo estes os únicos *peers* conhecidos pelo servidor. Internamente aos grupos, o *peer* $p_{2.1}$ está conectado diretamente ao servidor, e não está conectado nem possui informações sobre nenhum outro *peer*. Já o *peer* $p_{1.1}$ possui acordos de envio com os peers $p_{1.5}$ e $p_{1.3}$. $P_{1.3}$ possui um acordo de recebimento de $p_{1.1}$ e um acordo de envio para $p_{1.2}$. $P_{1.5}$ possui um acordo de recebimento com $p_{1.1}$ e dois acordos de envio, um com $p_{1.6}$ e outro com $p_{1.4}$. Neste cenário, $p_{1.2}$ não têm conhecimento de $p_{1.1}$, nem de $p_{1.5}$ e seus filhos, sendo o inverso também verdadeiro.

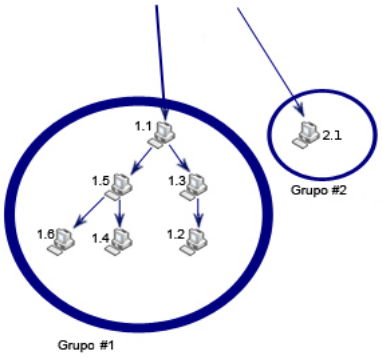


Figura 2. Topologia interna ao grupo.

3.3. Organização do Conteúdo

A organização do conteúdo compreende o seu mecanismo de geração e como ele é distribuído. O fluxo é inicialmente dividido em pedaços, chamados fatias, de tamanhos idênticos e é distribuído de acordo com a árvore de compartilhamento interna a cada *grupo*. Para cada par pai-filho na árvore, denomina-se *acordo de envio* o acordo que o pai tem para com o filho de retransmitir o fluxo, e *acordo de recebimento* o acordo que o filho tem para com o pai de receber o fluxo. A Figura 3 demonstra um exemplo desse conjunto de acordo entre os *peers*. Nesta figura, pode-se observar a propagação da fatia 1, transmitida pelo servidor aos *peers* $p_{1.1}$ e $p_{2.1}$. Os peers $p_{1.1}$, $p_{1.2}$ e $p_{1.3}$ pertencem ao

grupo g_1, enquanto os *peers* $p_{2.1}$ e $p_{2.2}$ pertencem ao grupo g_2. Quando $p_{1.1}$ recebe a fatia 1, ele a propaga para $p_{1.2}$, que por sua vez propaga para $p_{1.3}$. O mesmo ocorre no grupo g_2, onde $p_{2.1}$, ao receber a fatia 1, propaga-a para $p_{2.2}$.

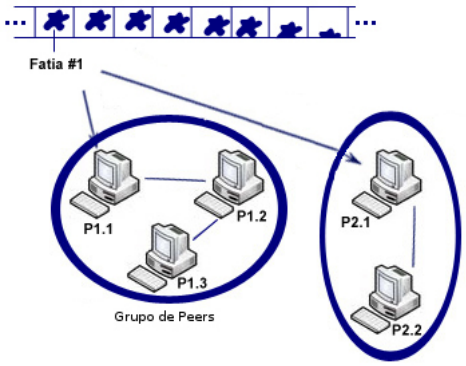


Figura 3. Distribuição de fluxo segundo acordos realizados.

4. Avaliação Experimental

Esta seção descreve a avaliação experimental realizada através de simulação do protocolo proposto. As simulações foram realizadas utilizando o simulador Java PeerSim [Jelasity et al.], no qual as entidades do sistema são representadas por objetos Java. Apesar de executadas localmente, o ambiente de simulação reflete a Internet. A topologia da rede simulada leva em consideração a distribuição de graus de nodos *power law* apresentados em [Faloutsos et al. 1999, Siganos et al. 2003, Bu and Towsley 2002], assim como conceitos de redes *small world* apresentadas em [Watts and Strogatz 1998]. Como falha entre os roteadores foram desconsideradas, a rede de roteadores gerada apenas auxilia na medição de latência entre os nós terminais.

Em todas as simulações, o comportamento dos nós foi simulado baseando-se em estudos sobre distribuição de fluxos presente em [Sripanidkulchai et al. 2004]. Sobre esse estudo, foi considerado que grande parte dos *peers* se conectam no início da transmissão, grande parte deles permanecem durante toda a transmissão e as inserções ocorrem segundo o efeito *flash mob*, onde num instante não há qualquer inserção e no instante seguinte podem ocorrer inserções múltiplas.

4.1. Experimento 1

O primeiro experimento objetiva demonstrar como a rede se comporta em um ambiente com *churn* (entrada e saída constante de *peers* na rede). O limite máximo de *peers* na rede foi de 256. A latência entre os roteadores foi estabelecida entre 5 e 15ms, sendo este valor adiciona em cerca de 10ms dos roteadores aos *peers*. Cada *peer* quando inserido foi designado a um roteador arbitrário, e se manteve conectado ao mesmo roteador até sua saída da rede. O número de *identificadores de peers* enviados pelo *tracker* foi limitado em 5 por requisição.

Este experimento teve a duração total de 600 segundos. A cada 1 segundo um novo elemento de conteúdo foi gerado e transmitido pela rede. A taxa de transmissão do conteúdo foi de 64Kbps, sendo a largura de banda dos *peers* suficiente para receber o

conteúdo em sua totalidade. Para cada *peer* foi designada uma banda de envio entre 512, 256, 128, 64 ou 0 (e neste caso o *peer* entra na rede como *leecher*, que não contribui para a propagação de conteúdo). Modificações na rede ocorreram em intervalos de 5 segundos, podendo acarretar na inserção de até 20 membros, na remoção de até 10 membros ou na manutenção no número de *peers*. A Figura 4 mostra a quantidade de *peers* ativos na rede durante a execução da simulação. A Figura 5 mostra a distribuição dos grupos no fim da execução.

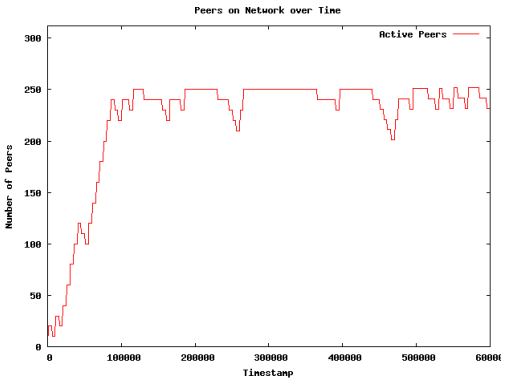


Figura 4. Peers por período de tempo.

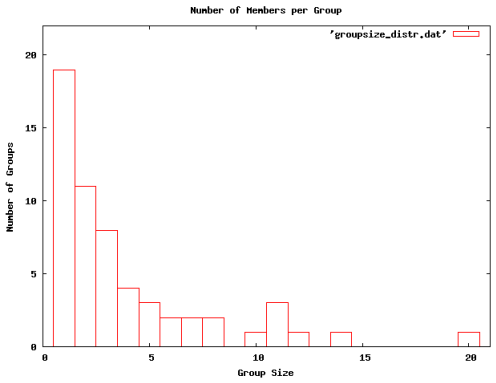


Figura 5. Distribuição de grupos por tamanho.

Pode-se observar pela Figura 5 que há uma grande quantidade de grupos pequenos. Isso se deve principalmente à saída de nós intermediários dos grupos durante a execução, o que acarreta a criação de um novo grupo pelos nós desconectados. Entretanto, espera-se que os novos grupos sejam preenchidos por outros *peers* a medida que estes são inseridos na rede, devido à aleatoriedade de escolha do conjunto de *peers* pelo *tracker*. Por outro lado, grupos demasiadamente grandes são também repartidos em grupos menores quando um *peer* pertencente ao grupo deixa a rede, resultando na formação de grupos menores.

4.2. Experimento 2

O segundo experimento têm por objetivo demonstrar como a simplicidade do PALMS resulta em comparação a técnicas de refinamento e controle do protocolo Narada.

Para este experimento, foi utilizada uma rede gerada pelo mesmo método do experimento 1, contendo no entanto 384 peers. Vale ressaltar que o protocolo Narada resulta na soma agregada de $O(N^2)$ (send N o número de nós na rede) mensagens de controle [Banerjee et al. 2002], dificultando simulações com grandes números de *peers*.

A taxa de geração e a largura de banda dos *peers* foram as mesmas adotadas no experimento 1, assim como a proporção de nós *leecher*. A taxa de ocorrência de inserção/remoção foi de 5 segundos, mas neste experimento foram realizadas inserções aleatórias de até 50 membros simultâneos e saída de até 5 membros simultâneos. A proporção entre entrada e saída se manteve igual ao do experimento 1.

Para o Narada, foram considerados os parâmetros de taxa de atualização (*refresh*) em 10 segundos, tempo mínimo de *timeout* em 20 segundos, tempo máximo de *timeout* em 30 segundos, taxa de checagem de utilidade em 20 segundos, taxa de atualização de rotas em 10 segundos, custo de consenso mínimo 5, utilidade mínima 40 e checagem de

utilidade em 5 outros *peers* por vez. Além disso, o *peer* de identificador 1 foi considerado a fonte de dados.

Para o PALMS, foram considerados os parâmetros de quantidade de nós enviada pelo *tracker* em 5, e largura máxima de banda em 512 Kbps(como no experimento 1).

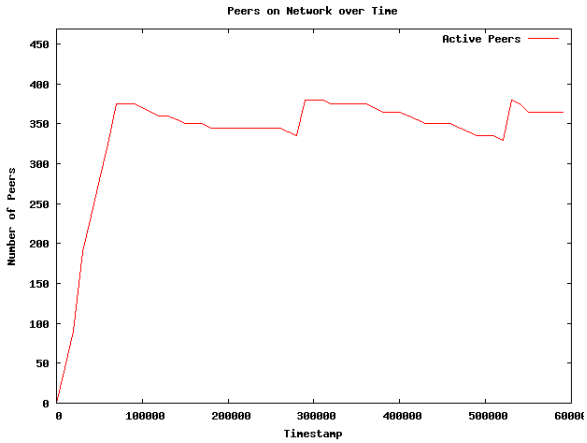


Figura 6. Peers por período de tempo.

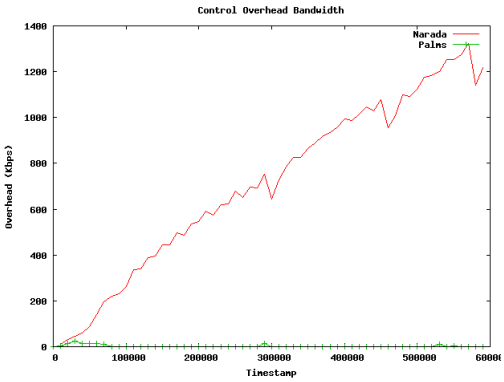


Figura 7. Banda utilizada para mensagens de controle.

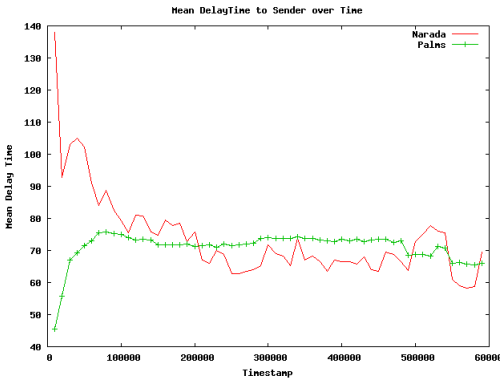


Figura 8. Atraso médio dos peers para a fonte de dados.

A Figura 6 mostra a quantidade de *peers* ativos durante a simulação. Como no experimento anterior, é necessário algum tempo até que a rede se torne estável pois ela é iniciada com nenhum nó presente. Após esse período, as inserções e remoções arbitrárias mantêm a rede próxima de seu limite.

A Figura 7 mostra a quantidade de banda utilizada para a troca de mensagens de controle na rede. No caso do PALMS, as mensagens de controle possuem papel importante no processo de entrada e saída dos nós, mantendo-se quase nulas enquanto os nós estiverem transmitindo conteúdo de maneira estável. Já no Narada, atualizações constantes e verificações de utilidade e custo de enlaces fazem com que a quantidade de mensagens de controle cresça até um ponto onde ela estabiliza-se. Caso a rede apresente *churn* constante (como é o caso desta simulação), os nós irão constantemente verificar por novas conexões e alertar sobre possíveis interrupções nos caminhos escolhidos.

A Figura 8 mostra o atraso médio de envio e recebimento das mensagens pelos *peers* até ou para o nó fonte (primeiro nó no Narada e servidor no Palms). É impor-

tante ressaltar que no protocolo Narada o atraso médio nesta simulação apresenta-se alto pois a rede ainda está em fase de descoberta. É importante observar também que após a considerável estabilização no número de *peers* na rede, o Narada inicia seus mecanismos de otimização da rede, melhorando constantemente a topologia da rede. Entretanto, é possível observar que o PALMS mantém o atraso médio constante e relativamente próximo aos apresentados pelo Narada, mesmo com a presença de *churn* na rede.

5. Conclusão

Este artigo apresentou uma solução simples para multicast em nível de aplicação. O protocolo PALMS permite a distribuição de conteúdo contínuo pela Internet a partir de uma única fonte. Utilizando técnicas P2P para realizar a difusão seletiva em nível de aplicação, o protocolo consegue diminuir a carga imposta ao servidor na distribuição de conteúdo dividindo os *peers* em *grupos*, construindo árvores de compartilhamento entre os *peers* pertencentes ao *grupo*. Experimentos demonstraram que o sistema impõe uma baixa carga de controle sobre os nós, e tira vantagem do *churn* (característico das redes de compartilhamento) para controlar o tamanho de seus *grupos*. A arbitrariedade dos *peers* enviados pelo *tracker* aumenta a efetividade de balanceamento dos grupos. Além disso, a pesquisa simultânea entre os *peers* enviados pelo *tracker* melhora a qualidade do caminho escolhido localmente, uma vez que o melhor *peer* (entre os enviados) é escolhido para retransmitir o tráfego.

Trabalhos futuros incluem a modificação na escolha dos *peers*, permitindo um maior equilíbrio entre o tamanho dos *grupos* e o refinamento dos *grupos* existentes pelos *peers* participantes. O simulador utilizado é altamente configurável, sendo possível realizar simulações com comportamento dos *peers* mais próximos à realidade, levando em consideração a localização dos *peers* e a interação entre os mesmos na rede se necessário.

Referências

Banerjee, S., Bhattacharjee, B., and Kommareddy, C. (2002). Scalable Application Layer Multicast. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 205–217, New York, NY, USA. ACM.

Bellovin, S. and Zinin, A. (2006). Standards Maturity Variance Regarding the TCP MD5 Signature Option (RFC 2385) and the BGP-4 Specification. RFC 4278 (Informational).

Bu, T. and Towsley, D. F. (2002). On distinguishing between internet power law topology generators. In *INFOCOM*.

Chu, Y.-h., Rao, S. G., and Zhang, H. (2000). A Case for End System Multicast (keynote address). In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA. ACM.

Cohen, B. (2003). Incentives Build Robustness in BitTorrent.

Dai, L., Cao, Y., Cui, Y., and Xue, Y. (2009). On Scalability of Proximity-aware Peer-to-Peer Streaming. *Comput. Commun.*, 32(1):144–153.

Diot, C., Levine, B. N., Lyles, B., Kassem, H., and Balensiefen, D. (2000). Deployment Issues for the IP Multicast Service and Architecture. *Network, IEEE*, 14(1):78–88.

Eriksson, H. (1994). MBONE: The Multicast Backbone. *Commun. ACM*, 37(8):54–60.

Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, New York, NY, USA. ACM.

Holbrook, H., Cain, B., and Haberman, B. (2006). Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast. RFC 4604 (Proposed Standard).

Hosseini, M., Ahmed, D. T., Shirmohammadi, S., and Georganas, N. D. (2007). A Survey of Application-Layer Multicast Protocols. *Communications Surveys & Tutorials, IEEE*, 9(3):58–74.

Jelasity, M., Montresor, A., Jesi, G. P., and Voulgaris, S. The Peersim simulator. `http://peersim.sf.net`.

Li, J. (2006). Peer-to-Peer Multimedia Applications. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 3–6, New York, NY, USA. ACM.

Mello, S. L. V. (2009). Hybrid Client-server Multimedia Streaming Assisted by Unreliable Peers. *The 15th International Conference on Distributed Multimedia Systems*, pages 1–6.

Moraes, I. M., Campista, M. E. M., Duffles, M., Rubinstein, M., Costa, L. H., and Duarte, O. C. M. B. (2008). Distribuição de Vídeo sobre Redes Par-a-Par: Arquiteturas, Mecanismos e Desafios.

Siganos, G., Faloutsos, M., Faloutsos, P., and Faloutsos, C. (2003). Power laws and the as-level internet topology. *IEEE/ACM Trans. Netw.*, 11(4):514–524.

Sripanidkulchai, K., Maggs, B., and Zhang, H. (2004). An analysis of live streaming workloads on the internet. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 41–54, New York, NY, USA. ACM.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10.

Zhang, X., Li, X., Luo, W., and Yan, B. (2009). An Application Layer Multicast Approach Based on Topology-Aware Clustering. *Comput. Commun.*, 32(6):1095–1103.

Zhang, X., Li, Z., and Wang, Y. (2008). A Distributed Topology-Aware Overlays Construction Algorithm. In *MG '08: Proceedings of the 15th ACM Mardi Gras conference*, pages 1–6, New York, NY, USA. ACM.

VI Workshop de Redes Dinamicas e Sistemas Peer-to-Peer

♦

Sessão Técnica 2

Simulação e Avaliação de Desempenho

SciMulator: Um Ambiente de Simulação de Workflows Científicos em Redes P2P*

Jonas Dias[1], Carla Rodrigues[1], Eduardo Ogasawara[1], Daniel de Oliveira[1], Vanessa Braganholo[2], Esther Pacitti[3], Marta Mattoso[1]

[1]Programa de Engenharia de Sistemas e Computação – COPPE / UFRJ
Caixa Postal 68.511, Rio de Janeiro, RJ, 21941-972

[2]Departamento de Ciência da Computação – UFRJ
Caixa Postal 68.530, Rio de Janeiro, RJ, 21941-590

[3]INRIA & LIRMM, Montpellier, France

{jonasdias, carlarod, ogasawara, danielc, marta}@cos.ufrj.br

braganholo@dcc.ufrj.br

pacitti@lirmm.fr

Abstract. The growth of large-scale scientific experiments motivates the search for computing environments that support the parallelization of computing activities, particularly those that complies to the Many Task Computing (MTC) paradigm. Peer-to-Peer (P2P) environments can meet this demand due to the easy access and distributed control. However, building a real P2P infrastructure to evaluate this solution is very costly. Within this context, we present the simulator SciMulator developed to evaluate P2P architectures. We present the modeling of the simulator and an initial assessment of its performance when using the SciMule architecture for submission of scientific workflows activities on P2P networks.

Resumo. O crescimento dos experimentos científicos em larga escala motiva a busca por ambientes computacionais que apoiem a paralelização de atividades computacionais, particularmente, as que atendem o paradigma *Many Task Computing* (MTC). Ambientes *Peer-to-Peer* (P2P) podem atender esta demanda, pelo fácil acesso e controle distribuído. Porém, construir uma infraestrutura P2P real para avaliar tal solução é muito custoso. Neste contexto, apresentamos o simulador SciMulator, desenvolvido para avaliar arquiteturas P2P. Neste trabalho, apresentamos a modelagem do simulador e uma avaliação inicial do seu desempenho ao utilizar a arquitetura SciMule para submissão de atividades de workflows científicos em redes P2P.

1. Introdução

Recentemente, o avanço do processamento de alto desempenho motivou a realização de experimentos científicos (Deelman et al. 2009). Tais experimentos são caracterizados pela grande movimentação de dados, dados heterogêneos e execução de um grande número de atividades com potencial de paralelização. Nestes experimentos, o encadeamento de atividades é popularmente modelado como um workflow científico

* Esse trabalho foi parcialmente financiado pelo CNPq e pelo INRIA dentro do projeto SARAVA, equipe associada.

(Brown et al. 2007). As atividades podem requerer alta complexidade computacional, o que faz com que a paralelização torne-se necessária para realização do experimento em tempo hábil. Em muitos casos, esta paralelização pode ocorrer por meio de varredura de parâmetros e fragmentação de dados. Estas características se encaixam no novo paradigma de computação conhecido como *Many Task Computing* (MTC) (Raicu et al. 2008).

Existem soluções para paralelizar atividades do workflow seguindo o paradigma MTC e executá-las em ambientes homogêneos como *clusters* (Abramson et al. 2008, Ogasawara et al. 2009). Entretanto, existem muitos outros ambientes heterogêneos que podem ser explorados para execução de workflows científicos como, por exemplo, *grids*, *desktop grids* (Anderson 2004) ou nuvens híbridas (Grid4All Consortium 2009). O principal problema é que cada um destes ambientes requer diferentes esforços, recursos e habilidades do cientista para definir e avaliar o paralelismo nas atividades do workflow.

A abordagem P2P também se mostra promissora para o processamento de atividades MTC pela alta escalabilidade, capacidade de lidar com distribuição de dados, além da tolerância a falhas. Segundo Pacitti et al. (2007), técnicas P2P também se mostram úteis em *grids* computacionais de larga escala. Recentemente, iniciativas como SciMule (Ogasawara et al. 2010) propõem a paralelização de atividades de workflows científicos em redes *peer-to-peer* (P2P). No entanto, faltam estudos que avaliem soluções para o processamento de atividades MTC de workflows científicos em redes P2P.

Uma rede P2P envolve milhares de computadores com arquiteturas distintas interligadas por uma rede também heterogênea. Construir uma infraestrutura real para realizar estudos iniciais de viabilidade é muito custoso e inviável para a grande maioria dos pesquisadores (Almeida et al. 2008). Portanto, antes de se realizar estudos em uma rede real, é necessário avaliar se há indícios de que a distribuição de atividades de workflows científicos em redes P2P é realmente vantajosa e em quais cenários há essa vantagem, uma vez que devem ser consideradas diversas características como latência entre pontos da rede e a dificuldade de transmissão de dados através de pontos com baixa largura de banda. A avaliação destes indícios pode ser feita através de estudos de simulação.

O objetivo deste trabalho é apresentar o modelo construído para o ambiente de simulação SciMulator, ressaltando os desafios encontrados, tais como: a modelagem de atividades MTC de um workflow; sua decomposição e escalonamento em uma rede com controle distribuído; seu processamento e retorno de resultados; captura de dados de proveniência (Davidson and Freire 2008); além do suporte à tolerância a falhas. Tais características não estão presentes em simuladores P2P tradicionais. O SciMulator foi construído a partir do PeerSim (Jelasity et al. 2010), um simulador de ambientes P2P muito utilizado em outros trabalhos como base na construção de outros cenários de pesquisa (Boudani et al. 2008, Dick et al. 2009). No SciMulator, cada nó da rede é capaz de submeter atividades de workflows científicos para serem executadas distribuidamente pelos outros nós da rede. As atividades são decompostas em tarefas seguindo estereótipos comuns no paradigma MTC.

Numa primeira avaliação, o SciMulator foi utilizado para avaliar a arquitetura do SciMule (Ogasawara et al. 2010). Ele mostrou-se ágil na simulação de cenários custosos e robusto no consumo de memória. Os resultados obtidos com o simulador foram positivos, uma vez que foi possível identificar componentes na arquitetura que precisam

ser aperfeiçoados, como, por exemplo, o escalonador de tarefas, o sistema de busca dos dados do experimento e o mecanismo de distribuição destes dados para execução de atividades, bem como possibilitar o desenvolvimento e avaliação de outras arquiteturas para apoiar MTC em ambientes P2P.

Este artigo está organizado da seguinte forma: a seção 2 descreve o SciMulator, um ambiente de simulação de workflows científicos em redes P2P, desenvolvido sobre o PeerSim; a seção 3 descreve sucintamente a arquitetura SciMule utilizada como primeiro estudo de caso do SciMulator; a seção 3 apresenta a análise de resultados; a seção 5 apresenta os trabalhos relacionados e a seção 6 conclui este artigo e apresenta trabalhos futuros.

2. SciMulator

O SciMulator é um simulador de um ambiente P2P onde cada nó da rede (*peer*) é capaz de submeter e executar tarefas distribuídas de atividades de workflows científicos seguindo o modelo MTC. O objetivo do simulador é permitir avaliações iniciais de arquiteturas voltadas ao processamento de atividades em redes heterogêneas com controle distribuído como as redes P2P. Por estas características, além do elevado grau de dinamismo, a solução P2P para processamento de tarefas deve se comportar de forma diferente de outras soluções para *clusters* e *grids*.

O SciMulator foi desenvolvido como uma extensão do PeerSim (Jelasity et al. 2010). O PeerSim é um simulador de redes P2P desenvolvido na linguagem Java. Ele possui quatro componentes principais extensíveis: (i) o nó, que representa uma máquina (*peer*) na rede P2P e é modelado pela classe *GeneralNode*; (ii) uma abstração do *overlay* que mantém informação sobre como os *peers* estão conectados; (iii) protocolos que são executados pelos nós e (iv) elementos de controle que executam ações globais na rede, tais como eventos de *churn* e registro de *log*. Estes componentes foram estendidos e outros foram criados para apoiar o modelo de submissão e execução de tarefas. Um desafio na construção do simulador foi, justamente, acrescentar componentes que permitissem a distribuição de atividades em tarefas MTC pela rede P2P, além de sua execução nos nós através de protocolos PeerSim. O PeerSim não oferece um modelo de controle de largura de banda dos *peers*, nem um sistema de *download* e *upload* entre eles. Portanto, o processo de transferência de dados também precisou ser modelado, pois influencia no tempo total de execução de uma atividade. Na Internet, a largura de banda entre nós da rede não pode ser prevista. O SciMulator busca simular esta característica estabelecendo valores estocásticos para a largura de banda disponível para cada nó da rede.

O SciMulator fornece um arcabouço base para avaliação de cenários e arquiteturas para submissão e paralelização de atividades MTC em redes P2P. Avaliar uma arquitetura no SciMulator significa avaliar a execução de um conjunto de protocolos pelos nós da rede dispostos em uma determinada topologia. Tal avaliação envolve a possível extensão de componentes referentes aos nós, à topologia e a alguns protocolos. Além dos componentes gerais do PeerSim, todo arcabouço de fragmentação e distribuição de atividades de workflow, escalonamento de tarefas, transmissão de dados, processamento, monitoramento, proveniência e recuperação de falhas pode ser reutilizado para modelar outras arquiteturas P2P para execução de workflows.

2.1 Submissão de Atividades

No SciMulator, os *peers* podem submeter atividades de workflows científicos na rede. Para isso, foi modelado um escalonador de tarefas que é responsável por todo o processo de decomposição e distribuição das atividades. Este componente entra em ação em um *peer* no momento em que este possui uma atividade para submeter.

Atividades são decompostas em tarefas de acordo com o seu tipo de paralelismo atrelado a MTC. Uma primeira implementação do SciMulator permite avaliação de atividades do tipo fragmentação de dados (*DataFragmentation*), nas quais as tarefas processam um fragmento do conjunto de dados da atividade, e varredura de parâmetros (*ParameterSweep*), nas quais as tarefas têm diferentes conjuntos de parâmetros para processar. O escalonador seleciona os *peers* para processar as tarefas considerando o balanceamento de carga. Características como o tamanho da fila de tarefas que cada *peer* possui, dados a serem transmitidos, uso do processador e a banda de rede utilizada, são levadas em consideração para que não haja sobrecarga em pontos da rede. O *peer* que submeteu a atividade também pode ser eleito para executar algumas tarefas de sua atividade. A Figura 1 apresenta um trecho do modelo UML do SciMulator exibindo os componentes do processo de submissão de uma atividade.

Os *peers* recebem tarefas de vizinhos, as quais precisam de um ou mais conjuntos de dados (*DataPackage*) para serem processadas. Se o *peer* não possuir os dados, ele efetua o *download* pelo nó que submeteu a atividade na rede. A transmissão de dados de uma tarefa é realizada através da troca de pacotes de dados simbólicos entre *peers* de acordo com a largura de banda do *link*. A taxa de *download* depende da velocidade do *link*, identificada pela menor largura de banda entre o cliente e o nó de execução. Um *peer* pode efetuar o *download* de várias tarefas simultaneamente. Assim, sua largura de banda é compartilhada entre todas as transmissões, segundo uma estratégia circular (*round-robin)*. Os *peers* armazenam os pacotes de dados localmente, pois uma futura tarefa pode precisar deste mesmo conjunto de dados.

