Número 01
Volume 04
Dezembro / 2011
ISSN 1983-4217

REVISTA BRASILEIRA DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS

Thaís Vasconcelos Batista
Lisandro Zambenedetti Granville
Cristiano Bonato Both

Número 1
Volume 4
Dezembro de 2011
ISSN 1983-4217

Revista Brasileira de Redes de Computadores e Sistemas Distribuídos

Laboratório Nacional de Redes de Computadores

Sociedade Brasileira de Computação

Revista Brasileira de Redes de Computadores e Sistemas Distribuídos /
Laboratório Nacional de Redes de Computadores (LARC), Comissão Especial de Redes de Computadores da Sociedade Brasileira de Computação.
Rio de Janeiro: LARC; SBC, 2011.
v. 4, n. 1 (jul-dez, 2011)
v. : il. cm.
Semestral.
Texto em português ou inglês.
ISSN 1983-4217

1. Redes de computadores 2. Sistemas distribuídos. I. Laboratório Nacional de Redes de Computadores (LARC) II. Sociedade Brasileira de Computação. Comissão Especial de Redes de Computadores. III. Título.

CDD 004.65

Laboratório Nacional de Redes de Computadores (LARC)
Contato: Luciano Paschoal Gaspary
Instituto de Informática - UFRGS
Av. Bento Gonçalves, 9500 - Porto Alegre, RS
Caixa Postal 15064 - CEP 91501-970

Sociedade Brasileira de Computação
Contato: Carlos André Guimarães Ferraz
Av. Bento Gonçalves, 9500 - Porto Alegre, RS
Caixa Postal 15012 - CEP 91501-970

Conteúdo

Carta dos Editores 7
Thais Vasconcelos Batista, Lisandro Zambenedetti Granville, Cristiano Bonato Both

Artigos

Roteamento, Atribuição e Adaptação Conjunta de Largura de Canais em Redes em Malha sem Fio IEEE 802.11 9
Celso Barbosa Carvalho, José Ferreira de Rezende

Detecção de *Spams* Utilizando Conteúdo Web Associado a Mensagens 19
Marco Ribeiro, Leonardo Teixeira, Pedro Guerra, Adriano Veloso, Wagner Meira Jr., Dorgival Guedes, Cristine Hoepers, Klaus Steding-Jessen, Marcelo Chaves

Mapeamento de Redes Virtuais em Substratos de Rede 29
Gustavo P. Alkmim, Daniel M. Batista, Nelson L. S. da Fonseca

Uma Análise do Impacto da Elasticidade no Lucro de Provedores de Computação na Nuvem 39
Rostand Costa, Francisco Brasileiro, Guido Lemos, Dênio Mariz

RouteFlow: Roteamento Commodity Sobre Redes Programáveis 51
Marcelo Nascimento, Christian Rothenberg, Rodrigo Denicol, Marcos Salvador, Maurício Magalhães

Carta dos Editores

A Revista Brasileira de Redes de Computadores e Sistemas Distribuídos é um periódico promovido conjuntamente pelo Laboratório Nacional de Redes de Computadores (LARC) e Sociedade Brasileira da Computação (SBC) através de sua Comissão Especial em Redes de Computadores e Sistemas Distribuídos (CE-RESD). Como resultado das ações da comunidade nacional de pesquisa em redes de computadores e sistemas distribuídos, a Revista tem por objetivo se estabelecer como um veículo de divulgação dos avanços científicos e tecnológicos da área, e assim estender o atual alcance do prestigioso e tradicional Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC).

Neste sexto número, a Revista apresenta versões atualizadas e estendidas de cinco artigos dentre os artigos ganhadores do prêmio de melhor trabalho da 29ª edição do SBRC, realizado em 2011 em Campo Grande. O primeiro artigo versa sobre roteamento em redes em malha sem fio. O segundo artigo versa sobre detecção de *spams* utilizando conteúdo Web. O terceiro artigo versa sobre mapeamento de redes virtuais. O quarto artigo versa sobre o impacto da elasticidade no lucro de provedores em computação nas nuvens. Finalmente, o quinto artigo versa sobre roteamento sobre redes programáveis.

No primeiro artigo, "Roteamento, Atribuição e Adaptação Conjunta de Largura de Canais em Redes em Malha sem Fio IEEE 802.11", de Celso Barbosa Carvalho, José Ferreira de Rezende, os autores apresentam uma extensão de um modelo de programação linear para roteamento, atribuição e adaptação conjunta de largura dos canais de comunicação como forma de aumentar a capacidade das redes em malha sem fio. Os resultados descritos no artigo mostram um aumento da capacidade para as redes sem fio que utilizam a extensão do modelo proposto.

No segundo artigo, "Detecção de *Spams* Utilizando Conteúdo Web Associado a Mensagens" de autoria de Marco Ribeiro, Leonardo Teixeira, Pedro Guerra, Adriano Veloso, Wagner Meira Jr., Dorgival Guedes, Cristine Hoepers, Klaus Steding-Jessen e Marcelo Chaves, apresenta uma estratégia de detecção de *spams* que explora o conteúdo das páginas Web. O artigo discute a utilização de um algoritmo de aprendizado para extrair as informações para a detecção de *spam*. Esse algoritmo utiliza páginas Web que foram coletadas através de uma metodologia de coleta proposta nesse trabalho. Os resultados mostram que a utilização de informações das páginas Web melhora significativamente a classificação de *spams* e *hams*.

No terceiro artigo, "Mapeamento de Redes Virtuais em Substratos de Rede", de Gustavo P. Alkmim, Daniel M. Batista, Nelson L. S. da Fonseca, os autores apresentam dois novos algoritmos para o problema de mapeamento de redes virtuais em substratos de rede que minimizam a utilização dos recursos. Os resultados mostram que os algoritmos encontram soluções em tempo viável para diversos cenários de requisição de redes virtuais.

No quarto artigo, "Uma Análise do Impacto da Elasticidade no Lucro de Provedores de Computação na Nuvem", de Rostand Costa, Francisco Brasileiro, Guido Lemos, Dênio Mariz, os autores apresentam uma análise para identificar as razões que levam os provedores de *Infrastructure-as-a-Service* (IaaS) a imporem limites que restringem a utilidade de seus serviços para a execução de aplicações fortemente paralelizáveis. Os resultados apresentados no artigo mostram que aumentos no limite imposto têm um

grande impacto na lucratividade dos provedores.

Por fim, no quinto artigo, "RouteFlow: Roteamento Commodity Sobre Redes Programáveis" de autoria de Marcelo Nascimento, Christian Rothenberg, Rodrigo Denicol, Marcos Salvador, Maurício Magalhães, apresenta o projeto RouteFlow que trata-se de uma arquitetura de roteamento IP para combinar o alto desempenho de hardware de prateleira com a flexibilidade de uma pilha de roteamento utilizada remotamente em computadores de uso geral. O resultado mostrado neste artigo corresponde a uma nova solução de roteamento IP com perspectivas promissoras do ponto de vista do custo e da flexibilidade. Além disso, a avaliação do protótipo apresentada no artigo comprova a viabilidade da arquitetura proposta.

Thais Vasconcelos Batista
Editor
Universidade Federal do Rio Grande do Norte
Departamento de Informática e
Matemática Aplicada
Av. Sen. Salgado Filho, 3000
59072-970 – Natal, RN

Lisandro Zambenedetti Granville
Editor
Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500
Caixa Postal 15064
91501-970 – Porto Alegre, RS

Cristiano Bonato Both
Editor Associado
Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500
Caixa Postal 15064
91501-970 – Porto Alegre, RS

Roteamento, Atribuição e Adaptação Conjunta de Largura de Canais em Redes em Malha sem Fio IEEE 802.11

Celso Barbosa Carvalho, José Ferreira de Rezende

GTA - PEE - COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68.504 – 21.945-970 – Rio de Janeiro – RJ – Brasil
{celso,rezende}@gta.ufrj.br

Abstract

Wireless Mesh Networks (WMNs) have become increasingly important due to its use as a network backbone or access to last mile users. In this context, increase the capacity of WMNs is a necessary topic. A large number of researches in this area consider the use of joint routing and channel assignment in order to increase the capacity in these networks. Recent studies show that the throughput performance of WMNs can be improved with the use of transmission channels of different widths. In this paper we extend a Linear Programming (LP) model and we consider joint routing, channel assignment and channel width adaptation in order to increase the capacity of WMNs. The results show an improvement of the capacity in networks that use the investigated approach.

Keywords: Wireless mesh networks, routing, channel assignment, channel width adaptation, capacity

Resumo

As redes em malha sem fio (Wireless Mesh Networks - WMNs) têm ganho considerável importância devido a sua crescente utilização como backbone de rede ou para acesso a usuários de última milha. Neste contexto, o aumento da capacidade das WMNs trata-se de uma questão de grande interesse. Uma grande parte das pesquisas nesta área considera o uso de roteamento e atribuição conjunta dos canais como forma de aumentar a capacidade destas redes. Estudos recentes mostram que o desempenho de vazão das WMNs pode ser melhorado com a utilização de canais de transmissão de diferentes larguras. Neste artigo estendemos um modelo de Programação Linear (Linear Programming-LP) e consideramos o roteamento, a atribuição e a adaptação conjunta de largura dos canais de comunicação como forma de aumentar a capacidade das WMNs. Os resultados mostram um aumento da capacidade para as redes que utilizam a abordagem investigada.

Palavras-chave: Redes em malha sem fio, roteamento, atribuição de canais, adaptação de largura de canal, capacidade

1. INTRODUÇÃO

As redes em malha sem fio (WMNs - *Wireless Mesh Networks*) têm sido cada vez mais utilizadas como solução para acesso sem fio a usuários de última milha ou como *backbone* de rede. Sendo assim, torna-se cada vez mais necessário aumentar a capacidade destas redes. Neste sentido, o roteamento e a atribuição de canais têm sido tema de diversas pesquisas. O uso do roteamento permite escolher caminhos, entre um par de roteadores origem e destino, que sofrem menor interferência. A atribuição de canais permite associar um canal específico a um enlace ou fluxo e desta forma eliminar ambas as interferências intra e inter-fluxo. Alguns trabalhos [18, 19] tratam da atribuição de canais como um tema isolado, outros trabalhos utilizam abordagens conjuntas que também envolvem roteamento como forma de gerenciar a atribuição de canais [1, 9].

Pesquisas recentes [4, 20, 10] mostram que a vazão das redes sem fio podem ser melhoradas ao utilizar canais de comunicação de diferentes larguras. Nesta situação, ao se utilizar canais de comunicação de menores larguras pode-se dividir a banda total de frequência em uma quantidade maior de canais não sobrepostos de maior ca-

pacidade conjunta, com maior alcance de comunicação e maior alcance de interferência. Em contrapartida, utilizar canais de maior largura permite aumentar a capacidade de um enlace individual e diminuir a distância de interferência.

No presente trabalho consideramos WMNs com múltiplos canais de diferentes larguras e roteadores equipados com múltiplos rádios de comunicação. Nosso estudo examina cenários de WMNs, tais como as do tipo TDMA (*Time Division Multiple Access*), onde há a atribuição de canais livre de conflito o que evita a interferência entre enlaces. No artigo estendemos uma formulação de Programação Linear (*Linear Programming*-LP) e consideremos roteamento multi-caminho, atribuição e escolha conjunta de largura de canais. O modelo estendido tem como base nossos trabalhos anteriores [2, 3] e que considera a existência de canais com diferentes larguras (ex: 5, 10 e 20MHz) e com diferentes alcances de transmissão/interferência. O modelo LP pode ser utilizado para determinar a capacidade das WMNs com as características comentadas.

Para abordar o tema em estudo, o artigo é organizado conforme a seguir: A Seção 2 lista trabalhos relacionados com atribuição de canais, roteamento e adaptação de largura de canal; a Seção 3 apresenta, respectivamente, os modelos de capacidade dos enlaces, alcances de transmissão, modelo de matrizes e modelo LP utilizados para simular as redes estudadas; a Seção 4 mostra os cenários de redes simulados e os resultados obtidos. Por fim, a Seção 5 apresenta as conclusões e relaciona trabalhos futuros.

2. TRABALHOS RELACIONADOS

Em [18, 19, 16, 17] elaboram-se modelos LP para atribuir canais com o objetivo de determinar a capacidade de WMNs com múltiplos canais. No entanto, estes trabalhos elaboram modelos LP que tratam da atribuição de canais como um tema isolado e, portanto, não abordam o uso de roteamento em conjunto.

Os autores em [8] propõem um algoritmo distribuído para realizar a atribuição de canais em cenários de Redes de Rádios Cognitivos (*Cognitive Radio Networks*-CRNs). O trabalho tem como objetivo reduzir as interferências com os Usuário Primários (*Primary Users*-PRs) da rede. Assim como nos trabalhos anteriores, não é tratado o roteamento em conjunto com a atribuição de canais.

Em [14] é elaborado um modelo LP que envolve ambos o roteamento e a atribuição de canais em WMNs. Em [13] desenvolve-se um modelo não linear inteiro e, em seguida, são aplicadas técnicas de linearização para realizar a atribuição de canais e determinar a capacidade da rede. Os mesmos autores em [12] utilizam uma abordagem algorítmica para tratar do problema. Nestes trabalhos, os autores consideram apenas a existência de canais ortogonais todos de mesma largura.

Em [4] realizam-se experimentos e identificam-se as vantagens de estreitar a largura do canal em redes locais IEEE 802.11 [7]. No trabalho é desenvolvido um mecanismo dinâmico de adaptação da largura de canal para melhorar o desempenho de uma rede local e, portanto, não é empregada a possibilidade de alteração da largura de canal em cenários de WMNs.

A pesquisa desenvolvida em [10] elabora um modelo e uma heurística de roteamento que envolvem programação linear em cenários de WMNs. Este trabalho, utiliza a possibilidade de dividir a banda total de frequências em diversos canais de diferentes larguras. No entanto, e assim como em [20], o trabalho desconsidera que os canais ao utilizarem diferentes larguras possuem diferentes alcances de transmissão. Além disto, em [10] é elaborado um modelo de interferência entre enlaces que não considera a existência das cliques de interferência (*Interference Cliques*-ICs [21]) e, que assim, pode gerar valores sub-ótimos de capacidade.

Em nossas propostas anteriores [2, 3] propomos métrica de roteamento e a possibilidade de alterar a largura de canal com o objetivo de aumentar a vazão das WMNs. Na proposta atual, aplicamos abordagem baseada em LP para determinar a capacidade das redes com as mesmas características.

Em relação aos trabalhos de [18, 19, 16, 17, 14, 13], estes não consideram a possibilidade de alteração de largura de canal em seus modelos LP. Em comparação ao trabalho de [10] nosso modelo LP considera em suas restrições, alcances de transmissão e capacidades dos enlaces dependentes da largura de canal utilizada. Além disto, tomamos como base a proposta de [18, 19] para modelar os ICs através da atribuição de canais livre de conflito, considerando a interferência par a par entre os enlaces da rede, conforme representado na Equação (13) da Seção 3.4.

Considerando os dois parágrafos anteriores e com base em [10, 19] nossa contribuição é estender o modelo LP de fluxo máximo para determinar a capacidade de WMNs IEEE 802.11 através do roteamento, atribuição e adaptação conjunta de largura de canais.

3. MODELAGEM DO SISTEMA

Nas Subseções a seguir apresentamos a modelagem utilizada para simular WMNs no trabalho. A Subseção 3.1 apresenta os cálculos utilizados para representar os tempos de transmissão e capacidade dos enlaces ao empregarem múltiplas larguras de canal e diferentes modulações. A Subseção 3.2 apresenta os cálculos de sensibi-

lidade mínima utilizados para determinar os alcances de comunicação entre roteadores e o alcance de interferência entre os enlaces da rede. Na Subseção 3.3 apresentamos o modelo de matrizes, utilizados para representar a existência de enlaces, as suas capacidades e as interferências existentes. Por fim, na Subseção 3.4 explicam-se as equações do modelo LP utilizado para determinar a capacidade da rede.

3.1. Capacidade dos Enlaces em Redes 802.11 Multi-Taxa e Multi-Largura de Canal

A modulação OFDM (*Ortogonal Frequency Division Multiplexing*) do 802.11 permite a utilização das larguras de canal de 5, 10 e 20MHz [7]. A seguir são apresentados os cálculos dos tempos de transmissão e capacidade dos enlaces na camada de Controle de Acesso ao Meio (*Medium Access Control*-MAC), ao serem empregados canais com diferentes larguras. As equações a seguir (Equações (1) e (2) [4]) foram utilizadas para determinar os valores das matrizes Cap e T e que serão apresentadas na Subseção 3.3.

Na Equação (1), a variável Cap ($bits/$s) é a capacidade obtida na camada MAC do IEEE 802.11 ao utilizar a largura de canal w_{ϖ} e o modo de transmissão m_n. Sendo que m_n representa uma combinação entre modulação e taxa de codificação de canal, conforme mostrado na Tabela 2 [7]. Na mesma equação, L_{data} é o tamanho do pacote de dados em bytes.

$$Cap^{w_{\varpi},m_n} = (8 \cdot L_{data})/T \quad (1)$$

$$T = CW + DIFS + T_{data} + SIFS + T_{ack} \quad (2)$$

$$T_{\alpha} = T_{pr} + T_{si} + T_{sym} \cdot ceil(\frac{L_{ser} + L_{tail} + 8L_{\alpha}}{N_{DBPS}}) + T_{SE} \quad (3)$$

Na Equação (2), $CW = [0, 31] * t_{slot}$ é a janela de contenção. Neste artigo, e assim como em [4], foi utilizado o valor médio da janela de contenção, o qual possui o valor $16 \times t_{slot}$. As variáveis $t_{slot} = 20\mu$s, $DIFS = 50\mu$s e $SIFS = 10\mu$s assumem valores conhecidos no 802.11[7]. As variáveis T_{data} e T_{ack} representam, respectivamente, os tempos de transmissão dos quadros de dados da camada MAC e quadro de reconhecimento (*acknowledgment*-ACK) ambos chamados de T_{α} na Equação (3). Nos cálculos deste artigo foi utilizado o mesmo valor de taxa de transmissão para ambos os quadros de dados da camada MAC e quadro de ACK.

Na Equação (3) e Tabela 1 [7], T_{pr}, T_{si} and T_{sym} representam, respectivamente, os tempos de transmissão do preâmbulo de sincronização, o tempo de transmissão do campo sinal que possui a função de indicar para a camada física qual o modo de transmissão utilizado e o tempo de duração do símbolo OFDM. As variáveis L_{SER} ($16\ bits$) e L_{tail} ($6\ bits$) representam o tamanho do campo serviço (reservado para aplicações futuras) e o campo *tail* marca o fim de um quadro OFDM. A variável L_{α} assume o valor L_{MAC} ($34\ bytes$) mais L_{data} ($bytes$) que correspondem ao cabeçalho MAC mais o campo de dados ou assume o valor L_{ACK} ($14bytes$) de um quadro de ACK.

Tabela 1. Tempos da camada física OFDM para as larguras de canal de 5, 10 e 20MHz [7]

Parâmetro	20MHz	10MHz	5MHz
T_{pr}	16µs	32µs	64µs
T_{si}	4µs	8µs	16µs
T_{sym}	4µs	8µs	16µs

A variável N_{DBPS} (coluna 4 da Tabela 2) representa o número de bits de informação transmitidos em um símbolo OFDM e seus valores dependem do modo de transmissão empregado [7].

Tabela 2. Modos de transmissão da camada física OFDM 802.11

Modo	Modulação	Taxa de Codificação	N_{DBPS}
m_1	BPSK	$1/2$	24
m_2	BPSK	$3/4$	36
m_3	QPSK	$1/2$	48
m_4	QPSK	$3/4$	72
m_5	16-QAM	$1/2$	96
m_6	16-QAM	$3/4$	144
m_7	64-QAM	$2/3$	192
m_8	64-QAM	$3/4$	216

3.2. Alcances dos Sinais Transmitidos com Diferentes Larguras de Canal

A mudança de largura de canal causa variações na sensibilidade mínima do receptor para cada um dos modos de transmissão da camada física OFDM do IEEE802.11. Os cálculos desta subseção foram utilizados para determinar o valor, em metros (Equação 7), do alcance de transmissão e interferência dos roteadores da rede. Estes valores de alcance foram utilizados para determinar a existência de enlaces de comunicação e a existência de interferência entre enlaces, representados respectivamente pelas matrizes E e IN da Subseção 3.3.