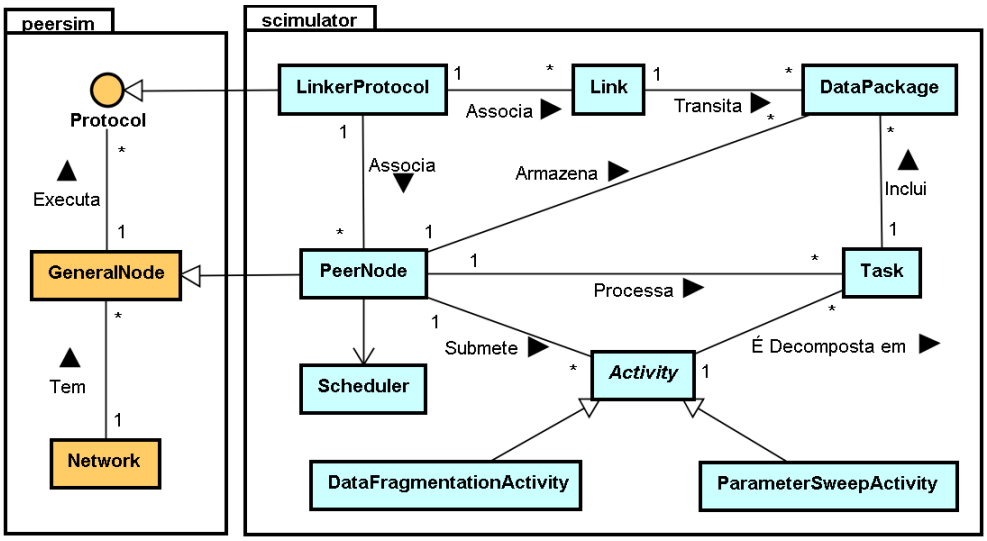


Figura 1: Modelo UML do SciMulator dos componentes de submissão

2.2 Execução de Atividades

Quando um *peer* detecta que foi escolhido para a execução de uma tarefa, ele verifica se precisa efetuar o *download* de algum conjunto de dados. Após obter todos os dados que compõem a tarefa, o *peer* começa a processá-la. Ele pode receber tarefas de diferentes

vizinhos, porém o protocolo de processamento é FIFO, isto é, as primeiras tarefas obtidas são as primeiras a serem processadas. Quando termina de processar a tarefa, o *peer* envia os resultados e informação de proveniência ao *peer* que submeteu a atividade.

A simulação do processamento de tarefas envolve o conceito de unidade de processamento. Cada *peer* posui *x* unidades de processamento por ciclo de simulação e cada tarefa possui um custo *y* de processamento. Sendo assim, uma tarefa deve demorar *y/x* ciclos para ser processada por um *peer* ocioso.

2.3 Configuração da Rede

Como o SciMulator é modelado para redes heterogêneas, durante o processo de configuração da rede, características como poder de processamento e largura de banda dos *peers* devem variar. Além disso, o *peer* não é uma máquina dedicada e tem seus recursos utilizados para outros fins. Ou seja, a capacidade total de processamento e largura de banda de um *peer* não é exclusiva da rede P2P. Portanto, ao longo da simulação, a medida de quanto uma máquina está ociosa em um determinado momento também varia. A porcentagem de quanto estes recursos estão disponíveis é estimada a cada ciclo. Os eventos que ocorrem durante a simulação, como a submissão de novas atividades e a entrada e saída de nós da rede (c*hurn*) também devem seguir uma regularidade variável. Para modelar este cenário, estudou-se um conjunto de distribuições estatísticas capazes de representar a variação destas características e eventos. Tais distribuições foram escolhidas com base no comportamento observado em experimentos reais realizados em clusters (Ogasawara et al. 2009) e parametrizadas para o modelo do simulador. A Tabela 1 mostra as distribuições escolhidas na modelagem do SciMulator. A capacidade de processamento de um *peer* varia seguindo uma distribuição Gamma, com fator de escala 30 e forma 2 (Freedman et al. 2007), com uma média de 80 unidades de processamento por ciclo. A largura de banda segue uma distribuição Gamma com média 1,5 sob uma função logarítmica de base dois definida de 7 (128 kbps) a 16 (64Mbps). A ociosidade da máquina segue uma distribuição normal com uma média de 0,5 e desvio padrão de 0,1875, indicando que cerca de cinquenta por cento dos recursos de cada *peer* estão disponíveis para a rede P2P em um determinado ciclo. As médias das distribuições de Poisson para Submissão de Atividades (*x*) e Ocorrência de Churns (*y*) são fatores da simulação e definem a frequência de submissão de atividades e de saída e entrada de *peers* na rede.

Tabela 1: Distribuições adotadas para modelar características heterogêneas dos nós e eventos da simulação.

Característica	Distribuição	Média	Desvio Padrão	Escala	Forma
Capacidade do Processador	Gamma	80,0	-	30	2
Largura de Banda	Gamma	1,5	-	1	2
Ociosidade da Máquina	Normal	0,5	0,1875	-	-
Submissão de Atividades	Poisson	*x*	-	-	-
Ocorrência de *Churns*	Poisson	*y*	-	-	-

Além da configuração interna dos nós e dos controles, a topologia da rede também é um fator decisivo no bom desempenho de uma rede P2P. Entretanto, esse fator é muito atrelado à arquitetura que está sendo avaliada pelo simulador. O PeerSim já oferece um arcabouço que facilita a implementação de novas topologias. Nos primeiros estudos com o SciMulator, foi avaliada a arquitetura SciMule (Ogasawara et al. 2010). As medições realizadas com o simulador são apresentadas na seção 4 e foram obtidas utilizando a arquitetura SciMule, que é descrita na seção 3.

2.4 O processo de simulação

A simulação foi modelada usando uma abordagem síncrona (Barbosa 1996), baseada em ciclos, do PeerSim. Esta abordagem foi escolhida por garantir maior escalabilidade, embora possua simplificações na camada de transporte. Entretanto, o modelo de transferências de dados desenvolvido no SciMulator visa compensar tais simplificações. O processo de simulação transcorre em um número fixo de ciclos. A Figura 2 mostra o diagrama de atividades do simulador. A simulação inicia com a configuração de uma topologia, que dispõe os *peers* na rede. Em seguida, cada ciclo inicia com o processamento dos controles, que são ações globais realizadas na rede, tais como: registros de observação dos eventos (*Observadores*), definição de quais nós saem e ingressam na rede (*Churn*), e quais nós submetem atividades (*Controle de Submissão de Atividades*) naquele ciclo. Logo após, segue a execução dos protocolos por cada um dos *peers*.

Cada *peer* inicia cada ciclo calculando o quanto de seus recursos está disponível para aquele ciclo. Em seguida, verifica se os vizinhos que estão processando alguma de suas tarefas ainda estão ativos. Caso o *peer* detecte a saída de um vizinho, a tarefa é reescalonada para outro nó. O próximo passo é a submissão de possíveis novas atividades, com o escalonamento das tarefas. Cada *peer*, então, requisita os dados de entrada para processar as tarefas recebidas e, após, responde às requisições iniciando o envio dos pacotes de dados. Cada *peer* segue com a execução do protocolo de *upload* e, depois, o de *download*. Ao final, cada *peer* processa o que for possível de suas tarefas e envia os resultados e dados de proveniência das tarefas concluídas naquele ciclo.

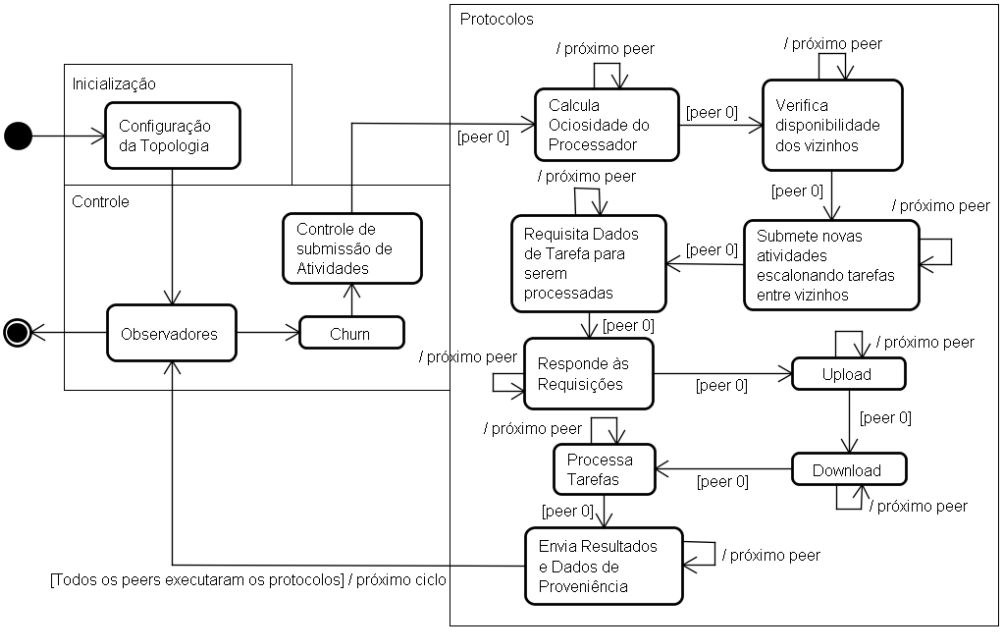


Figura 2: Diagrama de Atividades do Simulador

3. A Arquitetura SciMule

O SciMule é uma arquitetura projetada para distribuir atividades de workflows científicos em ambientes P2P. Em linhas gerais, as redes P2P podem ser classificadas quanto ao seu grau de centralização ou a forma de estruturação. Uma rede pode ser: centralizada, na qual um nó (*super-peer*) armazena informações sobre os outros nós e controla as buscas; descentralizada, caracterizada pela ausência do *super-peer* ou outro mecanismo de controle centralizado; e híbrida, que é uma combinação de ambas as

abordagens (Oram 2001). Já no que tange a topologia das redes descentralizadas, as redes podem ser não estruturadas, cujo armazenamento de dados e a vizinhança são aleatórios; ou estruturadas, cuja topologia é definida por algoritmos determinísticos, utilizados para otimizar a localização de recursos (Balakrishnan et al. 2003). Sob esta perspectiva, a arquitetura SciMule é dita híbrida, pois combina certas funcionalidades dos *super-peers* para facilitar o mecanismo de entrada de novos *peers* na rede, porém o escalonamento e a execução de tarefas são feitos de maneira distribuída sobre uma topologia não estruturada.

O objetivo do SciMule é estabelecer uma rede colaborativa entre cientistas para execução de tarefas de experimentos. Os principais desafios enfrentados pelo SciMule envolvem o escalonamento de tarefas, tolerância a falhas, o balanceamento de carga e a elaboração de uma topologia eficiente para troca de dados e processamento de tarefas de experimentos científicos de diferentes domínios, tudo em uma rede com controle distribuído. O SciMule é composto por três camadas: (i) uma camada de submissão que distribui atividades na rede, (ii) uma camada de execução que processa os pacotes de atividades recebidos, e (iii) a camada de *overlay* que armazena informações de posição dos nós na rede. Cada *peer* possui as três camadas e pode se comportar como cliente, quando submete atividades, como nó de execução e como um nó especial denominado *gate peer* (GP), que orienta o ingresso de novos *peers* na rede na camada de *overlay*.

O SciMule provê a análise de dois tipos de paralelização: o paralelismo de dados (Meyer et al. 2007) e o paralelismo por varredura de parâmetros (Samples et al. 2005). Para tal, a camada de submissão é capaz de decompor uma atividade MTC em um conjunto de tarefas e escaloná-las na rede. A Figura 3 apresenta a arquitetura em três camadas que caracteriza o SciMule. Tais camadas são detalhadas a seguir.

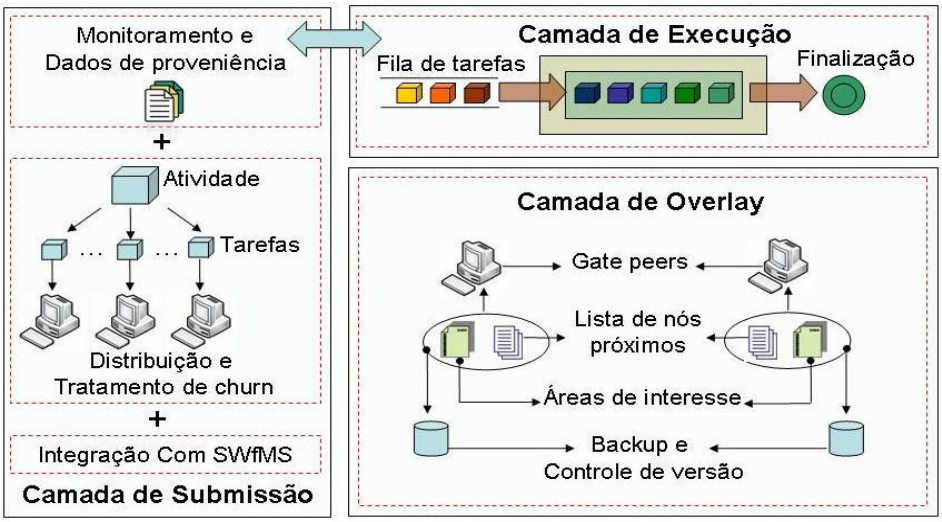


Figura 3: Arquitetura SciMule

Camada de Submissão. A camada de submissão é composta por componentes de workflow, que são módulos genéricos incluídos no SWfMS, tal como Kepler (Altintas et al. 2006), Taverna (Oinn et al. 2004) ou VisTrails (Callahan et al. 2006), e por componentes do mecanismo MTC do SciMule, responsáveis por distribuir tarefas, capturar dados de proveniência e tratar eventos de *churn* (entrada e saída intermitente dos nós na rede) (Wu et al. 2008). As atividades são divididas em tarefas de acordo com o tipo de paralelização que é definido pelo pesquisador no arquivo de configurações antes de iniciar o experimento.

Camada de Execução. Na camada de execução, as tarefas são recebidas e colocadas em uma fila. Quando os pacotes de dados de uma tarefa são recebidos por completo, elas são processadas. O *peer* confirma a conclusão da tarefa quando a execução tiver sido finalizada, retornando o controle para a camada de submissão. A saída de um *peer* de execução implica o reescalonamento da tarefa a outro *peer* ativo na rede.

Camada de Topologia. A camada de topologia (*overlay*) é apoiada por dois componentes: os *gate peers* (GP) e a lista de GP ativos. GP possuem a mesma estrutura que *peers* normais, porém armazenam uma lista de nós próximos (vizinhança) e suas áreas de interesse (*subjects*) (Dick et al. 2009), que estão relacionadas a programas e ferramentas em comum utilizados pelos *peers*. A lista de GP fica disponível na rede para facilitar a busca por GP próximos. Na topologia do SciMule, os *peers* são posicionados em um modelo unidimensional, de modo que a proximidade possa ser medida pela menor diferença entre dois pontos em uma circunferência. Um novo *peer* entra na rede sem qualquer conexão com os demais. O primeiro passo é obter a lista de GP para registrar-se em um deles. O nó recém-chegado se registra no GP mais próximo e, em seguida, adquire a lista de vizinhança. Ele também recebe a lista de um segundo GP adjacente para ter mais opções. Dado que, em uma rede com n nós e g GP, onde cada GP possui, em média, uma lista com n/g *peers*, um nó ingressante tem, inicialmente, no máximo $2n/g$ opções de vizinhos. Posteriormente, ele também deve estabelecer conexões com outros nós ingressantes registrados no mesmo GP e nos outros dois GP adjacentes, podendo atingir um máximo de $3n/g$ vizinhos. O SciMule também limita o número de conexões iniciais, de forma que *peers* novos precisam permanecer mais tempo na rede para adquirir mais vizinhos. Esta abordagem visa evitar nós que consomem muitos recursos, porém compartilham pouco (*free riders*).

Para um primeiro estudo do comportamento do SciMulator, a arquitetura SciMule foi utilizada como objeto de simulação. Cada componente descrito nesta seção foi codificado no ambiente SciMulator. A próxima seção traz os resultados dessa avaliação com detalhes. Apesar da avaliação do SciMulator ter sido realizada com a simulação dos componentes da arquitetura SciMule, o simulador não é restrito a esta arquitetura. Na prática, simulações de execuções de workflows científicos em redes P2P podem ser modeladas e executadas no SciMulator, independente da arquitetura proposta/utilizada.

4. Avaliação do SciMulator

As camadas da arquitetura SciMule foram utilizadas para avaliar o comportamento do SciMulator. Para realizar uma avaliação inicial do desempenho do SciMulator, executamos diversas simulações com ele, medindo o tempo de execução, a memória utilizada pelo simulador e quantas tarefas submetidas foram finalizadas variando o número de *peers* e a frequência de *churn*. A variação do tamanho da rede é interessante para avaliarmos a escalabilidade do simulador e a variação do *churn* indica quanto o dinamismo da rede influencia no desempenho do simulador ao processar as atividades. Os dados foram registrados por elementos (classes) de controle específico para registro de *log*, que registraram o tempo decorrido na simulação, a memória máxima utilizada pela máquina virtual Java e a taxa de atividades submetidas e executadas.

As simulações ocorreram por 14.400 ciclos, que equivalem a quatro dias. As atividades, submetidas na rede a uma frequência de 1%, custavam 8000 unidades de processamento com um conjunto de dados de 192MB. Estas atividades foram fragmentadas em 128 tarefas e escalonadas na rede. O número médio de vizinhos foi 32 e a conectividade máxima foi configurada para 64 *peers*. As instâncias do simulador

executaram em uma Altix ICE 8200 com processadores Intel Xeon 5355 de 2.66GHz com cerca de 1GB de memória RAM por núcleo de processamento. Cada instância do simulador foi executada por um núcleo. A Figura 4 apresenta os resultados obtidos em tempo de simulação e memória utilizada.

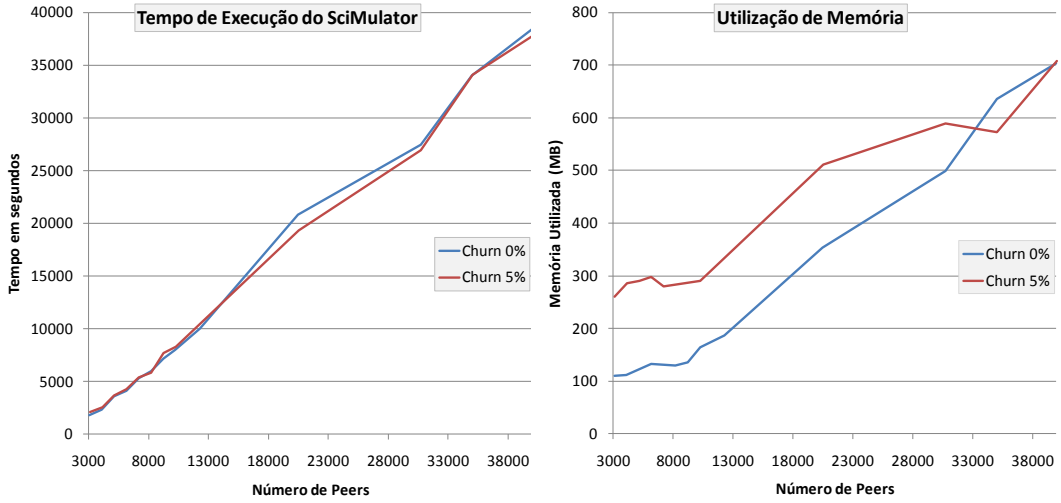


Figura 4: Medidas de desempenho do SciMulator

Ambos os resultados indicam que o simulador apresenta um comportamento linear no tempo de execução e no consumo de memória com o aumento do número de nós da rede. O cenário com *churn* apresenta um consumo maior de memória em função da estrutura de dinamismo do PeerSim, já que os objetos da classe *GeneralNode* não são descartados durante a simulação, mas apenas desligados da rede. Dessa forma, a memória não é liberada, pois é possível que o *peer* desligado volte à rede mais tarde. A aproximação do consumo de memória nos casos com maior número de *peers* na rede se deve à coleta de lixo da máquina virtual Java, que foi configurada para limitar o uso de memória em 768MB. Quando a utilização se aproxima deste valor, o sistema de coleta de lixo torna-se mais rigoroso e frequente.

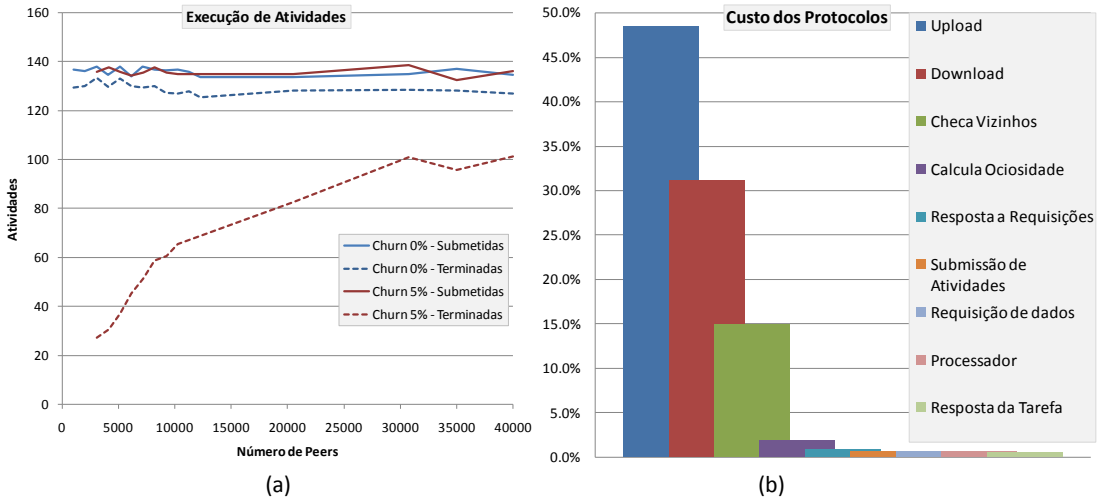


Figura 5: Medidas de (a) Submissão e finalização de atividades e (b) percentual de tempo de simulação gasto com cada protocolo.

Já nas medidas de atividades submetidas e finalizadas, os eventos de *churn* causam um grande impacto, reduzindo a taxa de finalização a cerca de 20% nas simulações com menos *peers*. A Figura 5 (a) mostra o gráfico de finalização de tarefas

variando o número de nós na rede. Nas simulações maiores, a taxa de finalização de atividades cresce, pois, como a frequência de submissão de atividades é a mesma, há mais *peers* ociosos. Dessa forma, a probabilidade de um *peer* ativo no processamento de uma tarefa sair da rede é menor. A queda de um *peer* ativo implica o reescalonamento de uma tarefa, o que pode aumentar significantemente o tempo de execução de uma atividade em função da retransmissão dos dados da tarefa.

A Figura 5 (b) mostra o percentual de tempo gasto com cada protocolo. O protocolo de *upload* é o mais custoso, pois o *peer* que submeteu uma tarefa necessita enviar o conjunto de dados para cada um dos vizinhos que deve processá-la. No protocolo de *download*, o *peer* faz o *download* apenas do dado da tarefa que precisa processar. Ainda assim, fica claro que os protocolos de transmissão de dados são os mais custosos para o simulador representando mais de 75% do tempo de execução de protocolos. Como os processos de transmissão de dados influenciam tanto no tempo de simulação como na taxa de finalização de tarefas, acreditamos que a melhoria de tais processos traga benefícios ao simulador. Técnicas de compactação, replicação e melhor distribuição de dados devem trazer um efeito positivo no desempenho do SciMulator.

5. Trabalhos Relacionados

A implantação de ambientes de computação distribuída em larga escala é bastante custosa, comprometendo a viabilidade de estudos para avaliar novos cenários e soluções. Como alternativa, simuladores vêm sendo cada vez mais utilizados. O GridSim (Buyya and Murshed 2002) é um simulador baseado em eventos e orientado para *grids.* Suas principais entidades são: (i) o usuário, que define parâmetros da simulação, tais como frequência de submissão de tarefas e estratégias de escalonamento, (ii) o escalonador, (iii) o recurso compartilhado, (iv) o serviço de informação, que armazena a lista de recursos disponíveis na *grid* e (v) as entradas e saídas que caracterizam as tarefas. Embora simule uma rede heterogênea, o GridSim não oferece uma rede dinâmica com controle totalmente distribuído, dificultando a construção de um simulador P2P.

CloudSim (Buyya et al. 2009) é um arcabouço para a simulação de computação em nuvem que provê componentes que simulam servidores, políticas de escalonamento e alocação de recursos. O CloudSim é independente de plataforma e pode ser caracterizado como uma extensão do GridSim, cuja infraestrutura principal é modelada pelo componente *DataCenter* que recebe as requisições e aloca os recursos, de acordo com a política definida que pode ser do tipo espaço compartilhado ou tempo de processamento compartilhado. Entretanto, o CloudSim também não oferece um ambiente de controle distribuído, pois centraliza o gerenciamento no componente *DataCenter*. Esta abordagem o descaracteriza como simulador para um ambiente P2P.

O PeerSim (Jelasity et al. 2010) é dedicado à simulação de redes P2P e oferece duas abordagens: o modelo síncrono, baseado em ciclos, nos quais cada nó obtém o controle e executa ações (protocolos) de forma sequencial comunicando-se diretamente com outros nós sem uma camada de transporte e, o modelo assíncrono, baseado em eventos, que controla a ordem em que os nós executam os protocolos através da troca de mensagens. O PeerSim oferece uma arquitetura P2P básica satisfatória, mas a avaliação de cenários mais complexos exige a extensão do simulador. A ausência da camada de transporte e dos mecanismos de execução de tarefas nos *peers* tornou necessária a construção do SciMulator.

6. Conclusões

A utilização de redes P2P para processamento distribuído de atividades pode ser uma solução interessante, em função da sua maior acessibilidade. Entretanto, a complexidade estrutural para construir uma rede deste tipo para realizar avaliações iniciais de arquiteturas é muito elevada. Por tal motivo, apresentamos o SciMulator, um simulador de ambientes P2P voltado à submissão e execução de atividades de workflows científicos seguindo o paradigma de paralelização MTC.

O SciMulator é uma extensão do PeerSim e modela a paralelização, o escalonamento, a distribuição e a execução de tarefas em uma rede P2P. As características da rede heterogênea e os eventos estocásticos, como a submissão de atividades e eventos de *churn*, variam seguindo distribuições estatísticas com o objetivo de representar a realidade. Inicialmente, modelamos a arquitetura do SciMule no SciMulator para, então, avaliá-lo.

Na avaliação inicial de desempenho, o SciMulator mostrou-se escalável no tempo de execução e consumo de memória, mesmo nos cenários com *churn.* A taxa de finalizações de atividade, como esperado, sofre impacto negativo na presença de *churn*, mas o efeito é amenizado com o aumento da rede. Com estes resultados, acreditamos que o SciMulator é um arcabouço promissor para avaliações de arquiteturas para execução de atividades em redes P2P.

Como trabalhos futuros, pretendemos aprimorar o mecanismo de *churn* do simulador, permitindo que nós que tenham saído da rede anteriormente, possam reingressar e dar continuidade ao processamento de tarefas. Também planejamos implementar soluções para compactação, replicação e distribuição de dados mais eficiente com mecanismos de indexação para melhorar a taxa de finalização de tarefas, uma vez que a retransmissão de dados é o maior gargalo da abordagem.

Referências

Abramson, D., Enticott, C., Altinas, I., (2008), "Nimrod/K: towards massively parallel dynamic grid workflows". In: *SC 08*, p. 1-11, Austin, Texas.

Almeida, E. C. D., Sunyé, G., Traon, Y. L., Valduriez, P., (2008), "A Framework for Testing Peer-to-Peer Systems". In: *Proceedings of the 2008 19th International Symposium on Software Reliability Engineering*, p. 167-176

Altintas, I., Barney, O., Jaeger-Frank, E., (2006), "Provenance Collection Support in the Kepler Scientific Workflow System", *Provenance and Annotation of Data*, , p. 132, 118.

Anderson, D., (2004), "BOINC: a system for public-resource computing and storage". In: *Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing*, p. 10, 4

Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., Stoica, I., (2003), "Looking up data in P2P systems", *Commun. ACM*, v. 46, n. 2, p. 43-48.

Barbosa, V. C., (1996), *An Introduction to Distributed Algorithms*. The MIT Press.

Boudani, A., Chen, Y., Straub, G., Simon, G., (2008), "Internet-Scale Simulations of a Peer Selection Algorithm". In: *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, p. 531-535, Los Alamitos, CA, USA.

Brown, D. A., Brady, P. R., Dietz, A., Cao, J., Johnson, B., McNabb, J., (2007), "A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis", *Workflows for e-Science*, Springer, p. 39-59.

Buyya, R., Ranjan, R., Calheiros, R., (2009), "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges". In: *Proc.*

of HPCS 2009HPCS 2009, Leipzig, Germany.

Buyya, R., Murshed, M., (2002). GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *ArXiv Computer Science e-prints*. Disponível em: http://adsabs.harvard.edu/abs/2002cs.....3019B. Acesso em: 17 Mar 2010.

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., Vo, H. T., (2006), "VisTrails: visualization meets data management". In: *Proc. SIGMOD 2006*, p. 745-747, USA.

Davidson, S. B., Freire, J., (2008), "Provenance and scientific workflows: challenges and opportunities". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, p. 1345-1350, Vancouver, Canada.

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

Dick, M. E., Pacitti, E., Kemme, B., (2009), "Flower-CDN: a hybrid P2P overlay for efficient query processing in CDN". In: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, p. 427-438, Saint Petersburg, Russia.

Freedman, D., Pisani, R., Purves, R., (2007), *Statistics, 4th Edition*. 4 ed. W. W. Norton.

Grid4All Consortium, (2009), "Towards Hybrid Clouds - A Grid4All perspective on cloud computing". , White paper. www.grid4all.eu.

Jelasity, M., Montresor, A., Jesi, G. P., Voulgaris, S., (2010), *The PeerSim simulator*, http://peersim.sourceforge.net.

Meyer, L., Scheftner, D., Vöckler, J., Mattoso, M., Wilde, M., Foster, I., (2007), "An Opportunistic Algorithm for Scheduling Workflows on Grids", *High Performance Computing for Computational Science - VECPAR 2006*, , p. 1-12.

Ogasawara, E., Dias, J., Oliveira, D., Rodrigues, C., Pivotto, C., Antas, R., Braganholo, V., Valduriez, P., Mattoso, M., (2010), "A P2P Approach to Many Tasks Computing for Scientific Workflows". In: *9th International Meeting on High Performance Computing for Computational Science*, Berkeley, CA, USA.

Ogasawara, E., Oliveira, D., Chirigati, F., Barbosa, C. E., Elias, R., Braganholo, V., Coutinho, A., Mattoso, M., (2009), "Exploring many task computing in scientific workflows". In: *MTAGS 09*, p. 1-10, Portland, Oregon.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., et al., (2004), "Taverna: a tool for the composition and enactment of bioinformatics workflows", *Bioinformatics*, v. 20, p. 3045-3054.

Oram, A., (2001), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc.

Pacitti, E., Valduriez, P., Mattoso, M., (2007), "Grid Data Management: Open Problems and New Issues", *Journal of Grid Computing*, v. 5, n. 3, p. 273-281.

Raicu, I., Foster, I., Yong Zhao, (2008), "Many-task computing for grids and supercomputers". In: *Workshop on Many-Task Computing on Grids and Supercomputers*, p. 1-11

Samples, M. E., Daida, J. M., Byom, M., Pizzimenti, M., (2005), "Parameter sweeps for exploring GP parameters". In: *2005 workshops on Genetic and evolutionary computation*, p. 212-219, Washington, D.C.

Wu, D., Tian, Y., Ng, K., Datta, A., (2008), "Stochastic analysis of the interplay between object maintenance and churn", *Computer Communications*, v. 31, n. 2 (Feb.), p. 220-239.

Proposta de uma metodologia de avaliação de sistemas peer-to-peer baseados em tabelas hash distribuídas

Paulo Ricardo Zanoni[1], **Luis Carlos Erpen de Bona**[1], **Eduardo Cunha de Almeida**[1]

[1]Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – CEP 81.531-980 – Curitiba – PR – Brasil

{paulo,eduardo,bona}@c3sl.ufpr.br

Abstract. *Among the most used types of peer-to-peer (P2P) networks are the distributed hash tables (DHTs), which are data structures that allow the insertion of data indexed by keys. There is a big amount of DHTs, which may have multiple implementations and be configured through many parameters. However, there is no consensus about the best way to evaluate a DHT performance, which complicates the choice of the ideal implementation for each case. This paper proposes a methodology for evaluating the performance of DHTs that uses a standard set of performance tests based on the most used evaluations found in the literature. An implementation for this methodology and its initial results are also presented.*

Resumo. *Dentre os tipos de redes par-a-par (P2P,* peer-to-peer*) mais utilizados estão as tabelas hash distribuídas (DHTs,* distributed hash tables*), que são estruturas de dados que permitem a inserção de dados indexados por chaves. Existe uma grande quantidade de DHTs, que podem possuir várias implementações e serem configuradas através de diversos parâmetros. Entretanto, não há um consenso sobre a melhor maneira de avaliar o desempenho de uma DHT, o que dificulta a escolha da implementação ideal para cada caso. Este trabalho propõe uma metodologia de avaliação de desempenho de DHTs que utiliza um conjunto padrão de testes de desempenho baseado nas avaliações mais utilizadas encontradas na literatura. É apresentada também uma implementação para essa metodologia e seus resultados iniciais.*

1. Introdução

Ao longo da última década as redes par-a-par (P2P, *peer-to-peer*) tornaram-se muito populares, surgindo uma grande variedade de casos de uso e implementações que possuem até milhões de participantes, como no caso da rede Gnutella [Gnutella Protocol Development]. Uma rede P2P é composta por um conjunto de participantes (nodos) dinâmicos sem o controle de uma autoridade central. Em geral as redes P2P são projetadas para gerenciarem uma grande quantidade de nodos entrando e saindo do sistema a todo momento.

As redes P2P podem ser classificadas de acordo com suas estruturas em três grupos: as redes P2P *estruturadas*, as *não-estruturadas* e as *fracamente estruturadas* [Androutsellis-Theotokis and Spinellis 2004]. Nas redes estruturadas os participantes mantém uma topologia pré-definida e seguem regras que definem como as informações são transmitidas e armazenadas. Nas redes P2P não-estruturadas não há regras para a

manutenção da topologia e local de armazenamento das informações, diminuindo o custo de manutenção da rede mas impossibilitando o estabelecimento de garantias quanto ao desempenho de certos aspectos da mesma. As redes P2P fracamente estruturadas são caracterizadas por não definirem estritamente a localização do conteúdo armazenado, mas afetarem a sua localização através de seus algoritmos de roteamento.

Os exemplos mais comuns de redes P2P estruturadas são as tabelas hash distribuídas (DHTs, *distributed hash tables*). As DHTs são estruturas de dados distribuídas que permitem a inserção de dados indexados por chaves. Cada chave é um identificador único, que deve ser guardado e utilizado para encontrar os dados previamente inseridos. Dentre as DHTs mais utilizadas estão Chord [Stoica et al. 2001], Pastry [Rowstron and Druschel 2001], Tapestry [Zhao et al. 2004] e CAN [Ratnasamy et al. 2001].

Existe uma grande quantidade de DHTs, cada uma com suas próprias características, parâmetros e topologia, o que dificulta o processo de escolha da DHT ideal para um determinado sistema. Além disso, cada DHT pode possuir diversas implementações, aumentando ainda mais o número de opções. Para facilitar o processo de escolha da DHT ideal para um determinado sistema, metodologias de avaliação de desempenho podem ser utilizadas. Nos últimos anos foram publicados uma série de trabalhos que propõem metodologias de avaliação de desempenho. Contudo, não existe um consenso de um conjunto mínimo de testes de desempenho suficientes para realizar esta avaliação de maneira imparcial, o que dificulta a avaliação e comparação dos resultados apresentados.

Neste artigo é proposta uma metodologia de avaliação de desempenho de DHTs. Essa metodologia é composta por diversos testes de desempenho, que por sua vez são definidos através dois elementos: cargas de trabalho e métricas de avaliação. O objetivo dos testes é cobrir de maneira imparcial os principais casos de uso das DHTs. As cargas de trabalho e métricas de avaliação são baseadas nos testes de desempenho mais utilizados pelos trabalhos que propõem a avaliação de desempenho de DHTs. Uma implementação para esta metodologia também é apresentada, bem como seus resultados iniciais.

A Seção 2 apresenta e compara alguns dos diversos trabalhos que avaliam desempenho de diferentes DHTs. Com base na análise realizada, a Seção 3 apresenta uma proposta de uma metodologia de avaliação de sistemas P2P baseados em DHTs, que tem sua implementação apresentada na Seção 4. A Seção 5 apresenta os resultados obtidos através dos experimentos iniciais realizados com a ferramenta. Por fim, a Seção 6 apresenta as considerações finais.

2. Avaliação de sistemas baseados em DHTs

Um teste de desempenho de DHT consiste na aplicação de uma carga de trabalho (*workload*) à rede e na análise de diversas métricas enquanto a mesma está sendo aplicada. A definição da carga de trabalho deve ser a mais completa possível, envolvendo informações como a DHT, seus parâmetros, o número de nodos, as ações executadas por cada um dos nodos (comportamento, o que inclui desde as operações `put` e `get` realizadas até os momentos em que o nodo entra e sai da rede) e até, se possível, a localização geográfica e organização topológica dos nodos, tanto nas camadas de aplicação quanto nas outras camadas de rede. Alguns trabalhos utilizam cargas de trabalho obtidas através da monitoração de redes P2P reais e outros criam suas próprias cargas de trabalho.

Apesar de ainda não haver um conjunto de testes de desempenho amplamente aceito como padrão para a avaliação de desempenho de DHTs, já foi apresentada uma grande quantidade de trabalhos que avaliam o desempenho de DHTs. Alguns desses trabalhos propõem também metodologias de avaliação de desempenho de DHTs, como [Oppenheimer et al. 2004], [Li et al. 2004] e [Kato and Kamiya 2007], porém nenhum deles foi amplamente utilizado em trabalhos subsequentes ou procurou estabelecer suas metodologias com base em trabalhos relacionados. Esta seção descreve alguns destes trabalhos (dos encontrados, os que possuem maior foco em avaliação de desempenho), analisando as avaliações de desempenho propostas e apresentando um resumo com as seguintes informações: as DHTs – ou redes – utilizadas, o ambiente sobre o qual as avaliações foram realizadas, o tamanho das redes em questão, o comportamento dos nodos e as métricas utilizadas. Esses dados são apresentados na Tabela 1.

Cada uma das redes utilizadas pelos trabalhos analisados foi associada a uma das seguintes categorias: (i) implementações de DHTs (i.e., prontas para serem utilizadas por aplicações reais); (ii) DHTs providas por simuladores; (iii) infraestruturas que possibilitam a criação de DHTs, como no caso de Plaxton, utilizado pelas DHTs Pastry e Tapestry; ou (iv) modelos teóricos de DHTs ou de infraestruturas que possibilitam a criação de DHTs, como no caso de [Kong et al. 2006]. As redes utilizadas nos trabalhos analisados são: Chord [Stoica et al. 2001], Pastry [Rowstron and Druschel 2001], Tapestry [Zhao et al. 2004], Kademlia [Maymounkov and Mazières 2002], CAN [Ratnasamy et al. 2001], Kelips [Gupta et al. 2003], Symphony [Manku et al. 2003], Plaxton [Plaxton et al. 1999], Accordion [Li et al. 2005] e Bamboo [Rhea et al. 2004].

Os ambientes utilizados nos diversos trabalhos analisados foram classificados como: (i) *reais*, para os casos mais próximos de uma utilização real de DHTs, como o PlanetLab [Chun et al. 2003]; (ii) *simulação*, para os casos nos quais simuladores de redes P2P foram utilizados; (iii) *emulação*, para os casos nos quais foram utilizadas implementações reais em ambientes restritos (e.g., centenas de nodos virtuais em um único nodo físico ou uma pequena rede local) ou (iv) *analíticos*, para os casos nos quais os estudos realizados foram puramente teóricos e os resultados apresentados são, por exemplo, provas matemáticas. Os simuladores utilizados nos trabalhos analisados são: p2psim [p2psim], ACME framework [Oppenheimer et al. 2003] e Microsoft Research Pastry Simulator v3.0A [Bjurefors et al. 2004]. Alguns trabalhos não mencionam os simuladores utilizados.

A coluna *tamanho da rede* descreve a quantidade de nodos utilizados nas avaliações realizadas pelos trabalhos apresentados. Atenção especial deve ser dada para o trabalho [Kong et al. 2006], onde a análise teórica é realizada para o caso no qual o tamanho da rede tende ao infinito.

A coluna *comportamento dos nodos* descreve o comportamento adotado pelos nodos na rede, o que inclui as operações `put` e `get`, além das entradas e saídas de cada nodo da rede (juntas, essas entradas e saídas definem o dinamismo dos nodos na rede, também conhecido como *churn*). Para os casos onde houve vários testes de desempenho com comportamentos diferentes, os diversos tipos de comportamento são listados. Na maioria dos casos analisados o comportamento dos nodos era simplesmente realizar um certo número de requisições em um determinado intervalo de tempo. Em alguns casos

os comportamentos utilizaram cargas de trabalho (*workloads*) observadas em aplicações reais, como em [Castro et al. 2005]. Em outros casos os comportamentos foram apenas inspirados nos possíveis comportamentos reais, ou então a escolha do comportamento adotado simplesmente não foi justificada. Houve também avaliações onde foram analisadas apenas as mensagens de controle geradas pelas DHTs, portanto os nodos não realizaram requisições, como em [Bjurefors et al. 2004].

Cada trabalho realizado possui seu próprio método para analisar desempenho e medir os resultados. Entretanto certas métricas como a latência foram analisadas de maneiras diferentes em diversos trabalhos (e.g., em [Li et al. 2004] a latência é simplesmente o intervalo de tempo entre uma requisição `get` e sua resposta, mas em [Rhea et al. 2003] a latência é expressa como a relação entre o intervalo de tempo obtido e o tempo de *ping* entre o nodo que busca a chave e o nodo que a possui). Apesar dessas pequenas diferenças, nós organizamos as diversas medições realizadas pelos trabalhos entre os seguintes grupos:

latência: mede o intervalo de tempo entre uma ou mais requisições e suas respostas;
tráfego de rede: mede a quantidade de mensagens – em valor absoluto ou em bytes – gerada pelos nodos durante suas operações;
taxa de sucesso: mede a quantidade de requisições completadas com sucesso pelos nodos;
proximidade de réplicas: compara a distância da réplica obtida através de uma requisição `get` com a distância da réplica mais próxima do nodo;
consistência: compara a consistência dos resultados de um conjunto de operações iguais – como buscas pela mesma chave – realizadas por nodos diferentes quase ao mesmo tempo;
caminhos falhos: mede a quantidade de nodos que pode ser alcançada por cada nodo da rede.

Observamos que algumas DHTs como Chord e Pastry foram utilizadas em quase todos os trabalhos realizados, portanto podem ser consideradas como as DHTs "mais populares". Além disso, quase todos os trabalhos analisados utilizaram ambientes simulados ou emulados, sendo apenas um deles baseado em um ambiente real. Outro fato observado foi que apesar das discussões sobre escalabilidade e aplicações que podem possuir até milhões de nodos, boa parte dos trabalhos não analisou mais do que apenas algumas centenas ou milhares de nodos. Ainda, apesar do comportamento dos nodos nas avaliações realizadas envolver quase somente a realização de buscas periódicas, a caracterização das redes envolvidas variou bastante, principalmente com relação aos algoritmos e técnicas utilizados internamente pelas DHTs. Por fim, observamos também que as métricas mais utilizadas nas avaliações de desempenho foram latência, tráfego de rede e taxa de sucesso.

3. Definição da metodologia

Em uma avaliação de desempenho de DHT há uma série de fatores que devem ser analisados, portanto cada avaliação deve ser composta por diversos testes de desempenho. Uma das maiores dificuldades na avaliação é que cada uma das inúmeras possíveis aplicações que utilizam DHTs pode apresentar uma carga de trabalho diferente. Portanto o conjunto

Trabalho	Rede(s)	Ambiente	Tamanho da rede	Comportamento dos nodos	Métricas
[Rhea et al. 2003]	Chord e Tapestry	Real: PlanetLab	79 – 83 nodos	(i) buscas sem churn	(i) latência, (ii) proximidade de réplicas
[Bjurefors et al. 2004]	Pastry	Simulação: Microsoft Research Pastry Simulator v3.0A	30-3000 nodos	(i) inicialização, (ii) *i* com buscas, (iii) *ii* com tabelas particionadas	(i) tráfego de rede
[Li et al. 2004]	Chord, Tapestry, Kelips e Kademlia	Simulação: p2psim	1024 nodos	(i) buscas com churn	(i) tráfego de rede, (ii) latência
[Oppenheimer et al. 2004]	Chord, Pastry e Tapestry	Emulação: ACME	150 nodos	(i) buscas com churn	(i) latência, (ii) taxa de sucesso, (iii) tráfego de rede, (iv) consistência
[Castro et al. 2005]	Pastry, Heterо-Pastry e Super-Pastry	Simulação	10000 – 37000 nodos	(i) workload real sem queries, (ii) workload real, (iii) churn	(i) tráfego de rede, (ii) taxa de sucesso, (iii) latência
[Kong et al. 2006]	CAN, Chord, Kademlia, Symphony e Plaxton	Analítico	Infinito	(i) rede com nodos falhos	(i) caminhos falhos
[Kato and Kamiya 2007]	Accordion, Bamboo, Chord e Pastry	Emulação: rede local	91 – 991 nodos	(i) buscas sem churn, (ii) buscas com churn, (iii) outros modelos mais complexos	(i) taxa de sucesso, (ii) latência, (iii) tráfego de rede
[Harvesf and Blough 2007]	Pastry	Simulação	1024 nodos	(i) buscas com nodos falhos e replicação	(i) taxa de sucesso, (ii) latência

Tabela 1. Trabalhos que medem o desempenho de DHTs

de testes de desempenho relevantes para uma determinada aplicação pode ser completamente diferente do conjunto de testes de outra. Com isso, a determinação de um conjunto a ser adotado como padrão para a maioria dos casos não é trivial. A proposta desta metodologia é, a partir da análise realizada, definir um conjunto de testes de desempenho mínimo que seja capaz de refletir as medições mais comuns entre os trabalhos que medem o desempenho de DHTs.

A metodologia proposta define cada teste de desempenho como uma combinação de dois elementos: a *carga de trabalho* e as *métricas* a serem analisadas. A carga de trabalho é a definição das ações a serem executadas pelos nodos da DHT. Como o objetivo é que cada teste possa ser utilizado por diversas DHTs e em diversos ambientes, algumas informações como por exemplo a disposição dos nodos na DHT ou os detalhes das camadas de rede mais inferiores devem ser omitidas da carga de trabalho. As métricas definem os dados que devem ser coletados pelos nodos, como latência das operações, taxa de sucesso e outras. O conjunto de dados a serem coletados deve ser independente da DHT a ser utilizada.

Identificamos dois grupos de cargas de trabalho: as cargas de trabalho baseadas em aplicações reais e as cargas de trabalho que visam exercitar alguma operação ou funcionalidade específica da DHT. Um exemplo muito comum do primeiro grupo são as cargas de trabalho de aplicações de compartilhamento de arquivos. Já sobre segundo grupo pode-se citar as cargas de trabalho com alto grau de *churn*, que visam testar a estabilidade das DHTs, e as cargas de trabalho onde os nodos efetuam grandes quantidades de buscas, visando obter medidas como latência, corretude e outras.

As métricas também podem ser divididas em dois grupos: as métricas gerais, que podem ser medidas em todos os tipos de testes de desempenho e as métricas específicas,

que só devem ser medidas em testes que possuem cargas de trabalho específicas. Do primeiro grupo pode-se citar as três métricas identificadas como mais analisadas: latência das mensagens, tráfego de rede gerado e taxa de sucesso das operações. Do segundo grupo pode-se citar, por exemplo, as métricas que avaliam a distância e disponibilidade das réplicas armazenadas em DHTs com suporte a réplicas.

O conjunto básico de testes de desempenho a ser escolhido deve apresentar ambos os tipos de cargas de trabalho e ambos os tipos de métricas a serem analisadas. Esses testes devem contemplar as funcionalidades essenciais das DHTs: escalabilidade, tolerância a *churn* (readaptação da topologia, propagação de chaves, roteamento), inserção de chaves e busca de chaves.

Com base no exposto e na análise feita na Seção 2, um conjunto básico de testes para compor uma avaliação de desempenho deve conter as seguintes cargas de trabalho: (i) sem buscas, com e sem *churn*, viabilizando métricas que envolvam a análise dos custos de manutenção da DHT, como o tráfego de rede, o tempo de estabilização e a transferência de chaves entre nodos; (ii) com buscas e diversos níveis de *churn*, viabilizando métricas que envolvam as operações `get` e `put`, como latência e taxa de sucesso de operações; e (iii) reais, obtidas através da análise de aplicações reais que utilizam DHTs.

4. A ferramenta proposta

A ferramenta proposta tem como objetivo permitir a aplicação de testes de desempenho de DHTs. Sua elaboração foi realizada com os seguintes objetivos: (i) ser facilmente adaptável aos diversos tipos de DHTs existentes; (ii) permitir facilmente a adição de diversos testes de desempenho; (iii) possuir uma boa escalabilidade; (iv) funcionar sem a necessidade de obter informações específicas das DHTs (e.g, lista de nodos vizinhos) e (v) funcionar sem a necessidade de alteração nas DHTs a serem avaliadas.

Com base nos requisitos estabelecidos acima um modelo para a implementação da ferramenta foi elaborado. Este modelo é formado por três entidades: o *mestre*, os *controladores* e os *nodos*, conforme ilustrado na Figura 1. As linhas contínuas representam a comunicação entre as entidades realizada através da ferramenta e as linhas pontilhadas representam a comunicação entre as entidades *nodo* realizada através das DHTs.

O mestre é a entidade hierarquicamente superior: ele possui a definição da carga de trabalho a ser aplicada na rede e deve aplicá-la comunicando-se com os controladores. Essa troca de informações deve funcionar de maneira externa à DHT a ser avaliada e ser rápida e confiável o suficiente para não afetar negativamente o resultado dos testes de desempenho realizados. Como o mestre é apenas um e deve gerenciar muitos controladores, cuidado especial deve ser tomado para que a sua implementação não comprometa a escalabilidade da ferramenta: quanto menos informações ele precisar enviar aos controladores durante a execução dos testes, maior será o seu grau de escalabilidade.

Para cada nodo existente na DHT deve existir uma entidade chamada controlador. O controlador é responsável por operar o funcionamento de um nodo participante da DHT a ser avaliada. Ele recebe do mestre as informações sobre como será seu comportamento durante a execução dos testes e quais dados deverá coletar para obter seus resultados. A partir dessas informações ele controla seu nodo associado para executar as ações necessárias. Cada implementação de DHT a ser utilizada exige a implementação de um controlador diferente.

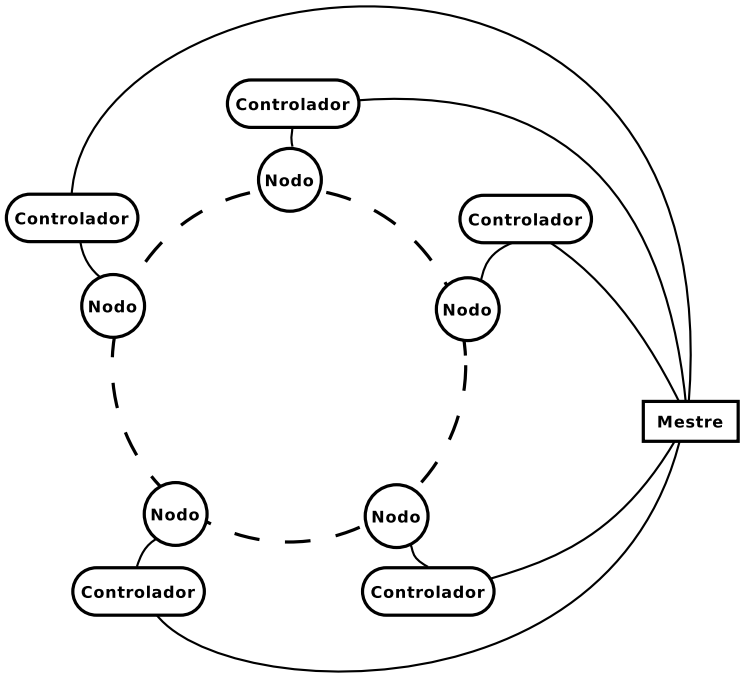


Figura 1. Representação esquemática da ferramenta proposta

Os nodos são as entidades que fazem parte da DHT, portanto podem possuir as mais variadas implementações. Apesar das possíveis diferenças, as implementações dos nodos precisam expor para os controladores as funcionalidades básicas das DHTs: iniciar uma nova DHT, entrar na DHT a partir de algum nodo que já faça parte da mesma, inserir chaves (`put`), buscar chaves (`get`) e sair da DHT. Todas as operações realizadas pelos controladores durante a avaliação devem depender apenas desse conjunto básico de funcionalidades.

Os resultados obtidos por cada controlador devem ser armazenados e enviados ao mestre após a realização da avaliação, para que então este possa processá-los e obter os resultados finais.

4.1. Multidhtshell

O Multidhtshell é a implementação realizada da ferramenta proposta e permite a utilização das DHTs providas pelo conjunto de ferramentas (*toolkit*) Overlay Weaver [Overlay Weaver]: Chord, Kademlia, Koorde, LinearWalker, Pastry e Tapestry. As entidades *mestre* e *controlador* são programas escritos na linguagem Ruby. A entidade *nodo* é representada pelo *owdhtshell*, uma ferramenta do Overlay Weaver que permite o controle de um nodo em uma DHT através de um pequeno *shell*. O Overlay Weaver foi escolhido por ser uma ferramenta que está em constante processo de desenvolvimento e possuir diversas DHTs que podem ser manipuladas através de uma única interface (o *owdhtshell*), o que torna necessária apenas uma implementação da entidade controlador. Através do Overlay Weaver pode-se utilizar tanto DHTs reais como simuladas, porém o Multidhtshell utiliza apenas DHTs reais.

Uma limitação da versão inicial dessa ferramenta é a comunicação entre as enti-

dades *mestre* e *controlador*. Essa comunicação não se dá através da rede e sim através de *threads*, portanto todos os nodos devem ser executados em um mesmo sistema. Com isso, pode-se afirmar que as avaliações realizadas são do tipo *emulação*. Uma vantagem desse método é que ele elimina parâmetros como conexão, latência e transmissão entre máquinas, aumentando a importância do desempenho da entidade *nodo* e aumentando também a reprodutibilidade das avaliações realizadas. É importante ressaltar que a comunicação entre os nodos das DHTs (pertencente ao código do Overlay Weaver) ainda assim é feita através dos protocolos de rede convencionais, portanto mesmo sendo realizada entre nodos de uma mesma máquina está sujeita a *bufferização*, espera de confirmações (ACK), etc.

A versão inicial do Multidhtshell possui dois testes de desempenho implementados, chamados de *estabilização* e *latência*. O teste *estabilização* serve para medir a taxa de propagação de chaves para nodos que entram na DHT. A aplicação de sua carga de trabalho consiste em quatro etapas: (i) um nodo é iniciado, criando a DHT, (ii) esse único nodo insere uma certa quantidade de chaves na DHT, (iii) outros nodos são adicionados na rede e (iv) paralelamente, cada um dos nodos tenta fazer `get` em cada uma das chaves inicialmente adicionadas até que consiga pelo menos um `get` com sucesso para cada chave. As métricas analisadas são o número de tentativas realizadas por cada nodo e o tempo decorrido. O teste *latência* tem como objetivo medir a latência das operações `get` das DHTs, dada pelo intervalo de tempo que ocorre do momento em que o nodo inicia a busca por uma chave até o momento em que ele a encontra, o que pode incluir comunicação com muitos nodos. Sua carga de trabalho é dividida da seguinte maneira: (i) um nodo é iniciado, criando a DHT, (ii) outros nodos são iniciados, (iii) espera-se até que todos os nodos entrem na DHT e (iv) paralelamente, cada nodo tenta fazer uma certa quantidade de requisições `get` em chaves aleatórias. A métrica analisada é o tempo de latência médio das requisições de cada nodo.

Versões futuras do Multidhtshell eliminarão a limitação acima citada, possuirão implementações de controladores capazes de controlar outros tipos de DHTs além daquelas providas pelo Overlay Weaver e possuirão mais testes de desempenho, de maneira a cumprir os objetivos propostos no início desta seção.

5. Resultados experimentais

Esta seção descreve os resultados obtidos nos experimentos iniciais realizados com o Multidhtshell. Como a versão inicial do Multidhtshell possui a limitação de que todos os nodos devem estar presentes no mesmo sistema, todos os nodos foram emulados em um computador com processador Intel(R) Core(TM) i5 3.20GHz, 4GB de memória RAM e sistema operacional Mandriva Linux 2010.1 Alpha 3 (Linux Kernel 2.6.33.1). A versão do Overlay Weaver utilizada foi a 0.9.9 com o *patch* oficial para melhorar a tolerância a *churn* (distribuído junto com o Overlay Weaver). A máquina virtual do Java utilizada pelo Overlay Weaver foi a Java HotSpot(TM) Server VM (build 16.0-b13, mixed mode). O objetivo dos experimentos realizados é apenas demonstrar que a ferramenta apresentada é funcional, portanto os valores escolhidos nos testes (como número de nodos e operações `get`) podem ser considerados pequenos.

O objetivo do teste *estabilização* é verificar o tempo de propagação das chaves para nodos que entram na DHT. O número de chaves presentes da DHT foi 100 e o número

de nodos inseridos – além do nodo inicial – foi 10. A Figura 2 mostra o tempo decorrido em cada DHT (i.e, tempo necessário para que todos os nodos da DHT realizassem pelo menos um `get` com sucesso em cada chave). A Figura 3 mostra o número máximo de buscas por uma única chave realizadas por um único nodo durante a avaliação (não necessariamente o nodo que mais demorou para obter todas as chaves).

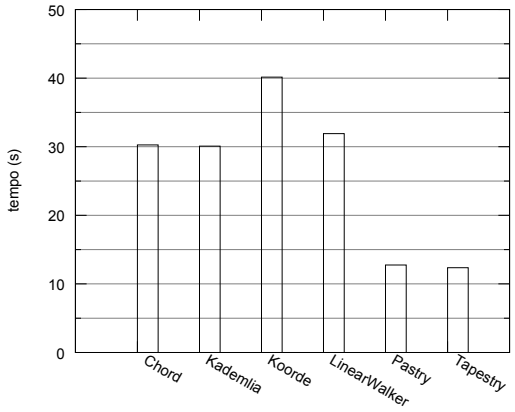


Figura 2. Estabilização: tempo de estabilização de cada DHT analisada. *Default reput interval*: 30s.