Conforme a Equação (4) [15], S_{min} (W) representa a mínima sensibilidade de um receptor, SNR_{min} (W) é

a mínima razão sinal-ruído requerida para a recepção do sinal para um dado modo de transmissão/recepção, K é a constante de Boltzmann ($1.38 \cdot 10^{-23}$J/K), T_0 (290K) representa a temperatura absoluta, a variável B (Hz) representa a largura do canal de comunicação, NF (W) é a figura de ruído que expressa o valor de deterioração do sinal causada pelo circuito do receptor. Sendo assim, ao utilizar a Equação (5), podemos estimar a redução no valor de sensibilidade mínima do receptor, obtido ao reduzir a largura de canal do valor $B2$ para o valor $B1$. Exemplificando, se $B1 = 10$MHz e $B2 = 20$MHz, para um dado modo de transmissão (e.g m_1) e seu respectivo valor de SNR_{min}, a relação assume o valor $R \cong -3dB$. Sendo assim, a cada vez que dividimos a largura de canal por dois, reduzimos em 3 dB o valor da sensibilidade mínima para o mesmo modo de transmissão.

$$S_{min} = SNR_{min} \cdot K \cdot T_0 \cdot B \cdot NF \quad (4)$$

$$R = 10 \cdot \log_{10}(S_{min}^{B1}/S_{min}^{B2}) \quad (5)$$

A partir desta redução de 3 dB no valor de sensibilidade mínima para um certo modo de transmissão, pode-se montar a Tabela 3 também encontrada em [7].

Tabela 3. Valores de Sensibilidade Mínima para Larguras de Canal de 5, 10 e 20MHz

	Larguras de Canal		
Modo	20MHz	10MHz	5MHz
m_1	-82	-85	-88
m_2	-81	-84	-87
m_3	-79	-82	-85
m_4	-77	-80	-83
m_5	-74	-77	-80
m_6	-70	-73	-76
m_7	–66	-69	-72
m_8	–65	-68	-71

$$PL = P_T - P_R = 20log_{10}(\frac{4\pi f d0}{c}) + 10nlog_{10}(\frac{d}{d0}) \quad (6)$$

$$d = 10^{\frac{P_T - P_R - 20 \cdot log_{10}(4 \cdot \pi \cdot f \cdot d0/c)}{10 \cdot n}} \quad (7)$$

Os valores da Tabela 3 aplicados a formula de perda de propagação log-distância, foram utilizadas no modelo do sistema deste artigo para determinar a máxima distância d de separação entre um roteador fonte e um destino. De acordo com a Equação (6) em [6], pode-se derivar a Equação (7), onde P_T é a potência de transmissão (utilizamos o valor 17 dBm) e P_R é a potência de recepção (aplicamos os valores de sensibilidade mínima da Tabela 3). A subtração $P_T - P_R$ representa a perda de propagação do sinal no meio. A variável f denota a frequência do sinal (utilizamos o valor 2.4GHz), $d0$ é a distância de referência (aplicado o valor 1m), c é a velocidade da luz no vácuo ($\simeq 3 \cdot 10^8$m/s), n é o expoente de perda de propagação (utilizamos o valor 2.5). Ao aplicar a Equação (7), podemos estimar o aumento proporcional no alcance de transmissão que é obtido ao empregar larguras de canal de valores menores. Nós chamamos $d^{w_1,m_n}/d^{w_2,m_n}$ a razão entre os alcances de transmissão obtidos ao utilizar as larguras de canal w_1 e w_2, ambos utilizando a mesma modulação m_n. Neste caso, se fizermos $w_1 = 10$MHz, $w_2 = 20$MHz e $m_n = 1$ (Tabela 3), verificamos um alcance de transmissão em torno de 1.32 vezes maior para os canis de 10MHz. Os mesmos cálculos anteriores com $w_2 = 5$MHz geram um alcance de transmissão de aproximadamente 1.74 vezes maior para o canais de largura 5MHz. Com estes resultados, percebe-se que ao utilizar larguras de canal menores é possível reduzir o número de saltos para um comunicação fim-a-fim, em contra partida, aumenta-se o alcance de interferência entre os enlaces das rotas existentes.

3.3. MODELO DE MATRIZES DO SISTEMA

Tal como em [2, 3], modelamos uma rede sem fio estática através de um grafo $G(V, E)$ formado por um conjunto de vértices (roteadores sem fio) $V = \{v_i\}_{1\times|V|}$ e um conjunto de arestas (enlaces) $E = \{e_{i,j,c}\}_{|V|\times|V|\times|C|}$. As arestas podem ser estabelecidas em um conjunto de canais C. Existe um conjunto de demandas $F = \{f_k\}_{k\times|K|}$ cada uma, originada em um vértice v_o e com destino a um vértice v_d.

Assume-se uma banda total de frequência (B_{TOT}) dividida em um conjunto de canais ortogonais de largura w_ϖ ($\varpi = 1, ..., |W|$, onde W é o conjunto de larguras de canal disponíveis). Para cada largura de canal w_ϖ, pode-se dividir a banda total disponível em B_{TOT}/w_ϖ canais ortogonais de igual largura, e que formam o conjunto $C^{w_\varpi} = \{c_d^{w_\varpi}\}_{1\times|C^{w_\varpi}|}$. Neste caso, a quantidade total de canais disponíveis em todas as larguras de canal existentes é dado por $C = \bigcup_{\varpi=1}^{|W|} C^{w_\varpi}$, sendo que canais de diferentes larguras w_ϖ podem ser parcialmente sobrepostos um ao outro.

Para exemplificar o modelo, para uma banda total de frequência B_{TOT} de valor 40 MHz e larguras de canal $W = \{w_1, w_2\}$ de 10 e 20 MHz. Para os canais de 20 MHz, tem-se dois canais não sobrepostos que formam o conjunto $C^{w_1} = \{c_1^{w_1}, c_2^{w_1}\}$. A mesma banda total de frequência é dividida em quatro canais não sobrepostos de 10 MHz que compõe o conjunto $C^{w_2} = \{c_1^{w_2}, ..., c_4^{w_2}\}$.

Nota-se que o canal $c_1^{w_1}$ de largura 20 MHz é sobreposto aos canais $c_1^{w_2}$ e c_2^{w2} de largura 10 MHz.

A seguir relacionamos as demais notações utilizadas neste artigo:

- Matriz de Enlaces: $E = \{e_{i,j,c_d^{w\varpi}}\}_{|V|\times|V|\times|C|}$, $\forall\, e_{i,j,c_d^{w\varpi}} \in \{0,1\}$. O valor 1 representa que dois vértices i e j estão no alcance de transmissão um do outro no canal c_d de largura w_ϖ. Um elemento $e_{i,j,c_d^{w\varpi}}$ assume o valor 1, caso a distância entre os vértices i e j ($d_{i,j}$), seja menor ou igual a um valor de distância determinado a partir da Equação (7). Para determinar este valor de distância, atribuímos o valor 17 dBm para P_T. Para a variável P_R utilizamos o valor de sensibilidade mínima do modo de transmissão de maior alcance ($ex: m_1 = -88\,dBm$ para a largura de 5MHz) para a largura de canal w_ϖ em questão.

- Matriz de Atribuição de Canais: $A = \{a_{i,j,c_d^{w\varpi}}\}_{|V|\times|V|\times|C|}$, $\forall\, e_{i,j,c_d^{w\varpi}} \in \{0,1\}$. Se $a_{i,j,c_d^{w\varpi}} = 1$, o canal $c_d^{w\varpi}$ foi atribuído para a comunicação entre os vértices i e j.

- Matriz de Tempos de Transmissão: $T = \{t_{i,j,c_d^{w\varpi}}\}_{|V|\times|V|\times|C|}$, $\forall\, e_{i,j,c_d^{w\varpi}} \in \mathbb{R}$, representa o tempo de transmissão de um pacote de dados e respectiva recepção do pacote de reconhecimento (ACK) no canal $c_d^{w\varpi}$. Os valores dos elementos desta matriz são calculados a partir da Equação (2).

- Matriz de Capacidades: $CAP = \{Cap_{i,j,c_d^{w\varpi}}\}_{|V|\times|V|\times|C|}$, $\forall\, e_{i,j,c_d^{w\varpi}} \in \mathbb{R}$, representa a capacidade do enlace, em $\mathrm{M}bits/\mathrm{s}$, e é dada pela Equação (1).

- Matriz de Interferências: $IN = \{in_{i,j,c_d^{w\varpi},u,v,c_g^{w\gamma}}\}_{|E|\times|E|\times|C|}$, $\in \{0,1\}$. Esta matriz representa a existência de interferência entre enlaces. O valor 1 indica que os enlaces $e_{i,j}$ e $e_{u,v}$, não podem ser atribuídos simultaneamente aos canais $c_d^{w\varpi}$ e $c_g^{w\gamma}$, uma vez que os vértices $(i \wedge u) \vee (i \wedge v) \vee (j \wedge u) \vee (j \wedge v)$ estão no alcance de interferência nos canais comentados. Considerou-se que a distância de interferência entre vértices é calculada da mesma forma que a distância de comunicação, utilizada para dar valores para a Matriz de Enlaces (considerado que a distância de comunicação é igual a distância de interferência).

O modelo de matrizes anterior foi codificado em um programa escrito em linguagem Matlab [11] e que foi utilizado para simular WMNs, onde os roteadores são capazes de alterar dinamicamente a largura do canal de comunicação. A partir da execução do programa matlab gerou-se como saída um conjunto de dados que serve como entrada para o modelo LP apresentado na Subseção 3.4. Este conjunto de dados continha, por exemplo, informações sobre o número de roteadores da rede, a quantidade de interfaces de rádio de cada roteador, a capacidade de cada enlace e cada restrição de interferência representada pela matriz IN da Subseção 3.3.

3.4. MODELO DE PROGRAMAÇÃO LINEAR (*Linear Programming*-LP)

Nesta seção descreveremos nossa extensão do modelo LP de fluxo máximo para o roteamento, atribuição e adaptação conjunta de largura de canais em WMNs. Na Equação (8) temos a função objetivo, onde queremos maximizar a soma dos valores das demandas de 1 até $|K|$. As restrições de (9) até (11) representam a lei de conservação de fluxo, onde a variável $x_{i,j,c_d^{w\varpi},f_k}$ significa o valor de fluxo escoado entre os vértices i e j no canal c_d de largura w_ϖ para a demanda f_k. A primeira restrição representa o valor da demanda f_k em um vértice intermediário na rota entre o par origem v_o e destino v_d. As restrições (10) e (11), representam, respectivamente, os valores das demandas originadas e terminadas em vértices que são fonte $v_i = v_o$ e destino $v_i = vd$. A restrição (12) representa o valor de fluxo $x_{i,j,c_d^{w\varpi},f_k}$ escoado entre os vértices i e j, caso o canal $c_d^{w\varpi}$ seja atribuído ao enlace $e_{i,j}$, através do valor 1 da variável $a_{i,j,c_d^{w\varpi}}$. A restrição (13) significa que somente um dos enlaces $e_{i,j}$ ou $e_{u,v}$ pode ser atribuído, respectivamente, aos canais $c_d^{w\varpi}$ ou $c_g^{w\gamma}$, uma vez que existe interferência (representada pela Matriz de Interferências da Subseção 3.3) entre estes enlaces nos canais comentados. A restrição (14) representa que a quantidade de enlaces atribuídos a um vértice v_i, não pode ultrapassar o seu número total de rádios de comunicação $q(R)$. Ambas as restrições (15) e (16) representam o limite inferior dos valores de fluxos nos enlaces. A restrição (17) representa a capacidade dos enlaces. A restrição (18) representa o limite inferior dos valores de demanda e, por fim, a restrição (19) representa os possíveis valores da variável $a_{i,j,c_d^{w\varpi}}$ de atribuição de canais.

$$Max: \sum_{k=1}^{|K|} f_k \tag{8}$$

Subject to:

$$\sum_{\varpi=1}^{|W|}\sum_{d=1}^{|C^{W\varpi}|} x_{j,i,c_d^{w\varpi},f_k} - \sum_{\varpi=1}^{|W|}\sum_{d=1}^{|C^{W\varpi}|} x_{i,j,c_d^{w\varpi},f_k} = 0 \quad ,\forall v_i \neq \{v_o, v_d\}, \forall f_k \tag{9}$$

$$\sum_{\varpi=1}^{|W|}\sum_{d=1}^{|C^{W\varpi}|} x_{j,i,c_d^{w\varpi},f_k} - \sum_{\varpi=1}^{|W|}\sum_{d=1}^{|C^{W\varpi}|} x_{i,j,c_d^{w\varpi},f_k} = -f_k \quad ,\forall v_i = v_o, \forall f_k \tag{10}$$

$$\sum_{\varpi=1}^{|W|}\sum_{d=1}^{|C^{W_\varpi}|} x_{j,i,c_d^{w_\varpi},f_k} - \sum_{\varpi=1}^{|W|}\sum_{d=1}^{|C^{W_\varpi}|} x_{i,j,c_d^{w_\varpi},f_k} = f_k \quad ,\forall v_i = v_d, \forall f_k \tag{11}$$

$$\sum_{k=1}^{|K|} x_{i,j,c_d^{w_\varpi},f_k} = Cap_{i,j,c_d^{w_\varpi}} \cdot a_{i,j,c_d^{w_\varpi}} \quad ,\forall c_d^{w_\varpi} \in C, \forall e_{i,j,c_d^{w_\varpi}} = 1 \tag{12}$$

$$a_{i,j,c_d^{w_\varpi}} + a_{u,v,c_g^{w_\gamma}} <= 1 \qquad ,\forall in_{i,j,c_d^{w_\varpi},u,v,c_g^{w_\gamma}} = 1 \tag{13}$$

$$\sum_{\varpi=1}^{|W|}\sum_{d=1}^{|C^{W_\varpi}|} a_{i,j,c_d^{w_\varpi}} + a_{j,i,c_d^{w_\varpi}} <= q(R) \qquad ,\forall v_i \tag{14}$$

$$x_{i,j,c_d^{w_\varpi},f_k} >= 0 \qquad ,\forall c_d^{w_\varpi} \in C, \forall e_{i,j,c_d^{w_\varpi}} = 1 \tag{15}$$

$$\sum_{k=1}^{|K|} x_{i,j,c_d^{w_\varpi},f_k} >= 0 \qquad ,\forall c_d^{w_\varpi} \in C, \forall e_{i,j,c_d^{w_\varpi}} = 1 \tag{16}$$

$$x_{i,j,c_d^{w_\varpi},f_k} <= Cap_{i,j,c_d^{w_\varpi}} \quad ,\forall c_d^{w_\varpi} \in C, \forall e_{i,j,c_d^{w_\varpi}} = 1 \tag{17}$$

$$f_k \geq 0 \qquad ,\forall f_k \tag{18}$$

$$a_{i,j,c_d^{w_\varpi}} \in \{0,1\} \tag{19}$$

O modelo LP anterior foi escrito em linguagem GNU-*MathProg* e aplicado ao programa de resolução de modelos lineares glpsol, ambos parte do GNU *Linear Programming Kit* [5]. A partir da execução do modelo obtivemos os resultados de capacidade apresentados na Seção 4.

4. AVALIAÇÃO DE DESEMPENHO

Nesta seção obtemos os valores de capacidade da rede utilizando cenários de avaliação onde existem somente canais 5, 10 ou $20\mathrm{MHz}$, ou quando estas três larguras de canal são utilizadas em conjunto. Para isto aplicamos o modelo LP da Seção 3.4.

Como primeiro cenário utilizamos uma área de 450 x 450 metros com 16 roteadores pseudo-aleatoriamente espalhados. O posicionamento dos roteadores utiliza números aleatórios uniformemente distribuídos, porém, utiliza-se um conjunto de valores de posicionamento dos vértices, somente se não existem vértices com grau maior que 4 ao tomar como referência o alcance da largura de canal de $20\mathrm{MHz}$ (largura de canal de menor alcance). Utilizamos índice de perda de propagação no meio com valor 2.85 o que resulta, respectivamente, em alcances de transmissão de $\cong 117$, $\cong 149$ e $\cong 190\mathrm{m}$ para as larguras de canal de 20, 10 e $5\mathrm{MHz}$. Estes valores de alcance significam que temos rotas de até 5, 4 e 3 saltos nas larguras de canal comentadas. Avaliamos o desempenho do modelo LP em cenários onde foram variados os valores da quantidade máxima de rádios de comunicação dos roteadores (variável $q(R)$) e a quantidade de pares de roteadores origem/destino $|K|$. Para cada configuração de $q(R)$ e $|K|$ foram realizadas 30 execuções do modelo LP com intervalo de confiança 95%. Os resultados são apresentados a seguir:

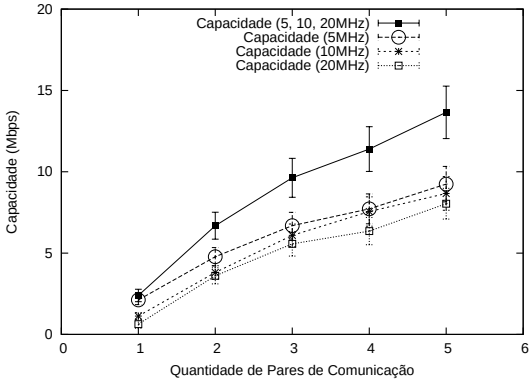


Figura 1. Capacidade com $f_k = 1, ..., 5$, $B_{TOT} = 40\mathrm{MHz}$ e $q(R) = 2$, área de $450\mathrm{m}\times450\mathrm{m}$ e 16 roteadores

Conforme observado na Figura 1 obtêm-se maiores valores de capacidade ao se utilizar a formulação LP que emprega todas as larguras de canal.

Para a Figura 2 utilizamos os mesmos parâmetros empregados anteriormente, exceto que aumentamos de 2 para 4 o número de rádios de comunicação disponíveis por roteador. Conforme observado, obtivemos maiores valores de capacidade para todas as configurações quando comparado com os resultados obtidos para a Figura 1. A maior quantidade de rádios de comunicação dos roteadores permite estabelecer uma maior quantidade de enlaces de menor largura de canal e, que somados resultam em uma maior capacidade.

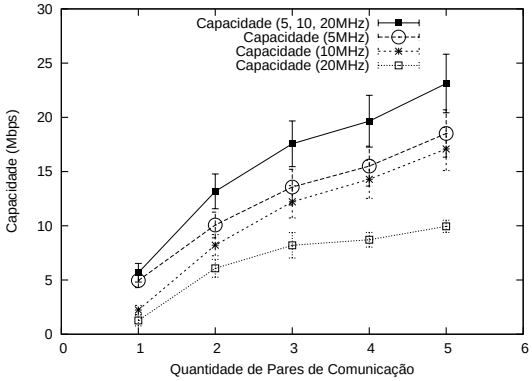


Figura 2. Capacidade com $f_k = 1, ..., 5$, $B_{TOT} = 40\mathrm{MHz}$ e $q(R) = 4$, área de $450\mathrm{m}\times450\mathrm{m}$ e 16 roteadores

Tabela 4. Exemplo de enlaces ocupados ao utilizar o modelo LP e canais com largura de 5 MHz

Largura de Canal	5MHz
Demanda f_1	-
Demanda f_2	(3,4,16,20,2.6) (3,4,31,35,2.6) (3,13,26,30,2.8),(13,4,1,5,3.8) (3,13,36,40,4.8),(13,4,6,10,3.8)
Demanda f_3	-
Demanda f_4	(7,8,11,15,4.8) (7,8,26,30,4.8) (7,8,31,35,4.8) (7,8,36,40,4.8)

Em um segundo cenário utilizamos uma área de 650 x 650 metros com 25 roteadores pseudo-aleatoriamente espalhados e equipados com 4 rádios de comunicação cada. De acordo com o observado na Figura 3, obtêm-se maiores valores de capacidade ao utilizar o modelo LP onde são utilizadas as larguras de 5, 10 e 20MHz. Observa-se, também, menores valores de capacidade para todas as quantidades de pares de comunicação quando comparado ao resultado obtido na Figura 2 que emprega os mesmos parâmetros. Isto ocorre devido a menor densidade de roteadores existentes no cenário atual. Apesar desta característica, percebe-se que é mantida a conectividade da rede devido ao emprego das larguras de canal mais estreitas e de maior alcance de transmissão.

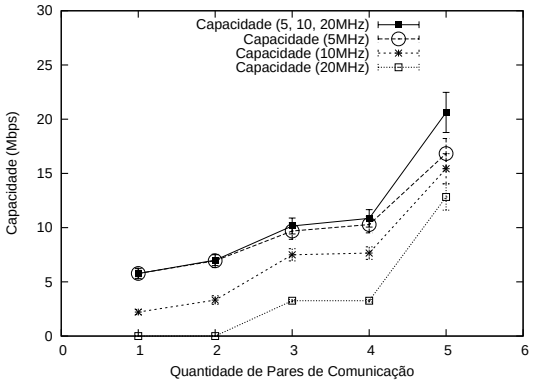


Figura 3. Capacidade com $f_k = 1, ..., 5$, $B_{TOT} = 40$MHz, $q(R) = 4$, área de 650m×650m e 25 roteadores

Para exemplificar quais enlaces e canais, podem ser ocupados através do uso modelo LP, utilizamos o mesmo cenário de avaliação empregado para gerar os resultados da Figura 2. Neste cenário utilizou-se uma banda de 40 MHz, 4 interfaces por roteador, área de 450 m×450 m e 16 roteadores. Para apresentar os dados de enlaces e canais ocupados, fazemos uso das informações contidas nas Tabelas 4, 5, 6 e 7 e Figura 4.

Na coluna 2 de cada uma das tabelas citadas, temos as possíveis configurações de larguras de canal empregaTabela 5. Exemplo de enlaces ocupados ao utilizar o modelo LP e canais com largura de 10 MHz

Largura de Canal	10 MHz
Demanda f_1	-
Demanda f_2	(3,4,1,10,2.7) (3,4,11,20,2.7) (3,4,21,30,2.7) (3,4,31,40,2.7)
Demanda f_3	-
Demanda f_4	(7,8,1,10,6.9) (7,8,11,20,6.9) (7,8,21,30,6.9) (7,8,31,40,6.9)

Tabela 6. Exemplo de enlaces ocupados ao utilizar o modelo LP e canais com largura de 20 MHz

Largura de Canal	20MHz
demanda f_1	-
demanda f_2	(3,13,21,40,7,1),(13,4,1,20,7.1)
demanda f_3	-
demanda f_4	(7,8,1,20,7.1) (7,8,21,40,7.1)

Tabela 7. Exemplo de enlaces ocupados ao utilizar o modelo LP ao utilizar e canais com largura de 5, 10 ou 20 MHz

Largura de Canal	5,10 e 20MHz
demanda f_1	-
demanda f_2	(3,4,16,20,2.6) (3,4,36,40,2.6) (3,13,11,15,4.8),(13,4,1,10,4.8) (3,13,31,35,4.8),(13,4,21,30,4.8)
demanda f_3	-
demanda f_4	(7,8,1,10,6.9) (7,8,11,20,6.9) (7,8,21,30,6.9) (7,8,31,40,6.9)

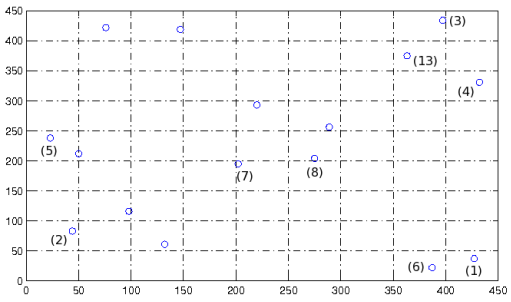


Figura 4. Posicionamento dos roteadores utilizado no exemplo de ocupação dos enlaces e canais

das. Na coluna 1 e linhas de 2 até 5 de cada tabela, temos o conjunto de demandas $F = \{f_k\}_{k \times |K|}$ a serem atendidas. Para as demandas numeradas de f_1 até f_4, foram utilizados como par origem-destino, respectivamente, os vértices (1,2), (3,4), (5,6) e (7,8). Como conteúdo da tabela, são apresentados um conjunto de dados na forma de uma tupla $(i, j, freq1, freq2, cap)$, onde i, j são os roteadores do enlace, $freq1$ e $freq2$ são as frequências inicial e final (em MHz dadas em valores inteiros no intervalo de 1 a 40 MHz, que é o valor de banda utilizada) do canal utilizado no enlace e cap representa a capacidade ($\mathrm{M}bits/\mathrm{s}$) escoada no enlace.

Utilizamos a linha 3 e coluna 2 da Tabela 4 como exemplo de leitura dos dados contidos nas tabelas 4 a 7. Neste caso, a demanda f_2 tem como origem o roteador 3 e como destino o roteador 4. Para esta demanda foram estabelecidas 4 rotas entre o par de roteadores origem/destino. A primeira rota, por exemplo, possui 01 salto e ocupa o canal com $freq1 = 16$ e $freq2 = 20$. A terceira rota, por exemplo, possui 02 saltos, sendo que o primeiro salto (enlace) é estabelecido entre os roteadores 3 e 13 no canal com $freq1 = 26$ e $freq2 = 30$ e o segundo salto é estabelecido entre os roteadores 13 e 4 no canal no canal com $freq1 = 1$ e $freq2 = 5$.

Optamos por apresentar as frequências inicial e final de cada canal, e não o seu índice (ex: c_1 para canal 1), para facilitar a visualização da concorrência entre os enlaces na ocupação dos canais.

Na Figura 4, temos os índices i dos roteadores utilizados nos enlaces os quais, também, aparecem nas tuplas das Tabelas de 4 a 7. Observa-se nestas tabelas que as demandas 1 e 3 não tiveram fluxos estabelecidos, uma vez que o modelo LP tem como objetivo maximizar a capacidade durante o período de um *slot* de tempo em uma rede do tipo TDMA, sem se preocupar com que todas as demandas sejam atendidas neste mesmo *slot*.

Observamos que para o fluxo 4, é obtido o mesmo valor de capacidade para os enlaces ao utilizar o modelo LP somente com canais de largura 10MHz (Tabela 5) e ao empregar o modelo LP para selecionar entre as larguras de 5, 10 e 20 MHz (Tabela 7). Neste caso, a largura de canal de valor 10 MHz é a que proporciona o maior valor de capacidade agregada através dos 4 enlaces (cada roteador é equipado com 4 rádios) estabelecidos entre os roteadores 7 e 8. Para a largura de 20MHz (Tabela 6) são estabelecidos somente 2 enlaces embora existam 4 rádios de comunicação. Nesta caso, foi utilizado o total de banda disponível (40 MHz) somente com o estabelecimento de 2 enlaces entre os roteadores 7 e 8. Para a largura de 5 MHz (Tabela 4), apesar de terem sido estabelecidos um total de 4 enlaces, suas capacidades somadas possuem valor inferior a capacidade somada de 4 enlaces de 10 MHz.

Para o fluxo 2 e canais de 20 MHz (Tabela 6), observa-se que são utilizados 2 saltos para comunicação entre os roteadores 3 e 4. Isto ocorre, uma vez que ao utilizar o roteador 13 como intermediário da rota, emprega-se um modo de transmissão mais veloz quando comparado ao modo de transmissão que seria utilizado para uma comunicação direta entre os roteadores 3 e 4. Ainda para o fluxo 2 e canais de 5 MHz (Tabela 4), embora obtenha-se maior capacidade com o emprego do roteador 13 como intermediário, somente 2 enlaces utilizam este roteador devido a limitação de número de rádios de comunicação (menor ou igual a 4).

Observa-se na Tabela 7 que o modelo LP, ao selecionar entre todas as possíveis larguras de canal, escolhe enlaces que possuem as larguras de 5 e 10 MHz. Com o objetivo de aumentar a capacidade, o modelo LP utiliza o roteador 13 como intermediário da rota, até o limite de sua quantidade de rádios de comunicação.

5. Conclusões

Neste artigo apresentamos uma formulação LP para roteamento, atribuição e adaptação conjunta de largura de canais em WMNs com tecnologia IEEE 802.11. Conforme observado, a utilização da mudança dinâmica da largura do canal de comunicação pode aumentar a capacidade das redes estudadas. Neste sentido, é possível aproveitar o balanceamento entre o uso de canais de menor largura e que possuem maior alcance de transmissão e, também, utilizar os canais mais largos, os quais possuem menor alcance de transmissão e favorecem o reuso espacial de frequências. Como trabalhos futuros pretendemos propor algoritmos para tratar do problema de alocação de canais nos cenários estudados.

Referências

[1] S. Avallone and I.F. Akyildiz. A channel assignment algorithm for multi-radio wireless mesh networks.

In *Computer Communications and Networks, 2007. ICCCN 2007.*, pages 1034–1039, 2007.

[2] Celso Barbosa Carvalho and José Ferreira de Rezende. Roteamento em redes em malha sem fio ieee 802.11 com adaptação de largura de canal. In *XXVIII Simpósio Brasileiro de Redes de Computadores (SBRC 2010)*, pages 161–174, 2010.

[3] Celso Barbosa Carvalho and José Ferreira de Rezende. Routing in ieee 802.11 wireless mesh networks with channel width adaptation. In *Med-Hoc-Net 2010 (Med-Hoc-Net'2010)*, 2010.

[4] Ranveer Chandra, Ratul Mahajan, Thomas Moscibroda, Ramya Raghavendra, and Paramvir Bahl. A case for adapting channel width in wireless networks. In *SIGCOMM Comput. Commun. Rev.*, pages 135–146, 2008.

[5] GNU-glpk. Glpk (gnu linear programming kit). `http://www.gnu.org/software/glpk`, 2010.

[6] Qizheng Gu. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2001.

[7] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard 802.11, 2007.

[8] Paulo R. Walenga Jr., Mauro Fonseca, Anelise Munaretto, Aline Carneiro Viana, and Artur Ziviani. Zap: Um algoritmo de atribuição distribuída de canais para mitigação de interferências em redes com rádio cognitivo. In *XXVIII Simpósio Brasileiro de Redes de Computadores (SBRC 2010)*, 2010.

[9] Bong-Jun Ko, V. Misra, J. Padhye, and D. Rubenstein. Distributed channel assignment in multi-radio 802.11 mesh networks. In *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pages 3978–3983, 2007.

[10] Li Li and Chunyuan Zhang. Optimal channel width adaptation, logical topology design, and routing in wireless mesh networks. *EURASIP Journal on Wireless Communications and Networking*, 2009, 2009.

[11] MATLAB. version 7.5 (r2007b), 2007.

[12] A.H.M. Rad and V.W.S. Wong. Logical topology design and interface assignment for multi-channel wireless mesh networks. In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pages 1–6, 2006.

[13] A.H.M. Rad and V.W.S. Wong. Joint channel allocation, interface assignment and mac design for multi-channel wireless mesh networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1469–1477, 2007.

[14] A.H.M. Rad and V.W.S. Wong. Joint logical topology design, interface assignment, channel allocation, and routing for multi-channel wireless mesh networks. *IEEE Wireless Communication*, 2007:4432–4440, 2007.

[15] Theodore Rappaport. *RF Systems Design of Transceivers for Wireless Communications*. Springer, 2005.

[16] Srikrishna Sridhar, Jun Guo, and Sanjay Jha. Channel assignment in multi-radio wireless mesh networks: a graph-theoretic approach. In *Proceedings of the First international conference on COMmunication Systems And NETworks*, pages 180–189, 2009.

[17] Srikrishna Sridhar, Jun Guo, and Sanjay Jha. Static channel assignment in multi-radio multi-channel 802.11 wireless mesh networks: issues, metrics and algorithms. In *Proceedings of the 49th IEEE Global Telecommunications Conference (Globecom)*, 2009.

[18] Fei Ye, Qing Chen, and Zhisheng Niu. End-to-end throughput-aware channel assignment in multi-radio wireless mesh networks. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 1375 –1379, 2007.

[19] Fei Ye, Sumit Roy, and Zhisheng Niu. Flow oriented channel assignment for multi-radio wireless mesh networks. *EURASIP Journal on Wireless Communications and Networking*, 2010, 2010.

[20] Yuan Yuan, Paramvir Bahl, Ranveer Chandra, Philip A. Chou, John Ian Ferrell, Thomas Moscibroda, Srihari Narlanka, and Yunnan Wu. Knows: Kognitiv networking over white spaces. In *Proceedings of IEEE DySPAN 2007*, 2007.

[21] Hongqiang Zhai and Yuguang Fang. Impact of routing metrics on path capacity in multirate and multihop wireless ad hoc networks. In *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 86–95. IEEE Computer Society, 2006.

Detecção de *Spams* Utilizando Conteúdo Web Associado a Mensagens

Marco Ribeiro[1], **Leonardo Teixeira**[1], **Pedro Guerra**[1]
Adriano Veloso[1], **Wagner Meira Jr.**[1], **Dorgival Guedes**[1]
Cristine Hoepers[2], **Klaus Steding-Jessen**[2], **Marcelo Chaves**[2]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais, Belo Horizonte, MG
{marcotcr, vilela, pcalais, adrianov, meira, dorgival}@dcc.ufmg.br

[2]CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
NIC.br - Núcleo de Informação e Coordenação do Ponto br, São Paulo, SP
{cristine, jessen, mhp}@cert.br

Abstract

In this paper we propose a strategy of spam classification that exploits the content of the Web pages linked by e-mail messages. We describe a methodology for extracting pages linked by spam and we characterize the relationship among those pages and the spam messages. We then use a machine learning algorithm to extract features found in the web pages that are relevant to spam detection. We demonstrate that the use information from linked pages can significantly outperforms current spam classification techniques, as portrayed by Spam Assassin. Our study shows that the pages linked by spams are a very promising battleground, where spammers do not hide their identity, and that this battleground has not yet been used by spam filters.

Keywords: spam filtering, spam detection, security

Resumo

Neste trabalho propomos uma estratégia de detecção de spams *que explora o conteúdo das páginas Web para as quais mensagens apontam. Descrevemos uma metodologia para a coleta dessas páginas, caracterizamos a relação entre as páginas e as mensagens de* spam*, e, em seguida, utilizamos um algoritmo de aprendizado de máquina para extrair as informações relevantes para a detecção de* spam*. Mostramos que a utilização de informações das páginas mencionadas melhora significativamente a classificação de* spams *e* hams*, gerando um baixo índice de falsos positivos. Nosso estudo revela que as páginas apontadas pelos* spams *ainda são um campo de batalha não explorado pelos filtros, onde os* spammers *não se preocupam em esconder a sua identidade.*

Palavras-chave: fintro de spam, detecção de spam, segurança

1. INTRODUÇÃO

Spam é um problema que tem acompanhado o desenvolvimento e popularização da Internet [8] e tem sido um meio usual de enviar mensagens relacionadas à obtenção de dados pessoais com objetivos ilícitos (*phishing*) e para a disseminação de códigos maliciosos [14]. O fato do custo de envio de *e-mails* ser muito baixo serve como incentivo ao uso do correio eletrônico para o envio de *e-mails* comerciais não solicitados em grandes quantidades [2], e os servidores de correio eletrônico têm que lidar com o fato de que entre 82% e 92% das mensagens recebidas são *spam* [12]. O prejuízo que essa prática acarreta a empresas e à sociedade é avaliado em bilhões de dólares [18].

O problema do *spam* é análogo a uma corrida armamentista (chamada comumente de *spam arms race* [7]). Isso significa que há uma evolução constante tanto de técnicas de detecção de mensagens indesejadas como da sofisticação das tecnologias adotadas pelos *spammers*. Nesta corrida, cada um tenta se sobrepor ao outro e a

mudança na estratégia de um lado induz a mudanças na estratégia do adversário. Os filtros anti-*spam* adotam, comumente, estratégias baseadas em filtragem de conteúdo de mensagens, como o *Spam Assassin* [19] e listas de bloqueio [3]. As duas estratégias são complementares, uma vez que a primeira trata do conteúdo da mensagem em si e a segunda trata das estratégias que o *spammer* utilizou para disseminar a mensagem. Vale notar que o objetivo final do *spammer* é ser atrativo o suficiente para o receptor tomar alguma ação — seja esta comprar algum produto ou seguir algum elo de navegação. Os filtros baseados em conteúdo obrigam o *spammer* a ofuscar suas mensagens, de forma que o *spammer* tem um compromisso entre manter o e-mail legível (e atingir menos caixas de entrada) e comprometer a legibilidade, possivelmente atingindo mais usuários. As estratégias dos *spammers* para evitar as listas de bloqueio, por outro lado, não comprometem a "qualidade" das suas mensagens.

Com o advento e popularização de técnicas de contaminação de máquinas por códigos maliciosos que podem transformar qualquer máquina de usuário em um *bot*, uma ferramenta para redistribuição de *spam* (entre outros usos), estratégias baseadas em listas de bloqueio têm se tornado menos eficientes [16]. A iminente troca de versão do protocolo IP (da versão 4 para a versão 6) provavelmente criará dificuldades ainda maiores para o sucesso das listas de bloqueio, uma vez o aumento da faixa de endereços disponíveis tornará mais difícil manter as listas de bloqueio atualizadas.

Em um trabalho recente, [7] caracterizaram a adaptação dos filtros em relação às estratégias utilizadas por *spammers* e mostraram como certas características são exploradas ao longo do tempo. Um aspecto do *spam* que ainda não é explorado pelos filtros e, consequentemente, não é ofuscado pelos *spammers*, é o conteúdo das páginas Web apontadas pelas *URLs* contidas nos *spams*. Muitas vezes essas páginas estão até mesmo fora do controle dos *spammers*, pertencendo a empresas externas que os contratam para divulgar seus produtos. [15] mostraram que pelo menos uma *URL* aparece em 85% a 95% dos *spams* presentes em todos os meses analisados por eles. Já [6] reportam que 96,5% das campanhas de *spam* observadas por eles continham pelo menos uma *URL*. Esse números indicam que técnicas que considerem o conteúdo das páginas como evidência para detecção e mitigação do *spam* pode ter impacto bastante positivo. Neste trabalho, mostramos que essas páginas podem oferecer informações valiosas acerca da natureza dos *spams*. Apesar de a obtenção dessas páginas implicar em um custo potencialmente alto para ser incluído em todos os servidores de correio eletrônico, essa informação pode contribuir para o desenvolvimento de novas ferramentas de identificação de *spam*.

Utilizamos duas bases de dados históricas de *spams* e mensagens legítimas (também chamadas *hams*), *SpamArchive* [5] e *Spam Assassin*, para construir uma base de dados que relaciona mensagens e páginas. Utilizando essa base como estudo de caso, mostramos que a utilização das páginas melhora a detecção de *spam* em aproximadamente 10%, sem causar um aumento no índice de falsos positivos. As contribuições deste trabalho, portanto, são (i) a disponibilização de uma base de dados que relaciona mensagens de *spam* às páginas apontadas por elas e (ii) a proposta de uma metodologia para a detecção de *spam* através do conteúdo das páginas apontadas pelas mensagens. Mostramos que as páginas mencionadas nas mensagens de correio eletrônico são um campo de batalha promissor e que a informação proveniente das páginas ainda não é explorado pelos filtros.

O restante deste trabalho é organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados, a Seção 3 descreve como a base de dados utilizada no trabalho foi obtida e apresenta uma caracterização da mesma; em seguida, a Seção 4 detalha a metodologia utilizada e os resultados são apresentados e discutidos na Seção 5; finalmente, as conclusões são apresentadas na Seção 6.

2. TRABALHOS RELACIONADOS

O comportamento dinâmico dos *spammers* já foi discutido em diversos trabalhos. Novas técnicas de envio de *spam* são documentadas em relatórios periódicos gerados por empresas de segurança, com estatísticas sobre as inovações e tendências do *spam*. O *spam arms race* tem sido caracterizado em trabalhos como [4], que identificaram algumas estratégias que *spammers* começaram a utilizar em 2002, como ofuscações de palavras para reduzir a eficácia de filtros bayesianos. [7] caracterizou a natureza evolutiva tanto do ponto de vista dos *spammers* quanto do ponto de vista dos filtros anti-*spam*.

Em [20], é apresentado um *survey* sobre técnicas de identificação de *spam* baseadas em classificação de texto. Um filtro que faz uso de grande parte das estratégias conhecidas atualmente é o *Spam Assassin* [19], que utiliza filtros bayesianos e listas de bloqueio DNS. Além disso, o *Spam Assassin* também conta com um conjunto de regras, geralmente representadas por expressões regulares, que são comparadas com os campos *body* ou *header* de cada mensagem. Ou seja, o *Spam Assassin* é um filtro que lida tanto com características do corpo da mensagem quanto características de rede.