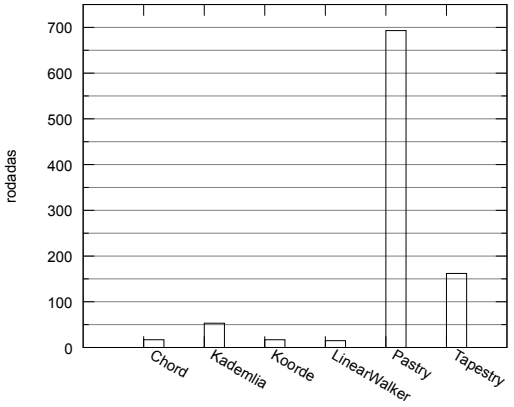


Figura 3. Estabilização: máximo de buscas por uma única chave por nodo. *Default reput interval*: 30s.

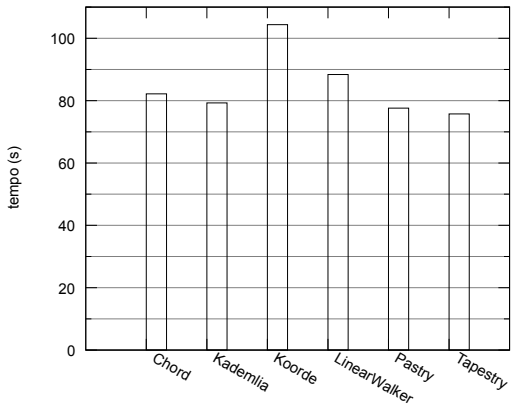


Figura 4. Estabilização: tempo de estabilização de cada DHT analisada. *Default reput interval*: 100s.

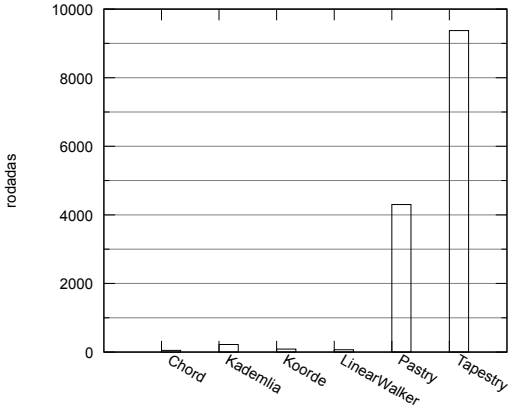


Figura 5. Estabilização: máximo de buscas por uma única chave por nodo. *Default reput interval*: 100s.

A implementação do Overlay Weaver utilizada possui um parâmetro chamado `DEFAULT_REPUT_INTERVAL`, que define a frequência com a qual as chaves são reinseridas na DHT (visando aumentar a tolerância a *churn*). As figuras 4 e 5 mostram os resultados obtidos quando o valor do parâmetro é modificado de 30 segundos para 100 segundos. Como esperado, o tempo de estabilização das DHTs analisadas aumentou. Observa-se que esse parâmetro exerce grande influência na capacidade de estabilização da rede, porém sem a presença de mais testes de desempenho não é possível medir seus outros efeitos. Além disso, as figuras 3 e 5 mostram uma grande diferença das

implementações das DHTs Pastry e Tapestry para as outras, pois estas são capazes de realizar uma quantidade de buscas muito maior em um intervalo de tempo menor. Maiores investigações sobre a razão deste comportamento devem ser realizadas através de análises no código-fonte do Overlay Weaver.

O teste *latência* tem como objetivo calcular a latência média das mensagens `get` em cada DHT. Em uma primeira avaliação foi utilizada uma DHT com 10 nodos, onde cada nodo realizou 100 buscas. A Figura 6 mostra, para cada DHT, a latência média dos nodos que apresentaram a menor e a maior latência, além da latência média entre todos os nodos. Outra avaliação foi realizada, porém com 50 nodos e 500 buscas para cada nodo. Seus resultados, apresentados na Figura 7, mostram que mesmo com um número muito maior de nodos e buscas o desempenho relativo entre as DHTs analisadas permanece o mesmo, com exceção das DHTs Kademlia, que apresentou um melhor desempenho relativo se comparado com a avaliação anterior e Pastry, que não pode ser avaliada devido a um *bug* na implementação do Overlay Weaver.

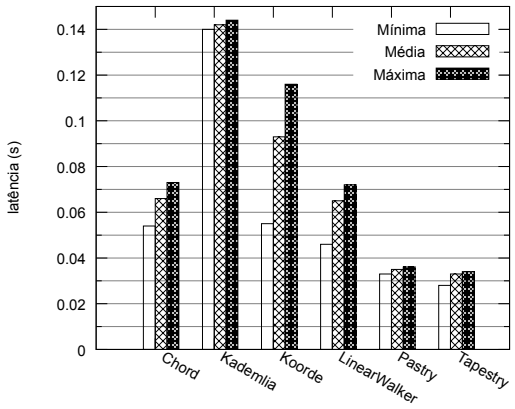


Figura 6. Latência: 100 buscas para cada um dos 10 nodos

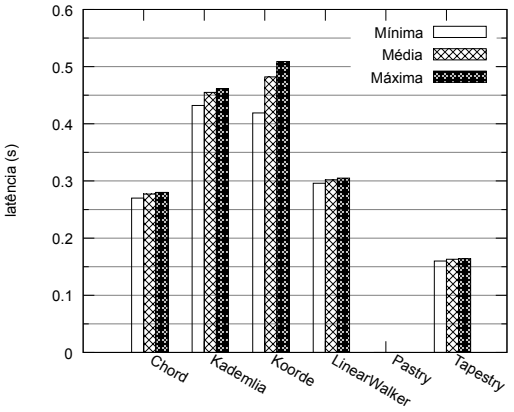


Figura 7. Latência: 500 buscas para cada um dos 50 nodos

6. Conclusão

Ao longo da última década as redes P2P tornaram-se bastante populares, sendo as DHTs os exemplos mais comuns de redes P2P estruturadas. Cada DHT pode possuir diversas implementações, que por sua vez podem ser ajustadas através de diversos parâmetros. Essa grande diversidade torna difícil a escolha da DHT ideal para cada aplicação. Uma possível solução para esse problema é a utilização de avaliações de desempenho. Nos últimos anos foram apresentados diversos trabalhos que propõem métodos para a avaliação de desempenho de DHTs, porém não há um consenso de um conjunto mínimo de testes de desempenho suficientes para realizar, de maneira imparcial, uma avaliação de desempenho de DHTs.

Este artigo analisou os trabalhos que efetuaram avaliações de desempenho de DHTs, visando identificar os seus pontos em comum. Com base nessa análise foi proposta uma metodologia de avaliação de desempenho de DHTs que utiliza um conjunto de testes de desempenho baseado nos testes mais utilizados pelos trabalhos analisados. Cada teste de desempenho foi definido como uma combinação de uma carga de trabalho e um conjunto de métricas a serem analisadas.

Uma implementação inicial para a metodologia proposta foi apresentada. Essa implementação é capaz de avaliar as DHTs implementadas pelo Overlay Weaver e possui dois testes de desempenho. Os resultados iniciais da utilização da ferramenta foram os esperados.

Referências

Androutsellis-Theotokis, S. and Spinellis, D. (2004). A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4):335–371.

Bjurefors, F., Larzon, L. A., and Gold, R. (2004). Performance of pastry in a heterogeneous system. In *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*, pages 278–279.

Castro, M., Costa, M., and Rowstron, A. (2005). Debunking some myths about structured and unstructured overlays. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation*, pages 85 – 98, Berkeley, CA, USA. USENIX Association.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. (2003). PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12.

Gnutella Protocol Development. `http://rfc-gnutella.sourceforge.net/`. Acessado em 9 de dezembro de 2009.

Gupta, I., Birman, K., Linga, P., Demers, A., and Van Renesse, R. (2003). Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead. *Lecture Notes in Computer Science*, pages 160–169.

Harvesf, C. and Blough, D. M. (2007). The Design and Evaluation of Techniques for Route Diversity in Distributed Hash Tables. In *Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing*, pages 237 – 238. Citeseer.

Kato, D. and Kamiya, T. (2007). Evaluating DHT Implementations in Complex Environments by Network Emulator. In *Proceedings of the IPTPS 2007*.

Kong, J. S., Bridgewater, J. S. A., and Roychowdhury, V. P. (2006). A General Framework for Scalability and Performance Analysis of DHT Routing Systems. *Arxiv preprint cs/0603112*.

Li, J., Stribling, J., Gil, T. M., Morris, R., and Kaashoek, M. F. (2004). Comparing the performance of distributed hash tables under churn. In *Proceedings of the 2nd Bertinoto Workshop on Future Directions in Distributed Computing (FuDiCo II): Survivability: Obstacles and Solutions, Bertinoro, Italy*, pages 87–99. Citeseer.

Li, J., Stribling, J., Morris, R., and Kaashoek, M. F. (2005). Bandwidth-efficient management of DHT routing tables. In *Proc. 2nd Symposium on Networked Systems Design and Implementation*.

Manku, G. S., Bawa, M., and Raghavan, P. (2003). Symphony: Distributed Hashing in a Small World. In *USENIX Symposium on Internet Technologies and Systems (USITS)*. Stanford InfoLab.

Maymounkov, P. and Mazières, D. (2002). Kademlia: A Peer-to-peer Information System Based on the XOR Metric. *Proceedings of the IPTPS02*, 1:53 – 65.

Oppenheimer, D., Vatkovskiy, V., and Patterson, D. A. (2004). Towards a framework for automated robustness evaluation of distributed services. In *Proceedings of the 2nd Bertinoto Workshop on Future Directions in Distributed Computing (FuDiCo II): Survivability: Obstacles and Solutions, Bertinoro, Italy*. Citeseer.

Oppenheimer, D., Vatkovskiy, V., Weatherspoon, H., Lee, J., Patterson, D. A., and Kubiatowicz, J. (2003). Monitoring, Analyzing, and Controlling Internet-scale Systems with ACME. Technical report, UC Berkeley Technical Report UCB-CSD-03-1276.

Overlay Weaver. `http://overlayweaver.sourceforge.net`. Acessado em 27 de março de 2010.

p2psim. `http://pdos.csail.mit.edu/p2psim/`. Acessado em 27 de março de 2010.

Plaxton, C. G., Rajaraman, R., and Richa, A. W. (1999). Accessing Nearby Copies of Replicated Objects in a Distributed Environment. *Theory of Computing Systems*, 32(3):241–280.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. (2001). A Scalable Content-Addressable Network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161 – 172. ACM.

Rhea, S., Geels, D., Roscoe, T., and Kubiatowicz, J. (2004). Handling Churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference*, pages 127–140.

Rhea, S. C., Roscoe, T., and Kubiatowicz, J. (2003). Structured peer-to-peer overlays need application-driven benchmarks. *Lecture notes in computer science*, pages 56–67.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, volume 11, pages 329 – 350. Citeseer.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149 – 160. ACM.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. D. (2004). Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on selected areas in communications*, 22(1):41 – 53.

Avaliação de redes P2P baseadas no fenômeno *Small World* para o compartilhamento de conteúdos similares gerados por funções LSH

Rodolfo S. Villaça[1], Luciano B. de Paula[2], Maurício F. Magalhães[2], Fábio Luciano Verdi[3]

[1] Departamento de Engenharias e Computação (DECOM/CEUNES)
Universidade Federal do Espírito Santo (UFES) – São Mateus, ES – Brasil

[2]Departamento de Engenharia de Computação e Automação Industrial (DCA/FEEC)
Universidade Estadual de Campinas (UNICAMP) – Campinas, SP – Brasil

[3]Universidade Federal de São Carlos (UFSCAR) – Sorocaba, SP – Brasil

rodolfovillaca@ceunes.ufes.br, {luciano,mauricio}@dca.fee.unicamp.br
verdi@ufscar.br

Abstract. *The organization of content sharing P2P networks greatly influences how to search and discover contents. Among several works in this area, we highlight the use of LSH (Locality Sensitive Hash) functions in the creation of keys used to index semantically similar contents. When it comes to structured P2P networks, such as Chord, the use of these functions creates an undesirable imbalance among peers participating in the network. This paper proposes an Overlay network exploiting the Small World phenomenon to share semantically similar contents. Using Oversim, semantically similar contents are hashed with LSH and MD5 functions and distributed in a Chord network and in a Small World based network, and the results compare the number of hops needed to recover similar contents, showing that the latter network is more appropriate in this scenario.*

Resumo. *A organização de uma rede P2P influencia na busca e recuperação de conteúdos. Dentre muitos trabalhos nesta área destacamos o uso de funções LSH para a criação de chaves utilizadas na indexação de conteúdos similares. Em redes P2P estruturadas, como Chord, o uso destas funções gera um desbalanceamento na distribuição das chaves entre os nós participantes da rede. Este trabalho propõe uma rede Overlay construída segundo o fenômeno Small World para o compartilhamento de conteúdos semanticamente semelhantes. Utilizando o Oversim, conteúdos semanticamente semelhantes são indexados através de funções LSH e MD5 e distribuídos em uma rede Chord e em uma rede Small World, e os resultados comparam o número de saltos (hops) necessários à sua recuperação, mostrando que essa última é mais adequada neste cenário.*

1. Introdução

Dentre as principais causas do declínio na utilização de redes P2P [Labovitz et al. 2009] podemos citar as dificuldades encontradas pelos usuários na busca e recuperação de conteúdo de seu interesse, frequentemente espalhados em grandes áreas e em muitos nós.

As redes P2P organizadas sob a forma de DHTs (*Distributed Hash Tables*) facilitam a recuperação de conteúdo, com a desvantagem de haver um custo associado à manutenção dessa organização. Além disso, geralmente os conteúdos indexados nestas DHTs através de funções de *hash* tradicionais, como o MD5, não mantêm relação de similaridade entre si, dificultando a busca por conteúdos semelhantes.

As funções de *hash* sensíveis à localidade, ou LSH (*Locality Sensitive Hash*), procuram manter a proximidade entre os identificadores gerados para conteúdos similares e, se aplicadas em DHTs, estas funções podem facilitar a busca inexata. Para exemplificar, podemos citar [Zhu 2005] que utiliza funções LSH para indexação de textos e [Haghani et al. 2009] para imagens, ambos aplicadas em DHTs. Nesses dois trabalhos, textos e imagens similares são indexados e mantidos próximos, possibilitando a busca por um grupo de elementos similares. Neste sentido utilizamos uma função LSH com similaridade baseada em ontologias para gerar identificadores de conteúdos semanticamente relacionados. Os resultados mostram que a função LSH utilizada concentra as chaves geradas em uma região do espaço de identificadores comparando os resultados com o espalhamento gerado por uma função de *hash* tradicional, nesse caso o MD5.

Entretanto, essa distribuição não-uniforme das chaves geradas por funções LSH leva a um desbalanceamento do espaço de identificadores. Com o objetivo de propor e avaliar uma forma adequada para a recuperação de um conjunto desbalanceado de identificadores gerados por conteúdos similares através da função LSH mencionada anteriormente, este trabalho propõe a construção de redes P2P baseadas no fenômeno *Small World*.

O fenômeno *Small World* [Kleinberg 2000] aplicado em redes P2P privilegia o estabelecimento de contatos (*fingers*) próximos em relação ao estabelecimento de contatos distantes, onde essa noção de proximidade está relacionada à distância lógica entre dois pontos no espaço de identificadores. Estas redes exibem ao mesmo tempo características de *clusterização* e baixo número de saltos no roteamento entre dois pontos quaisquer da rede. Tais características fazem com que estas redes sejam candidatas naturais para a distribuição de identificadores gerados por funções LSH. Em redes como o *Chord* o estabelecimento de contatos não possui relação com a similaridade entre os conteúdos.

Inspirado em [Girdzijauskas 2009], este trabalho propõe alterações no algoritmo de estabelecimento de contatos do *Chord* com o objetivo de construir uma DHT que possua as características do modelo *Small World*. Através do software de simulação de redes P2P *Oversim*[1] implementamos uma rede com estas características. O resultado mostra que o número de saltos necessários para a recuperação de conteúdos similares em uma rede *Small World* é menor quando comparado com uma DHT *Chord*. Esta redução acontece tanto para conteúdos indexados com uma função LSH quanto para conteúdos indexados com funções de *hash* tradicionais, como o MD5, devido à garantia de existência de poucos saltos entre quaisquer dois pontos de uma rede *Small World*. Entretanto, o melhor resultado e o menor número de saltos acontece com a utilização do conjunto LSH e *Small World*. Tais resultados mostram que é possível facilitar a busca inexata por conteúdos similares em uma rede P2P estruturada através do uso de redes *Small World* e funções LSH, sendo esta a principal contribuição deste trabalho.

[1]The Oversim P2P Simulator - http://www.oversim.org/

Este artigo está organizado da seguinte maneira: na Seção 2 há uma apresentação dos conceitos principais e trabalhos relacionados envolvendo redes P2P e do modelo *Small World*. A Seção 4 apresenta um exemplo de função LSH com similaridade baseada em ontologias, responsável pela geração dos identificadores de conteúdo utilizados neste trabalho. Em seguida, na Seção 3, estão detalhadas as modificações no algoritmo de estabelecimento de contatos do *Chord* de tal forma a montar uma rede que se assemelhe ao modelo *Small World*. Na seção seguinte (Seção 5), a metodologia de testes utilizada neste trabalho é descrita, junto com os resultados obtidos da comparação entre o *Chord* e a rede *Small World*. Por fim, na Seção 6, apresentamos as conclusões finais e perspectivas de trabalhos futuros.

2. Redes P2P e o Modelo *Small World*

Esta seção descreve um modelo de referência para redes P2P e o modelo *Small World* aplicado a essas redes. A proposta desse trabalho visa as redes P2P descentralizadas, porém estruturadas. Nesse tipo de rede P2P, o roteamento segue alguma regra baseada na topologia, que pode ser organizada em anel (*Chord*), árvore (*KAD*), *plaxon-style mesh* (*Pastry*) ou em um plano cartesiano de várias dimensões (*CAN*) [Lua et al. 2005].

2.1. Conceitos básicos

Podemos descrever uma rede P2P como sendo um conjunto de nós $\mathcal{P}$ que provê acesso a um conjunto de recursos (conteúdos) $\mathcal{R}$ através de uma função que mapeie $\mathcal{P}$ e $\mathcal{R}$ em um espaço virtual de identificadores $\mathcal{I}$ utilizando-se de duas funções $\mathcal{F}_{\mathcal{P}} : \mathcal{P} \rightarrow \mathcal{I}$ e $\mathcal{F}_{\mathcal{R}} : \mathcal{R} \rightarrow \mathcal{I}$. Estas funções permitem o estabelecimento de relações de proximidade entre nós e recursos através de métricas no espaço virtual de identificadores $\mathcal{I}$. Para permitir o acesso aos nós, e consequentemente aos recursos compartilhados em uma rede P2P, torna-se necessário embutir um grafo no espaço de identificadores proposto $\mathcal{I}$. Este grafo auxilia o roteamento, facilitando a recuperação de conteúdos associados às suas chaves (identificadores) na rede.

Um projeto de redes P2P deve considerar os seguintes aspectos:

- a escolha do espaço de identificadores $\mathcal{I}$;
- a escolha das funções de mapeamento $\mathcal{F}_{\mathcal{P}}$ e $\mathcal{F}_{\mathcal{R}}$;
- associação de pertinência entre os nós e os recursos;
- o grafo embutido no espaço de identificadores;
- a estratégia de roteamento;
- a manutenção da rede em função da entrada e saída de nós.

O atual trabalho não aborda as questões relacionadas à manutenção da rede e utiliza uma organização dos nós em anel, como no *Chord*.

2.1.1. Espaço de Identificadores e Roteamento

Uma característica fundamental na escolha do espaço de identificadores é a existência de alguma métrica de proximidade d: $\mathcal{I}\ x\ \mathcal{I} \rightarrow$ R, onde R denota o conjunto dos números reais. Essa métrica d deve satisfazer as propriedades 1, 2 e 3, apresentadas a seguir, e se possível também as propriedades 4 e 5.

$\forall x,y \in \mathcal{I} : d(x,y) \geq 0$ (Propriedade 1)

$\forall x \in \mathcal{I}$: *d(x,x)* = 0 (Propriedade 2)

$\forall x,y \in \mathcal{I}$: *d(x,y)* = 0 $\rightarrow$ *x=y* (Propriedade 3)

$\forall x,y \in \mathcal{I}$: *d(x,y)* = *d(y,x)* (Propriedade 4)

$\forall x,y,z \in \mathcal{I}$: *d(x,z)* $\leq$ *d(x,y)* + *d(y,z)* (Propriedade 5)

A escolha adequada do espaço de identificadores dos nós e recursos da rede é uma questão relevante pois, a partir desta escolha, dependerão fatores como escalabilidade, separação de identificadores e localizadores, roteamento, *clusterização* e manutenção da semântica dos conteúdos mapeados neste espaço. Estes três últimos são os fatores principais no desenvolvimento da proposta deste trabalho, e todos são diretamente dependentes desta métrica de distância *d*.

2.1.2. Funções de Mapeamento $\mathcal{F}_\mathcal{P}$ e $\mathcal{F}_\mathcal{R}$

A função de mapeamento $\mathcal{F}_\mathcal{P}$ associa um nó a um ponto no espaço $\mathcal{I}$. Na maioria dos casos este ponto é único, tal que $\forall p,q \in \mathcal{R} : p \neq q \rightarrow \mathcal{F}_\mathcal{P}(p) \neq \mathcal{F}_\mathcal{P}(q)$. Entretanto, em casos de replicação de conteúdo nos nós como uma solução de tolerância a falhas, admite-se o relaxamento desta condição. Essa mesma condição pode ser aplicada à função $\mathcal{F}_\mathcal{R}$, associando um recurso na rede a um ponto no espaço de identificadores. Neste caso o relaxamento desta condição representa a probabilidade de conteúdos similares ocuparem o mesmo ponto no espaço de identificadores.

A probabilidade de um nó ou recurso ser mapeado em um determinado ponto ou região do espaço virtual $\mathcal{I}$ também é uma característica importante na escolha das funções $\mathcal{F}_\mathcal{P}$ e $\mathcal{F}_\mathcal{R}$. Essa distribuição pode ser homogênea (probabilidades iguais) ou não-homogênea (probabilidades diferentes). Funções de *hash* como MD5 e SHA-1 possuem distribuição com probabilidades iguais, enquanto que funções LSH, como a que será apresentada na seção 4, possui distribuição com probabilidades diferentes pois a mesma possui a característica de manter a similaridade dos recursos $\mathcal{R}$ mapeados em $\mathcal{I}$. A função proposta neste artigo deverá satisfazer às relações 6 e 7 apresentadas a seguir, onde *sim* é a similaridade:

$\forall\ s_1,s_2 \in \mathcal{R}$: *d*($\mathcal{F}_\mathcal{R}(s_1)$,$\mathcal{F}_\mathcal{R}(s_2)$) $\propto$ *sim*(s_1,s_2) (6)

$\forall\ s_1,s_2 \in \mathcal{R}$: *P*[$\mathcal{F}_\mathcal{R}(s_1)$ = $\mathcal{F}_\mathcal{R}(s_2)$] = *sim*($s_1$,$s_2$) (7)

Em (6) e (7), definimos uma função de similaridade *sim(a,b)* : $\mathcal{R}$ *x* $\mathcal{R}$ $\rightarrow$ [0..1], onde 0 significa nenhuma similaridade, e 1 significa similaridade total. A relação (6) implica que a distância entre dois conteúdos quaisquer (s_1,s_2) é proporcional à similaridade entre eles, ou seja, quanto mais similares, mais próximos no espaço de endereçamento eles se encontram, e vice-versa. A relação (7) indica que a probabilidade de que ambos tenham um mesmo identificador é igual à similaridade entre eles, ou seja, quanto maior a similaridade, maior a probabilidade de possuírem o mesmo identificador, e vice-versa.

2.1.3. Roteamento em Redes P2P estruturadas

O serviço básico provido por uma rede P2P estruturada é encaminhar a requisição de um identificador *i* ($i \in \mathcal{I}$) a partir de um nó *p* ($p \in \mathcal{P}$). A estratégia de roteamento influencia no número de saltos necessários no encaminhamento da requisição entre *p* e *i* e pode ser definida como uma função não determinística $\mathcal{R} : \mathcal{P}\ x\ \mathcal{I} \rightarrow \mathcal{P}$ que seleciona, no conjunto

$\mathcal{N}$ de nós vizinhos a p ($\mathcal{N}_\mathcal{P}$), um outro nó r ($r \in \mathcal{N}_\mathcal{P}$) para encaminhar a requisição. O nó escolhido deverá satisfazer a seguinte relação: $d(i,\mathcal{F}_\mathcal{P}(r)) < d(i,\mathcal{F}_\mathcal{P}(p))$ (8)

2.2. O fenômeno *Small World*

As primeiras pesquisas envolvendo as características deste fenômeno, popularmente conhecido como "os seis graus de separação", pertencem à década de 60. Desde então, muitos outros trabalhos se dedicaram a explicar e modelar este fenômeno através de grafos e algoritmos. Dentre estes trabalhos destacam-se [Watts and Strogatz 1998] e [Kleinberg 2000].

Podemos afirmar que uma população possui as características do fenômeno *Small World* se, para quaisquer pares de indivíduos pertencentes a esta população, houver um caminho curto que interligue ambos através de uma sequência de outros indivíduos vizinhos que possuam algum conhecimento em comum. Em uma rede, esse "conhecimento em comum" pode ser traduzido em uma agregação, fazendo com que conteúdos que possuam características similares estejam armazenados próximos entre si, sendo esta proximidade medida no espaço virtual de identificadores.

[Watts and Strogatz 1998] propuseram um modelo para a construção de redes baseadas no fenômeno *Small World* onde o grafo de construção destas redes possui vizinhos divididos entre dois tipos: locais e de longa distância. Em seu trabalho, supondo um conjunto $\mathcal{I}$ de pontos pertencentes ao espaço virtual de identificadores, para cada nó $p \in \mathcal{P}$ mapeado no espaço virtual $\mathcal{I}$, interliga-se o mesmo com k vizinhos próximos, aleatoriamente escolhidos segundo alguma métrica de proximidade d. Estes são os vizinhos locais de p. Feito isso, escolhe-se aleatoriamente, um pequeno número de nós distantes. Estes são os vizinhos de longa distância de p.

[Kleinberg 2000] demonstrou que dentre uma grande variedade de redes *Small World* existe uma forma de construção onde o roteamento é mais eficiente e o número de nós intermediários que interligam dois pontos quaisquer da rede é mínimo. A principal contribuição deste trabalho é mostrar que o estabelecimento dos vizinhos não deve ser realizado de forma puramente aleatória, mas sim de forma inversamente proporcional à distância, conforme veremos a seguir.

Na proposta de Watts e Strogatz existe um número fixo k de vizinhos locais, e um pequeno número de vizinhos de longa distância, enquanto Kleinberg relaxa a necessidade desse valor fixo. Kleinberg estabelece que um nó p possui contatos curtos com seus vizinhos adjacentes e contatos de longa distância com nós pertencentes à rede segundo uma probabilidade inversamente proporcional a $d(p,q)^r$, onde r é um parametro estrutural e q é um nó qualquer pertencente à rede. Para pequenos valores de r o grafo formado é puramente aleatório (*Random Networks*), enquanto que a medida que r aumenta, o grafo torna-se cada vez mais clusterizado. Em nenhum destes casos extremos a rede apresentará as características de *Small World* desejadas.

Os resultados apresentados em nosso trabalho baseiam-se na construção de redes *Small World* proposta por [Girdzijauskas 2009], [Girdzijauskas et al. 2010]. Nela, o espaço virtual é organizado em um anel, e cada nó u possui contato direto com seus vizinhos adjacentes, à direita, e à esquerda. A principal diferença entre esta proposta e a proposta de Kleinberg é que cada nó deverá possuir $\log_2 N$ contatos, onde N representa o número de nós na rede. [Girdzijauskas 2009] mostra que esta construção possui um custo

máximo de $O(\log_2(N))$ saltos, além de mostrar um algoritmo de descoberta que permite que seja realizada uma estimativa do valor de *N*.

3. Construção e simulação de uma Rede P2P *Small World*

Esta seção descreve uma maneira de modificar o algoritmo de aquisição de contatos (*fingers*) do *Chord* de tal forma a construir um grafo que possua as características do modelo *Small World*. A rede P2P gerada a partir destas modificações possui as seguintes características:

- espaço virtual formado por identificadores binários de *k* bits;
- identificadores dispostos em um anel, com roteamento no sentido horário;
- distribuição balanceada dos nós, cujos identificadores são gerados através de funções *hash* comuns;

O algoritmo de roteamento guloso (*greedy routing*) baseado na distância entre identificadores utilizado no *Chord* é mantido, assim como a forma de associação entre um nó e os identificadores pelos quais ele é responsável no anel (em resumo, um nó é responsável por todos os identificadores localizados na faixa compreendida por ele mesmo e seu antecessor no anel). Os testes descritos na Seção 5 foram realizados somente em uma rede estática, após a conclusão do procedimento de entrada de todos os nós. Além disso, fixamos o número de nós em cada teste de tal forma que não houve necessidade de adotar nenhum procedimento para estimar este valor.

3.1. Espaço virtual de identificadores

Conforme dito anteriormente, o espaço de identificadores é binário e apresenta uma organização circular. Considerando-se um espaço de *k* bits, admite-se identificadores na faixa compreendida entre 0 e 2^k no domínio dos números naturais. Considere também que a rede seja composta por *N* nós, e estes sejam dispostos aleatoriamente neste espaço segundo uma função de distribuição de probabilidade uniforme.