Em [16], foi feito um estudo sobre a efetividade de listas de bloqueio baseadas em *DNS* em relação a *botnets*. Os resultados preliminares indicam que apenas 5% de todos os *IP*s dos bots estudados apareciam na lista de bloqueio utilizada. Em [17], é feita uma avaliação de várias listas de bloqueio, e mostra-se que as listas de blo-

queio apresentam um número significativo de falsos negativos e falsos positivos. Devido aos problemas potenciais de listas de bloqueio, é necessário a descoberta de novas técnicas.

Um método de detecção de *spam* baseado em características das *URLs*, como propriedades do endereço *IP* (incluindo a presença do mesmo em uma lista de bloqueio), propriedades de *WHOIS*, propriedades de domínio e propriedades geográficas é proposto em [11]. O autor não utiliza o conteúdo das páginas apontadas pelas *URLs*.

Em [23] construiu-se uma base de dados com páginas apontadas por *spams* da base de dados *Spam Archive* [5] no período entre novembro de 2002 e janeiro de 2006. Porém, essa base de dados tem como foco *Web Spam*, e não relaciona cada página a uma mensagem de *spam*, de forma que não pudemos utilizá-la.

Dessa forma, nosso trabalho é o primeiro, até onde sabemos, a empregar o conteúdo de páginas como evidência para identificação de *spams*, e o primeiro a disponibilizar uma base de dados que relaciona *spams* a páginas *Web*.

3. Base de Dados

Em [15], mostrou-se que pelo menos uma *URL* aparece em 85% a 95% das mensagens de *spam* no *Spam Archive* no período entre 2004 e 2006, enquanto [6] reportaram que 96,5% de suas campanhas continham pelo menos uma *URL*. Apesar disso, a coleta de páginas de *spam* ainda é uma tarefa desafiadora. [1] mostra que poucas páginas têm um tempo de vida maior do que 13 dias, ou seja, a coleta das páginas tem que ser feita em um período próximo do instante que a mensagem foi disseminada.

Tabela 1. Descrição da base de dados obtida

Número de mensagens	63.034
Número de páginas	157.114
Número médio de páginas baixadas por mensagem	2,49

Entre julho e dezembro de 2010, nós obtivemos as mensagens de *spam* da base de dados *Spam Archive* diariamente (a base também é atualizada diariamente), de forma a obter as mensagens de *spam* mais recentes. Em seguida, extraimos as *URLs* do corpo das mensagens (utilizando os módulos Perl URI::Find e HTML::LinkExtor) e utilizamos expressões regulares simples para remover imagens e executáveis. Em seguida, carregamos e armazenamos as páginas (utilizando a biblioteca de transferências de *URLs libcurl* [9]). No caso de mensagens que continham múltiplas *URLs*, todas as *URLs* foram carregadas e armazenadas. Várias *URLs* continham redirecionamentos; nesse caso, seguimos todos os redirecionamentos e armazenamos o conteúdo final da página.

Para cada uma das 157.114 páginas obtidas com sucesso, armazenamos dois arquivos: o primeiro contém o conteúdo HTML da página e o outro contém as informações da sessão HTTP associada ao carregamento da página, contendo vários cabeçalhos. Além disso, associamos a página baixada com a mensagem correspondente. As características da base de dados obtidas são mostradas na Tabela 1, e a distribuição do número de páginas baixadas por mensagem é mostrada na Figura 1. Percebe-se que a grande maioria das mensagens contém poucas *URLs*. Vale notar que só consideramos parte da base de dados, em particular as mensagens para as quais pelo menos uma página foi baixada.

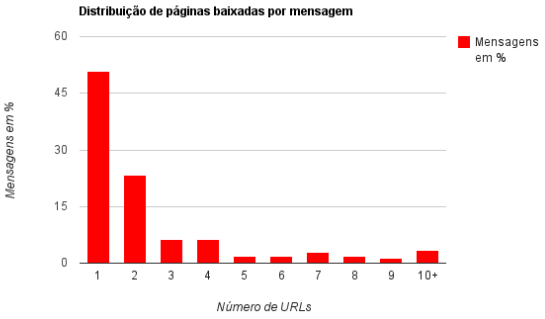


Figura 1. Distribuição do número de páginas baixadas por mensagem

4. Metodologia

A técnica para detecção de *spam* que propomos se baseia nas páginas apontadas por *URLs* em mensagens de *spam*. Apesar de o acesso a essas páginas implicar em um custo extra, em uma implementação em produção nossa técnica poderia funcionar de forma complementar a outras estratégias de classificação de *spam*. Ao se analisar uma mensagem, carrega-se as páginas identificadas por URLs contidas na mesma e verifica-se se essas páginas possuem conteúdo que seja associado com campanhas de *spam* — da mesma forma que um filtro de conteúdo avalia o corpo da mensagem, mas nesse caso considerando o conteúdo da página. Esse conteúdo é então combinado com as outras características da mensagem (dadas pelo *Spam Assassin*) e o par (mensagem, página) é classificado para identificar *hams* e *spams*. Nesta seção descrevemos as operações em cada uma dessas etapas. O processo é ilustrado pelo diagrama apresentado na Figura 2 e pelo exemplo apresentado ao final.

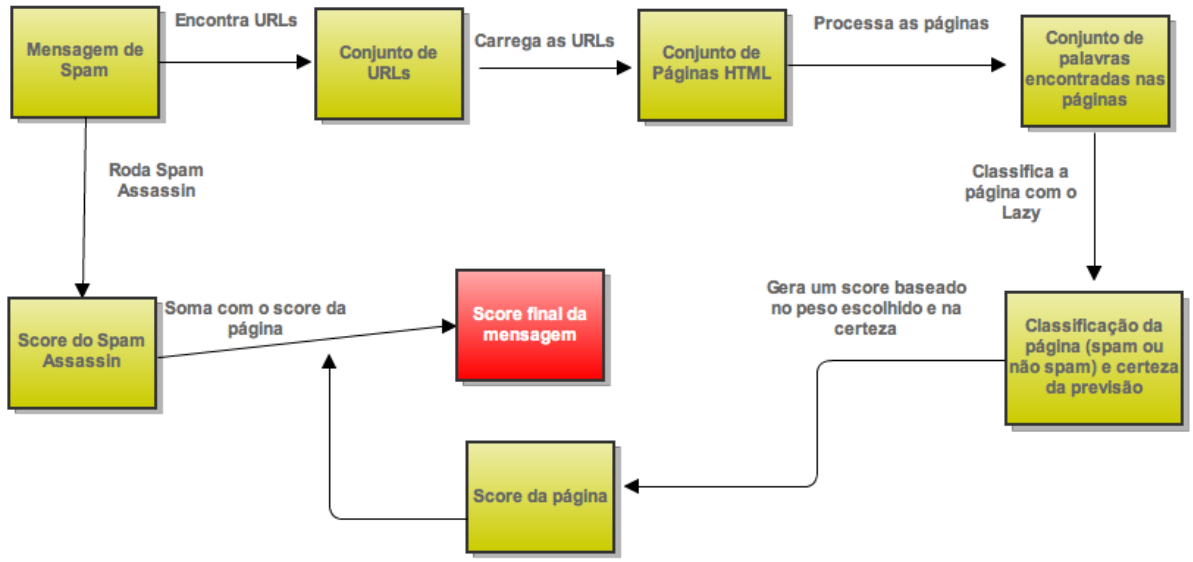


Figura 2. Diagrama ilustrativo da metodologia

4.1. Processamento da página

Após a identificação das URLs nas mensagens e o acesso às páginas por elas identificadas, utilizamos o navegador *lynx* [10] como um filtro para formatar a página em modo texto, retirando *tags* HTML e javascripts. Essa etapa é importante para eliminar ruídos devidos a pequenas mudanças de formatação e facilitar a representação de cada página para fins de detecção. Utilizando o *lynx*, temos uma representação bem próxima da representação que um usuário que visitasse a página teria. O programa gera um *dump* da página formatada para visualização, que é então utilizada pelo classificador associativo.

4.2. Classificação da página

Para classificar a página, utilizamos um algoritmo de aprendizado associativo sob demanda [21]. Optamos por esse algoritmo por (i) ter bom desempenho para utilização em tempo real (o algoritmo consegue classificar em média 111 páginas por segundo), (ii) gerar um modelo de boa legibilidade (que pode ser facilmente transformado em um conjunto de expressões regulares, como as do *Spam Assassin*) e (iii) ser bem calibrado [22]. Essa última característica significa que o algoritmo gera uma probabilidade de cada previsão estar certa, ou seja, as previsões com mais certeza são mais confiáveis. O algoritmo produz regras do tipo $\chi \rightarrow c$, onde χ é um conjunto de termos e c é a classe (*spam* ou *ham*). Cada uma dessas regras tem uma certa frequência (que chamamos de suporte) e uma confiança, que é dada pelo número de instâncias que são classificadas corretamente pela regra dividido pelo número de instâncias que contém o conjunto de termos χ. O resultado final da classificação de cada página é a classe predita pelo algoritmo e a certeza da predição, medida entre 0 e 1. Como o algoritmo é bem calibrado, a certeza da predição é confiável, e pode ser levada em consideração ao avaliar-se o peso associado à classificação de uma página.

Uma das dificuldades da detecção de *spam* é a assimetria entre o custo de se classificar um *spam* incorretamente e o custo associado aos diferentes tipos de erro. Um falso negativo simplesmente causa alguma irritação, *i.e.,* o usuário recebe uma mensagem indesejada. Por ou-

tro lado, um falso positivo pode ser crítico: uma mensagem importante pode nunca chegar à caixa de entrada do usuário, se for filtrada pelo servidor [4]. Em virtude do alto custo de se classificar uma mensagem legítima como *spam*, empregamos também a noção de *custo de classificação*. O custo de uma classe mede o quão caro é classificar incorretamente uma instância dessa classe. Ao ponderar todas as regras obtidas para uma determinada instância, o algoritmo faz uma soma ponderada das regras levando em conta a confiança e o custo de cada classe, de forma a valorizar mais regras que apontam para classes de custo mais alto. Isso implica que quanto maior o custo, mais certeza o algoritmo precisa ter para classificar uma página como *spam*.

Como o classificador associativo necessita de instâncias de ambas as classes (*spam* e *ham*), utilizamos uma base de dados pública de *ham*s, fornecida pelo próprio *Spam Assassin*, como já citado. Para instâncias da classe *spam*, utilizamos a base de dados descrita na seção anterior.

4.3. Classificação da mensagem

Há várias maneiras de se ponderar o resultado da classificação das páginas. Uma delas é associar um peso p à classificação da página, e ponderar p com as outras características já obtidas na mensagem. No *Spam Assassin*, por exemplo, uma mensagem é considerada *spam* se ela atinge 5 ou mais pontos (que são obtidos através das regras e listas de bloqueio). Uma forma de incorporar a nossa técnica ao *Spam Assassin* seria adicionar x pontos a uma mensagem se o algoritmo descrito na subseção anterior classificou a página como *spam*. Outra forma seria adicionar $x*c$ pontos à mensagem, onde x é um valor pré-determinado e c é a certeza da predição. Dessa forma, páginas com maior chance de serem identificadas como *spam* acarretariam em pontuações mais altas para as suas respectivas mensagens.

Outra forma seria eliminar completamente o uso de listas de bloqueio, e substituí-las pela nossa técnica. Essa forma pode ser interessante quando os recursos de rede disponíveis são limitados, uma vez que tanto as listas de bloqueio quanto a estratégia baseada em páginas exigem uma consulta a algum servidor externo.

Naturalmente, mensagens que não possuam *URL*s não podem ser classificadas pela nossa técnica. Essas mensagens podem ser filtradas pelos métodos convencionais de detecção de *spam*. Porém, como já foi mostrado, mais de 85% das mensagens contém *URLs* [15] [6].

4.4. Exemplo ilustrativo

Apresentamos um exemplo da aplicação da nossa técnica passo a passo. Escolhemos uma mensagem de *spam* obtida do *Spam Archive* no mês de outubro de 2010. A Figura 3 mostra o corpo da mensagem (omitimos grande parte dos cabeçalhos por questão de espaço). Percebe-se que a mensagem é bem concisa, e ofuscada. O *Spam Assassin* sem listas de bloqueio encontra apenas a regra apresentada na Tabela 2.

```
From: Discount Rolex_Etc. <0-hef@hef.fr>
To: ----
Subject: 75% Off on Gucci, Rolex, And Loius Vuitton Handbags and Various Other items
Date: Tue, 26 Oct 2010 19:20:16 +0300
MIME-Version: 1.0
Content-Type: multipart/alternative;
   boundary="----=_hhnlkay_21_61_99"
X-Priority: 3
X-Mailer: ouddj 10
Message-ID: <4281462511.JBNTUZ0E381515@hwvxuglmwirrhn.fnezaq.biz>

------=_hhnlkay_21_61_99
Content-Type: text/plain;
   charset="windows-1250"
Content-Transfer-Encoding: quoted-printable

Stop paying more than you have to!
http://migre.me/1JHMb
------=_hhnlkay_21_61_99
Content-Type: text/html;
   charset="windows-1250"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=3DContent-Type content=3D"text/html; charset=3Dwindows-1=
250">
<STYLE></STYLE>
</HEAD>
<BODY>
<a href=3D"http://su.pr/6uuaSy">Stop paying more than you have to!</a>
</BODY></HTML>

------=_hhnlkay_21_61_99--
```

Figura 3. Mensagem de *spam* extraída do *Spam Archive*

Regra	Significado	Pontuação
HTML_MESSAGE	Há HTML na mensagem	0.001

Tabela 2. Regra encontrada para mensagem ofuscada

A pontuação resultante, portanto, é 0.001. O *Spam Assassin* com listas de bloqueio ativadas encontra as regras apresentadas na Tabela 3.

Regra	Significado	Pontuação
HTML_MESSAGE	Há HTML na mensagem	0.001
RCVD_IN_ BRBL_LASTEXT	Lista de bloqueio DNS BRBL	1.644
URIBL_BLACK	Há alguma URL contida em uma lista de bloqueio	1.775

Tabela 3. Regras encontradas com listas de bloqueio ativadas

A pontuação resultante é 3,4 – ainda insuficiente para classificar a mensagem como *spam*. Um excerto da página apontada pelas *URLs* dessa mensagem é ilustrado na Figura 4.

Figura 4. Pagina de *spam* apontada pela mensagem extraída do *Spam Archive*

Percebe-se, neste caso, que o conteúdo da mensagem e o conteúdo da página são totalmente diferentes. O conteúdo da página é transformado, então, em um conjunto de palavras (através do navegador *lynx*), que é entregue ao classificador associativo, que já dispunha de um conjunto de páginas de *spam* e não-*spam* como treino (no caso, as outras páginas armazenadas do *Spam Archive* e a base de dados de *ham* do *Spam Assassin*). O classificador associativo encontra um conjunto de regras, das quais alguns exemplos são:

Regra	Suporte	Confiança
$viagra \rightarrow Spam$	36.70%	99.84%
$levitra \rightarrow Spam$	34.01%	99.90%
$rather \rightarrow Ham$	2.97%	67.30%

Por fim, o resultado final do classificador associativo é que a página é *spam*, com 90% de certeza. Supondo que tenhamos pré-definido que o peso das páginas seria $4 * c$, sendo c a certeza do classificador associativo. Percebe-se que o valor de c é determinante na pontuação final da mensagem, de forma que as páginas que o classificador associativo tem menos certeza recebem uma pontuação menor. Essa página, portanto, teria pontuação igual a 3,6. Somando-se a pontuação obtida pelo *Spam Assassin* com a pontuação da página, temos uma pontuação igual a 7,0 – mais do que suficiente para classificar a mensagem como *spam*.

5. Resultados e Discussão

Para avaliar a aplicabilidade de se construir filtros anti-*spam* a partir do conteúdo das páginas, selecionamos todas as páginas únicas da base de dados. Optamos por avaliar apenas as páginas únicas para impedir que uma campanha de mensagens apontando para a mesma página enviesasse os nossos resultados. Quando várias mensagens diferentes apontavam para a mesma página, uma

delas foi selecionada aleatoriamente para a avaliação, de forma que apenas uma instância de cada página permanecesse na avaliação. Ao final, portanto, avaliamos a nossa técnica em 32929 páginas *spam*, apontadas por 12111 mensagens de *spam* e 11134 páginas *ham*, apontadas por 4927 mensagens retiradas da base de *ham* do *Spam Assassin*. Utilizamos validação cruzada para a avaliação, dividindo as páginas em 5 partições. Utilizamos nossa técnica em conjunto com o filtro *Spam Assassin*, com suas regras e consultas a listas de bloqueio. Para combinar as pontuações do *Spam Assassin* e o resultado da classificação das páginas, multiplicamos um valor de peso pela certeza da previsão do classificador associativo e somamos esse resultado à pontuação dada pelo *Spam Assassin*. Vale notar que se o classificador associativo classifica uma página como *ham*, a pontuação da página, que é somada à pontuação do *Spam Assassin*, é negativa. Nas subseções seguintes mostramos a relação entre a certeza da classificação das páginas e a pontuação das mensagens dado pelo *Spam Assassin* e o impacto da variação dos parâmetros peso e custo. O classificador associativo foi executado com confiança 0.3, o custo das duas classes foi igual e o peso escolhido foi 4, exceto quando indicado diferente. Esses valores foram ajustados na validação cruzada.

5.1. Comparação com SpamAssassin

A Figura 5 mostra a relação entre a pontuação das mensagens dado pelo *Spam Assassin* e a certeza do classificador associativo de que a página é *spam*. As linhas azuis indicam as divisórios entre *hams* e *spams*, como dadas pelos dois classificadores. Os pontos verdes representam os *spams*, e os vermelhos representam os *hams*. Vale notar que há uma grande quantidade de *spams* no quadrante inferior direito – ou seja, *spams* que não são identificados pelo *Spam Assassin*, mas são identificados pela nossa técnica. Percebe-se também que a maioria dos *hams* no quadrante inferior direito possuem uma pontuação muito baixo no *Spam Assasin*, além de uma certeza baixa dada pelo classificador associativo, e portanto mesmo tendo sido incorretamente classificados pela nossa técnica, não seriam considerados como *spams* quando a combinação entre os scores fosse feita.

A métrica de McNemar [13] comparando o *Spam Assassin* com a classificação dada através das páginas tem um valor 677.6, e nos permite afirmar que os dois classificadores são diferentes com pelo menos 99.99% de certeza.

5.2. Impacto do parâmetro peso

Mostramos na Figura 6 o impacto de diferentes valores de peso (que é multiplicado com a certeza da previsão do classificador associativo) no índice de falsos negativos

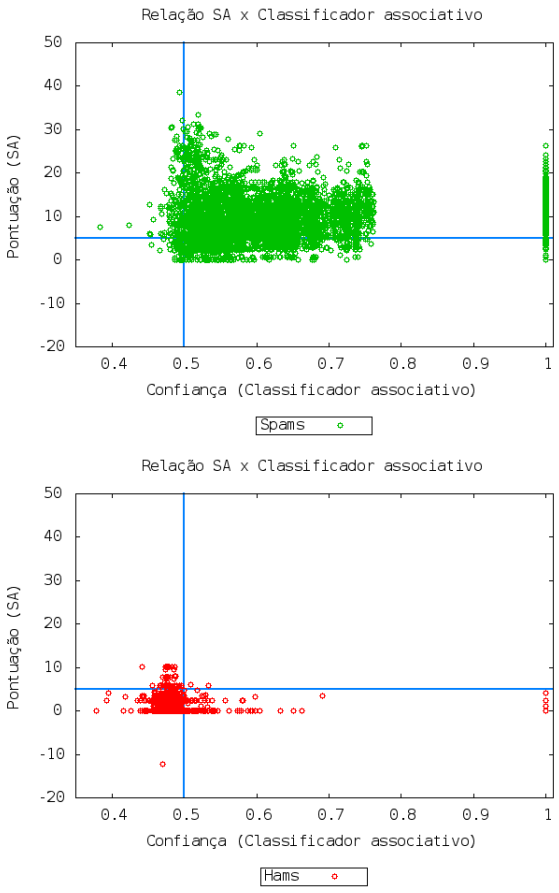


Figura 5. Pontuação do *Spam Assassin* x certeza da página ser *spam* do classificador associativo

e falsos positivos. Mostramos também na figura o índice de falsos positivos e falsos negativos gerados através da utilização do *Spam Assassin* sem a nossa técnica, para fins de comparação. Percebe-se que com um peso de até 4, o índice de falsos positivos permanece praticamente igual ao índice de falsos positivos do *Spam Assassin*, embora o índice de falsos negativos seja consideravelmente mais baixo. Selecionamos para os experimentos seguintes, portanto, o valor de peso 4, que representa o menor índice de falsos negativos sem aumentar o índice de falsos positivos.