Desta forma, dividindo o espaço $\mathcal{I}$ em $\log_2 N$ partições logarítmicas, conforme proposto em [Girdzijauskas 2009], define-se as partições A_1, que contêm aproximadamente a metade da população $\mathcal{P}$ de nós da rede, correspondendo aos nós cujos identificadores pertencem ao intervalo $[2^{k-1}..2^k)$; A_2, que contém aproximadamente $\frac{1}{4}$ da população $\mathcal{P}$ de nós da rede, correspondendo aos nós cujos identificadores pertencem ao intervalo $[2^{k-2}..2^{k-1})$; e assim, sucessivamente.

A Figura 1 mostra um exemplo contendo um espaço de endereçamento de k = 4 bits com 8 nós. É importante ressaltar, observando a mesma figura, que a partição A_3 contém somente um nó, correspondente ao sucessor e ao antecessor de *p* no espaço $\mathcal{I}$, considerando ambos os sentidos no anel circular onde os identificadores foram distribuídos.

Esta divisão é importante pois dela extrairemos a função de probabilidade usada para estabelecimento dos contatos de cada nó na rede proposta. Supondo um nó *u* com identificador $\mathcal{F}_{\mathcal{P}}(u)$, o objetivo é fazer com que contatos pertencentes às partições mais próximas estejam em maior número na tabela de roteamento deste nó. É importante esclarecer que não se trata de privilegiar uma região em detrimento de outra, trata-se apenas de privilegiar os nós que estão mais próximos em $\mathcal{I}$ em relação aos nós mais distantes.

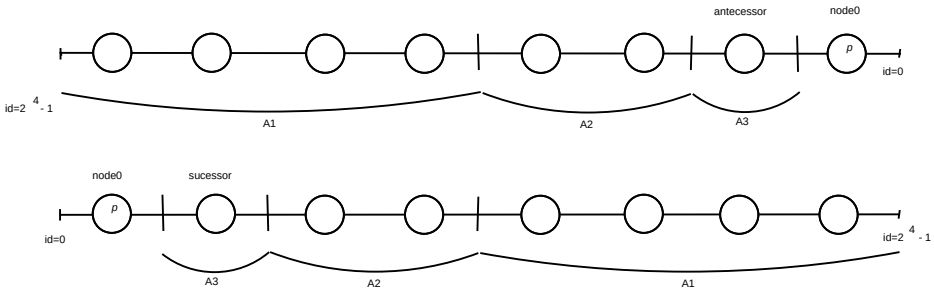


Figura 1. Espaço $\mathcal{I}$ com 8 nós e identificadores de *k*=8 bits

A maneira encontrada para realizar a implementação desse procedimento está diretamente relacionada com a divisão do espaço em partições. Com a mesma probabilidade (P=$1/\log_2 N$), sorteia-se uma dentre as $\log_2 N$ possíveis partições. O resultado deste sorteio determinará qual partição receberá uma solicitação de estabelecimento de contato para inserção na tabela de roteamento de u. Após a determinação de qual partição terá uma entrada na tabela de roteamento de u, sorteia-se novamente, com probabilidade igual (P=$1/\frac{N}{2^n}$), um identificador i pertencente à partição sorteada (relação (8)). Segue-se normalmente o procedimento adotado pelo *Chord* para o estabelecimento da ligação. Já foi dito na Seção 2.2 que este procedimento garante que o custo máximo do roteamento nesta rede, em termos do número de saltos, é $O(\log_2(N))$.

Todo o procedimento descrito no parágrafo anterior está perfeitamente adequado para um nó u com identificador 0, ou seja, localizado no início do espaço de endereçamento. No caso de um nó u qualquer, com identificador $\mathcal{F}_{\mathcal{P}}(u) \neq 0$ basta utilizar o valor sorteado como um deslocamento δ a ser adicionado ou subtraído ao identificador do nó (u), de tal forma que o nó v a ser inserido na tabela de roteamento de u possuirá identificador $\mathcal{F}_{\mathcal{P}}(v) = \mathcal{F}_{\mathcal{P}}(u) \pm \delta$. Isto levará em conta a proximidade à direita e à esquerda no espaço circular com probabilidade igual pois selecionaremos, aleatoriamente e com probabilidades iguais, o sinal do deslocamento δ durante a rotina de preenchimento da tabela de roteamento de cada nó.

3.2. Roteamento dos identificadores

O algoritmo de roteamento usado na rede *Small World* proposto neste trabalho baseia-se no roteamento guloso (*greedy routing*) usado no *Chord*, onde a cada encaminhamento aproxima-se cada vez mais do destino procurado. Essa aproximação é feita sempre levando-se em consideração o sentido horário e a distância em um espaço de identificadores circular. A Figura 2 mostra um exemplo de uma rede com k = 4 bits e 8 nós construída segundo a proposta deste trabalho. Na figura podemos observar os contatos locais de um nó u qualquer, representados pelo seu antecessor e sucessor no espaço $\mathcal{I}$ e três contatos de longa distância, representados por f_1, f_2 e f_3.

A partir da Figura 2 é possível explicar as diferenças conceituais existentes entre o roteamento no *Chord* e o roteamento na rede *Small World*. No *Chord*, embora exista a proximidade entre identificadores no sentido anti-horário do espaço virtual, esta proximidade não é importante para a escolha dos nós a serem adicionados na tabela de roteamento.

De acordo com a Figura 2, no *Chord* podemos interpretar os contatos f_1 e f_2 e o antecessor do nó u sendo contatos distantes, enquanto que o sucessor e o contato f_3

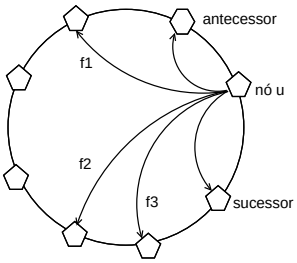


Figura 2. Contatos locais e de longa distância de um nó *u* em $\mathcal{I}$, *k*=4 e *N*=8

como sendo contatos próximos ao nó *u*. No *Chord*, o estabelecimento dos contatos é feito utilizando-se um algoritmo que considera somente o tamanho do anel e a distância no sentido horário, enquanto na rede baseada em *Small World* o estabelecimento de contatos é feito levando-se em consideração a proximidade dos nós no anel virtual em ambos os sentidos. Quanto mais próximos, maior a chance de dois nós estabelecerem contatos entre si.

A próxima seção discute um exemplo de construção de funções LSH cuja similaridade é baseada em ontologias para classificação de conteúdos.

4. Funções LSH com Similaridade Baseada em Ontologias

Neste trabalho estamos interessados não somente em avaliar o comportamento de uma rede P2P estruturada segundo o fenômeno *Small World*, comparando-a com o *Chord*, mas também estamos interessados em avaliar o comportamento destas redes em situações geradas pela agregação dos identificadores segundo a semântica dos conteúdos neles representados.

Neste contexto, uma das formas de se classificar dados, dando-lhes significado, é feita através da utilização de ontologias. Essa prática tem crescido recentemente devido à expansão da Web Semântica e do número de aplicações que consomem dados semanticamente classificados.

No escopo desse trabalho, uma ontologia é um modelo de dados que define um domínio e que pode ser usada para raciocinar sobre conceitos ou termos daquele domínio, estabelecendo relações entre eles [Uschold and Gruninger 2004]. Geralmente são utilizadas pequenas ontologias, com dúzias ou centenas de conceitos, para a definição de um domínio. Estas ontologias possuem menos expressividade mas são de fácil gerência. É possível utilizar essa classificação ontológica como base para uma função de similaridade e, a partir daí, criar uma família de funções LSH, utilizadas na criação de identificadores que mantenham essa relação entre si.

Em [Indyk and Motwani 1998] é definido que uma família de funções de *hash F* é classificada como sensível à localidade (LSH) correspondente a uma função de similaridade *sim(a, b)* se, para toda função $h \in F$, temos: $\mathbf{Pr}_{h \in F}[h(a) = h(b)] = sim(a, b)$, onde **Pr** é a probabilidade e *sim(a, b)* é uma função de similaridade que varia entre 0 e 1, sendo 1 para *a* e *b* completamente relacionados e 0 caso contrário.

Para a criação de uma função LSH baseada na similaridade extraída de uma ontologia, em primeiro lugar é utilizada a técnica *Enhanced Topic-based Vector Space Model* (eTVSM) [Polyvyanyy 2007] para obter a similaridade dos conceitos de uma ontologia

simples (também conhecida como ontologia *lightweight*). Para cada conceito da ontologia, o eTVSM gera um vetor que relaciona o mesmo com os demais termos.

Para todo conceito folha de uma ontologia, um vetor de conceito $\overrightarrow{\tau_i}$ = $\{\tau_{i,1}, \tau_{i,2}, ..., \tau_{i,k}\}$, onde *k* é o número total de conceitos da ontologia, é criado da seguinte maneira:

$$\tau_{i,k} = \begin{cases} 1 & \text{se } \tau_k \in \text{ ao caminho de } \tau_i \text{ até a raiz ou } i = k \\ 0 & \text{caso contrário} \end{cases}$$

Para todo conceito interno temos:

$$\overrightarrow{\tau_i} = |\sum \overrightarrow{\tau_s}|$$

onde $\overrightarrow{\tau_s}$ são todos os vetores de conceitos dos filhos diretos de τ_i. A similaridade entre cada par de conceitos pode ser medida pelo cálculo do cosseno entre os seus respectivos vetores, quanto maior o valor do cosseno, mais similares eles são.

Em seguida aplica-se a função *Random Hyperplane Hash* (RHH) [Charikar 2002] compondo um identificador de *n* bits no espaço $\mathcal{I}$. O RHH, uma família de funções LSH que correspondem à similaridade de cosseno, consiste em criar um vetor $\overrightarrow{r}$, sendo cada coordenada deste vetor gerada a partir de uma distribuição normal e, para cada vetor de conceito, contabiliza-se o produto escalar entre eles. Se o resultado do produto escalar for igual ou maior que 0, o resultado do *hash* é 1, caso contrário, 0. Para uma chave de *n* bits, são criados *n* vetores $\overrightarrow{r}$ e os resultados são concatenados.

A partir deste ponto, o identificador gerado pela aplicação deste método será utilizado como chave semântica daquele conceito da ontologia. Para uma maior ou menor agregação é possível criar *m* grupos de *n* vetores $\overrightarrow{r}$ e, pelo princípio da similaridade de Jaccard, escolhe-se o menor dentre os identificadores gerados como chave do conceito.

5. Testes e Resultados

Nos testes realizados com a função LSH com similaridade baseada em ontologias, comparamos a capacidade de agregação das chaves no espaço de identificadores com o espalhamento ocasionado pela geração de chaves através de uma função de hash tradicional, no caso o MD5. Esta função LSH foi implementada em linguagem Java.

As modificações propostas na Seção 3.2 foram implementadas no simulador *Oversim*, o qual usa a plataforma *Omnet++* e possibilita a simulação de grandes números de nós em uma única estação de trabalho PC usando o sistema operacional Linux. A implementação foi feita através de modificações realizadas na implementação do *Chord* disponível no Oversim, alterando-se as rotinas de aquisição de contatos e roteamento.

Também foram realizadas modificações na aplicação de busca e recuperação de chaves disponibilizadas no próprio *Oversim*, denominada *KBRTestApp*. As modificações feitas permitem realizar buscas por chaves específicas, determinadas pelo usuário.

Os testes feitos possuem dois objetivos principais:

1. Apresentar o comportamento da função LSH com similaridade baseada em ontologias, verificando a agregação dos conteúdos similares em uma determinada área do espaço de identificadores. A análise será feita em função do tamanho desta área, comparada à área utilizada por uma função de *hash* MD5, aplicada sobre os mesmos conteúdos;

2. Avaliar a implementação da rede *Small World* proposta neste artigo através da realização de buscas por chaves criadas por uma função LSH (agregadas no espaço dos identificadores) e por chaves criadas por um *hash* tradicional (MD5), medindo-se o número de saltos necessários para que estas buscas sejam concluídas. Essas buscas também são realizadas no *Chord* e os resultados são comparados.

O resultado esperado no primeiro teste é que, com o uso de uma função LSH, os identificadores sejam confinados em uma mesma região do espaço, enquanto que usando-se uma função *hash* MD5 estes mesmos identificadores encontrem-se distribuídos uniformemente no espaço.

Para o segundo teste espera-se que o custo, medido no número de saltos, para recuperação dos conteúdos gerados pela função LSH seja menor em uma DHT baseada no modelo *Small World* do que no *Chord*. Isto deve-se ao fato de que no *Small World* estes conteúdos estarão confinados em uma mesma região, e os nós pertencentes a esta região terão contatos diretos entre si com uma maior probabilidade. Para as buscas por conteúdos gerados pelo MD5, é desejado que os resultados obtidos com a rede *Small World* sejam, no mínimo, próximos àqueles obtidos utilizando o *Chord*. Isto deve-se à característica de baixo número de saltos presente no fenômeno *Small World*.

A seguir descrevemos os cenários de teste e as ontologias utilizadas para classificação do conteúdo. Na realização dos testes o seguinte cenário foi padronizado:

- Uma ontologia *lightweight* organizada sob a forma de árvore binária de 127 tópicos *t*, representando a classificação de algum conteúdo não especificado;
- Adoção de um espaço de identificadores de 128 bits e *N* variando em 1000, 2000, 5000 e 10000 nós, com uma distribuição aleatória e uniforme neste espaço;
- Comportamento estático das DHTs, de tal forma que durante as simulações não haveria entrada ou saída de nós.

A ontologia usada é representada por uma árvore binária de 7 níveis e 127 tópicos. A classificação segundo uma ontologia possui a granularidade de um conceito, ou seja, um conjunto de dados que possam ser classificados como similares e deveriam estar localizados próximos no espaço de identificadores. Assume-se, também, que todos os dados classificados segundo uma mesma ontologia, mas em conceitos distintos, possuem alguma relação entre si e, portanto, estarão próximos no espaço de identificadores.

5.1. Agregação de conteúdo usando LSH e ontologias para classificação de conteúdos

Utilizando a ontologia descrita na seção anterior são gerados 127 identificadores, cada um deles representando um tópico t_x através da função LSH descrita na Seção 4. Outros 127 identificadores foram gerados utilizando-se uma função *hash* MD5 no nome de cada tópico. De forma aleatória e uniforme distribuimos 1000, 2000, 5000 e 10000 nós no espaço de identificadores.

O resultado dos testes mostra o intervalo de armazenamento dos conceitos classificados segundo a ontologia para os dois conjuntos (LSH e MD5) de 127 identificadores. Esse intervalo é calculado pelo número de nós existentes entre o primeiro que tenha

chaves armazenadas e o último, seguindo o sentido horário no anel. Quanto menor o intervalo, mais próximos os conceitos estarão uns dos outros.

A Figura 3 mostra que o conteúdo indexado com a função LSH foi agregado no espaço de identificadores. Em uma rede com 1000 nós, a função de *hash* MD5 utilizou aproximadamente todos os nós na distribuição dos identificadores de conteúdo (989 nós), enquanto que a função LSH concentrou os identificadores em pouco mais de 25% dos nós (261). Resultados semelhantes podem ser observados na mesma figura em redes com 2000, 5000 e 10000 nós.

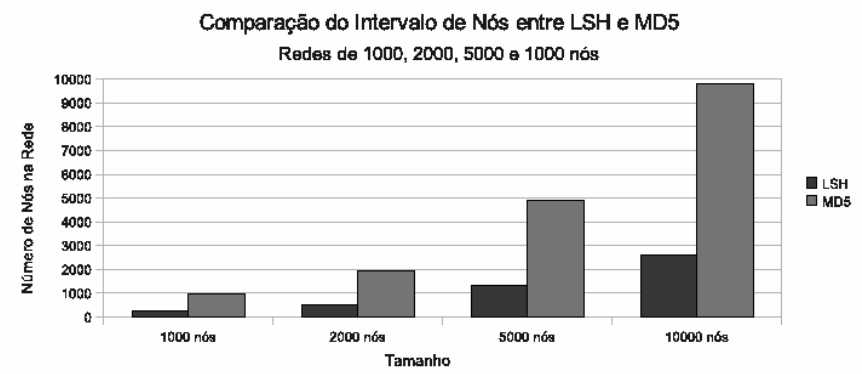


Figura 3. Intervalo de agregação para 1000, 2000, 5000 e 10000 nós

Como esperado, os resultados nos permitem afirmar que a função LSH utilizada mantêm a similaridade entre os identificadores gerados para conteúdos classificados segundo mesma ontologia, enquanto os identificadores criados pelo *hash* MD5 não possuem nenhuma agregação por similaridade, espalhando-se uniformemente no espaço.

Essa métrica de distância obedece a todas as propriedades de (1) a (5), pois mede o comprimento de uma região, o qual não pode ser negativo (1), pode ser nula se e somente se todo o conteúdo for agregado em um único identificador (2) e (3), é uma medida de distância obtida através da diferença entre o maior e o menor identificador (4) e, por ser unidimensional, necessariamente obedece à propriedade (5).

5.2. Comparações entre *Chord* e *Small World* quanto a busca de conteúdos similares

O segundo conjunto de testes realizados neste trabalho tem como objetivo verificar se houve redução no custo de recuperação de conteúdos similares na rede P2P *Small World*, comparando-se o custo de recuperação dos mesmos conteúdos em uma DHT *Chord*.

Nestes testes utilizamos o mesmo conjunto de identificadores gerados no teste anterior, criados pela função LSH e pelo *hash* MD5. O procedimento empregado na realização dos testes é descrito a seguir:

- Para cada tópico t_x ($x \in [1..127]$) da ontologia foram realizadas 127 buscas por cada um dos demais tópicos t_y ($y \in [1..127]$), incluindo o próprio t_x. Estas buscas eram originadas a partir do nó responsável pelo armazenamento de t_x;
- Ao alcançar o tópico de destino t_y, pertencente a algum nó da rede P2P, o número de saltos desde t_x até t_y foi armazenado em um arquivo de *log*;

Este procedimento foi repetido variando-se a função de *hash* (e consequentemente o conjunto de 127 identificadores), o modelo de rede utilizado (*Chord* e *Small World*) e o número de nós na rede P2P (1000, 2000, 5000 e 10000 nós). Em cada caso, calculamos a média e o seu respectivo intervalo de confiança, mediana, desvio padrão, máximo, mínimo. Os resultados obtidos para a média de saltos estão apresentados nas Figuras 4(a), 4(b), 4(c), 4(d).

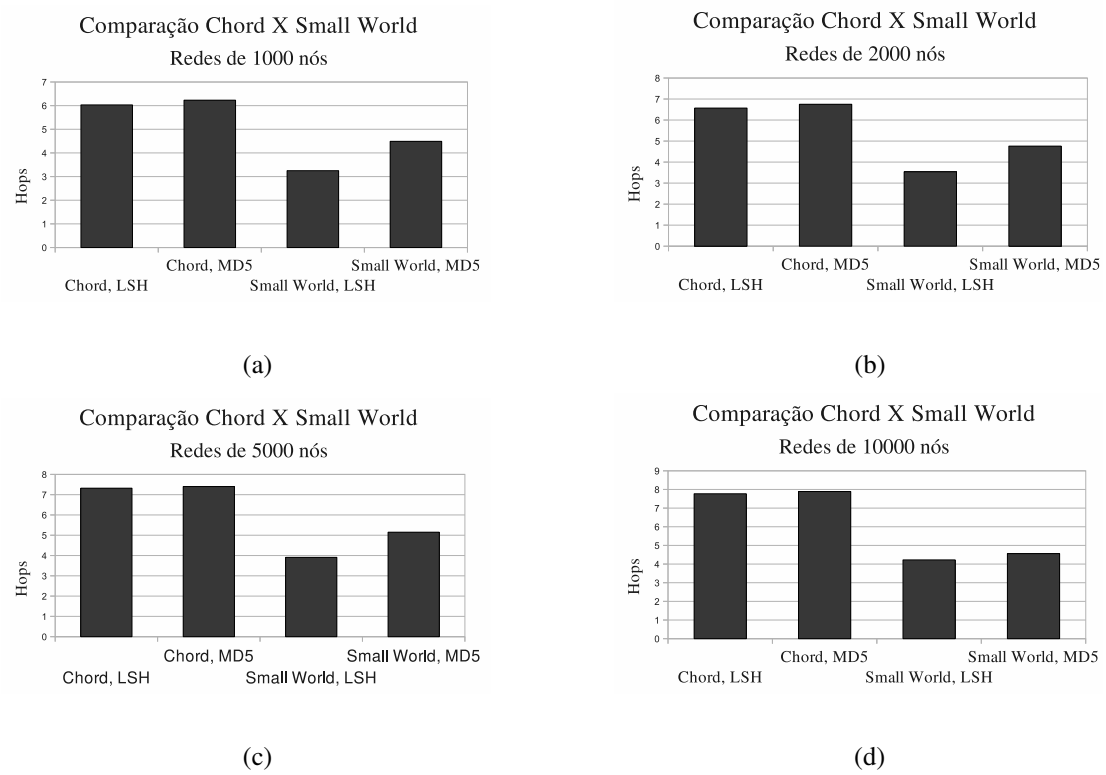


Figura 4. Comparação do número de saltos para redes de 1000(a), 2000(b), 5000(c) e 10000(d) nós

Em todos os testes, o número mínimo de saltos obtido foi 0, correspondendo às situações nas quais tópicos distintos estavam armazenados em um mesmo nó na rede. Em redes *Chord* de 1000 nós o número máximo obtido para o *hash* MD5 foi de 10 saltos, mesmo valor obtido com a função LSH. Usando *Small World* estes números caem para 8 e 6, respectivamente. Em redes *Chord* de 2000 nós o número máximo obtido para o *hash* MD5 foi de 11 saltos, mesmo valor obtido com a função LSH. Usando *Small World* estes números caem para 7 e 6, respectivamente. Em redes *Chord* de 5000 nós o número máximo obtido para o *hash* MD5 foi de 12 saltos, mesmo valor obtido com a função LSH. Usando *Small World* estes números caem para 8 e 7, respectivamente. Em redes *Chord* de 10000 nós o número máximo obtido para o *hash* MD5 foi de 13 saltos, mesmo valor obtido com a função LSH. Usando *Small World* estes números caem para 9 e 8, respectivamente.

Estes resultados mostram que o *Chord*, pela característica de estabelecimento de contatos somente no sentido horário, não aproveita adequadamente, na média, o ganho oferecido pela agregação dos identificadores gerados por uma função LSH. Isso acontece pois a agregação permite que dois identificadores estejam próximos segundo a métrica de distância utilizada porém, no sentido anti-horário do anel. Neste caso a agregação não terá

efeito na redução do número de saltos entre estes dois identificadores e a média tende a se aproximar do resultado obtido no MD5. Para comprovar esta hipótese, realizamos testes de tal forma que as buscas pelos identificadores dos conceitos na DHT *Chord* fossem realizadas sempre em ordem crescente, obedecendo o sentido horário no anel. Nesta situação específica comprovamos que houve uma redução no número médio de saltos no *Chord* usando-se uma função LSH, comparando com o resultado do MD5 nas mesmas condições. Estes resultados não são mostrados aqui por falta de espaço.

Observa-se também que a média é sempre menor utilizando-se o conjunto LSH e *Small World*, independente do número de nós da rede P2P e que, à medida em que a rede cresce, a média na rede *Chord* se distancia cada vez mais da média na rede *Small World*, tanto para a função de *hash* LSH quanto para o *hash* MD5, comprovando a característica de baixo número de saltos desta rede, independente do seu tamanho.

A medida do número de saltos também é válida e obedece às propriedades (1), (2), (3) e (5): independente do sentido, o número de saltos é sempre positivo (1) e possui valor 0 (nenhum salto) caso o identificador de destino esteja armazenado no próprio nó que origina a busca (2) e (3); em se tratando de um algoritmo de roteamento guloso, deve-se necessariamente obedecer a regra do triângulo em um espaço geométrico (5). Entretanto, pela assimetria no estabelecimento dos contatos, a regra (4) não é obedecida, comprovado principalmente nos resultados obtidos pelo *Chord*.

Como conclusão destes testes pode-se afirmar que houve uma redução no número de saltos necessários à recuperação de conteúdos similares utilizando-se uma rede *Small World* e funções LSH. A redução da média do número de saltos tende a ser maior com o aumento do número de nós na rede.

6. Conclusão

A primeira parte dos resultados obtidos neste trabalho mostra que a geração de identificadores utilizando uma função LSH, mantém a similaridade dos conteúdos classificados segundo uma ontologia, o que implica em um armazenamento mais próximo no espaço de endereçamento. Essa proximidade propicia uma redução no custo necessário para a recuperação de conteúdos similares, facilitando-se assim as buscas em redes P2P.

Para confirmar essa afirmação, o artigo propôs a criação de uma rede P2P que exiba as duas principais características do modelo *Small World*: 1) baixo número de saltos entre quaisquer dois pontos; 2) um pequeno grau de clusterização, sem que isso transforme a rede em um grafo regular. A rede proposta foi baseada em [Girdzijauskas 2009]. Os resultados obtidos compararam o número médio de saltos necessário para recuperação de conteúdos similares em uma rede P2P *Chord* e na rede P2P *Small World* proposta neste trabalho, confirmando-se a hipótese inicial de redução no custo de recuperação de conteúdos similares através do conjunto LSH e redes *Small World*.

Para realização de trabalhos futuros, pretendemos reconstruir a rede *Small World* segundo uma função de distribuição de probabilidade obtida através de dados extraídos em redes sociais utilizadas como referência. Isso nos permitirá que os contatos sejam estabelecidos não somente em função da distância, mas também em função da distribuição do conteúdo nesta rede, de tal forma que exista uma maior probabilidade de estabelecimento de contatos em regiões do espaço de identificadores que possuam conteúdos similares.

Além disso, pretendemos avaliar os resultados em outras formas de organização de redes estruturadas.

Nas simulações realizadas neste trabalho não foi considerada nenhuma dinâmica para representar a entrada e saída de nós na rede P2P. Sabemos que esta é uma limitação deste trabalho e uma característica importante presente nestas redes. Entretanto, o objetivo principal deste trabalho era a confirmação das hipóteses de agregação de identificadores de conteúdo gerados por uma função LSH com similaridade baseada em ontologias e a redução no custo de recuperação de conteúdos similares utilizado-se uma rede P2P *Small World*.

Referências

Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, New York, NY, USA. ACM.

Girdzijauskas, S. (2009). *Designing Peer-to-Peer Overlays: a Small-World Perspective*. PhD thesis, Ecole Polytechnique Federale de Lausanne (EPFL), Lausane, CH.

Girdzijauskas, v., Datta, A., and Aberer, K. (2010). Structured overlay for heterogeneous environments: Design and evaluation of oscar. *ACM Transactions on Autonomous and Adaptative Systems*, 5(1):1–25.

Haghani, P., Michel, S., and Aberer, K. (2009). Distributed similarity search in high dimensions using locality sensitive hashing. In *Proceedings of 12th International Conference on Extending Database Technology*.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA. ACM.

Kleinberg, J. (2000). The small-world phenomenon: an algorithm perspective. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170, New York, NY, USA. ACM.

Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J., Jahanian, F., and Karir, M. (2009). Internet Observatory 2009 Annual Report. Technical report, 47th North American Network Operators' Group (NANOG47), Dearborn, MI.

Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93.

Polyvyanyy, A. (2007). Evaluation of a Novel Information Retrieval Model: eTVSM. Master's thesis, Universitat of Potsdam, Hasso Plattner Institut, Potsdam, Deutschland.

Uschold, M. and Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *SIGMOD Rec.*, 33(4):58–64.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442.

Zhu, Y. (2005). *Enhancing Search Performance in Peer-to-Peer Networks*. PhD thesis, Computer Science and Engineering, University of Cincinnati.

VI Workshop de Redes Dinamicas e Sistemas Peer-to-Peer

♦

Sessão Técnica 3

Redes P2P em Redes Móveis

CDS-BitTorrent: Um Sistema de Disseminação de Conteúdo para a Melhoria do Desempenho de Aplicações BitTorrent sobre MANETs

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. *This paper introduces a content dissemination system, namely, CDS-BitTorrent. This system extends BitTorrent in order to improve download performance in MANETs that are composed by mobile devices. To improve performance, CDS-BitTorrent adopts specific strategies for selecting some content and disseminating it through a limited broadcast. Simulation results obtained by using NS-2 indicate that CDS-BitTorrent is able to reduce the download time by up to 30% and the overhead of segments by up to 50% when compared to the traditional BitTorrent approach and in accordance with the scenarios studied.*

Resumo. *Este artigo introduz o CDS-BitTorrent como um sistema de disseminação de conteúdo que estende o BitTorrent para a melhoria de seu desempenho de download em MANETs formadas por dispositivos móveis. Para alcançar tal objetivo, o CDS-BitTorrent adota estratégias específicas de seleção de parte do conteúdo e de disseminação dessa parte através de um broadcast controlado. Resultados de simulação com o NS-2 mostram que o CDS-BitTorrent permite uma redução de até 30% no tempo de download e uma redução de até 50% no overhead de segmentos enviados, quando comparado ao BitTorrent tradicional e de acordo com os cenários estudados.*

1. Introdução

As MANETs (*Mobile Ad Hoc Networks*) são redes móveis que possuem como características a autonomia dos nós, a auto-organização, o dinamismo da topologia, o roteamento em múltiplos saltos e a formação em caráter tipicamente temporário. Todas essas características também estão presentes na maioria das redes P2P (*Peer-to-Peer*). Tais semelhanças entre MANETs e redes P2P têm culminado em pesquisas que buscam explorar suas sinergias e melhorar o desempenho de aplicações P2P onde a adoção de redes sem fio se faz necessária [Hu et al. 2005].