5.3. Impacto do parâmetro custo

Mostramos na Figura 7 o impacto da variação do custo no índice de falsos positivos e falsos negativos da nossa técnica. Os valores do eixo x representam a diferença entre o custo de se classificar um *ham* como *spam* e o custo de se classificar um *spam* como *ham*. Portanto, se o valor no eixo x é 50%, isso significa que é 50% mais custoso classificar um *ham* como *spam* do que vice-versa. Naturalmente. um aumento no custo gera uma redução do

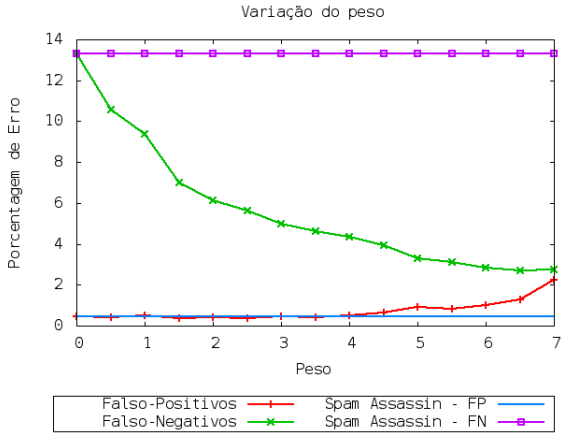


Figura 6. Falsos positivos e falsos negativos x Peso

índice de falsos positivos e um aumento no índice de falsos negativos. Com um custo maior do que 70%, nossa técnica passa a classificar *spams* com menos eficácia do que o *Spam Assassin*, embora o número de falsos positivos chegue a 0. É interessante notar que esse compromisso é ajustável na nossa técnica, através do parâmetro custo. Cabe ao usuário da técnica definir o custo de acordo com a sua necessidade.

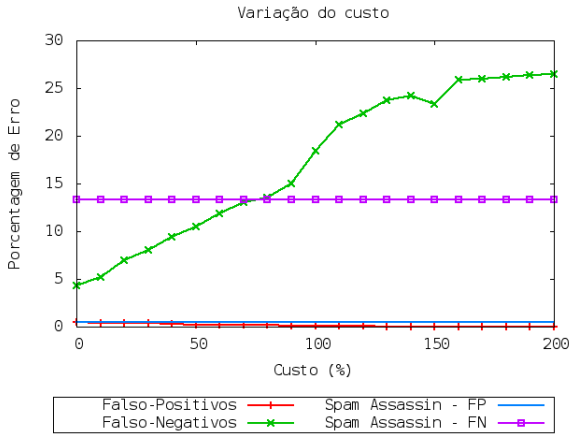


Figura 7. Falsos positivos e falsos negativos x Custo

6. CONCLUSÕES

Neste trabalho, mostramos que as páginas *Web* apontadas por mensagens de *spam* podem ser utilizadas com sucesso para a classificação dessas mensagens. Nossa proposta consiste em utilizar as páginas como complemento a outras estratégias já utilizadas de classificação de mensagens de *spam*.

Mostramos na seção de trabalhos relacionados que estratégias de filtragem de *spam* convencionais não fazem uso das páginas. A grande maioria dos *spams* contém *URL*s [15], e portanto podem ser filtrados pela nossa técnica. Mostramos também que uma das estratégias mais comuns para a filtragem de *spams*, o uso de listas de bloqueio, esta perdendo sua efetividade [16] [17], e portanto é necessário que novas técnicas de filtragem sejam estudadas e utilizadas. Neste trabalho propomos uma técnica que explora um aspecto no qual os *spammers* ainda não escondem a sua identidade. Além disso, as páginas muitas vezes não pertencem aos *spammers*, e portanto são um campo de batalha no qual os *spammers* estão em desvantagem.

Avaliamos o uso de um algoritmo de aprendizado de máquina sob demanda [21] para a classificação das páginas, e propomos uma forma de se agregar a classificação das páginas com a classificação tradicional das mensagens com o filtro *Spam Assassin* [19]. Mostramos que a utilização da nossa técnica melhora a filtragem de *spam* em mais de 10%, sem inserir um número significativo de falsos positivos. Mostramos também que a quantidade de falsos positivos pode ser ajustada com a variação do parâmetro custo.

Acreditamos que o trabalho abre novas possibilidades para o desenvolvimento de estratégias de filtragem de *spam*, introduzindo um aspecto totalmente novo e ainda não explorado na literatura. Em outras palavras, as páginas apontadas pelas mensagens constituem-se em um novo campo de batalha, com o qual hoje os *spammers* não precisam se preocupar. Neste novo campo de batalha, diferentes algoritmos podem ser utilizados para a classificação das páginas, e o resultado da classificação pode ser combinado com outras técnicas. Por fim, servidores de correio eletrônico poderiam utilizar as técnicas descritas em [6] para agrupar as mensagens em campanhas, de forma a diminuir o número de páginas a serem classificadas.

Agradecimentos

O presente trabalho foi realizado com o apoio do UOL (`www.uol.com.br`) , através do Programa UOL Bolsa Pesquisa, Processo Número 20110215235100, e também do CNPq, CAPES, FAPEMIG e FINEP.

Referências

[1] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. pages 135–148, 2007.

[2] Vinton G. Cerf. Spam, spim, and spit. *Commun. ACM*, 48(4):39–43, 2005.

[3] Duncan Cook, Jacky Hartnett, Kevin Manderson, and Joel Scanlan. Catching spam before it arrives: domain specific dynamic blacklists. In *ACSW Frontiers '06: Proceedings of the 2006 Australasian workshops on Grid computing and e-research*, pages 193–202, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[4] Tom Fawcett. "in vivo"spam filtering: a challenge problem for kdd. *SIGKDD Explor. Newsl.*, 5:140–148, December 2003.

[5] B Guenter. Spam archive, 2010. `http://untroubled.org/spam/`.

[6] Pedro H. Calais Guerra, Dorgival Guedes, Wagner Meira Jr., Cristine Hoepers, and Klaus Steding-Jessen. Caracterização de estratégias de disseminação de spams. In *26o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Rio de Janeiro, RJ, 2008.

[7] Pedro H. Calais Guerra, Dorgival Guedes, Jr. Wagner Meira, Cristine Hoepers, Marcelo H. P. C. Chaves, and Klaus Steding-Jessen. Exploring the spam arms race to characterize spam evolution. In *Proceedings of the 7th Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, Redmond, WA, 2010.

[8] Brian Hayes. Spam, spam, spam, lovely spam. *American Scientist*, 91(3):200–204, May–June 2003.

[9] Libcurl, 2010. `http://curl.haxx.se/libcurl/`.

[10] Lynx, 2010. `http://lynx.browser.org/`.

[11] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1245–1254, New York, NY, USA, 2009. ACM.

[12] MAAWG. Email Metrics Program: Report #5 – Third and Fourth Quarter 2008. `http://www.maawg.org/about/MAAWG_2008-Q3Q4_Metrics_Report.pdf`, March 2009.

[13] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157, 1947. 10.1007/BF02295996.

[14] Jason Milletary. Technical trends in phishing attacks. Technical report, CERT Coordination Center, Carnegie Mellon University, October 2005. `http://www.cert.org/archive/pdf/Phishing_trends.pdf`.

[15] Carlton Pu and Steve Webb. Observed trends in spam construction techniques: a case study of spam evolution. *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS)*, 2006.

[16] Anirudh Ramachandran, David Dagon, and Nick Feamster. Can dns-based blacklists keep up with bots? In *In Proceedings of the 3rd Conference on Email and AntiSpam (CEAS) (Mountain View)*, 2006.

[17] S. Sinha, M. Bailey, and F. Jahanian. Shades of grey: On the effectiveness of reputation-based blacklists. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 57 –64, Oct. 2008.

[18] Janice C. Sipior, Burke T. Ward, and P. Gregory Bonner. Should spam be on the menu? *Commun. ACM*, 47(6):59–63, 2004.

[19] SpamAssassin, 2008. `http://spamassassin.apache.org`.

[20] Upasana and S. Chakravarty. A survey on text classification techniques for e-mail filtering. In *Machine Learning and Computing (ICMLC), 2010 Second International Conference on*, pages 32 –36, Feb 2010.

[21] Adriano Veloso, Wagner Meira Jr., and Mohammed J. Zaki. Lazy associative classification. In *ICDM*, pages 645–654. IEEE Computer Society, 2006.

[22] Adriano Veloso, Wagner Meira Jr., and Mohammed Javeed Zaki. Calibrated lazy associative classification. In Sandra de Amo, editor, *SBBD*, pages 135–149. SBC, 2008.

[23] Steve Webb. Introducing the webb spam corpus: Using email spam to identify web spam automatically. In *In Proceedings of the 3rd Conference on Email and AntiSpam (CEAS) (Mountain View)*, 2006.

Mapeamento de Redes Virtuais em Substratos de Rede

Gustavo P. Alkmim[1], **Daniel M. Batista**[2], **Nelson L. S. da Fonseca**[1]
Instituto de Computação
Universidade Estadual de Campinas (UNICAMP) – Campinas, SP – Brasil
email: alkmim@lrc.ic.unicamp.br, nfonseca@ic.unicamp.br

Departamento de Ciência da Computação
Universidade de São Paulo (USP) – São Paulo, SP – Brasil
email: batista@ime.usp.br

Abstract

Network virtualization is a promising technique for building the Future Internet since it allows the introduction of new functionalities in network elements at low costs. To this aim, an open issue is the problem of efficient mapping of virtual network elements to those of the physical (real) network (also called network substrate). Such type of problem is NP-hard and existing solutions ignore a series of real network characteristics in order to solve the problem in reasonable time frames. This paper introduces two algorithm for the problem of mapping virtual networks on network substrates considering a set of network parameters which has not been considered before and which makes the solution more realistic.

Keywords: Network virtualization, Mapping virtual networks, Network substrates

Resumo

A virtualização de redes é uma técnica promissora para a Internet do futuro, pois facilita a adição de novas funcionalidades nos elementos de rede e diminui os custos das organizações. Uma das questões que ainda precisa ser investigada para a implantação eficiente desta tecnologia é a alocação de recursos físicos para as redes virtuais. Por se tratar de um problema NP-Difícil, os algoritmos propostos, até o momento, desconsideram vários parâmetros, a fim de obter soluções em tempo viável. Este artigo apresenta dois novos algoritmos para o problema de mapeamento de redes virtuais em substratos de rede. Os algoritmos minimizam a utilização dos recursos e consideram diversos parâmetros negligenciados por outros trabalhos na literatura. Experimentos mostram que os algoritmos encontram soluções em tempo viável para diversos cenários de requisição de redes virtuais.

Palavras-chave: Virtualização de redes, Mapeamento de redes virtuais, Substratos de rede

1. INTRODUÇÃO

Uma das principais características da arquitetura da Internet, que facilitou a sua proliferação em escala global, é seu aspecto generalista e minimalista. Dessa forma, a pilha de protocolos TCP/IP pôde ser implementada sobre diferentes tipos de tecnologias e o núcleo da rede foi construído de modo a ser o mais simples possível. No entanto, a diversificação das aplicações e o intenso uso da Internet como infraestrutura global de comunicação acarretou em uma série de adições de protocolos e mecanismos ua arquitetura, a fim de que se pudesse prover funcionalidades inexistentes na pilha TCP/IP original. A impossibilidade de alterações no núcleo da Internet para inclusão de novas funcionalidades, embora mantenha o núcleo simples, dificulta o desenvolvimento de novas aplicações, motivo pelo qual o termo "ossificação da Internet" é utilizado para representar esse fato.

Para superar essas restrições, novas arquiteturas e mecanismos vêm sendo propostos para promover a evolução da Internet do futuro [18] [9] [22] [21] [10] [3]. Várias destas soluções baseiam-se na virtualização da rede. Através da virtualização é possível definir redes virtuais, compostas de roteadores e enlaces virtuais, que fazem uso de roteadores e enlaces da rede física; conjunto de recursos, normalmente, chamado de substrato da rede. A virtualização de redes permite a coexistência de diferentes pilhas e arquiteturas de redes no mesmo núcleo da Internet, sem a necessidade de modificá-lo e sem restringir as características destes protocolos e arquiteturas.

Dentre as diversas questões em aberto na área de vir-

tualização de redes, uma das mais importantes é a busca por mapeamentos eficientes de redes virtuais nos substratos da rede física [4] [19]. O mapeamento consiste em determinar a alocação de roteadores e enlaces da rede física para os roteadores e enlaces de uma rede virtual. No entanto, mesmo tendo-se o conhecimento prévio de todas as requisições de redes virtuais, o mapeamento ótimo é um problema NP-difícil [11], já que ele pode ser reduzido ao Problema de Separação de Multi-caminhos (*Multipath Separator Problem*)[2], que é NP difícil.

Diversas soluções têm sido propostas para realizar o mapeamento de redes virtuais [19] [4] [14] [20] porém, a maioria destas propostas assume hipóteses restritivas para tornar o problema tratável. Algumas das hipóteses são: (i) considerar que todas as requisições de estabelecimento de redes virtuais são conhecidas antecipadamente [14] [20], (ii) assumir que o substrato tem capacidade infinita [20] [8] e (iii) particularizar a topologia da rede virtual [14].

Este artigo apresenta soluções que visam tratar os pontos negligenciados pelos trabalhos anteriores no mapeamento de redes virtuais. As novas propostas para mapeamento de redes deste trabalho consideram a existência de 3 classes de provedores: os provedores de infraestrutura, os provedores de conectividade e os provedores de serviços [22]. Os provedores de infraestrutura são os responsáveis pela rede física (roteadores, cabeamento, etc), na qual serão instanciadas as redes virtuais. Os provedores de serviços são os responsáveis por fornecer os serviços da Internet para os usuários finais. São estes provedores que solicitam as redes virtuais, para suportar seus serviços. Os provedores de conectividade são responsáveis por instanciar as redes virtuais requisitadas pelos provedores de serviços na infraestrutura física. É papel dos provedores de conectividade mapear as redes virtuais requisitadas pelos provedores de serviços na infraestrutura física fornecida pelos provedores de infraestrutura.

O objetivo deste trabalho é, portanto, propor uma solução eficiente para que os provedores de conectividade possam mapear redes virtuais no substrato da rede, sem restrições nas diversas camadas de rede envolvidas. Em comparação com outros trabalhos propostos na literatura, este trabalho considera um cenário mais realista e processa uma quantidade maior de parâmetros que influenciam diretamente a complexidade da solução do problema. É necessário garantir, portanto, que o tempo de execução do algoritmo de mapeamento seja viável.

Dois algoritmos de mapeamento de redes virtuais em substratos de rede são propostos baseados em uma formulação de programação inteira 0-1 (ILP). O objetivo é minimizar a quantidade total de largura de banda alocada nos enlaces físicos para uma rede virtual. Ao se minimizar a quantidade de largura de banda alocada para uma rede virtual, tenta-se maximizar a quantidade de recursos disponíveis para as requisições seguintes. A diferença entre os dois algoritmos está no fato de um deles encontrar a solução exata do ILP, enquanto o outro utiliza técnicas de relaxação para diminuir o tempo de execução.

Experimentos mostram que a abordagem introduzida encontra soluções em tempo viável para diversos cenários de requisição de redes virtuais. Os algoritmos fornecem, também, resultados promissores considerando-se o bloqueio e a qualidade da solução.

O artigo está organizado da seguinte forma: a Seção 2 resume os trabalhos relacionados. A Seção 3 apresenta os dois algoritmos propostos. A avaliação de desempenho dos algoritmos é apresentada na Seção 4 e as conclusões e trabalhos futuros são apresentados na Seção 5.

2. TRABALHOS RELACIONADOS

Esta seção resume trabalhos relacionados sobre o mapeamento de redes virtuais em substratos de redes. Definições importantes para o entendimento do restante desta seção são a de nós de acesso (nós dehum borda) e a de nós de núcleo (nó do *backbone*). Os nós de acesso são as origens e os destinos dos fluxos de dados enquanto que os nós do *backbone* são responsáveis exclusivamente pelo roteamento das informações.

Em [9], propõe-se uma arquitetura chamada Cabo na qual os provedores de infraestrutura (PI) são responsáveis pelo controle da camada da rede física e os provedores de serviços (PS) são responsáveis pelo fornecimento dos serviços da rede. Diferentemente, a nossa abordagem considera a existência de uma camada entre o PS e o PI, na qual os algoritmos de mapeamento são executados.

Em [19], são apresentadas quatro razões que tornam o problema de mapear redes virtuais tão desafiador. A primeira razão é o grande número de restrições envolvendo os nós da rede, como processamento e memória, bem como envolvendo enlaces, como largura de banda e atraso da rede. Além disso, é necessário mecanismos de controle de admissão na rede, pois os recursos são limitados e, portanto, algumas requisições podem ser potencialmente negadas. O terceiro motivo é o fato de não se conhecer previamente as requisições, e o tempo de vida das redes virtuais. A quarta razão é a diversidade de topologias existentes. Os algoritmos apresentados no presente artigo consideram essas questões colocadas por [19] como desafiadores: a disponibilidade dos recursos (roteadores e enlaces) é levada em consideração na tomada de decisões; as redes virtuais especificam os recursos necessários e o seu tempo de vida apenas no momento da requisição e as soluções não são voltadas para uma topologia de redes específica. Embora o nosso trabalho não inclua, explicitamente, a existência de um mecanismo de controle de admissão, ele pode ser facilmente integrado na arquitetura proposta, de modo que apenas as requisições

aprovadas por ele sejam repassadas para os algoritmos.

Na maioria das soluções propostas na literatura [19] [4] [11], [14], os recursos alocáveis são a capacidade dos enlaces físicos e a capacidade de processamento dos nós físicos da rede. No entanto, outros recursos que possuem forte influência na eficiência da alocação, como memória e acesso a disco, não são incluídos na formulação do problema. Em alguns trabalhos [16] esta questão é sugerida como tema de trabalho futuro mas nenhuma solução foi proposta até o momento. A nossa proposta considera diversos parâmetros negligenciados por outros trabalhos; por exemplo, além das capacidades dos enlaces, considera a memória do roteador físico, a quantidade de elementos de processamento e o tempo de *boot* do roteador virtual.

Na abordagem em [19], o substrato físico possibilita a divisão de caminhos e a migração de caminho, o que permite que o problema seja resolvido em tempo polinomial. Diferentemente de [19], nossos algoritmos processam uma maior quantidade de características do problema, como a presença de imagens que devem ser utilizadas para instanciar os roteadores virtuais.

Em diversos trabalhos na literatura [19], separa-se as etapas de mapeamento de enlaces e de mapeamento de nós. Com o objetivo de introduzir uma correlação entre estas duas etapas, dois algoritmos foram propostos em [4]: o D-ViNE (*Deterministic VN Embedding*) e o R-ViNE (*Randomized VN Embedding*). Nestes, faz-se um mapeamento dos nós virtuais nos nós do substrato, de forma a facilitar o mapeamento dos enlaces virtuais. Estes algoritmos não consideram todos os parâmetros que nossos algoritmos consideram, como por exemplo a presença de imagens virtuais e o atraso nos enlaces da rede. Nos nossos algoritmos, o mapeamento de nós e de enlaces são também realizados de forma combinada. Desse modo, a probabilidade de soluções serem encontradas aumenta.

Em [14], caracteriza-se uma rede virtual somente pela demanda de transporte entre para origem-destino considerando apenas restrições de tráfego entre os nós definido por um conjunto de restrições genéricas.

Os trabalhos existentes na literatura levam em consideração um conjunto restrito de características dos nós ou dos enlaces da rede. Além disso, não tratam de questões como a presença de imagens na rede física que devem ser utilizadas para instanciar os roteadores virtuais. Na Seção 3, os algoritmos introduzidos incluem todas estas questões, que são importantes para se ter um algoritmo de mapeamento realista.