Em particular às abordagens P2P existentes, o protocolo BitTorrent [Cohen 2006] tem chamado a atenção da comunidade de pesquisa por sua eficiência no compartilhamento de arquivos extensos na Internet. Trazer tal eficiência para ambientes de MANETs é um grande desafio de pesquisa que, por sua vez, vem incentivando estudos recentes que buscam soluções para prover um funcionamento eficiente do protocolo BitTorrent, considerando as características das MANETs e dos cenários de uso dessas redes.

Este artigo introduz um sistema de disseminação de conteúdo, denominado CDS-BitTorrent (*Content Dissemination System - BitTorrent*). Ele estende o protocolo BitTorrent com o objetivo de melhorar o desempenho de seu processo de *download* em um ambiente de MANETs, usando estratégias específicas de seleção de parte do conteúdo e de disseminação dessa parte através de um *broadcast* controlado. O grande diferencial do CDS-BitTorrent é a busca por uma solução totalmente focada na camada aplicação, permitindo que usuários de dispositivos móveis, como *smartphones* e PDAs, se beneficiem de melhorias de desempenho através da adoção de um sistema que não requer alterações de *firmware*. Este artigo avalia as ideias de seleção e disseminação de conteúdo introduzidas pelo CDS-BitTorrent, focando em cenários onde há, inicialmente, apenas uma fonte de conteúdo (como em conferências e palestras) e deseja-se enviar arquivos extensos em um menor espaço de tempo possível para os *peers* da rede.

O restante deste artigo está organizado como segue: a Seção 2 apresenta os trabalhos relacionados à adaptação do BitTorrent em MANETs. A Seção 3 detalha o protocolo BitTorrent. A Seção 4 é dedicada à apresentação do CDS-BitTorrent. A Seção 5 apresenta os cenários de simulação e os resultados de avaliação de desempenho do BitTorrent tradicional e do CDS-BitTorrent. Finalmente, a Seção 6 apresenta as conclusões deste trabalho.

2. Trabalhos Relacionados

A melhoria do desempenho do BitTorrent em cenários de MANETs é um tópico importante que vem sendo abordado recentemente em diversas pesquisas [Rajagopalan and Shen 2006] [Krifa et al. 2009] [Souza and Nogueira 2008]. Essas propostas são apresentadas a seguir, enfatizando-se como as mesmas se diferenciam do CDS-BitTorrent.

Rajagopalan *et al*. [Rajagopalan and Shen 2006] buscam uma maior eficiência do BitTorrent em MANETs com o uso de técnicas de *cross-layering* ao se integrar funcionalidades da camada aplicação com uma camada de roteamento baseada no protocolo denominado ANSI.

Krifa *et al.* [Krifa et al. 2009] propõem o BitHoc, uma solução *cross-layer* para BitTorrent em redes *ad hoc*. A solução se divide em um componente de gerenciamento de *peers* no papel de *Tracker* e um componente de compartilhamento de conteúdo, adaptando a seleção de peças de acordo com a topologia da MANET. O BitHoc conta com informações provenientes do protocolo de roteamento OLSR (*Optimized Link State Routing*) para atualizar o Tracker sobre os *peers* disponíveis.

Em [Souza and Nogueira 2008], é proposto um conjunto de modificações no protocolo BitTorrent para lidar com a localidade espaço-temporal, uma propriedade presente em redes *ad hoc* críticas, nas quais nós próximos tendem a baixar o mesmo conteúdo ao mesmo tempo. A proposta apresentada explora recursos de diversas camadas, em uma abordagem *cross-layering*, e consiste em agregar os *peers* da rede em *clusters*, cada qual com seu líder. O líder é responsável por fazer o *download* do conteúdo e por repassá-lo em *multicast* aos *peers* de seu *cluster*. Esses *peers* são desconectados da rede BitTorrent e aguardam passivamente pelos conteúdos do líder.

O CDS-BitTorrent se difere dos trabalhos relacionados por focar apenas na camada aplicação para melhoria do desempenho de *download*. Assim, o CDS-BitTorrent

não interfere na arquitetura da pilha de protocolos e é independente da escolha do protocolo de roteamento da MANET. Uma importante consequência é permitir que usuários de dispositivos móveis, principalmente em MANETs formadas por *smartphones* e PDAs, se beneficiem de melhorias de desempenho sem a necessidade de alteração do *firmware* desses dispositivos. Atualmente, o *firmware* da grande maioria desses dispositivos é fechado e não permite, por parte dos usuários, alterações de camadas inferiores à camada aplicação. Adicionalmente, as soluções puramente baseadas na camada aplicação não dependem de políticas específicas de suporte por parte dos fabricantes e facilita uma adoção maior e mais rápida por parte dos usuários.

3. O Protocolo P2P BitTorrent

O BitTorrent [Cohen 2006] é atualmente um dos principais protocolos P2P adotados na Internet para a obtenção de arquivos extensos. Sua arquitetura híbrida mistura a simplicidade obtida com o uso de entidades centralizadas e a flexibilidade garantida pela autonomia dos *peers* da rede.

No BitTorrent, a rede faz uso de uma entidade denominada *Tracker*, a qual é responsável por responder a consultas feitas via HTTP por parte daqueles que desejam ter informações sobre quais *peers* possuem interesse no mesmo arquivo. Os dados para a consulta são obtidos a partir de um arquivo `.torrent` que descreve o conteúdo desejado e fornece a URL do *Tracker*. Uma vez obtida essa informação, a comunicação com o *Tracker* não é mais obrigatória e é possível a comunicação direta entre os *peers* por meio de um protocolo específico. Ocasionalmente, a comunicação com o *Tracker* pode ser retomada para atualização de informações.

A garantia do equilíbrio das cooperações é uma característica marcante do BitTorrent. Os *peers* que mais colaboram conseguem obter conteúdo mais rapidamente, incentivando o compartilhamento entre usuários. Isso é possível graças ao mecanismo de *choking*, o qual permite que alguns *uploads* sejam bloqueados temporariamente para evitar a degradação do desempenho daqueles que mais contribuem na rede.

A eficiência no *download* de arquivos extensos se deve à forma como eles podem ser fracionados, permitindo que a obtenção dos mesmos seja feita a partir de múltiplas fontes simultaneamente, aumentando a tolerância a falhas do sistema. Um arquivo é dividido logicamente em frações denominadas *pieces*, que são subdivididas em blocos. Tanto os *pieces* quanto os blocos possuem tamanho fixo, cujo valor depende da implementação. Um *piece* é identificado por um índice que define a ordem que o mesmo ocupa no arquivo. Já o bloco é identificado pelo seu *offset*, ou seja, pelo seu deslocamento em *bytes* dentro de um *piece*. Ao longo do *download*, os *peers* fazem requisições por blocos e quando um *peer* completa um *piece*, ele informa aos demais sobre sua nova aquisição.

Na implementação clássica do BitTorrent, prioriza-se a requisição de *pieces* mais raros ainda não obtidos. Já o bloco, é escolhido aleatoriamente entre os que ainda precisam ser obtidos [Cohen 2006]. Um *peer* que possui apenas parte do arquivo é denominado *leecher* enquanto o *peer* que possui o arquivo completo é chamado de *semente*. O processo de *download* e as mensagens do protocolo são detalhados nas próximas seções.

3.1. Processo de *download*

O processo de *download* de um arquivo, por parte de um *peer* interessado, ocorre de acordo com os seguintes passos [Rajagopalan and Shen 2006]:

1. O *peer* faz uma consulta ao *Tracker*, por meio do protocolo HTTP, usando os parâmetros contidos no arquivo `.torrent`. Esse arquivo é previamente criado e disseminado por uma semente inicial;
2. O *Tracker* responde com uma lista de *leechers* e/ou sementes em processo de *download/upload* do arquivo naquele momento;
3. O *peer* se torna parte desse conjunto de *peers* interessados no arquivo e tem condições de trocar mensagens com os demais, tanto para enviar quanto para receber blocos do arquivo;
4. Durante a transferência, os *peers* podem trocar *bitfields*; o *bitfield* é uma informação que indica os *pieces* que ainda não foram baixados de um arquivo em um *peer* particular; cada bit dessa informação está relacionado a um *piece*;
5. Ao fim do *download*, o *peer* pode decidir se tornar uma semente do arquivo. Se decidir se tornar semente, precisará informar o *Tracker* sobre isso.

3.2. Mensagens do Protocolo

O diálogo entre os *peers* da rede BitTorrent é feito por meio de um protocolo próprio que permite fazer requisições por blocos, bloquear o *upload* para algum *peer* e compartilhar informações com os demais *peers* sobre o andamento do *download*. Na Internet, as implementações desse protocolo contam tipicamente com o TCP na camada de transporte [Cohen 2006]. As mensagens do BitTorrent são descritas a seguir:

- `HANDSHAKE` - usada para o primeiro contato entre 2 *peers*;
- `BITFIELD` - carrega informação sobre *pieces* já baixados, ou seja, de *bitfield*;
- `INTERESTED` - informa que um *peer* está interessado nos *pieces* de outro *peer*;
- `REQUEST` - usada para requisitar um *piece*, especificando o *offset* do bloco desejado;
- `PIECE` - traz consigo um bloco de um *piece*, informando o índice do *piece* e o *offset* do bloco;
- `HAVE` - informa que um *peer* acabou de completar um *piece*;
- `CHOKE` - informa a um *peer* que este está impedido de requisitar *pieces* de um outro *peer*;
- `UNCHOKE` - informa a um *peer* que este pode requisitar *pieces* de um outro *peer*.

4. O Sistema de Disseminação CDS-BitTorrent

O sistema de disseminação CDS-BitTorrent introduz estratégias específicas de seleção e disseminação de conteúdo para a melhoria do desempenho do processo de *download*. Tal sistema permite que uma pequena parcela do conteúdo seja entregue através de um *broadcast* controlado, isto é, que ocorre periodicamente somente para vizinhos diretos do nó emissor e com um intervalo adequado entre envios. No CDS-BitTorrent, um *peer* pode assumir dois papéis:

***Peer* Disseminador** - responsável por escolher e disseminar algumas mensagens de `PIECE` via *broadcast* controlado, alcançando somente seus nós vizinhos na MANET.

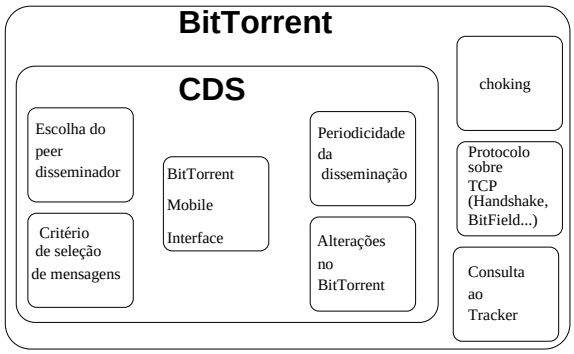


Figura 1. Arquitetura do CDS-BitTorrent

Como semente, o disseminador também poderá atender requisições por conteúdo de outros *peers* via *unicast*. É estabelecido que apenas uma das sementes deve ser o disseminador. O acréscimo de mais disseminadores do mesmo interesse poderia ser inconveniente, pois levaria a trocas extras de mensagens entre eles no intuito de evitar o envio de conteúdo redundante, sobrecarregando a rede.

***Peer* comum** - não realiza disseminações via *broadcast* e pode receber mensagens de `PIECE`, sejam elas provenientes do disseminador, sejam resultantes de suas próprias requisições, mantendo sua autonomia na rede. Ele também pode se tornar semente e atender requisições por conteúdo de outros *peers* via *unicast*.

A Figura 1 mostra uma espécie de visão arquitetural do CDS-BitTorrent. O CDS-BitTorrent é composto por uma interface denominada BMI (*BitTorrent Mobile Interface*) e por elementos responsáveis por aplicar o critério de seleção de mensagens, escolher o *peer* disseminador e disseminar periodicamente mensagens selecionadas. Além disso, são necessárias algumas alterações no protocolo BitTorrent. Tudo isso é detalhado nas próximas seções.

4.1. Interface de Disseminação BMI

A BMI (*BitTorrent Mobile Interface*) é uma subcamada da camada aplicação. É responsável por lidar com mensagens de `PIECE` recebidas ou a serem enviadas via *broadcast*. As mensagens a serem enviadas pela BMI contam com o serviço não-orientado à conexão do UDP que, ao contrário do TCP, viabiliza a entrega de mensagens recebidas por *broadcast* à aplicação. A Figura 2 ilustra a comunicação entre as camadas aplicação, transporte e a interface BMI. Do lado emissor, a BMI recebe as mensagens a serem enviadas em modo *broadcast*. Essas mensagens são encapsuladas junto com um identificador de interesse do conteúdo (por exemplo, um valor de *hash* do arquivo compartilhado) em segmentos UDP. A BMI também sinaliza ao *socket* UDP o endereço de *broadcast* da rede como destino das mensagens. Do lado receptor, a BMI atua como um filtro, desencapsulando e repassando mensagens para a aplicação BitTorrent somente quando a mesma possuir interesse no conteúdo recebido. Caso não haja interesse, as mensagens correspondentes são descartadas pela BMI.

As mensagens destinadas ao envio em *unicast* são encaminhadas diretamente ao TCP sem passar pela BMI. Assim, um dispositivo utilizando o CDS-BitTorrent se serve tanto do TCP quanto da BMI/UDP para envio de mensagens. Os critérios de seleção de

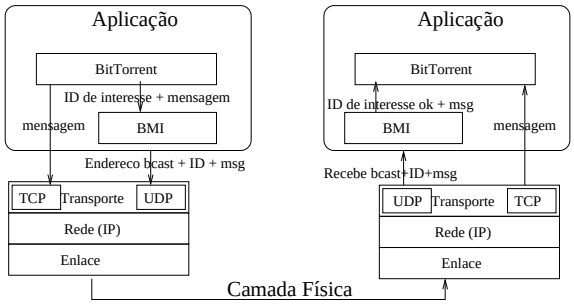


Figura 2. Pilha de protocolos com a interface BMI

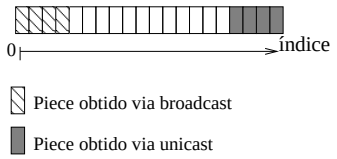


Figura 3. *Pieces* com blocos baixados via *broadcast* e *unicast*

mensagens de `PIECE` a serem enviadas para a BMI é apresentado na próxima seção.

4.2. Critérios de Seleção de Mensagens

O CDS-BitTorrent dissemina parte das mensagens de `PIECE` em modo *broadcast*, uma vez que esse tipo de mensagem é o mais recorrente durante o *download*. Conforme observado na Seção 3.2, uma mensagem de `PIECE` carrega consigo um bloco de conteúdo. Assim, na prática, deve ocorrer uma seleção de quais blocos devem ser enviados em *broadcast* pelo disseminador e quais blocos devem ser requisitados por um *peer* comum. Tais seleções devem possuir regras distintas para evitar a chegada de mensagens redundantes que sobrecarregariam a rede. Assim, o CDS-BitTorrent estabelece as seguintes regras:

Seleção de blocos para disseminação - o disseminador escolhe preferencialmente os blocos mais raros dentre os *pieces* mais raros, do menor para o maior índice;

Seleção de blocos a serem requisitados - os *peers* comuns requisitam preferencialmente os blocos do menor para o maior *offset* dentre os *pieces* mais raros, do maior para o menor índice.

O disseminador armazena informações de blocos já disseminados e não envia um mesmo bloco mais de uma vez. Os *peers* conseguem estimar os blocos e *pieces* mais raros graças a uma modificação na mensagem de `BITFIELD` feita para o CDS-BitTorrent. Neste contexto, a mensagem de `BITFIELD` agora traz informação no nível de bloco. A mensagem de `HAVE` é utilizada para atualizar a informação de *pieces* baixados pelos demais *peers*. Com as estratégias adotadas, o preenchimento dos *pieces* em um *peer* próximo ao disseminador tende a ocorrer de acordo com a Figura 3. Contudo, a mobilidade pode potencialmente impactar essa tendência. Assim sendo, é necessária uma avaliação do desempenho do CDS-BitTorrent em cenários com mobilidade. Essa avaliação será apresentada na Seção 5.4.

4.3. Escolha do *Peer* Disseminador

A semente com maior identificador é escolhida como *peer* disseminador. Para o CDS-BitTorrent, o identificador do *peer* equivale ao seu endereço IP. Assim, quando uma se-

mente recebe a lista de *peers* vinda do *Tracker*, ela tem acesso aos seus identificadores. Para verificar se um *peer* é semente, são utilizadas as mensagens de `BITFIELD`, já presentes na implementação tradicional do BitTorrent. Tal critério confere simplicidade ao algoritmo por não exigir comunicação extra entre os *peers* para essa escolha. A cada mensagem `BITFIELD` recebida, a semente verifica se o emissor da mensagem também é uma semente. Se sim, compara os identificadores e, caso o seu seja menor, encerra o algoritmo. Caso contrário, se constatar que é a semente com maior identificador, assume o papel de disseminador. Uma vez que o CDS-BitTorrent é direcionado à situações de reuniões e conferências, é assumido que, inicialmente, todos os nós estão presentes. O procedimento é realizado apenas uma vez, nas primeiras trocas de `BITFIELD`, e é convencionado que o *Tracker* retorna todos os *peers* interessados no conteúdo. Ao contrário do que se pode imaginar, a existência de um único disseminador não afeta a tolerância a falhas do sistema, pois, mesmo que o disseminador deixe a rede ou que surjam partições na mesma, o *download* pode continuar através da comunicação *unicast*, quando possível.

4.4. Periodicidade de Disseminação

O envio em excesso de mensagens via *broadcast* pode gerar muitas colisões no meio de comunicação compartilhado, afetando negativamente o processo de *download*. Por isso, a periodicidade da disseminação por *broadcast* torna-se um fator fundamental para um bom desempenho do *download*. No CDS-BitTorrent, essa periodicidade é modelada como uma variável aleatória distribuída uniformemente no intervalo de tempo [0, M], sendo M um parâmetro de projeto. A escolha do valor desse parâmetro será comentada na Seção 5.

4.5. Alterações no Protocolo BitTorrent

Conforme apresentado, o CDS-BitTorrent requer pequenas alterações no protocolo BitTorrent no tocante ao tratamento de mensagens `BITFIELD` e `HAVE`. Além disso, é necessário estender o BitTorrent com novas funcionalidades, como a escolha do *peer* disseminador e os algoritmos de seleção e disseminação de blocos.

5. Avaliações de Desempenho

Esta seção avalia, através de simulações com o NS-2 [Fall and Varadhan 2007], o desempenho do CDS-BitTorrent e do BitTorrent tradicional quando executados em um ambiente de MANETs. O módulo para NS-2 desenvolvido por *Eger et al.* [Eger et al. 2007] foi utilizado para a avaliação do BitTorrent tradicional[1]. Para a avaliação do CDS-BitTorrent foram feitas diversas modificações nessa implementação, dentre elas, as apresentadas na Seção 4.5. Além delas, foi acrescido um módulo com protocolo de roteamento OLSR para as avaliações [Paquereau and Helvik 2006]. A modelagem das camadas física e MAC (*Medium Access Control*) segue a especificação IEEE 802.11g [IEEE Std 802.11g 2003]. O raio de alcance de comunicação dos nós é de *50 m* para simular o alcance típico desse rádio em ambientes internos. A área onde os nós estão confinados é de *150m* x *150m* para simular um espaço para eventos. O modelo de propagação de sinais utilizado é o *Two Ray Ground*. As métricas de avaliação, os cenários, os parâmetros de simulação e os resultados obtidos são apresentados nas próximas seções.

[1]Nessa implementação, o *Tracker* é abstraído em um objeto visível a todos os nós da rede. Tal decisão não afeta os resultados deste trabalho, uma vez que o objetivo é a avaliação do desempenho do processo de *download* em si e não a do processo de localização de *peers*.

5.1. Métricas de Avaliação

As seguintes métricas foram utilizadas nas avaliações de desempenho:

***Overhead* de segmentos -** número total de segmentos de transporte gerados até a entrega do conteúdo a todos os *peers*;

Fração de segmentos UDP e TCP - a fração de segmentos UDP e TCP gerados em comparação ao total de segmentos enviados com o uso do CDS-BitTorrent;

Pacotes perdidos - número de pacotes perdidos até a entrega total do conteúdo a todos os *peers*;

Tempo de *download* - tempo médio que um *peer* leva para finalizar o *download*;

***Overhead* de roteamento -** número de mensagens de roteamento transmitidas, incluindo transmissões entre saltos;

Atraso médio fim a fim - tempo médio decorrido entre o envio e recebimento de segmentos de transporte até a completude do *download*, considerando o TCP para a abordagem tradicional e adicionalmente o UDP para o caso do CDS-BitTorrent.

5.2. Cenários e Demais Parâmetros de Simulação

Os cenários de simulação foram concebidos de forma a imitar uma situação de compartilhamento de um vídeo em um espaço para eventos, onde pessoas com dispositivos móveis formando uma MANET podem se mover ou permanecer paradas. Assim, dois cenários, um com mobilidade e outro sem, foram definidos para as avaliações de desempenho. Especificamente para o cenário sem mobilidade, os nós estão dispostos em uma topologia de grade, distribuídos uniformemente à distância de *1 m* entre vizinhos de uma mesma fileira, imitando o espaçamento físico entre pessoas sentadas assistindo a uma palestra. Em específico para o cenário com mobilidade, 25 nós estão inicialmente dispostos em uma topologia de grade (5x5), distribuídos uniformemente à distância de *1 m* entre vizinhos de uma mesma fileira.

Para simular a mobilidade, foi adotada uma variação do modelo *Random Waypoint* (RWP), conforme descrito a seguir: após um tempo de espera, os nós se movem a uma velocidade constante para um destino de coordenadas aleatórias para posteriormente voltarem a ficar imóveis durante o mesmo tempo de espera. Mais uma vez, após esse tempo de espera, os nós seguem para um novo destino aleatório, repetindo o processo descrito até o fim da simulação. O tempo de espera utilizado foi de 10 segundos. A diferença entre o modelo adotado e o RWP consiste no uso de uma velocidade constante, diferente deste último, onde, a cada mudança de direção, uma velocidade aleatória é escolhida em uma distribuição uniforme com zero como valor mínimo. Essa mudança foi feita motivada pelo fato de estudos mostrarem que a forma como o RWP tradicionalmente lida com a questão da velocidade é controversa [Yoon et al. 2003] e que a proximidade das velocidades a um valor estacionário em longas simulações levam a resultados mais realísticos, sendo este o caso do presente experimento.

Ambos os cenários avaliados possuem características em comum, as quais são descritas como segue. Cada nó executa uma instância da aplicação, ou seja, cada nó corresponde a um *peer* BitTorrent ou CDS-BitTorrent conforme abordagem avaliada. A simulação se inicia com apenas uma semente. Os nós entram na rede em instantes

aleatórios entre 0 e 1s após o início da simulação. Após a entrada de todos os nós na rede, não há saída ou entrada de novos nós. A simulação termina quando todos os nós concluem o *download*. O *peer* que finaliza um *download* continua na aplicação para colaborar com os demais. Durante a simulação, os *peers* compartilham um arquivo de 100MB, considerando-se *pieces* de 512KB, divididos em blocos de 16KB cada. No caso do CDS-BitTorrent, o parâmetro de periodicidade da disseminação M é igual a $1s$. Tal valor foi obtido por meio de um estudo aqui omitido por limitações de espaço. No referido estudo, foi observado que um valor grande para M tornaria a disseminação pouco eficiente. Por outro lado, um valor de M menor que $1s$ desencadearia em uma sobrecarga de mensagens, degradando o desempenho na rede. O valor de $M = 1s$ garantiu em todos os cenários estudados um desempenho melhor do CDS-BitTorrent do que o do BitTorrent tradicional, sem sobrecarregar a rede. Os parâmetros específicos adotados para o BitTorrent foram os seguintes: intervalo de *choking* de 10 segundos e número máximo de *uploads* simultâneos igual a 4.

Todos os resultados apresentados a seguir possuem um intervalo de confiança de 99%. Os intervalos são representados por barras de erro nos gráficos, sendo algumas delas imperceptíveis. Cada ponto simulado corresponde à média obtida a partir de 20 simulações.

5.3. Cenário sem Mobilidade

No cenário sem mobilidade, as métricas de avaliação definidas foram estudadas em função do número de *peers* na rede. Foram realizadas simulações para 4 (2x2), 9 (3x3), 16 (4x4), 25 (5x5), 36 (6x6) e 49 (7x7) *peers* na rede.

A Figura 4(a) apresenta a fração de segmentos gerados pelo TCP e a partir de mensagens de `PIECE` enviadas à interface BMI durante o processo de *download* com o CDS-BitTorrent. Observa-se que apenas 5,4% do total de segmentos é enviado em *broadcast* no caso de haver 4 *peers* na rede. Esse valor diminui com o aumento do número de *peers*, atingindo 1,3% do total de segmentos para o caso de 49 *peers*. Essa diminuição ocorre porque com o aumento do número de *peers* há mais conexões TCP ativas, gerando mais tráfego em *unicast*, enquanto a geração de tráfego em *broadcast* pelo disseminador não varia em função do número de *peers* na rede.

A Figura 4(b) mostra o *overhead* de segmentos em função do número de *peers*. O CDS-BitTorrent apresenta um menor *overhead* de segmentos em relação à abordagem tradicional. Em particular, observa-se que ele gera 50% a menos de segmentos do que a abordagem tradicional com 36 *peers* na rede. Adicionalmente, observa-se que a taxa de crescimento de *overhead* de segmentos do CDS-BitTorrent também é menor do que a da abordagem tradicional. O melhor desempenho do CDS-BitTorrent se justifica pelo fato do pequeno percentual de segmentos enviados em *broadcast* servir todos os *peers* da rede e assim permitir uma redução na quantidade de requisições por blocos e no consequente tráfego em *unicast* na rede.

A Figura 4(c) mostra como o número de pacotes perdidos na rede varia em função do número de *peers*. Nota-se que a perda de pacotes com o CDS-BitTorrent é até 40% menor do que a perda observada com a abordagem tradicional. Isso é consequência direta da menor quantidade de segmentos que o mesmo gera e envia na rede.

A Figura 4(d) apresenta o tempo de médio *download* em função do número de

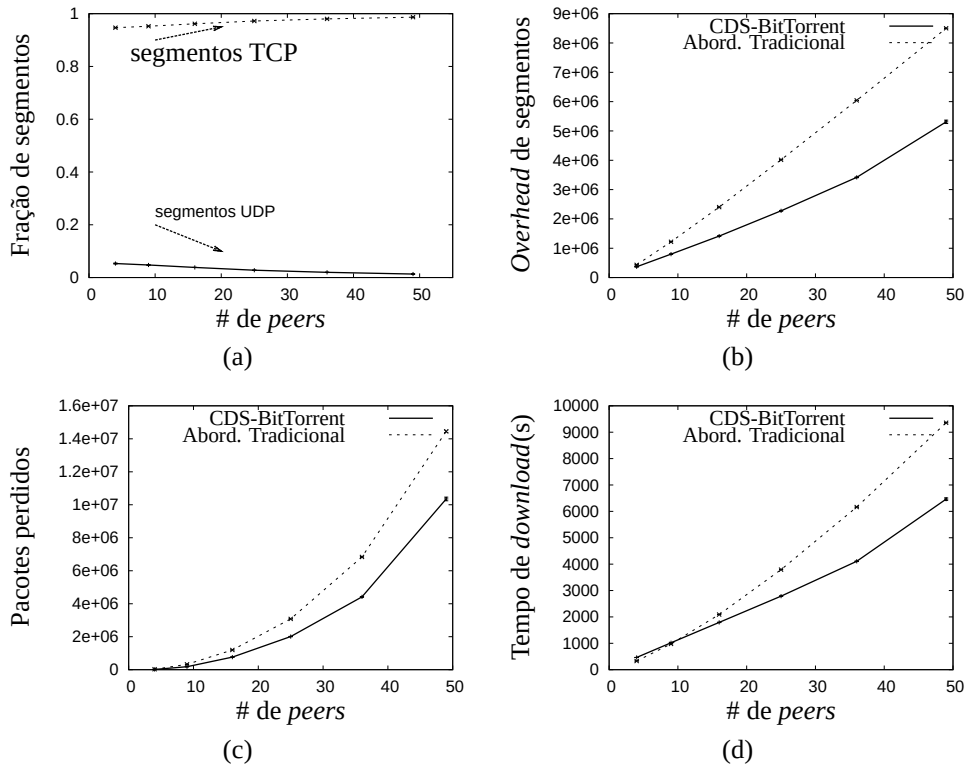


Figura 4. Cenário sem Mobilidade: (a) Fração de segmentos (b) *Overhead* de segmentos (c) Pacotes perdidos (d) Tempo de *download*

peers. Nota-se que a taxa de crescimento do tempo de *download* é menor com o uso do CDS-BitTorrent. Observa-se ainda que não há diferença significativa entre as abordagens até 9 *peers* na rede. Entretanto, a partir de 16 *peers*, o tempo de *download* passa a ser menor para o CDS-BitTorrent. No caso de 49 *peers*, por exemplo, observa-se uma melhora de 30% em comparação à abordagem tradicional. O melhor tempo de *download* observado com o uso do CDS-BitTorrent é consequência direta do menor *overhead* de segmentos que ele gera para a entrega do conteúdo.