3. ALGORITMOS PROPOSTOS

Propõem-se dois algoritmos para realizar o mapeamento de redes virtuais em substratos de redes. As principais características dos algoritmos são a inclusão de parâmetros essenciais para garantir um mapeamento realista, bem como a existência de imagens de roteadores na rede utilizadas para instanciar as redes virtuais, dado que a instanciação de um roteador virtual exige a transferência da imagem do *software* de roteador de um repositório até o roteador físico e a inicialização dessa imagem.

Os dois algoritmos são baseados em uma formulação de programação inteira. Para a formulação do problema, considera-se que a rede física é representada por um grafo (N, F), onde N é o conjunto dos roteadores físicos e F é o conjunto dos enlaces físicos. Analogamente, a rede virtual é representada por um grafo (M, V), onde M é o conjunto dos roteadores virtuais e V o conjunto dos enlaces virtuais. As entradas para o problema são:

- $N \in \mathbb{Z}$ - Conjunto de roteadores da rede física.
- $F \in \mathbb{Z}$ - Conjunto de enlaces da rede física.
- $M \in \mathbb{Z}$ - Conjunto de roteadores da rede virtual.
- $V \in \mathbb{Z}$ - Conjunto de enlaces da rede virtual.
- $I \in \mathbb{Z}$ - Conjunto de imagens.
- $A \in \mathbb{Z}$ - Quantidade de núcleos disponíveis nos roteadores da rede física.
- $P \in \mathbb{Z}$ - Quantidade de núcleos requisitados pelos roteadores da rede virtual.
- $C \in \mathbb{R}$ - Largura de banda disponível nos enlaces da rede física.
- $Q \in \mathbb{R}$ - Largura de banda requisitada pelos enlaces da rede virtual.
- $D \in \mathbb{R}$ - Atraso nos enlaces da rede física.
- $R \in \mathbb{R}$ - Atraso máximo permitido nos enlaces da rede virtual.
- $L_{n,m} \in \{0, 1\}$ - Define se o roteador virtual m pode ser alocado no roteador físico n. Assume o valor 1 caso a alocação seja permitida e 0 caso contrário.
- $R_{n,i} \in \{0, 1\}$ - Assume o valor 1 caso a imagem i estiver localizada em um repositório conectado diretamente ao roteador físico n, caso contrário assume o valor 0.
- $E_{m,i} \in \{0, 1\}$ - Assume o valor 1 caso a imagem i contiver todos os requisitos necessários pelo roteador virtual m, caso controle assume o valor 0.
- $B \in \mathbb{R}$ - Quantidade de memória de armazenamento disponível nos roteadores da rede física.
- $G \in \mathbb{R}$ - Quantidade de memória de armazenamento necessária para carregar as imagens.

- $S \in \mathbb{R}$ - Tempo máximo requisitado para a instanciação da rede virtual.

- $T_{n,i} \in \mathbb{R}$ - Tempo necessário para a imagem i ser inicializada no roteador físico n.

A inclusão de I, D, R, $L_{n,m}$, $R_{n,i}$, $E_{m,i}$, B, G, S e $T_{n,i}$ em uma mesma formulação é o diferencial deste trabalho em relação aos demais encontrados na literatura.

A função objetivo minimiza a quantidade total de largura de banda alocada nos enlaces físicos para uma rede virtual, o que implica em maximizar a largura de banda disponível para as próximas requisições, aumentando, assim, a chance de se conseguir recursos para efetuar as próximas requisições. Isso significa que o algoritmo retornará a solução em que a banda alocada seja a menor possível, sem ignorar os requisitos das redes virtuais.

A solução do problema é dada pelos valores das seguintes variáveis:

- $X_{n,m,i}$ - Retorna 1 caso o roteador virtual m tenha sido alocado no roteador físico n utilizando a imagem i. Caso contrário, retorna 0. Esta variável retornar o mapeamento dos nós virtuais nos roteadores físicos e a imagem que foi utilizada para cada roteador virtual. Para facilitar o entendimento, cada tupla (n, m, i) pode ser vista como uma possível alocação para o roteador virtual m.

- $Y_{u,v,w}$ - Retorna 1 caso o enlace físico (u, v) seja alocado para o enlace virtual w. Caso contrário, retorna 0. Pode-se a partir dos valores dessa variável, determinar-se o caminho na rede física que foi alocado para cada enlace virtual.

- $Z_{u,v,n,m,i}$ - Retorna 1 caso o enlace físico (u, v) seja utilizado para transferir para n a imagem i, utilizada pelo roteador virtual m, que foi alocado no roteador físico n. Caso contrário, retorna 0. Esta variável retorna quais enlaces físicos foram utilizados para transferir cada imagem para o roteador físico no qual será instanciada. A partir dos valores obtidos por esta variável, tem-se o caminho na rede física pelo qual a imagem foi transferida até chegar no roteador físico destino.

O problema formulado é apresentado a seguir:

$$\text{Minimize} \sum_{u \in N} \sum_{v \in N} \sum_{w \in V} Y_{u,v,w} \times Q(w)$$

sujeito a

$$\sum_{n \in N} \sum_{i \in I} X_{n,m,i} = 1 \quad \forall m \in M \tag{R1}$$

$$\sum_{m \in M} \sum_{i \in I} P(m) \times X_{n,m,i} \leq A(n) \quad \forall n \in N \tag{R2}$$

$$X_{n,m,i} = 0 \quad \forall n \in N, \forall m \in M, \forall i \in I | L_{n,m} = 0 \text{ ou } E_{m,i} = 0 \tag{R3}$$

$$\sum_{w' \in V} Y_{u,v,w'} \times Q(w') \leq C(w) \quad \forall w = (u, v) \in F \tag{R4}$$

$$\sum_{u \in N} \sum_{v \in N} Y_{u,v,w} \times D(u, v) \leq R(w) \quad \forall w \in V, (u, v) \in F \tag{R5}$$

$$\sum_{m \in M} \sum_{i \in I} X_{n,m,i} \times G(i) \leq B(n) \quad \forall n \in N \tag{R6}$$

$$\sum_{u \in N} \sum_{v \in N} \sum_{n \in N} \sum_{i \in I} Z_{u,v,n,m,i} \times D(u, v) + \sum_{u \in N} \sum_{v \in N} \sum_{n \in N} \sum_{i \in I} \frac{Z_{u,v,n,m,i} \times G(i)}{C(u, v)} + \sum_{n \in N} \sum_{i \in I} X_{n,m,i} \times T_{n,i} \leq S \quad \forall m \in M, (u, v) \in F \tag{R7}$$

$$Z_{u,v,n,m,i} \leq X_{n,m,i} \quad \forall u, \forall v, \forall n \in N, \forall m \in M, \forall i \in I \tag{R8}$$

$$Y_{u,v,w}, Z_{u,v,n,m,i} = 0 \quad \forall u, \forall v, \forall n \in N, \forall w \in V, \forall m \in M, \forall i \in I | (u, v) \notin F \tag{R9}$$

$$\sum_{f \in N} Y_{n,f,w} - \sum_{f \in N} Y_{f,n,w} = \quad (R10)$$
$$\forall w = (a,b) \in V, \forall n \in N$$

$$\sum_{i \in I} X_{n,a,i} - \sum_{i \in I} X_{n,b,i}$$
$$\sum_{j \in N} Z_{u,j,n,m,i} - \sum_{j \in N} Z_{j,u,n,m,i} = \quad (R11)$$
$$X_{n,m,i} \times R_{u,i} - X_{n,m,i} \times (1 - \lceil \frac{|u-n|}{\alpha} \rceil)$$
$$\forall m \in M, \forall i \in I, \forall n, u \in N, \alpha = |N|$$

$$X_{n,m,i}, Y_{u,v,w}, Z_{u,v,n,m,i} \in \{0,1\} \quad (R12)$$
$$\forall u, \forall v, \forall n \in N, \forall m \in M, \forall w \in V, \forall i \in I$$

A restrição $(R1)$ garante que um roteador virtual seja alocado em um roteador físico e que uma imagem seja utilizada por ele.A restrição $(R8)$ impede que a imagem i seja transferida para o roteador físico n, caso ela não seja utilizada. A restrição $(R10)$ garante que o conjunto de enlaces físicos alocados para um dado enlace virtual $w = (a,b)$ é um caminho válido na rede física. Para isto, analisa-se o grau de entrada e de saída de cada roteador n da rede física, representados, respectivamente, por $\sum_{f \in N} Y_{f,n,w}$ e $\sum_{f \in N} Y_{n,f,w}$, considerando-se somente os enlaces físicos alocados para o enlace virtual.

A restrição $(R11)$ garante que o conjunto de enlaces físicos selecionados para transferir a imagem i, utilizada pelo roteador virtual m, para o roteador físico n é um caminho válido na rede física. Esta restrição é análoga estrição 10. Assim como naquela restrição, analisa-se o grau de entrada e de saída de cada roteador u da rede física, representados, respectivamente, por $\sum_{j \in N} Z_{j,u,n,m,i}$ e $\sum_{j \in N} Z_{u,j,n,m,i}$, considerando-se somente os enlaces físicos alocados para a transferência da imagem i. A restrição $(R9)$ garante que se o enlace físico (u,v) não existir, então nenhum enlace virtual pode ser alocado utilizando (u,v). A restrição $(R12)$ garante que as variáveis X, Y e Z são binárias.

As restrições $(R2)$ e $(R6)$ referem-se a restrições de limitação dos nós físicos. A restrição $(R2)$ garante que o número de núcleos de cada roteador físico é suficiente para atender o requisito de número de núcleos de cada roteador virtual alocado nele. Cada roteador físico deve possuir mais núcleos do que a soma de todos os núcleos de todos os roteadores virtuais que foram alocados nele. A restrição $(R6)$ garante que a quantidade de memória de armazenamento disponível em cada roteador físico será suficiente para armazenar as imagens de todos os roteadores virtuais que foram alocados nele.

A restrição $(R3)$ garante que os roteadores virtuais só serão instanciados com imagens que atendam todos os seus requisitos de software e em nós físicos que possuam as características permitidas pelo cliente que requisitou a rede virtual.

As restrições $(R4)$ e $(R5)$ referem-se limitações dos enlaces físicos. A restrição $(R4)$ garante que a largura de banda disponível nos enlaces da rede física é suficiente para atender o requisito de largura de banda dos enlaces virtuais. A restrição $(R5)$ garante que o atraso total em um caminho físico alocado para um enlace virtual é menor ou igual ao atraso máximo permitido nos enlaces virtuais alocados.

A restrição $(R7)$ garante que o tempo total gasto para instanciar a rede virtual seja menor que o tempo requisitado pela rede virtual. O tempo necessário para alocar cada roteador virtual é dado pela soma dos seguintes fatores: atraso para transferir a imagem, tempo gasto para transferir a imagem e o tempo gasto para alocar a imagem no roteador físico. A formulação proposta considera que mais de uma imagem pode ser transmitida pelo mesmo enlace físico simultaneamente.

O primeiro algoritmo proposto (algoritmo ótimo) busca a solução exata do problema. Já o segundo algoritmo proposto (algoritmo relaxado) emprega heurísticas para diminuir o tempo de execução do algoritmo. A heurística empregada é a de relaxação. Nesse caso, as variáveis binárias são consideradas como variáveis reais.

Os algoritmos propostos utilizam o CPLEX [12] para resolver o ILP proposto. Algumas etapas realizadas pelo CPLEX ao resolver um problema são: (1) o pré-processamento (CPLEX Presolve), que simplifica e reduz o tamanho do problema, (2) o descobrimento (CPLEX Probe), que analisa as implicações lógicas de se fixar os valores (1 ou 0) das variáveis do problema e (3) a busca pela solução do problema utilizando o método *Branch-and-cut* [12]. Na etapa (3) é criada uma árvore de busca de soluções, sendo que no nó raiz são determinadas soluções iniciais para o problema através de heurísticas, que incluem etapas de relaxação [12]. A diferença entre os dois algoritmos propostos é que um algoritmo procura pela solução ótima do problema através de todos os nós da árvore de busca e o outro termina sua execução no nó raiz da árvore de busca.

4. AVALIAÇÃO DE DESEMPENHO

Experimentos de simulação foram realizados com o objetivo de avaliar 3 métricas: tempo de execução do algoritmo, quantidade de banda passante do substrato que foi alocada para a requisição de rede virtual e taxa de bloqueio das requisições. Logo, a análise dos resultados

busca avaliar o quão úteis para aplicações reais os algoritmos propostos podem ser e quais os aspectos que precisam ser aprimorados em trabalhos futuros. Os algoritmos propostos foram avaliados e comparados em cenários estáticos, nos quais apenas uma requisição de rede virtual é feita, e em cenários dinâmicos, nos quais várias requisições de redes virtuais são feitas ao longo do tempo. Nesse último caso, a disponibilidade dos recursos do substrato varia com o passar do tempo por conta das alocações que vão sendo feitas para as redes virtuais.

Todas as simulações do cenário estático foram realizadas no sistema operacional Debian GNU/Linux Squeeze. O computador utilizado para os experimentos foi um Intel Xeon de 2,27GHz com 2 processadores de 8 núcleos cada, e 6GB de memória RAM. Os dois algoritmos foram implementados em C++ e utilizaram a biblioteca de otimização CPLEX versão 12.0.

As simulações foram realizadas em um simulador desenvolvido em C pelos autores. O simulador recebe como entrada um modelo de rede física e gera eventos de chegada de requisição de redes virtuais. De posse da descrição da rede física e da rede virtual, o simulador invoca os códigos dos algoritmos para que seja devolvida uma alocação e devolve como saída todos os dados de interesse: informação sobre se a requisição foi ou não bloqueada, tempo de execução do algoritmo e banda alocada pelo algoritmo. O mapeamento realizado pelos algoritmos também é devolvido pelo simulador para fins de depuração do seu funcionamento.

4.1. CONFIGURAÇÃO DOS EXPERIMENTOS

Vários cenários estáticos foram avaliados nos experimentos. Avalia-se nestes experimentos se os tempos de execução do algoritmo relaxado são realmente menores do que os do algoritmo ótimo. Em relação ualidade das alocações, o resultado esperado é que o algoritmo ótimo aloque menos largura de banda do que o algoritmo relaxado, mas que o ganho no tempo de execução do algoritmo relaxado seja tal que compense a perda na qualidade da alocação. O conjunto de cenários pode ser obtido através da combinação dos seguintes valores:

- Número de nós do substrato: 5, 7, 10, 12, 15, 17, 20, 22 e 25. A utilização destes valores permite uma análise do desempenho dos algoritmos em função do aumento da rede física. O objetivo com essa faixa de valores é avaliar o desempenho dos algoritmos com substratos de diversos tamanhos. Devido limitações do hardware utilizado nos experimentos, a quantidade de nós físicos foi limitada a 25 nós. Simulações com mais de 25 nós físicos terminavam abruptamente por falta de memória.
- Número de imagens na rede: 20% relativo ao número de nós físicos da rede.
- Quantidade de núcleos disponíveis nos roteadores da rede física: 6 núcleos. Valor baseado em roteadores disponíveis no mercado [6].
- Largura de banda disponível nos enlaces do substrato: uniformemente distribuído entre 1Gbps e 10Gbps. Esta faixa de valores é comum para redes físicas robustas [17].
- Quantidade de memória de armazenamento disponível nos roteadores da rede física: 256MB. Este valor baseia-se na quantidade de memória *flash* presente em roteadores reais encontrados no mercado [5].
- Quantidade de memória de armazenamento necessária para carregar cada imagem: 128MB. Este valor foi baseado na quantidade de memória *flash* recomendada para o *software* definido em [7], um sistema operacional para roteadores.
- Quantidade de tempo necessário para uma imagem ser instanciada em um roteador físico: 10 segundos. Utilizando os roteadores atuais como referência, o sistema operacional dos mesmos demora pouco tempo para ser iniciado tendo em vista que são sistemas bastante reduzidos.
- Tempo limite para instanciar cada rede virtual: 60 segundos.
- Tipos de requisição: Foram definidas três tipos de requisições de redes virtuais: Tipo 1, Tipo 2 e Tipo 3. As requisições diferem entre si em termos da quantidade de recursos requisitada, de tal forma que as requisições do Tipo 3 requisitam mais recursos do que as do Tipo 2 que, por sua vez, requisitam menos recursos do que as do Tipo 1. A Tabela 1 descreve os requisitos de cada um dos tipos das redes virtuais utilizadas nos experimentos.

Espera-se que a quantidade de redes virtuais do Tipo 1 alocadas pelos algoritmos propostos seja maior do que a quantidade de alocações do Tipo 2, que por sua vez devem ser maiores do que a quantidade de alocações do Tipo 3.

A topologia do substrato da rede e das redes virtuais foram geradas aleatoriamente através da ferramenta `BRITE` [15], utilizando o algoritmo BA-2 [1]. Para a rede física, os atrasos nos enlaces da rede permitidos são os próprios valores padrão retornados pelo `BRITE`. Como o atraso permitido nas redes virtuais deve ser maior que o atraso dos enlaces da rede física, o atraso dos enlaces das redes virtuais foi obtido multiplicando o valor retornando pelo `BRITE` por um valor que é dependente do tipo da requisição de rede virtual. Para as requisições de redes virtuais do Tipo 1, o atraso é o valor retornado pelo `BRITE` multiplicado por 15 (o que permite, aproximadamente a

Tabela 1. Descrição dos tipos de redes virtuais.

Tipo	# de nós virtuais	# de núcleos	Largura de banda (uniformemente distribuído)	Probabilidade de alocação de um roteador físico (Restrição de localidade)	Probabilidade de utilização de uma imagem virtual (Restrição de software)
1	5	2	100Mbps–200Mbps	100%	100%
2	8	3	200Mbps–300Mbps	100%	100%
3	10	6	300Mbps–400Mbps	100%	100%

utilização de 15 enlaces físicos por enlace virtual). Para as requisições do Tipo 2, o atraso é o valor retornado pelo `BRITE` multiplicado por 10. Para as requisições do Tipo 3, o atraso é o valor retornado pelo `BRITE` multiplicado por 5.

Para os cenários dinâmicos, o tempo de simulação de cada cenário foi de 10000 segundos. O intervalo de chegada e a duração das requisições foram definidos como um número aleatório exponencialmente distribuído com média de 25 e 1000 segundos, respectivamente. O substrato possui 25 nós físicos e a banda disponível em cada enlace é um valor aleatório uniformemente distribuído entre 1Gbps e 10Gbps. Da mesma forma que no cenário estático, foram definidos os três tipos de requisições, similares aos da Tabela 1. A única diferença é com relação probabilidades referentes estrição de localidade e estrição de software. Ao invés de 100% para as requisições do Tipo 2 e do Tipo 3, esses valores são 80% e 60%, respectivamente. Em todos os tipos de requisições, o número de nós virtuais é um número uniformemente distribuído entre 1 e 5.

Em todos os experimentos realizados cada cenário foi simulado 5 vezes. Quando não especificado, os resultados apresentados a seguir correspondem médias dos valores encontrados. Os resultados das diversas instâncias foram bastante próximos entre si e por isso os intervalos de confiança não são exibidos nos gráficos para facilitar a visualização. O nível de confiança considerado foi de 95%.

4.2. Resultados e Discussões

A Tabela 2 e o gráfico da Figura 1 apresentam os resultados obtidos nos cenários estáticos.

A Tabela 2 exibe a média dos tempos de execução dos algoritmos para cada um dos 3 tipos de requisição. Em média o tempo de execução do algoritmo relaxado é menor do que a do algoritmo ótimo em todos os casos, conforme esperado. Para as requisições do Tipo 1, 2 e 3, o tempo de execução do algoritmo relaxado foi, respectivamente, em média, menor do que 92%, 74% e 27%.

Com relação ariação do tempo de execução em função da quantidade de nós, observou-se que eles são diretamente proporcionais. Os gráficos não são exibidos dado limitação de espaço.