A Figura 5(a) apresenta o *overhead* de roteamento em função do número de *peers*. Observa-se que o uso do CDS-BitTorrent contribui para a diminuição do *overhead* de roteamento. Em particular, esse *overhead* chega a ser 20% menor para o caso de 49 *peers*. O menor *overhead* em comparação à abordagem tradicional é consequência direta do menor tempo de *download* proporcionado pelo uso do CDS-BitTorrent já que, nesse período de tempo mais curto, haverá menos pacotes de roteamento gerados.

A Figura 5(b) mostra o atraso médio fim a fim em função do número de *peers*. Observa-se que tal atraso é similar em ambas as abordagens para até 36 *peers*. A partir desse valor, o atraso médio fim a fim é ligeiramente maior com o CDS-BitTorrent, sendo enfatizado pela nítida mudança na inclinação da curva. Note que isso se reflete também na mudança de inclinação que ocorre no mesmo ponto da curva do tempo de *download* apresentada na Figura 4(d). O aumento no atraso médio fim a fim a partir de 36 *peers* se justifica pelo aumento de colisões de transmissões por causa do *broadcast* já que a disputa pelo meio de comunicação se torna mais intensa à medida que o número de *peers* aumenta. Apesar disso, o ligeiro aumento observado no atraso médio fim a fim não tem

impacto significativo no tempo médio de *download* com o CDS-BitTorrent já que com o mesmo, a quantidade de segmentos gerados é bem menor.

5.4. Cenário com Mobilidade

No cenário com mobilidade, as métricas de avaliação definidas foram estudadas em função da velocidade dos nós da rede. Foram realizadas simulações para as velocidades de *0,0 m/s* a *1,5 m/s* em passos de *0,5 m/s*, considerando 25 *peers*.

A Figura 6(a) mostra como a fração de segmentos gerados a partir de mensagens de `PIECE` enviadas à interface BMI varia em função da velocidade do nós durante o *download* com o CDS-BitTorrent. Observa-se que, em média, de 2,8% (para *0,0 m/s*) a 2,0% (para *1,5 m/s*) do total de segmentos gerados são originados a partir de conteúdo proveniente da interface BMI. A diminuição do percentual ocorre por causa do aumento do volume de tráfego TCP, já que o volume de tráfego enviado em *broadcast* pelo disseminador não depende da velocidade dos nós. Como a mobilidade espalha os nós pela área de $150m$ x $150m$, diminuindo o número de nós vizinhos do nó disseminador, menos *peers* são favorecidos continuamente pela disseminação. Isso obriga os demais *peers* a fazerem mais requisições por conteúdo, gerando um maior volume de tráfego TCP na rede em resposta a tais requisições.

A Figura 6(b) mostra o *overhead* de segmentos em função da velocidade dos nós. Observa-se que esse *overhead* é significativamente menor com o uso do CDS-BitTorrent. Porém, o *overhead* de segmentos é aproximadamente o mesmo com o aumento da velocidade para o caso da abordagem tradicional ao passo que para o caso de uso do CDS-BitTorrent, há um aumento quando há mobilidade. Isso ocorre com o uso do CDS-BitTorrent por causa do aumento da fração de tráfego TCP na rede como explicado anteriormente. Apesar disso, o *overhead* de segmentos com o CDS-BitTorrent no pior dos casos foi 20% menor do que o alcançado com a abordagem tradicional.

A Figura 6(c) mostra o número de pacotes perdidos em função da velocidade. Observa-se que há uma menor quantidade de pacotes perdidos com uso do CDS-BitTorrent para todas as velocidades estudadas. Observa-se também que para ambas as abordagens, o número de pacotes perdidos diminui com o aumento da velocidade. A explicação é como segue: o aumento da velocidade faz com que os *peers* deixem de compartilhar mais rapidamente o mesmo domínio inicial de colisões. Em consequência, a

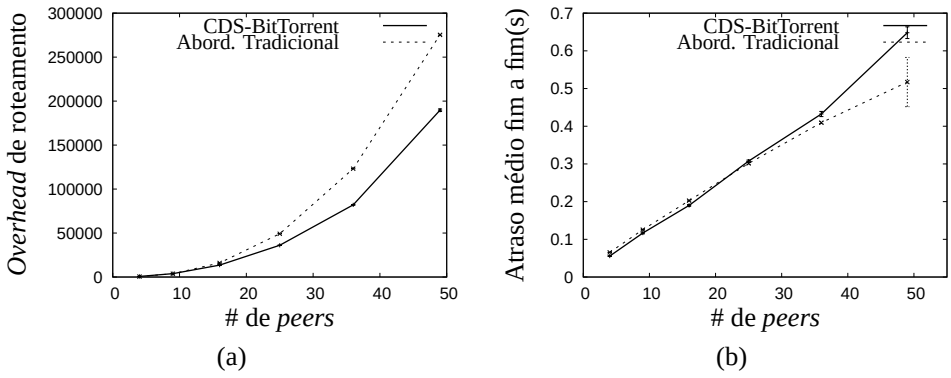


Figura 5. Cenário sem Mobilidade: (a) *Overhead* de roteamento (b) Atraso médio fim a fim

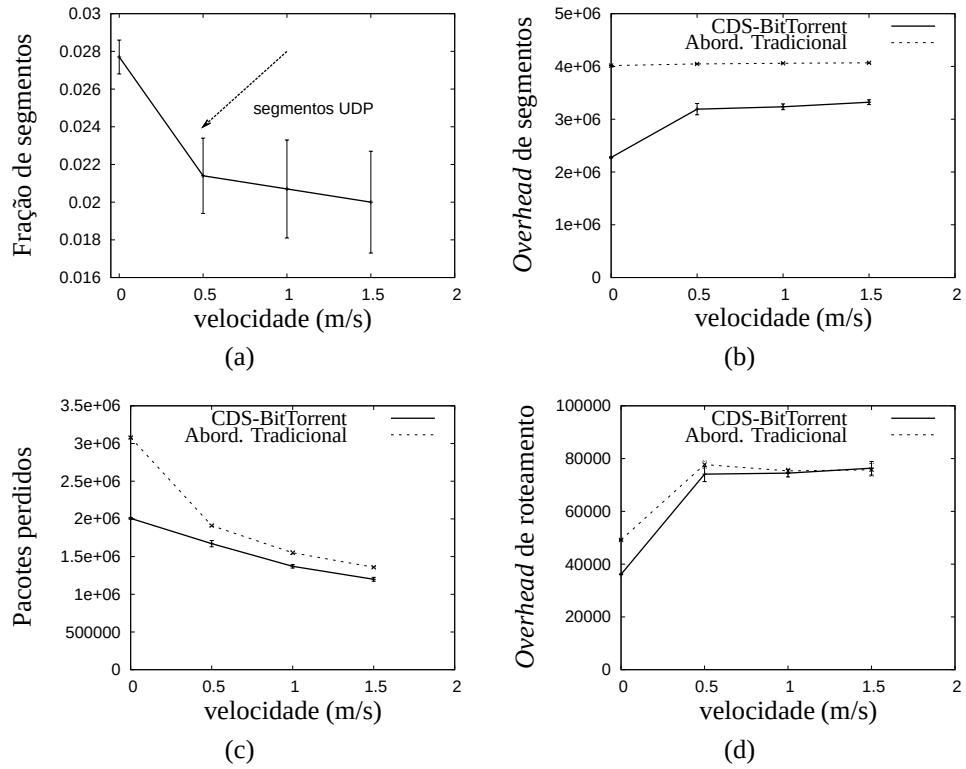


Figura 6. Cenário com Mobilidade: (a) Fração de segmentos (b)*Overhead* de segmentos (c) Pacotes perdidos (d) *Overhead* de roteamento

disputa inicial pelo meio de comunicação sem fio é amenizada mais rapidamente, reduzindo perdas por colisão de transmissões.

A Figura 6(d) mostra o *overhead* de roteamento em função da velocidade. Nota-se que para ambas as abordagens há um aumento abrupto do *overhead* de roteamento com a mobilidade, passando o mesmo a ser similar nos dois casos a partir da velocidade de *0,5 m/s*. O aumento abrupto é consequência da reação do OLSR à mudanças de topologia da rede por causa da mobilidade, o que o faz reduzir o intervalo de envio periódico de mensagens de controle na rede. Já a similaridade do *overhead* a partir de *0,5 m/s*, se justifica pelo tempo de *download* similar (Figura 7(a)) entre ambas as abordagens, tornando o tempo de execução do protocolo de roteamento para envio de mensagens periódicas também similar durante as simulações.

A Figura 7(a) mostra o tempo médio de *download* em função da velocidade. Observa-se que este tempo não varia significativamente com a velocidade dos nós quando a abordagem tradicional é utilizada. Por outro lado, no caso do CDS-BitTorrent, o tempo de *download* apresenta um aumento quando se adiciona mobilidade ao cenário. O aumento é consequência do espalhamento dos nós que, por sua vez, diminui o número de *peers* continuamente servidos pelo conteúdo enviado em *broadcast* e faz com que o tráfego *unicast* tenha participação maior no processo de obtenção do arquivo, aumentando o tempo de *download*. Vale notar que o tempo de *download* com o CDS-BitTorrent é similar ao da abordagem tradicional para velocidades de $0,5\ m/s$ a $1,5\ m/s$. A princípio, esse resultado não seria esperado pois, conforme apresentado, o *overhead* de segmentos e a perda de pacotes são menores com o uso do CDS-BitTorrent para tais velocidades. A

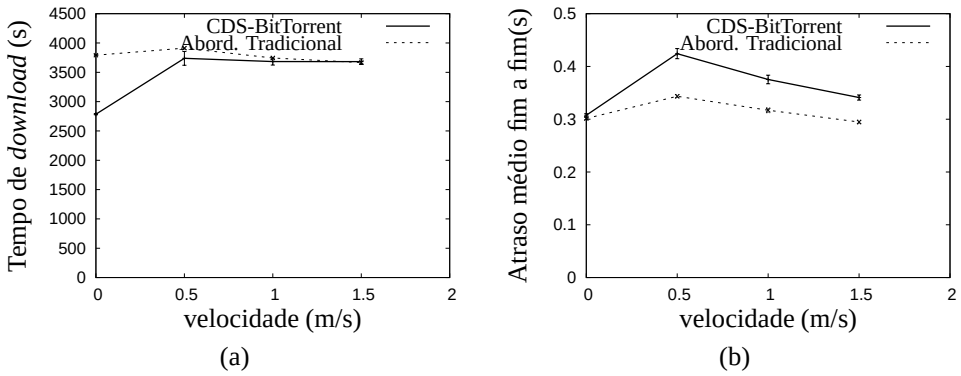


Figura 7. Cenário com Mobilidade: (a) Tempo de *download* (b) Atraso médio fim a fim

razão para o ocorrido se deve ao fato do tempo médio de *download* passar a ser dominado pelos atrasos na entrega de pacotes em consequência de atrasos para a reconstrução de rotas e do aumento abrupto do *overhead* de roteamento com a mobilidade, gerando mais tráfego na rede. Apesar do exposto, o tempo médio de *download* com o CDS-BitTorrent é sempre menor ou igual ao obtido com a abordagem tradicional.

A Figura 7(b) mostra o atraso médio fim a fim de segmentos em função da velocidade. Observa-se que tal atraso aumenta para ambas as abordagens avaliadas quando se passa de *0,0 m/s* a *0,5 m/s*. Isto ocorre pois passam a existir atrasos para a reconstrução de rotas e atrasos consequentes do tráfego adicional na rede gerado pelo crescimento abrupto do *overhead* de roteamento. Observa-se também uma ligeira queda no atraso médio fim a fim para ambas as abordagens a partir de *0,5 m/s*. Essa queda é decorrente da redução da perda de pacotes observada com o aumento da mobilidade conforme Figura 6(c). Nota-se ainda que o impacto da mobilidade no atraso médio fim a fim é maior com o uso do CDS-BitTorrent. O maior impacto é consequência direta do fato do aumento relativo do *overhead* de roteamento ser maior com o uso do CDS-BitTorrent do que com o uso da abordagem tradicional ao se passar de um cenário sem mobilidade para um com mobilidade conforme mostra a Figura 6(d). Em suma, os resultados mostram que o atraso médio fim a fim é maior com o uso do CDS-BitTorrent no cenário estudado. Entretanto, isso não trouxe impacto suficiente no tempo de *download* para que o mesmo fosse maior do que o obtido utilizando-se a abordagem tradicional já que houve uma compensação pelo fato do CDS-BitTorrent gerar um menor *overhead* de segmentos.

6. Conclusões

Este trabalho introduziu o sistema CDS-BitTorrent que se propõe à melhoria do processo de *download* do BitTorrent em MANETs através da adoção de estratégias específicas de seleção e disseminação de conteúdo. A redução no tempo de *download* com o uso do CDS-BitTorrent se mostrou mais efetiva nos cenários sem mobilidade e de baixa mobilidade (até *0,5 m/s*), sendo adequado aos cenários para os quais foi proposto. Ainda sim, é importante ressaltar que o CDS-BitTorrent obteve em todos os cenários avaliados, independente da mobilidade, um menor *overhead* de segmentos e uma menor perda de pacotes na rede, contribuindo diretamente para um menor consumo de energia dos nós da MANET. De forma geral, os resultados mostraram que disseminar seletivamente apenas

de 1,3% a 5,4% do volume total de segmentos via um *broadcast* "racional", traz impactos positivos de desempenho, enfatizando a boa relação custo/benefício das estratégias adotadas pelo CDS-BitTorrent. O CDS-BitTorrent ainda mantém a tolerância a falhas do sistema, ao permitir que *peers* atuem de forma independente da disseminação.

O BitTorrent foi concebido para a Internet e considera que todas as conexões possuem um mesmo custo. Ao se valer de difusões via UDP, o CDS-BitTorrent diminui a importância de conexões TCP para diversos *peers* e consegue assim melhorar o desempenho. Os resultados aqui apresentados abrem caminho para discussões sobre até que ponto os princípios do BitTorrent para Internet são adequados em MANETs. Vale ressaltar que o CDS-BitTorrent é um sistema em evolução, mas que as ideias aqui apresentadas são promissoras conforme sugerem os resultados apresentados. Em trabalhos futuros, extensões ao CDS-BitTorrent podem ser consideradas, como o ajuste automático do parâmetro M em função do tráfego na rede e o controle da disseminação baseado em Teoria dos Jogos. Além disso, pode-se ampliar a variedade de cenários de estudo, incluindo situações com diferentes velocidades entre os nós e diferentes interesses por conteúdo.

Referências

Cohen, B. (2006). BitTorrent Specification. Technical report, BitTorrent.org.

Eger, K., Hossfeld, T., Binzenhofer, A., and Kunzmann, G. (2007). Efficient Simulation of Large-scale P2P Networks: Packet-level vs. Flow-level Simulations. In *Proceedings of the 2nd UPGRADE-CN*, pages 9–16, New York, NY, USA.

Fall, K. and Varadhan, K. (2007). NS Notes and Documentation. Technical report, The VINT Group.

Hu, Y. C., Das, S. M., and Pucha, H. (2005). Peer-to-Peer Overlay Abstractions in MANETs. In Wu, J., editor, *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, volume 1, pages 857–874. CRC Press.

IEEE Std 802.11g (2003). IEEE Standard.

Krifa, A., Sbai, M. K., Barakat, C., and Turletti, T. (2009). BitHoc: A Content Sharing Application for Wireless Ad hoc Networks. In *Proceedings of the IEEE Percom*.

Paquereau, L. and Helvik, B. E. (2006). A Module-based Wireless Node for NS-2. In *Proceedings of the WNS2*, page 4, New York, NY, USA.

Rajagopalan, S. and Shen, C.-C. (2006). A Cross-layer Decentralized BitTorrent for Mobile Ad hoc Networks. In *Proceedings of the 3rd MobiQuitous*, pages 1–10.

Souza, C. and Nogueira, J. M. (2008). Um Estudo do BitTorrent em Redes ad hoc sem Fio Críticas com Localidade Espaço-temporal. In *Anais do 25º SBRC*, pages 329–342.

Yoon, J., Liu, M., and Noble, B. (2003). Random Waypoint Considered Harmful. In *Proceedings of the IEEE INFOCOM*, volume 2, pages 1312–1321.

REPI: Rede de comunicação Endereçada Por Interesses

Renato C. Dutra, Rodrigo S. Granja, Heberte F. Moraes, Claudio L. Amorim

Laboratório de Computação Paralela – COPPE
Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal – 68.511 – Rio de Janeiro – RJ – Brasil

{rcdutra, rodrigos, hmoraes, amorim}@lcp.coppe.ufrj.br

Resumo. *O problema de roteamento de mensagens em redes sem fio ad hoc é abordado, utilizando uma estrutura de mensagem composta de prefixo e payload. O prefixo contém informações (Interesses) do usuário, permitindo o encaminhamento das mensagens pelo casamento de interesses, no lugar do endereçamento fim-a-fim. Para avaliar esse original modelo de comunicação, implementamos uma rede endereçada por interesses (REPI) sobreposta a uma rede ad hoc com 20 nós. Resultados de uma avaliação preliminar confirmaram a viabilidade do uso da REPI em redes ad hoc, cujas taxa e custo de entrega de mensagens foram equivalentes aos do algoritmo "Gossip" usando até* 75% *de probabilidade, com a vantagem de utilizar informação útil como endereçamento da mensagem.*

Abstract. *The problem of message routing in ad hoc wireless networks is addressed using a message structure composed of prefix and payload. The prefix contains information (Interests) of the user, allowing routing of messages by matching of interests, instead of end-to-end addressing for routing. To evaluate this original communication model, we implemented an overlay network addressed by interests (REPI) in a 20-node ad hoc network. Results of a preliminary assessment confirmed the feasibility of using REPI in ad hoc networks, whose rate and cost of delivering messages were equivalent to the Gossip algorithm using up to 75% of probability, with the advantage of using useful information as addressing of the message.*

1. Introdução

Rede sem fio ad hoc vem sendo objeto de pesquisa nos últimos anos devido à sua aplicabilidade, particularmente em situações onde a infraestrutura disponível de rede tenha se deteriorado. Um exemplo expressivo de uso das redes ad hoc está em um ambiente aonde um desastre natural ocorreu. Nesse ambiente, os equipamentos de comunicação (inclusive os celulares) podem utilizar comunicação multihop beneficiando-se da localidade dos equipamentos, do envio das informações locais para processamento remoto, do armazenamento dos dados para futuro treinamento, da descentralização e, principalmente, do envolvimento das pessoas nas regiões críticas [Rao et al. 2007].

Porém, o envio de mensagens em redes sem fio ad hoc é um desafio devido à dificuldade de identificação dos nós e ao custo de manutenção do roteamento. O uso de identificação única em redes sem fio, especificamente para solucionar problemas de roteamento em redes ad hoc, é uma questão de pesquisa ainda em aberto [Intanagonwiwat

et al. 2000, Spyropoulos et al. 2008]. O custo de manutenção do roteamento é alto quando ocorre o emprego de *overlay networks* Par-a-Par (P2P) por meio de *spanning trees* [Carzaniga e Hall 2006], devido à sobreposição de uma rede "cabeada" em uma rede sem fio ad hoc. Uma possível solução para os dois problemas, são as tabelas *hash* distribuídas P2P, usadas para a identificação dos nós e a manutenção das *spanning trees* dinâmicas, ainda utilizando P2P [Zahn e Schiller 2006, Heer et al. 2006]. A solução de interpretar redes sem fio ad hoc como redes P2P é antiga [Sohrabi et al. 2000], e procura unir a dinâmica de associação dos pares P2P à dinâmica das redes sem infraestrutura, com dezenas ou milhares de nós, possuindo características de multihoping, auto-organização, economia de energia e escalabilidade.

Por outro lado, a crescente aplicabilidade de modelos de comunicação distribuída baseados na arquitetura P2P simplifica a cooperação entre os usuários, basicamente pela redução da carga de comunicação no servidor central e pela delegação da administração dos conteúdos.

Embora o uso de soluções de redes cabeadas aplicadas a redes sem fio ad hoc seja promissor, as características distintas destas têm estimulado a pesquisa e desenvolvimento de novos mecanismos e técnicas que as explorem efetivamente, evitando adaptações diretas de soluções de redes cabeadas. Nesses casos, geralmente, não há a preocupação de o escopo das aplicações ser o mesmo de redes cabeadas, permitindo, inclusive, o desenvolvimento de novas e originais aplicações.

Com esse intuito, propusemos e avaliamos um novo modelo de comunicação para redes ad hoc [Dutra e Amorim 2010], implementado por meio de uma Rede Endereçada Por Interesses (REPI), visando uma solução eficaz para a identificação dos nós e a manutenção do roteamento de mensagens.[1]

Uma possível aplicação da REPI é no caso de ocorrência de um desastre natural em uma dada região, onde exista a necessidade de localidade, agilidade, armazenamento da informação centrada na pessoa, empregando eficientemente o endereçamento por interesses. A população local poderia ser encontrada pela localização dos celulares nesta rede endereçada por interesses. Os interesses poderiam ser definidos tais como: médicos, bombeiros, policiais e voluntários. Nestes canais de interesse, os profissionais poderiam comunicar-se em uma rede distribuída, sem necessitar de centralização, promovendo uma intervenção rápida na região.

A REPI possui três propriedades distintas. A primeira é a rede ser endereçada por termos, ou seja, a estrutura da mensagem é composta apenas de termos, sem o uso de outras formas ou campos de identificação. A segunda propriedade é a rede formar-se somente quando uma entidade enviar mensagens através de uma ação pró-ativa, ou seja, a rede se materializa para uma entidade quando ela enviar uma mensagem. A terceira propriedade é a ausência de endereçamento convencional fim-a-fim permitindo à rede ser volátil e independente do roteamento clássico para difusão de mensagens.

Para uma avaliação preliminar da REPI, foram executados experimentos em uma implementação REPI em uma rede ad hoc com 20 nós, variando os parâmetros: número de nós transmissores, intervalo de tempo no envio das mensagens e número de campos

[1]Por simplicidade, doravante usaremos REPI para denominar também o protocolo de rede e o algoritmo de roteamento, indiscriminadamente.

na parte B do prefixo. Foram também medidos os valores de porcentagem de mensagens entregues nos nós com interesse, custo de envio destas mensagens, número de nós colaboradores e número de saltos por onde as mensagens passaram da fonte ao(s) destino(s). Em uma avaliação preliminar, a REPI alcançou uma taxa de entrega de mensagens e custo associado equivalentes aos do algoritmo de referência *Gossip* para redes ad hoc usando probabilidade de 75%; ainda, com a vantagem da REPI utilizar informação útil para endereçamento.

A principal contribuição deste trabalho reside na comprovação da viabilidade da REPI como rede ad hoc baseada em endereçamento por interesses. Outras contribuições importantes incluem: uma nova métrica para medir colaboração em redes ad hoc, devido ao uso dos interesses implicarem na colaboração entre os nós; ampliação do conceito de encaminhamento probabilístico de mensagens, pelo uso de vários campos no prefixo; experimentos e medidas em um cenário realista utilizando 20 nós, com envio de 1000 mensagens por cada nó e 20 repetições em cada experimento.

O ótimo desempenho da REPI foi confirmado em um cenário real, onde uma aplicação de mensagens instantâneas baseada na REPI foi utilizada no Laboratório de Computação Paralela por seis alunos e três pesquisadores com interesses fixos (Projeto, Tráfego, Alimentação, Seminário) e interesses momentâneos. A monitoração da capacidade da rede foi realizada através da ferramenta SAMCRA [Granja et al. 2010], pelo intervalo de uma semana.

Além desta introdução, este artigo está organizado como segue. Na Seção 2, revisamos os trabalhos relacionados, e na Seção 3, descrevemos a REPI. Na Seção 4, os resultados de uma avaliação experimental preliminar da REPI são discutidos. Na Seção 5, apresentamos as conclusões e na Seção 6, delineamos os trabalhos em andamento.

2. Trabalhos Relacionados

A proposta da REPI foi construída sobre trabalhos anteriores com vários focos de pesquisa descritos a seguir, tais como, a colaboração entre usuários, o encaminhamento de mensagens pelos interesses dos usuários, a rede ser orientada ao usuário e o endereçamento por termos.

Em redes ad hoc, Kortuem et al. 1999 propõem a colaboração baseada no uso do perfil do usuário, a troca de informação do perfil e a possibilidade de pessoas desconhecidas se encontrarem. Porém, os encontros são físicos, o usuário é identificado e precisa se locomover para se encontrar, não existe o uso do multihop e a mensagem é convencional, de acordo com o protocolo utilizado (por exemplo, TCP/IP). Rantanen et al. 2004 utilizam o encaminhamento de mensagens de contexto em redes ad hoc, contudo os equipamentos são identificados, as mensagens são convencionais e o GPS é utilizado para localização. Borcea et al. 2007 propõem o conceito da rede ser orientada ao usuário, entretanto é um experimento voltado para o reconhecimento de padrões em redes sociais, utilizando mecanismos convencionais.

Awad et al. 2009 propõem o esquema de endereçamento virtual para roteamento em WSN *Virtual Cord Protocol*, onde DHTs associam dados com nós com específicos endereços. Neste caso, os nós são identificados, e exige o conhecimento de vizinhança relativa. Cheng et al. 2008 propõem o *Rainbow* utilizando gerenciamento de conteúdo no

nível do protocolo MAC por meio de um mecanismo chamado *innovation reporting*, com o qual o nó verifica se o dado será enviado caso seja recente e novo. As duas propostas diferem da nossa pelo uso de identificação e de redes sobrepostas.

Popescu e Liu 2006 modelam a comunicação de interesses em um vetor multidimensional, porém o trabalho se detém na análise matemática do problema ontológico. Ventresque et al. 2008 discutem o problema de ontologia inerente à WEB semântica, e propõem um mapeamento dos termos.

Spyropoulos et al. 2008 propõem uma técnica chamada *spray-and-wait* para *Disruption Tolerant Networks* (DTN), utilizando o mecanismo *Store-carry-forward* e negociação no envio da mensagem para o vizinho, garantindo entrega neste trecho do encaminhamento. A abordagem adotada em nosso caso é, também, diferente da adotada em redes DTN por não utilizar armazenamento, nem confirmação de entrega.

A REPI tem outras vantagens potenciais sobre as implementações existentes, que utilizam apenas probabilidade, e sobre as implementações que utilizam roteamento por meio de redes sobrepostas, como no caso das redes baseadas em conteúdo. No primeiro caso, a principal vantagem é o uso da informação como parâmetro de decisão para roteamento, diferente de usar uma semente aleatória como faz o *Gossip*, por permitir que o encaminhamento seja realizado pelo interesse do usuário. No segundo caso, nós não utilizamos endereçamento convencional e, portanto, eliminamos a necessidade de identificação dos nós e o *overhead* de manutenção do roteamento na rede.

3. Rede de Endereçamento por Interesses

Como visto, em uma região de ocorrência de um desastre natural, é fundamental a comunicação ágil e determinada por interesses momentâneos, formando grupos de socorro rapidamente e permitindo que as pessoas se manifestem, enviando mensagens atualizadas da situação local.

O uso da mensagem contendo interesses permite formação de grupos, encaminhamento e endereçamento, deslocando a decisão de encaminhamento da rede (dos equipamentos) para os usuários.

Uma REPI é constituída, basicamente, por mensagens contendo interesses em um prefixo, o equivalente ao cabeçalho em uma mensagem padrão, como apresentado na Figura 1. Este prefixo foi dividido em duas partes: a primeira parte, chamada B, contém dados biométricos dos usuários, seguindo uma distribuição normal multivariada; a segunda parte, chamada Y, assume que os interesses dos usuários obedecem a uma distribuição Zipf [Li 1992].

Desta forma podemos expressar o prefixo analiticamente de acordo com a Equação 1:

$$P = (B, Y) = (B_1, B_2, B_3...B_k; Y_1, Y_2, Y_3...Y_m) \; \forall \; k, m \; \in \; \mathbb{N}^* \tag{1}$$

Onde, para a parte B de um prefixo real com valores a, b, c, d, e, e para a parte Y, com valor y, de acordo com a Equação 2:

$$P = (B, Y) = (a, b, c, d, e; y) \tag{2}$$

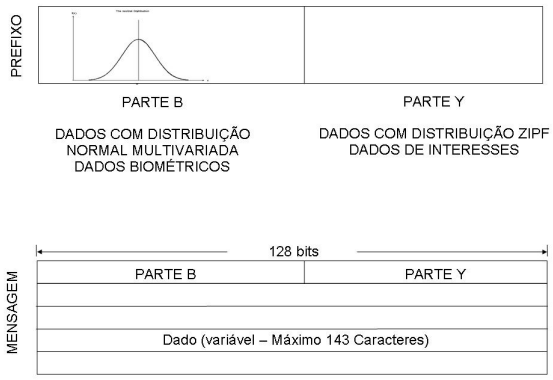


Figura 1. Prefixo e mensagem endereçada por interesses.

A parte B (a, b, c, d, e), dados biométricos, é utilizada para encaminhamento da mensagem, usando estes dados como critério de escolha de encaminhamento pelo nó por meio de um filtro de casamento. A parte Y (y), dados de interesses, é utilizada para endereçamento da mensagem. A parte B permite a colaboração entre os nós para encaminhamento, pois um nó, mesmo sem interesse y, poderá encaminhar mensagens provenientes de nós com interesse y. As partes B e Y podem ser utilizadas para identificação única da mensagem, devido à pequena probabilidade de casamento de todos os campos entre nós distintos.