O gráfico da Figura 1 plota a quantidade de largura de banda alocada pelos algoritmos em função do número

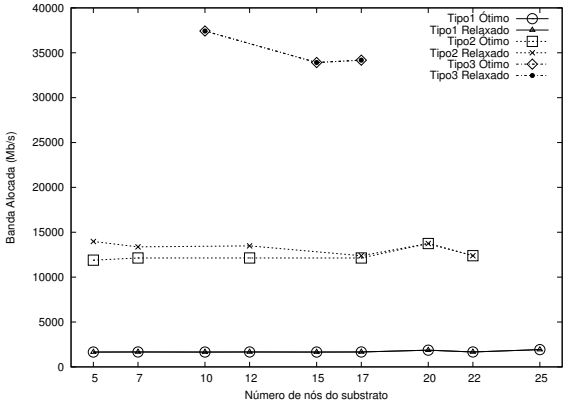


Figura 1. Banda alocada - Cenário estático.

de nós da rede física. A quantidade alocada pelo algoritmo relaxado foi muito próxima locada pelo algoritmo ótimo. Na maioria dos cenários, inclusive, os valores foram iguais. A maior diferença entre as quantidades alocadas pelos dois algoritmos ocorreu para a requisição do Tipo 2 quando haviam 5 nós no substrato, e foi apenas de 15%. Pode-se observar que a medida que as requisições se tornam mais rígidas, a quantidade de pontos apresentados no gráfico diminui. Isso se dá pelo fato de não haver solução para aqueles casos. Por exemplo, para as redes do Tipo 3, em apenas 3 configurações (10, 15 e 17 nós) uma solução foi encontrada.

Os gráficos das figuras 2, 3, 4 e 5 apresentam os resultados obtidos no cenário dinâmico. Os gráficos das figuras 2, 3 e 4 exibem o tempo de execução dos algoritmos para o três tipos de requisição ao longo do tempo (cada gráfico apresenta o resultado de uma única instância simulada. Os resultados de todas as instâncias, para uma mesma configuração, foram bem próximos). O tempo de execução apresentado equivale ao tempo gasto pelos algoritmos para devolver as alocações. Para requisições mais complexas, o tempo de execução do algoritmo ótimo é impraticável mas, de um modo geral, isso não ocorre com o tempo de execução do algoritmo relaxado, como pode ser visto pela curva das requisições do Tipo 3 (Figura 4). O motivo para tal é o fato do algoritmo relaxado parar sua execução no nó raiz da árvore de busca da programação inteira, enquanto o algoritmo ótimo continua realizando a busca até percorrer toda árvore ou até o tempo limite esta-

Tabela 2. Tempo de execução dos algoritmos – Cenário estático.

Tipo de requisição	Algoritmo	Média (s)
1	Ótimo	608,11
1	Relaxado	46,66
2	Ótimo	398,44
2	Relaxado	104,11
3	Ótimo	1202,55
3	Relaxado	882,44

belecido ser atingido. Desta forma, para requisições mais complexas ou para redes físicas com um maior número de nós, o número de soluções para o problema aumenta, fazendo com que a árvore de busca aumente também. É possível observar nos gráficos das figuras 2 e 3 que os tempos de execução dos algoritmos diminuiu com o passar do tempo. Isso ocorreu pelo fato da rede estar ocupada com as requisições anteriores. Dessa forma, os algoritmos não encontram solução, por não haverem recursos disponíveis, e devolvem essa informação rapidamente.

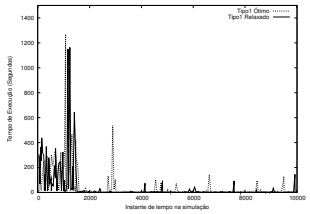


Figura 2. Tempo de execução dos algoritmos (Tipo 1) - Cenário dinâmico

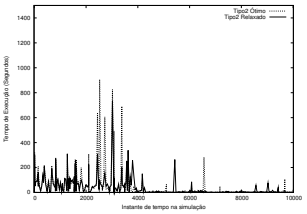


Figura 3. Tempo de execução dos algoritmos (Tipo 2) - Cenário dinâmico

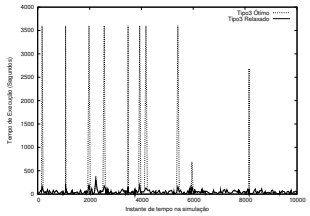


Figura 4. Tempo de execução dos algoritmos (Tipo 3) - Cenário dinâmico

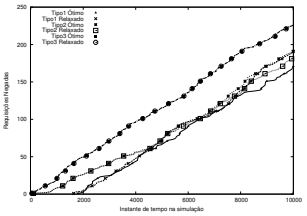


Figura 5. Requisições negadas - Cenário dinâmico

O gráfico da Figura 5 mostra o número total de requisições bloqueadas. Da mesma forma, pode-se perceber que o bloqueio produzido dos dois algoritmos são muito semelhantes. O resultado exibido no gráfico da Figura 5 confirma os baixos tempos de execução dos algoritmos nas figuras 2, 3 e 4 a medida que o tempo aumenta. Como a maioria dos recursos está ocupada, a quantidade de requisições bloqueadas aumenta, o bloqueio é influenciado pelo tipo de rede requisitada. Alocações de redes do Tipo 3 são mais bloqueadas do que as do Tipo 2, que por sua vez são mais bloqueadas do que as redes do Tipo 1, como esperado.

5. CONCLUSÕES

Neste trabalho, foram propostos dois algoritmos para o mapeamento de redes virtuais em um substrato da rede que consideram ambientes realistas. O algoritmo relaxado é capaz de encontrar a solução bem mais rapidamente do que o algoritmo ótimo, enquanto que a banda passante alocada pelos dois algoritmos é bem semelhante. Além disso, o bloqueio produzido pelos algoritmos é semelhante.

Durante as simulações foi constatado que os algoritmos utilizam muita memória RAM para serem executados. Mesmo o algoritmo relaxado pode levar um tempo consideravelmente alto em ambientes com muitos nós físicos e requisições complexas. Desta forma, tem-se como metas futuras a determinação dos valores númericos do consumo de memória dos algoritmos e o desenvolvimento de um algoritmo que seja capaz de encontrar uma solução satisfatória em um tempo menor do que o dos algoritmos propostos neste trabalho. Além disso, este novo algoritmo deve ser capaz de encontrar soluções utilizando pouca memória RAM.

Como os trabalhos presentes na literatura não levam em consideração a maioria dos parâmetros dos algoritmos propostos neste trabalho, comparações com algoritmos como os apresentados em [4] e [13] exigem adaptações para a entrada e a saída de métricas específicas. Essas comparações serão realizadas pelos autores com o objetivo de se quantificar o impacto decorrente de uma quantidade maior de parâmetros nos algoritmos de mapeamento.

6. AGRADECIMENTOS

Esta pesquisa foi parcialmente financiada pela Fundação de Amparo esquisa do Estado de São Paulo (FAPESP), processo 2010/03422-5.

Referências

[1] Réka Albert and Albert L. Barabási. Topology of Evolving Networks: Local Events and Universality. *Physical Review Letters*, 85(24):5234–5237, Dec 2000.

[2] David G. Andersen. Theoretical Approaches to Node Assignment. `http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps`. Último acesso em 20/12/2010, Dez 2002.

[3] R. Bless, C. Hiibsch, S. Mies, and O.P. Waldhorst. The Underlay Abstraction in the Spontaneous Virtual Networks (SpoVNet) Architecture. In *Next Generation Internet Networks (NGI 2008)*, pages 115–122, Abril 2008.

[4] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. In *IEEE INFOCOM*, pages 783–791, Abril 2009.

[5] Cisco Systems. Cisco 7200 Series Routers Overview [Cisco 7200 Series Routers], 2010. `http://www.cisco.com/en/US/prod/collateral/routers/ps341/product_data_sheet09186a008008872b.html`. Último acesso em 20/12/2010.

[6] Cisco Systems. Cisco Multiprocessor WAN Application Mode [Cisco Catalyst 6500 Series Switches], 2010. `http://www.cisco.com/en/US/prod/collateral/modules/ps5510/product_data_sheet0900aecd800f8965_ps708_Products_Data_Sheet.html`. Último acesso em 20/12/2010.

[7] Cisco Systems. Download Software, 2010. `http://www.cisco.com/cisco/software/release.html?mdfid=278807391&flowid=956&softwareid=280805680&release=12.4.2-XB11&rellifecycle=GD&relind=AVAILABLE&reltype=latest`. Último acesso em 20/12/2010.

[8] J. Fan and M. H. Ammar. Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies. In *IEE INFOCOM*, pages 1–12, Abril 2006.

[9] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to Lease the Internet in Your Spare Time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64, 2007.

[10] Jiayue He, Rui Zhang-Shen, Ying Li, Cheng-Yen Lee, Jennifer Rexford, and Mung Chiang. DaVinci: Dynamically Adaptive Virtual Networks for a Customized Internet. In *ACM CoNEXT '08*, pages 15:1–15:12, 2008.

[11] Ines Houidi, Wajdi Louati, and Djamal Zeghlache. A Distributed and Autonomic Virtual Network Mapping Framework. In *ICAS '08*, pages 241–247, 2008.

[12] IBM. IBM ILOG CPLEX Optimization Studio V12.2, 2010. `http://publib.boulder.ibm.com/infocenter/cosinfoc/v12r2/index.jsp?topic=/ilog.odms.cplex.help/Content/Optimization/Documentation/CPLEX/`. Último acesso em 20/12/2010.

[13] Jens Lischka and Holger Karl. A Virtual Network Mapping Algorithm Based on Subgraph Isomorphism Detection. In *ACM VISA '09*, pages 81–88, 2009.

[14] Jing Lu and Jonathan Turner. Efficient Mapping of Virtual Networks onto a Shared Substrate. Technical Report WUCSE-2006-35, Washington University, 2006. `http://www.arl.wustl.edu/~jst/pubs/wucse2006-35.pdf`. Último acesso em 20/12/2010.

[15] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. Brite, 2010. `http://www.cs.bu.edu/brite/`. Último acesso em 20/12/2010.

[16] Pradeep Padala, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, and Kenneth Salem. Adaptive Control of Virtualized Resources in Utility Computing Environments. In *ACM EuroSys '07*, pages 289–302, 2007.

[17] RNP. Mapa do backbone RNP, 2010. `http://www.rnp.br/backbone/`. Último acesso em 20/12/2010.

[18] Dirk Trossen. Invigorating the Future Internet Debate. *SIGCOMM Comput. Commun. Rev.*, 39(5):44–51, 2009.

[19] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, 2008.

[20] Y. Zhu and M. Ammar. Algorithms for Assigning Substrate Network Resources to Virtual Network

Components. In *IEEE INFOCOM*, pages 1–12, Abril 2006.

[21] Yaping Zhu, Andy Bavier, Nick Feamster, Sampath Rangarajan, and Jennifer Rexford. UFO: a resilient layered routing architecture. *SIGCOMM Comput. Commun. Rev.*, 38(5):59–62, 2008.

[22] Yaping Zhu, Rui Zhang-Shen, Sampath Rangarajan, and Jennifer Rexford. Cabernet: Connectivity Architecture for Better Network Services. In *ACM CoNEXT '08*, pages 64:1–64:6, 2008.

Uma Análise do Impacto da Elasticidade no Lucro de Provedores de Computação na Nuvem

Rostand Costa[1,2], Francisco Brasileiro[1], Guido Lemos[2], Dênio Mariz[2]

[1]Universidade Federal de Campina Grande Departamento de Sistemas e Computação
Av. Aprígio Veloso, s/n, Bodocongó 58.429-900 - Campina Grande, PB - Brasil
{rostand, fubica}@lsd.ufcg.edu.br

[2]Universidade Federal da Paraíba Departamento de Informática
Laboratório de Aplicações de Vídeo Digital 58.050-900, João Pessoa, PB
{rostand, guido, denio}@lavid.ufpb.br

Abstract

The cloud computing paradigm allows the provision of Information Technology infrastructure in the form of a service that clients acquire on-demand and paying only for the amount of service that they actually consume. Many embarrassingly parallel applications could potentially achieve an enormous benefit from the elasticity offered by cloud computing providers. The execution time of these applications is inversely proportional to the amount of computing resources available to process them. Unfortunately, current public cloud computing providers do impose strict limits on the amount of resources that a single user can simultaneously acquire. In this paper we analyze the reasons why traditional providers need to impose such a limit. We show that increases on the limit imposed have a severe impact on the profit achieved by the providers. This leads to the conclusion that new approaches to deploy cloud computing services are required to properly serve embarrassingly parallel applications.

Keywords: Cloud computing, elasticity, profit, bag-of-tasks

Resumo

O paradigma da computação na nuvem permite o fornecimento de infraestrutura de Tecnologia da Informação sob a forma de um serviço que os clientes adquirem sob demanda e pagam apenas pela quantidade de serviços que realmente consomem. Muitas aplicações que processam grandes cargas de trabalho em paralelo poderiam potencialmente se beneficiar da elasticidade oferecida pelos provedores de computação na nuvem. Essas aplicações têm seu tempo de execução inversamente proporcional à quantidade de recursos computacionais usados para processá-las. Infelizmente, os provedores públicos atuais de computação na nuvem precisam impor um limite estrito na quantidade de recursos que um único usuário pode adquirir concomitantemente. Neste trabalho nós analisamos por que esse limite precisa ser imposto. Nossos resultados mostram que aumentos no limite imposto têm um grande impacto na lucratividade do provedor. Desse modo, é preciso pensar em novas formas de oferecer computação na nuvem para que as aplicações estudadas possam ser atendidas de forma apropriada.

Palavras-chave: Computação na nuvem, elasticidade, lucratividade, utilização intensiva.

1. INTRODUÇÃO

Computação na nuvem é um paradigma em evolução que permite o fornecimento de Tecnologia da Informação (TI) como um serviço que pode ser adquirido *on line* e sob demanda pelos clientes. Os recursos utilizados

para prover serviço aos clientes podem ser rapidamente provisionados e liberados pelos provedores do serviço, que utilizam um modelo de tarifação onde o cliente paga apenas pelo que foi efetivamente consumido. Este paradigma pode ser usado em diferentes níveis da pilha de TI [15]. Por exemplo, no nível mais alto, clientes podem adquirir serviços que provêem uma funcionalidade particular de software. Este tipo de fornecimento de TI é normalmente chamado de SaaS (do inglês, *software-as-a-service*). Similarmente, o nível mais baixo da pilha, clientes podem adquirir máquinas virtuais totalmente funcionais executando um determinado sistema operacional, sobre o qual eles podem instalar e executar as suas próprias aplicações. Este tipo de serviço recebeu o nome de IaaS (do inglês, *infrastructure-as-a-service*) [15].

Este trabalho está focado no último tipo de serviço. Dessa forma, no restante deste artigo serão usados os termos computação na nuvem e IaaS com o mesmo propósito.

Ao adquirir recursos de TI de um provedor de computação na nuvem, os clientes podem desfrutar da elasticidade oferecida, podendo aumentar e diminuir o seu consumo de serviços de uma forma virtualmente ilimitada, sem qualquer custo adicional. Em teoria, essa elasticidade ilimitada permitiria aos usuários decidir livremente, por exemplo, se desejam usar 1 recurso por 1.000 horas ou 1.000 recursos por 1 hora, pagando o mesmo preço em ambos os casos.

A elasticidade do modelo de computação na nuvem é particularmente interessante para uma classe importante de aplicações que está se tornando cada vez mais popular. As chamadas aplicações de e-ciência são caracterizadas por cargas de trabalho que requerem computação intensiva. Muitas destas aplicações podem ser paralelizadas trivialmente, através da quebra do trabalho a ser realizado em várias tarefas menores que podem ser processadas independentemente. Esta classe de aplicação é referenciada na literatura como aplicações "embaraçosamente paralelas" ou simplesmente "saco-de-tarefas" (BoT, do inglês *bag-of-tasks*) [4]. Por exemplo, as simulações de Monte Carlo, que podem envolver a execução de milhares de cenários diferentes, podem ser paralelizadas simplesmente pela execução de cada cenário em uma unidade de processamento diferente. Aplicações que processam enormes quantidades de dados podem usualmente ser paralelizadas através da divisão dos dados entre um número de processos idênticos que executam a computação sobre cada bloco de dados independentemente; no final, pode ser necessário realizar algum tipo de consolidação dos processamentos individuais. A renderização de imagens complexas e vídeos se encaixa bem nesta descrição. A lista de aplicações BoT é vasta e engloba não apenas usuários da academia, mas também da indústria e do governo. Além disso, a quantidade crescente de dados gerada e consumida pela sociedade moderna deve aumentar a pressão para executar eficientemente estas aplicações [8].

Se o cliente que necessita executar uma aplicação BoT fosse capaz de requisitar de um provedor de computação na nuvem tantas máquinas virtuais quanto as necessárias para maximizar o nível de paralelização da execução da aplicação, isto lhe permitiria executar esta aplicação no menor tempo possível, sem que isso implicasse em um gasto extra com os recursos computacionais usados. A elasticidade do serviço oferecido por um provedor de IaaS é, obviamente, limitada pela quantidade física de recursos que ele dispõe. Acontece que, atualmente, esse limite é muito mais restritivo, uma vez que os provedores de computação na nuvem em operação restringem a quantidade de recursos que cada cliente pode demandar de cada vez a um número relativamente muito baixo, comparado com a capacidade dos provedores. Por exemplo, no momento em que este texto estava sendo escrito, um dos principais provedores comerciais em atividade limitava em 20 o número de máquinas virtuais que podem ser instanciadas de forma dedicada (*on-demand instances*) e em 100 o número de máquinas virtuais que podem ser instanciadas segundo um modelo *"best-effort"* (*spot instances*) [1]. Para este provedor em particular, clientes podem usar um canal paralelo de negociação para tentar aumentar este limite de forma *ad hoc*, mas como as condições sob as quais uma negociação é bem sucedida não são documentadas, nós consideramos neste artigo apenas o canal de comunicação automático. Aparentemente, o alvo do limite não é o volume da requisição em si, mas o exercício extremo da elasticidade através de grandes alocações com liberações logo em seguida. Como já existem serviços de alta demanda hospedados em provedores de computação na nuvem (ex. Gmail, Twitter, Bing etc.) e também a possibilidade de se negociar alocações superiores, é possível inferir que o limite serve como um regulador do uso intensivo de recursos por períodos curtos.

Embora os limites atualmente impostos pelos provedores de IaaS não impeçam que a maioria dos clientes enxerguem o serviço provido como uma fonte infinita de recursos, este não é o caso para a maioria das aplicações BoT. Estas aplicações requerem a instanciação de um sistema com milhares de máquinas virtuais. Além disso, quanto mais máquinas elas puderem usar, mais curto será o tempo de utilização das mesmas. O projeto Belle II Monte Carlo [14], por exemplo, requer de 20.000 a 120.000 máquinas virtuais para o processamento em tempo aceitável dos dados produzidos em três meses de experimentos. Ou seja, eles têm uma altíssima demanda por recursos de forma bastante esporádica. Esse padrão de consumo é muito comum entre os usuários que executam aplicações BoT.

Neste artigo nós fazemos uma análise que tenta identificar as razões que levam os provedores de IaaS a imporem limites que restringem a utilidade de seus serviços para a execução de aplicações da classe BoT. Nossa metodologia baseia-se no uso de simulação. Inicialmente definimos um modelo simplificado de provedores de IaaS, apresentado na Seção 3, e um gerador de cargas de trabalho sintéticas apropriadas para esse modelo, discutido na Seção 4. Em seguida, nós apresentamos o modelo de simulação utilizado (Seção 5.1). Para instanciar o modelo de simulação de forma adequada, nós realizamos um projeto de experimento para identificar as variáveis aleatórias do modelo que tinham um maior impacto na variável de resposta, e dessa forma definir os cenários de experimentação (Seção 5.2). Os resultados das simulações executadas que apresentamos na Seção 6 apontam que aumentos no limite imposto pelo provedor de IaaS levam a impactos substanciais na lucratividade do provedor. Dessa forma, é pouco provável que os provedores de IaaS atualmente em operação possam vir a oferecer um serviço adequado para os usuários que precisam executar aplicações BoT. Na seção de conclusão deste artigo nós indicamos uma possível alternativa para a implantação de um serviço de IaaS que possa atender apropriadamente essa classe de aplicações.

Antes de apresentar o nosso estudo sobre o impacto que o limite no número máximo de recursos que um cliente pode instanciar concomitantemente tem na lucratividade de um provedor de IaaS, na próxima seção nós fazemos uma breve revisão da literatura relacionada com o nosso trabalho.