Como um exemplo de encaminhamento considere a tabela 1. O nó origem n_1 envia mensagem para Defesa Civil (nós n_3 e n_5). Sendo nv os vizinhos[2] no raio de alcance para cada nó, o nó n_2 encaminha a mensagem recebida de n_1 por haver casamento no campo $Cabelo = castanho$ (V - verdadeiro). O nó n_3 encaminha a mensagem recebida de n_2 porque $Idade = 28$, ainda, n_3 é endereço da mensagem por $Interesse = DefesaCivil$. O nó n_4, apesar de ser vizinho de n_3, possui casamento dos campos falso (F) e, portanto, descarta a mensagem, enquanto n_5 recebe a mensagem de n_3, por ter interesse na mensagem ($interesse = DefesaCivil$).

Tabela 1. Tabela de encaminhamento

n	**nv**	**B**					**Y**	**Origem**	**Enc.**	**End.**
		Cabelo	**Olho**	**Sexo**	**Peso**	**Idade**	**Interesse**			
1	2	*castanho*	castanho	M	75	25	Defesa Civil	V		
2	3	*castanho*	verde	F	65	*28*			V	F
3	4,5	louro	castanho	M	70	*28*	Defesa Civil		V	V
4		castanho	azul	F	65	27			F	F
5		preto	azul	M	80	40	Defesa Civil		F	V

De acordo com o casamento do conteúdo da parte B e da parte Y das mensagens que chegam a um nó qualquer e seus prefixos, pode-se obter: (1) $B = 1$ significa o casamento dos campos B do prefixo da mensagem com os campos B do prefixo do nó que

[2]No contexto deste trabalho, nós vizinhos são aqueles que estão dentro do raio de alcance de potência emitida por um nó

recebe a mensagem; (2) $B = 0$ significa nenhum casamento dos campos B dos prefixos; (3) $Y = 1$ significa o casamento dos campos Y do prefixo da mensagem com os campos B do prefixo do nó que recebe a mensagem; e (4) $Y = 0$ significa nenhum casamento dos campos B dos prefixos. É possível definir, então, 4 tipos de nós nesta implementação da REPI:

- $B = 1$ e $Y = 1$ - Nós colaboradores e, receptores ou transmissores;
- $B = 0$ e $Y = 1$ - Nós receptores ou transmissores;
- $B = 1$ e $Y = 0$ - Nós colaboradores;
- $B = 0$ e $Y = 0$ - Nós não participantes da REPI.

A parte B da mensagem é uma expansão da classe de algoritmos probabilísticos de encaminhamento de mensagem da qual fazem parte os algoritmos do tipo *Gossip*, onde a escolha de encaminhamento é obtida por probabilidade de diversas formas, por exemplo, na escolha do vizinho que encaminhará a mensagem. No caso da REPI, os nós contêm dados com determinada probabilidade e estes dados podem ser definidos na fabricação dos equipamentos ou definidos pelo usuário. A escolha de qual vizinho irá encaminhar a mensagem é definida sob demanda baseado nestes dados. Esta escolha difere da do *Gossip* devido ao encaminhamento ocorrer na chegada da mensagem em um nó, o qual decide se encaminha ou não a mensagem de acordo com um filtro de casamento. Esta diferença, no caso da REPI, aumenta o número de mensagens na rede, possibilitando por um lado, mais caminhos de envio da mensagem e consequentemente uma maior taxa de entrega. e por outro lado, aumenta o custo de envio da mensagem.

Quanto à segunda parte do prefixo, esta é responsável pelo endereçamento de duas formas distintas: a primeira, pela possibilidade do usuário digitar qualquer sequência de caracteres, como uma senha. A mensagem somente será mostrada para o equipamento que contenha tal senha. Esta decisão não impede que outros usuários possam "escutar" o canal de comunicação e descobrir o conteúdo das mensagens, e também requer o uso de um dicionário, para reduzir ou eliminar problemas inerentes ao casamento dos termos. Entretanto, é possível utilizar o mecanismo de chave pública/privada para garantir a segurança da informação contida no payload, evitando que os equipamentos que encaminhem a mensagem tenham acesso ao conteúdo.

A segunda forma de endereçamento é devido à probabilidade de escolha ao acaso das mesmas palavras por usuários diferentes seguir a distribuição Zipf [Li 1992], o que torna a probabilidade de casamento entre as palavras, em prefixos de diferentes usuários, muito pequena caso o filtro de casamento seja restrito. Esta propriedade pode identificar unicamente um usuário.

Resumindo, as principais funcionalidades da REPI são:

- Fazer broadcast de mensagens contendo interesses;
- Decidir o encaminhamento das mensagens sob demanda, utilizando um filtro de casamento;
- Formar grupos de interesses sob demanda;
- Endereçar as mensagens por interesses.

Praticamente, o usuário pode inserir seus dados biométricos no prefixo. Para uma amostra grande, estas variáveis possuem uma distribuição de probabilidade normal, porém, para formação de grupos e encaminhamento da mensagem o uso de variáveis

quaisquer com distribuição de probabilidade normal terá o mesmo efeito. Obviamente, para uma amostra pequena esta aproximação não é válida, e o sistema poderá inserir automaticamente dados garantindo a distribuição.

Na atual implementação da REPI, o equipamento utiliza somente a parte B do prefixo para encaminhamento. Porém, a parte Y também poderia ser utilizada. Ainda, qualquer distribuição de probabilidade pode ser utilizada para a formação do prefixo, tanto da parte B quanto da parte Y.

4. Avaliação Experimental

Nesta Seção, nós apresentamos a avaliação experimental preliminar da REPI, utilizando uma rede ad hoc de 20 nós (equipamentos TmoteSky [Polastre 2005]) distribuídos de acordo com a Figura 2, em um ambiente fechado, comunicando-se por ZigBee com a programação do protocolo REPI utilizando o sistema operacional TinyOS.

4.1. Experimentos e Resultados

O tamanho de mensagem usado foi de 116 Bytes com 41 Bytes para o prefixo e 73 Bytes para payload. De acordo com o experimento, a parte B do prefixo variou de um a cinco campos, cada campo com distribuição de probabilidade normal, selecionados automaticamente para cada usuário. Na parte Y, os interesses foram selecionados automaticamente de um dicionário de palavras. Para isolar o funcionamento do algoritmo nos nós da sobrecarga da instrumentação, esta foi desviada pela rede cabeada para o Sistema de Automação, Monitoração e Configuração de Redes Ad hoc (SAMCRA) para não haver interferência com a comunicação sem fio, assim como, não houve processamento local dos resultados [Granja et al. 2010].

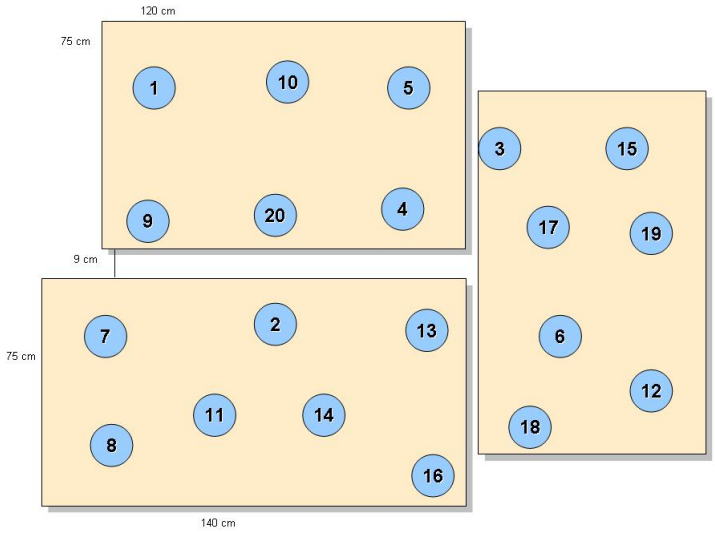


Figura 2. Distribuição dos vinte nós para os experimentos nº 1, nº 2 e nº 3.

O desempenho da REPI foi comparado com os dos algoritmos *Flooding* e *Gossip*, utilizando as métricas: (1)Taxa de Entrega de Mensagens (TEM), que mede o número de mensagens com interesse y que foram entregues ao nó com interesse em y; (2) Custo de

Entrega de Mensagens (CEM), definida como o número de mensagens com interesse y dividido pelo número de mensagens totais encaminhadas; (3) Taxa de Perda de Mensagens (TPM), que mede o número de mensagens perdidas; e a nova métrica (4) Número de Nós Colaboradores (NC), igual ao número de nós sem interesse em y que encaminharam a mensagem dos transmissores aos receptores. NC determina quantos nós sem interesse numa mensagem de interesses ($Y = 0$) participaram no seu encaminhamento ($B = 1$), avaliando a colaboração.

Três experimentos representativos foram realizados para avaliar a REPI, executados 20 vezes. O experimento nº 1, avalia os efeitos da distribuição de rádio-frequência (RF) dos nós na conectividade, no multihop e na contenção. Neste experimento, um nó transmite 1000 mensagens e a taxa de entrega de cada nó e as conexões com um salto são armazenadas. O experimento nº 2, avalia a contenção pelo protocolo ZigBee. Neste caso, variando o número de nós transmitindo 100 mensagens em intervalos de tempo variados, a taxa de perda de mensagens é armazenada. O experimento nº 3, mede o custo do protocolo REPI e a Taxa de Entrega de Mensagens (TEM) para os nós com interesse, avaliando o impacto da colaboração dos nós sem interesse na entrega das mensagens, para os destinos. Para isso, variou-se o número de nós transmitindo 1000 mensagens no intervalo de tempo de contenção mínima, medindo-se a TEM e o CEM.

No experimento nº 1, apresentado na Figura 3, o nó 4 transmitiu 1000 mensagens em intervalo de 2 a 4 segundos, aleatoriamente, para todos os outros nós. O número de campos na parte B do prefixo foi constante e igual a 5, com distribuição normal multivariada, e todos os nós tinham o mesmo interesse armazenado na parte Y. O percentual de mensagens recebidas pelos nós, diretamente ou por multihop, com número de saltos médio de 3 é apresentado em cada nó. Pode-se notar que a média e desvio padrão da TEM foi de $93,55 \pm 1,38$.

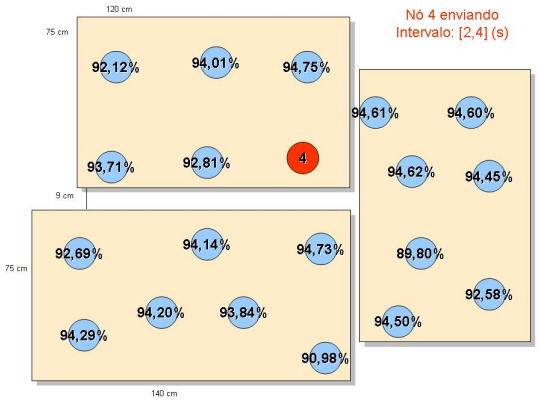


Figura 3. Taxa de Entrega de Mensagens (TEM) por nó para o experimento nº 1.

Ainda nesse experimento, foram medidos os percentuais de mensagens entregues em um salto, com o objetivo de avaliar a a distribuição de RF em função da distância relativa entre o nó 4 e os nós restantes, como ilustrada pelo grafo de conectividade da Figura 4.

Na Figura 4 são identificadas três regiões distintas de acordo com a probabilidade de entrega de mensagem em um salto: (1) região hachurada diagonal, com ocorrência de

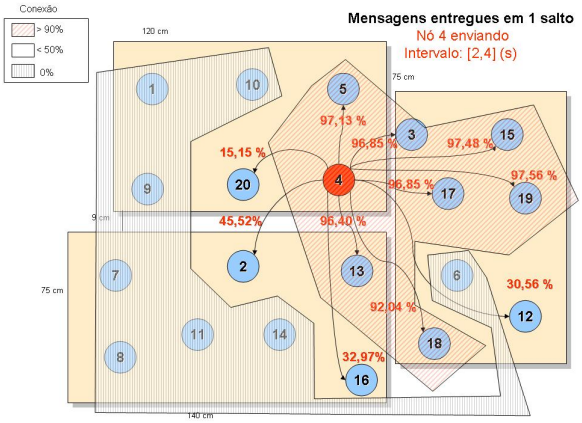


Figura 4. Grafo de conectividade para os experimentos nº 1, nº 2 e nº 3.

mais de 90% de conexões; (2) região sem hachurado, com ocorrência de menos de 50% de conexões, indicando instabilidade; e (3) região em hachurado vertical, com nenhuma ocorrência de conexões. Observa-se que o nó 6 não recebeu mensagens do nó 4, embora próximos fisicamente, enquanto que o nó 12, mais distante, recebeu mensagens do nó 4.

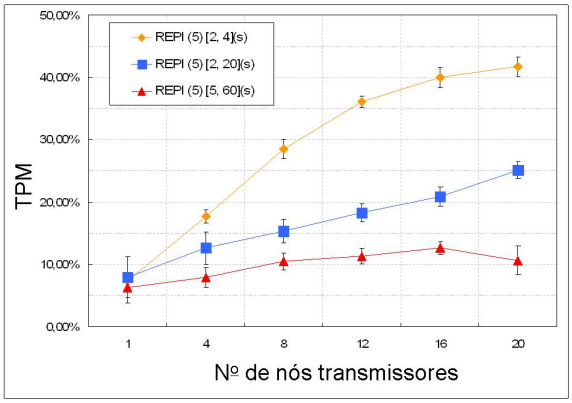


Figura 5. Taxa de Perda de Mensagens (TPM) x Variação do Número de nós transmissores

No experimento nº 2, a TPM foi medida, variando o número de nós transmitindo. O intervalo de tempo entre o envio das mensagens variou em três intervalos: $[2, 4]$; $[2, 20]$; $[5, 60]$ (s), com cinco campos no prefixo. Os resultados são mostrados na Figura 5. Com o intervalo de $[2, 4]$(s) foi obtida uma taxa de perda crescente com o aumento do número de nós transmissores, em torno de 10% para um nó transmitindo e 42% para os 20 nós. Por outro lado, com um intervalo de $[5, 60]$(s) a taxa de perda de mensagens se manteve constante e aproximadamente igual a 10%.

No experimento nº 3, cuja topologia é apresentada na Figura 6, foram avaliados o CEM e a TEM, com 5 nós transmissores, $(1, 7, 8, 9, 11)$, e 5 nós receptores, $(3, 5, 12, 15, 19)$, sendo os outros nós, colaboradores ou não participantes. O intervalo entre as mensagens foi de $[2, 4]$ (s).

Os valores de CEM e TEM, apresentados na Figura 7, foram medidos para REPI,

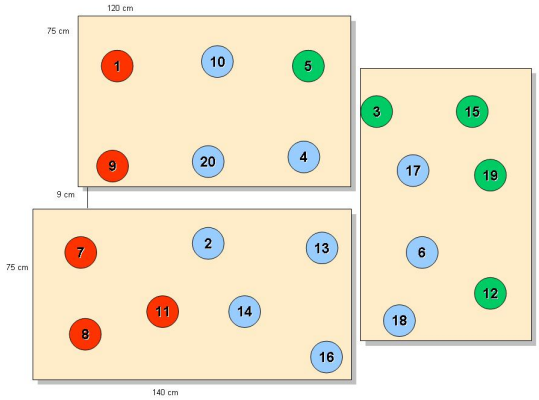


Figura 6. Topologia do experimento nº 3 (nós 1, 7, 8, 9, 11 transmissores de mensagens com interesse y e nós 3, 5 12, 15, 19 com interesse y).

variando o número de campos na parte B do prefixo entre $(1, 2, 4, 5, 6, 8)$ (REPI(1), REPI(2), etc.), *Flooding*, REPI$^-$, para quatro campos na parte B do prefixo, sem colaboração, e *Gossip*, com probabilidades de encaminhamento de mensagem em 25%, 50%, 75% e 85%[3].

Nota-se que os valores de CEM e TEM para REPI, com qualquer variação de campos na parte B do prefixo, são melhores que os do *Flooding*. Comparando-se TEM$_{Gossip(75\%)}$, igual a $75,65\%$, com TEM$_{REPI(5)}$, igual a $76,88\%$, verifica-se que ambos os valores são praticamente iguais. Quanto ao CEM$_{Gossip(75\%)}$, igual a $12,14$, comparado ao CEM$_{REPI(5)}$, igual a $13,70$, verifica-se que CEM$_{REPI(5)}$ é $10,48\%$ maior. Para a REPI$^-$, a TEM é igual a $33,88\%$ e o CEM é igual a $7,49$.

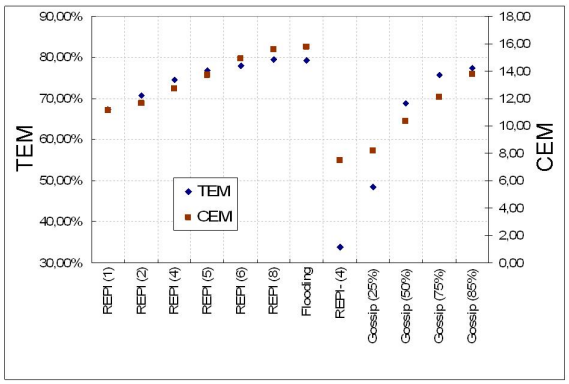


Figura 7. Comparação dos protocolos REPI, *Flooding*, REPI$^-$ e *Gossip*.

No experimento nº 3 também foi avaliada a colaboração, apresentada na Figura 8. O número de nós colaboradores é, para um campo, $13,6$, para 2 campos, $14,7$, enquanto para 4, 5, 6 e 8 campos, estável em $14,9$.

Ainda, no experimento nº 3, foi avaliado o número de saltos para a variação de

[3]o valor de probabilidade em que o Gossip atende quase todos os nós em qualquer execução fica no intervalo de $[60\%, 80\%]$ [Haas et al. 2006].

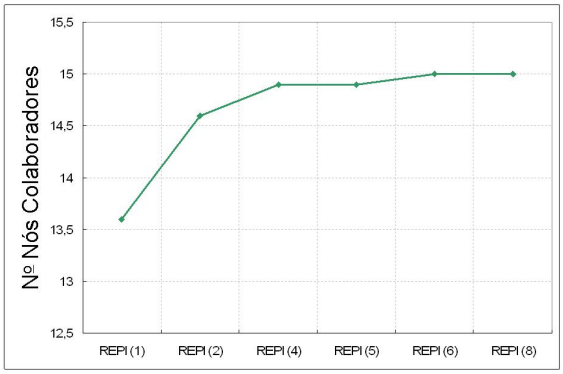


Figura 8. Colaboração na REPI x número de campos B do prefixo.

campos na parte B do prefixo da REPI, *Flooding*, REPI$^-$ e *Gossip* (25%, 50%, 75% e 85%), apresentado na Figura 9. Observa-se que o número de saltos para todas as mensagens na rede possui valor médio 3, para a REPI com os campos B do prefixo variando de 1 a 8, enquanto que o número de saltos para as mensagens entregues aos nós com interesses possui valor médio $2,4$ para a mesma variação de campos. Os mesmos valores foram encontrados para o *Flooding* e para o *Gossip* 75% e 85%, enquanto o número de saltos para o *Gossip* 25% foi igual a $2,5$ e para 50% foi igual a $2,75$. Nota-se que para quatro campos sem colaboração, os valores são iguais a $2,1$ para total de mensagens na rede e e $1,9$ para mensagens entregues, mas com TEMs inferiores.

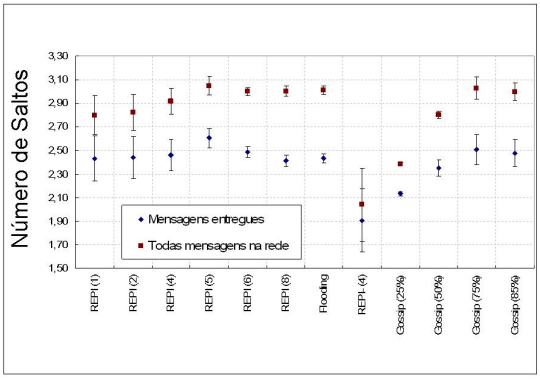


Figura 9. Número médio de saltos no experimento nº 3.

4.2. Discussão dos Resultados

De acordo com o experimento nº 1, a taxa de entrega de mensagens (TEM) da REPI, com somente um nó enviando e todos os outros nós com interesse na mensagem, foi alta e igual a $93,55 \pm 1,38\%$. Dado que em um sistema se comunicando por ZigBee a taxa de perda é de 10%, os alto valores encontrados são significativos.

Também, os resultados mostraram três regiões distintas em RF, obrigando à ocorrência de multihop para envio da mensagem, e que o número de saltos, considerando somente a RF, deveria ser no máximo 3. Notou-se ainda, a imprevisibilidade da

comunicação por RF, na medida em que a comunicação com vizinhos próximos pode ser preterida pelos mais distantes, devido à distribuição das ondas eletromagnéticas em um ambiente fechado.

O aumento no intervalo de envio ocasiona uma menor perda de mensagens, devida à diminuição da contenção gerada pelas mensagens trafegando na rede. Este intervalo é mais realista pelo tempo de digitação e envio da mensagem por um usuário. Porém, se houver um número grande de usuários, a contenção no protocolo ZigBee deve ser considerada. Como esta limitação depende apenas do protocolo, equipamentos utilizando protocolos mais eficientes podem se beneficiar da REPI. Por exemplo, em um ambiente de desastre natural, um canal de emergência (SOS), pode ser configurado automaticamente e a mensagem enviada em intervalos de $[5, 60]$(s).

Com relação à TEM e ao CEM, REPI com 5 campos é comparável ao *Gossip* 75%. Porém, a REPI tem potencial para melhorar com o aumento do número de nós. Ainda, o uso de informação dos interesses pelas aplicações, em muitos casos, é mais interessante do que a pura probabilidade do *Gossip*. A colaboração pode ser alta utilizando-se filtro de casamento com "pelo menos um", interesse coincidente, o que permitirá alta probabilidade de casamento, consequentemente, um maior número de nós encaminhando as mensagens. Para poucos nós, não há necessidade de muitos campos na parte B do prefixo, porém, é necessário considerar o tipo de filtro de casamento implementado. Pelos resultados muito baixos de TEM da REPI$^-$, vê-se que a REPI depende da colaboração para alcançar TEM com valores próximos aos do *Gossip*.

A colaboração foi alta, igual a $13,6$ para um campo na parte B do prefixo, aumentando para $14,7$ com 2 campos, e se estabilizando em $14,9$ para 4,5, 6 e 8 campos. Para um total de 20 nós, sendo cinco transmissores e cinco receptores, nota-se que para se alcançar 15 nós colaboradores, os próprios nós transmissores foram colaboradores uns dos outros. Pode-se observar ainda, que 4 campos na parte B do prefixo é suficiente para atender 20 nós, utilizando o filtro de "pelo menos um". Para avaliar o impacto do aumento de campos na parte B do prefixo na REPI, é necessário incluir mais nós, aumentando a densidade de nós na rede. Por outro lado, avaliar outras possibilidades de casamento no filtro pode tornar a colaboração mais eficiente, p.ex., tornando o filtro de casamento adaptável ao fluxo de mensagens em cada nó.

Ainda, as mensagens entregues aos nós com interesses possuem um atraso menor do que as mensagens totais na rede. Pode-se esperar que com o aumento do número de nós na rede este atraso aumente, mas seja menor do que o atraso médio das mensagens totais na rede. Na REPI, as conexões ocorrem quando as mensagens são enviadas. Não existe um mecanismo de *store-carry-forward*. Se não existir nó com a parte B do prefixo com pelo menos um campo casando, ou não existir vizinho, a mensagem é descartada ou perdida, devido a não haver mecanismo de confirmação entre os nós vizinhos. Naturalmente, o modelo permite o armazenamento das mensagens nos equipamentos, proporcionando o envio das mensagens quando ocorrer um encontro oportuno entre nós que estejam fora da área de cobertura um do outro, tornando a REPI uma DTN.

5. Conclusões

Nós mostramos a viabilidade do uso de interesses dos usuários como endereçamento de mensagens em uma rede ad hoc, e que o desempenho do endereçamento por interesses se

iguala aos alcançados pelo algoritmo *Gossip*, com a vantagem de se utilizar informação útil como cabeçalho da mensagem. A Rede de comunicação Endereçada por Interesses (REPI) pode ser utilizada em ambientes onde não há infra-estrutura de telecomunicações ou, essa foi atingida (p.ex., desastre natural), permitindo a formação de grupos de interesses e o encaminhamento e endereçamento de mensagens, com foco no usuário. Existem limitações intrínsecas às redes ad hoc, como segurança, garantia de entrega, e limitações específicas da REPI, como o projeto do prefixo, o uso de melhores distribuições para representar os dados B e Y, mas que podem ser contornadas pelas aplicações, de acordo com as necessidades do ambiente de utilização.

6. Trabalhos Futuros

A REPI, por ser uma nova e promissora rede ad hoc, há necessidade de aplicá-la e avaliá-la em um número maior de nós, verificando sua escalabilidade, bem como a mobilidade. Essas duas características de redes ad hoc serão avaliadas por meio de simuladores tradicionais (p.ex., NS-3 ou Glomosim). O uso da rede REPI em redes cabeadas, alterando o modelo dos nós P2P tradicional, está em desenvolvimento.

7. Agradecimentos

Este trabalho foi financiado pelas Agências: FINEP, CNPq, CAPES.

Referências

Awad, A., Shi, L., German, R., e Dressler, F. (2009). Advantages of virtual addressing for efficient and failure tolerant routing. In *Sensor Networks, IEEE/IFIP WONS 2009, Snowbird, UT*, pages 111–118.

Borcea, C., Gupta, A., Kalra, A., Jones, Q., e Iftode, L. (2007). The mobisoc middleware for mobile social computing: challenges, design, and early experiences. In *MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, pages 1–8, ICST, Brussels, Belgium. ICST.

Carzaniga, A. e Hall, C. P. (2006). Content-based communication: a research agenda. In *SEM '06: Proceedings of the 6th international workshop on Software engineering and middleware*, pages 2–8, New York, NY, USA. ACM.

Cheng, C., Kung, H. T., kwan Lin, C., yung Su, C., e Vlah, D. (2008). Rainbow: A wireless medium access control using network coding for multi-hop content distribution. In *Proc. MILCOM 08, San Diego, CA*.

Dutra, R. C. e Amorim, C. L. (2010). Modelo de comunicação endereçada por interesses. Technical report, Relatório Técnico ES 733 / PESC - COPPE - UFRJ.

Granja, R. S., Dutra, R. C., Moraes, H. F., e Amorim, C. L. (2010). Samcra: Um sistema para avaliação experimental de redes ad hoc. submetido à Seção de Ferramentas - XXVIII SBRC - SBC.

Haas, Z. J., Halpern, J. Y., e Li, L. (2006). Gossip-based ad hoc routing. *IEEE/ACM Trans. Netw.*, 14(3):479–491.

Heer, T., Gotz, S., Rieche, S., e Wehrle, K. (2006). Adapting distributed hash tables for mobile ad hoc networks. In *PERCOMW '06: Proceedings of the 4th annual IEEE international conference n Pervasive Computing and Communications Workshops*, page 173, Washington, DC, USA. IEEE Computer Society.

Intanagonwiwat, C., Govindan, R., e Estrin, D. (2000). Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67, New York, NY, USA. ACM.

Kortuem, G., Segall, Z., e Thompson, T. G. C. (1999). Close encounters: Supporting mobile collaboration through interchange of user profiles. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 171–185, London, UK. Springer-Verlag.

Li, W. (1992). Random texts exhibit zipf's-law-like word frequency distribution. In *Proceedings of IEEE Transactions on Information Theory*, volume 38(6), pages 1842–1845.

Polastre, J. (2005). Tmotesky. Technical report, MoteIV.

Popescu, G. V. e Liu, Z. (2006). Network overlays for efficient control of large scale dynamic groups. In *DS-RT '06: Proceedings of the 10th IEEE international symposium on Distributed Simulation and Real-Time Applications*, pages 135–142, Washington, DC, USA. IEEE Computer Society.

Rantanen, M., Oulasvirta, A., Blom, J., Tiitta, S., e Mäntylä, M. (2004). Inforadar: group and public messaging in the mobile context. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 131–140, New York, NY, USA. ACM.

Rao, R. R., Eisenberg, J., e Schmitt, T. (2007). *Improving Disaster Management: The Role of IT in Mitigation, Preparedness, Response, and Recovery*. The National Academies Press.

Sohrabi, K., J., G., Ailawadhi, V., e Pottie, G. J. (2000). Protocols for selforganization of a wireless sensor network. *IEEE Personal Communications*, 7:16–27.

Spyropoulos, T., Psounis, K., e Raghavendra, C. S. (2008). Efficient routing in intermittently connected mobile networks: the multiple-copy case. *IEEE/ACM Trans. Netw.*, 16(1):77–90.

Ventresque, A., Cazalens, S., Lamarre, P., e Valduriez, P. (2008). Improving interoperability using query interpretation in semantic vector spaces. In Hauswirth, M., Koubarakis, M., e Bechhofer, S., editors, *Proceedings of the 5th European Semantic Web Conference*, LNCS, Berlin, Heidelberg. Springer Verlag.

Zahn, T. e Schiller, J. (2006). *DHT*-based unicast for mobile ad hoc networks. In *PERCOMW '06: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, page 179, Washington, DC, USA. IEEE Computer Society.

Índice por Autor