2. Trabalhos Relacionados

O trabalho de Menascé e Ngo discute como os métodos tradicionais de planejamento de capacidade foram impactados com o advento da computação na nuvem e como o atendimento dos níveis de serviço negociados passam a exigir novos mecanismos, como técnicas de computação autonômica [12]. São considerados os pontos de vista tanto do cliente quanto do provedor, com a percepção de que todos os riscos e custos associados com o provisionamento de recursos migram dos ombros dos clientes para os ombros dos provedores. O aprofundamento que fazemos neste artigo nos aspectos de disponibilidade e regulação da demanda pelos provedores confirma esta condição, indicando que novas abordagens são necessárias para a construção de infraestrutura para computação na nuvem tanto para contornar as limitações atuais quanto para aliviar a carga dos provedores.

O estudo de Greenberg *et al.* [7] mostra que os custos típicos associados para a construção de centros de processamento de dados para nuvens possuem a seguinte distribuição: aquisição de servidores, incluindo hardware e software, respondem por 45% do custo total; montagem da infraestrutura, incluindo refrigeração e instalações lógicas e elétricas, consomem 25% dos recursos; equipamentos e canais de comunicação em geral são responsáveis por 15% do orçamento e os 15% restantes ficam por conta de fornecimento de energia e outras despesas. Um arcabouço para a análise detalhada destes investimentos e como eles compõem o custo total de propriedade [13] dos provedores foi proposto por Li *et al.* [11]. Na abordagem proposta, os quatro principais grupos de custos identificados por Greenberg *et al.* são expandidos para oito categorias e, em complemento aos investimentos iniciais, são também considerados os custos relacionados com a operação do centro de processamento de dados, considerando a elasticidade no consumo de recursos da nuvem. Para isto, o arcabouço permite a apuração do custo de amortização e do custo de utilização de cada recurso virtual. No primeiro caso, é definido quanto cada recurso custa por estar disponível para o uso e, no segundo caso, quanto custa ao ser efetivamente usado pelos clientes. No modelo usado no nosso trabalho nós utilizamos esses dois custos para computar o valor do lucro obtido por um provedor de IaaS.

No trabalho de Anandasivam *et al.* [2] é introduzida uma versão do conceito de preço auto-ajustável adaptada para computação na nuvem. Considerando um contexto com demandas dinâmicas e imprevisíveis, no qual o provedor precisa decidir como alocar os seus recursos finitos de forma a maximizar a sua lucratividade, a aplicação de um sistema de leilão pode atuar como um influenciador no comportamento de usuários sensíveis ao preço dos recursos. Este mecanismo pode ter impactos positivos no controle da capacidade do provedor. Através do nosso estudo é possível constatar que o limite imposto pelos provedores também é usado como um regulador da demanda dos usuários para conciliar os picos de utilização com a sua capacidade instalada.

3. Um Modelo Simplificado de um Provedor de IaaS

Os serviços oferecidos por provedores de computação na nuvem precisam, normalmente, fornecer garantias de qualidade de serviço (QoS, do inglês *Quality of Service*) que atendam plenamente os requisitos estabelecidos com os clientes que adquirem os seus serviços, expressos através de um acordo de nível de serviço (SLA, do inglês *service level agreement*). Muitas dessas garantias são providas através da manutenção de capacidade excedente pelo provedor. Por outro lado, os custos do provedor são reduzidos pelas vantagens que a economia de escala pode proporcionar-lhes. Por exemplo, a con-

centração de sua estrutura em grandes centros de processamento de dados, dedicados e centralizados, e o compartilhamento de recursos físicos através da virtualização são estratégias cruciais para efetivamente oferecer serviços de uma forma economicamente viável. Sua competitividade também é baseada na capacidade de realizar uma multiplexação estatística de picos e vales no uso simultâneo de recursos por um grande número de clientes. Outra vantagem é o nível de automação atingido pelos provedores de computação na nuvem que, entre outras coisas, permite que eles reduzam substancialmente a relação de funcionários por servidores. Adicionalmente, os provedores podem obter um aumento no nível de utilização dos seus serviços através da oferta de um portfólio de serviços que contemple diferentes modelos de precificação [1].

Dentre as muitas propriedade de QoS que um provedor de computação na nuvem precisa observar, neste trabalho nós iremos nos concentrar na disponibilidade do serviço (*service availability*), isto é, na probabilidade de que um cliente que solicita um serviço tenha o seu pedido plenamente atendido[1]. Esta propriedade não deve ser confundida com a confiabilidade do serviço (*service reliability*), que é representada pela probabilidade de que o serviço provido não irá falhar enquanto o cliente estiver usandoo. Por exemplo, no caso de um provedor de IaaS, a sua disponibilidade é afetada quando um cliente solicita uma nova máquina virtual e o provedor é incapaz de instanciar o recurso pedido, enquanto que a sua confiabilidade é afetada sempre que uma máquina virtual que tenha sido instanciada sofre uma falha.

Considerando que os recursos de um provedor são alocados a cada cliente por um intervalo de tempo mínimo, por exemplo, 1 hora, vamos assumir que o tempo é discretizado em intervalos de tamanho fixo (fatias), e modelaremos um provedor IaaS P em um determinado período de observação de tamanho ΔT como uma tupla:

$$P = \langle K, L, U, D, A, C_i, C_u, V, E \rangle,$$

onde:

- K é a quantidade de recursos disponíveis no servidor;
- L é a quantidade máxima de recursos que pode ser alocado em cada fatia de tempo;
- U é o conjunto de usuários registrados no servidor;
- D é a distribuição de demanda desses usuários;
- A é a estratégia de alocação de recursos usada pelo provedor;
- C_i é o custo incorrido pelo provedor para disponibilizar cada recurso individual por fatia de tempo, obtido pelo rateio da amortização do custo total de propriedade pelos recursos disponíveis e pelas fatias possíveis no mesmo período[2];
- C_u é o custo adicional incorrido pelo provedor, por fatia de tempo, gasto somente quando cada recurso individual está sendo efetivamente usado, baseado no conceito de custo de utilização proposto por Li *et al.* [11] e considerando que algum nível de eficiência energética é praticado [3];
- V é o valor cobrado dos usuários pela utilização efetiva de um recurso por uma fatia de tempo ou fração;
- E é o encargo para o provedor por cada violação cometida; ele pode ser tangível (ex. compensação para o usuário) ou intangível (ex. dano na imagem do provedor). Neste trabalho nós consideramos apenas o aspecto tangível dos encargos por violações.

Na próxima seção nós apresentamos em detalhes como a demanda D dos usuários U de um provedor P é descrita. Por enquanto, basta assumir que $d(u,t), 0 \leq d(u,t) \leq L, \forall u \in U, 1 \leq t \leq \Delta T$, é a quantidade de recursos demandada pelo usuário u na fatia t. Dependendo do padrão de demanda (D) dos usuários do provedor, da estratégia de alocação (A) e da capacidade (K) do provedor, cada usuário u que requisita $d(u,t)$ recursos na fatia t irá receber uma alocação de recursos associada que é expressa por $a(u,t), 0 \leq a(u,t) \leq d(u,t)$. Quando $a(u,t) < d(u,t)$ temos uma violação na disponibilidade do serviço do provedor. Dessa forma, o número total de violações ocorridas em uma fatia t é dado por:

$$v(t) = \sum_{u \in U} 1 - \lfloor \frac{a(u,t)}{d(u,t)} \rfloor \quad (1)$$

Seja $\alpha(t)$ a utilização do provedor na fatia de tempo t. $\alpha(t) = \sum_{u \in U} a(u,t)$. Uma maneira de aferir a eficiência do provedor é medir o seu lucro no período de tempo considerado, o qual é representado em nosso modelo através da fórmula:

$$\Lambda = \sum_{t=1}^{\Delta T} [(V - C_u) \times \alpha(t) - v(t) \times E] - K \times C_i \times \Delta T \quad (2)$$

[1] O foco em disponibilidade foi uma simplificação para tornar o modelo tratável, outras dimensões serão abordadas em trabalhos futuros.

[2] Embora os custos descritos possuam um comportamento linear e sejam uma simplificação dos custos reais, de perfil mais complexo, essa simplificação oferece uma boa aproximação e atende as necessidades do nosso modelo.

4. GERAÇÃO DE CARGAS DE TRABALHO SINTÉTICAS PARA UM PROVEDOR DE IAAS

Por causa da indisponibilidade de traços de execuções reais ou mesmo caracterizações da carga de trabalho de provedores de IaaS, nós tivemos que criar um gerador de cargas de trabalho sintéticas, para definir a demanda imposta ao provedor modelado na seção anterior.

O uso total do sistema em cada fatia de tempo t, representado por $\alpha(t)$, é resultante do perfil de uso de cada usuário individual ou, em outras palavras, da forma como ele requisita recursos dentro do limite imposto pelo provedor durante o seu tempo de permanência no sistema. Em princípio, todos os usuários podem, sob demanda e sem custos adicionais, se beneficiar da inerente elasticidade do serviço e usar qualquer quantidade de recursos (desde nenhum) até o limite L imposto pelo provedor, em qualquer fatia de tempo.

Considerando o comportamento do sistema no intervalo de tempo de duração ΔT, algumas categorias de usuários irão emergir. Uma classificação inicial dos usuários está relacionada com o nível de demanda observada no período considerado. Os usuários ativos são aqueles que fizeram alguma demanda por recursos do sistema em um dado intervalo, ou seja, $d(u,t) > 0$ para algum valor de $t, 1 \leq t \leq \Delta T$. Os outros usuários são ditos inativos. Seja U_a o conjunto de usuários ativos; $U_a = \{u | u \in U \wedge \exists t, 1 \leq t \leq \Delta T, d(u,t) > 0\}$.

O comportamento de cada categoria de usuário ativo é descrito através do uso das distribuições tradicionalmente associadas na literatura com classes de usuários e sessões de uso [6, 16, 9]. Para geração da carga de trabalho foi aplicada a abordagem de geração hierárquica, usando uma modelagem baseada no usuário [6]. Esta técnica baseia-se na separação do comportamento dos usuários em três níveis: perfil da população/duração da sessão/atividade dentro da sessão, contemplando aspectos como localidade e amostragem [6], além de auto-similaridade [6]. Com isto, é possível a inclusão na carga de trabalho gerada de longas permanências e ausências (cauda longa [9]) e também de comportamentos regulares. O sistema modelado é do tipo fechado, com um número conhecido e finito de usuários ($|U_a|$).

A população de usuários ativos pode ser dividida em dois grupos, considerando a regularidade de demanda dos mesmos. Usuários ativos regulares são aqueles com uso ininterrupto. O conjunto de usuários regulares é descrito da seguinte forma: $U_r = \{u | u \in U_a \wedge \forall t, 1 \leq t \leq \Delta T, d(u,t) > 0\}$. Os usuários eventuais são os usuários ativos não regulares, ou seja, $U_e = U_a - U_r$.

Nós assumimos que os usuários regulares têm apenas uma sessão, cuja duração engloba pelo menos todo o intervalo de ΔT fatias considerado. Por outro lado, para os usuários eventuais o tempo de sessão é governado pelas seguintes variáveis aleatórias:

- $\tilde{o}$: duração (em fatias) de cada sessão de um usuário eventual, seguindo uma distribuição uniforme com limite inferior l_o e limite superior u_o [9]; e
- $\tilde{i}$: intervalo entre sessões, seguindo uma distribuição pareto com parâmetros k_i e s_i [9].

Dentro de cada sessão o usuário pode estar"em atividade" ou em "espera" (*think time*), que indicam, respectivamente, se o usuário está efetivamente usando recursos, ou não. O comportamento de cada usuário em sessão pode ser definido pela quantidade de recursos que ele utiliza, pela duração deste uso e também pelo tempo que ele fica sem usar os recursos do sistema. Desta forma, cada atividade pode ser caracterizada pela tupla $\mathcal{A} = \langle r, n, e \rangle$, onde r e n representam a quantidade de recursos requisitados por fatia de tempo e a duração da atividade em número de fatias, respectivamente, e e representa o tempo de espera até a próxima fatia quando o usuário estará em atividade. A mudança na quantidade de recursos, embora possível, implica no início de outra atividade. A seguir, serão descritos os perfis de uso de cada categoria de usuário da nossa população.

O perfil de uso dos usuários regulares foi modelado de uma forma simplificada. Usuários regulares apresentam atividades ininterruptas (sem espera) que duram uma fatia de tempo. Em cada sessão o número de recursos demandados é baseado na variável aleatória $\tilde{m}$ com distribuição normal, média τ e variância σ, onde τ é o *ticket* médio dos usuários regulares, dado por:

$$\tau = \frac{\frac{\sum_{t=1}^{\Delta t} \sum_{u \in U_r} a(u,t)}{\Delta T}}{|U_r|} \tag{3}$$

O perfil de atividade dos usuários regulares é definido como:

$$\mathcal{A}_{regular} = \langle \tilde{m} \sim N(\tau, \sigma), 1, 0 \rangle \tag{4}$$

Esta abordagem permite contemplar eventuais aumentos ou diminuições do tamanho das atividades dos usuários regulares que, através de compensação, não alterem substancialmente a utilização total dos usuários regulares do sistema em cada fatia de tempo. Mudanças mais abruptas no comportamento de usuários regulares que afetam este relacionamento serão tratadas adiante.

O comportamento em sessão dos usuários eventuais, por sua vez, é baseado em três variáveis aleatórias:

- $\tilde{s}$: quantidade de recursos alocados em cada atividade, seguindo uma distribuição uniforme entre 1 e L [9];

- $\tilde{d}$: duração (em fatias) de cada atividade, seguindo uma distribuição exponencial com média λ_d [9]; e
- $\tilde{t}$: intervalo (em fatias) entre atividades (*think time*), seguindo uma distribuição exponencial com média λ_t [9].

O perfil de atividades dos usuários eventuais é definido como:

$$\mathcal{A}_{eventual} = \langle \tilde{s} \sim U(1, L), \tilde{d} \sim E(\lambda_d), \tilde{t} \sim E(\lambda_t) \rangle \quad (5)$$

Dois perfis particulares de usuários eventuais foram também modelados para cobrir as seguintes situações: a) usuários regulares apresentando uma demanda não usual por recursos motivada por *flurries* ou *flashmobs* em seus serviços [10]; b) usuários eventuais com utilização intensiva e sensível ao tempo (aplicações BoT) [14] que consomem todos os recursos disponíveis. Estes perfis são definidos da seguinte forma:

$$\mathcal{A}_{flashmob} = \langle U(\tau+1, L), \tilde{d} \sim E(\lambda_d), \tilde{t} \sim E(\lambda_t) \rangle \quad (6)$$

$$\mathrm{A}_{BoT} = \langle L, \tilde{d} \sim E(\lambda_d), \tilde{t} \sim E(\lambda_t) \rangle . \quad (7)$$

5. Descrição dos Experimentos

O principal objetivo dos experimentos de simulação é observar: i) a capacidade mínima necessária para atendimento de todas as solicitações dentro do limite; ii) a ociosidade do sistema em cada cenário; e iii) o resultado operacional com diferentes limites. Em seguida apresentaremos como o modelo de simulação foi implementado e como os cenários de simulação foram instanciados.

5.1. Implementação do Modelo de Simulação

Para ser resolvido por simulação, o modelo proposto foi implementado usando a ferramenta Möbius [5]. Esta plataforma permite a realização de simulação de eventos discretos e resolução numérica ou analítica de modelos de sistemas que podem ser descritos em uma variedade de formalismos diferentes.

Um dos formalismos suportados são os modelos compostos *Replicate/Join* que permitem a definição de modelos por composição na forma de uma árvore, na qual cada folha da árvore pode ser um modelo atômico, descrito em um dos formalismos suportados, ou outro modelo composto. Cada nó da árvore que não é uma folha é classificado ou como um nó *Join* ou como um nó *Replicate*. Um nó do tipo *Join* é usado para compor dois ou mais submodelos através do compartilhamento de estado e um nó do tipo *Replicate* é usado para construir um modelo consistindo de um determinado número de cópias idênticas do seu submodelo filho. Recursivamente, cada nó filho de um nó *Replicate* ou *Join* pode ser um *Replicate*, um *Join*, ou um modelo atômico ou composto.

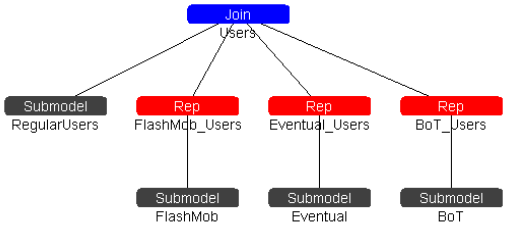


Figure 1. O Modelo Composto dos Usuários Ativos de um Provedor IaaS

Nós usamos o formalismo *Replicate/Join* para a criação do modelo composto representantivo dos usuários ativos de um provedor IaaS *ActiveUsers* (Figura 1). Este modelo contém quatro submodelos atômicos, modelados no formalismo *Stochastic Activity Network* (SAN), representando os quatro perfis de usuários descritos: Regular, Eventual, FlashMob e BoT. O uso dos nós *Replicate* permite a criação do número desejado de instâncias de cada perfil de usuário associado e também o compartilhamento de estado entre as instâncias de um mesmo tipo de submodelo. O nó *Join*, por sua vez, permite o compartilhamento de estado entre instâncias de submodelos de tipos diferentes. Desta forma, a carga de trabalho sintética foi construída através da atividade autônoma e combinada de uma instância do submodelo Regular, cuja demanda em cada fatia é multiplicada por $|U_r|$, e $|U_e|$ instâncias dos submodelos Eventual, FlashMob e BoT, de acordo com a distribuição de atividade configurada para cada tipo de perfil.

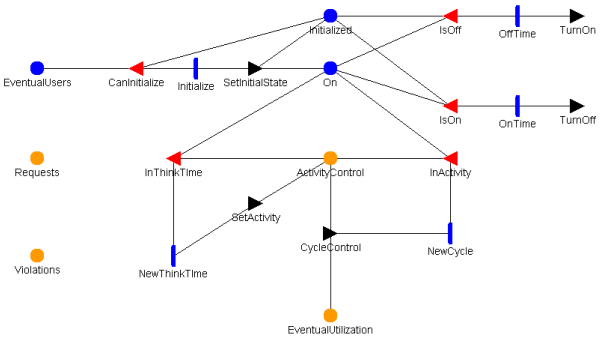


Figure 2. O modelo atômico (SAN) de um usuário do perfil Eventual

O submodelo Eventual (Figura 2) representa o com-

portamento de um usuário do perfil Eventual. Conforme descrito na seção anterior, um usuário consome recursos através de uma série de estágios. Estes estágios foram modelados em um submodelo SAN como *places* e *extended places* (círculos azuis e laranja, respectivamente). Cada *place* mantém um contador (representado como *tokens*) de como cada usuário está em cada estágio e os *input gates* (triângulos vermelhos) são usados para inspecionar estes valores e habilitar (ou não) a transição do sistema através da execução de atividades temporizadas (barras verticais). Cada atividade temporizada tem uma duração que impacta na dinâmica do sistema modelado e também uma distribuição (e parâmetros associados) que regula o seu comportamento. Os *output gates* (triângulos pretos) são executados após o tempo de duração de uma atividade ter sido completada e permite a alteração do estado do sistema através da alteração dos *tokens* nos *places*. Os arcos (linhas pretas) controlam o fluxo de transição de estágios.

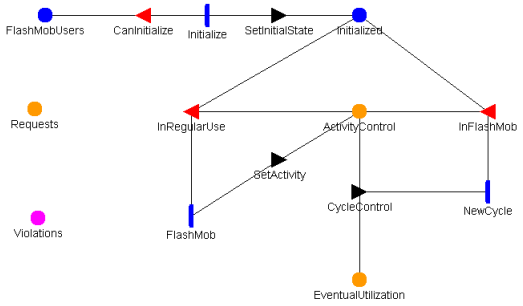


Figure 3. O modelo atômico (SAN) de um usuário do perfil Regular

Cada usuário de perfil Eventual é inicializado randomicamente em um dos estágios possíveis (*OnSession* ou *OffSession*) que é controlado pelo lugar *On*. A partir deste momento, as atividades *OffTime* e *OnTime* começam a regular a alternância do usuário em sessões de uso e períodos de inatividade, controlados pelas variáveis aleatórias $\tilde{o}$ e $\tilde{i}$, respectivamente. Uma nova atividade para o usuário em sessão é atribuída (conforme descrito no perfil Eventual e usando as variáveis aleatórias $\tilde{d}$ e $\tilde{s}$) através da porta de saída *SetActivity* após um período de espera (*think time*) ser cumprido. A duração de cada espera é gerida pela atividade temporizada *NewThinkTime* (variável aleatória $\tilde{t}$). O lugar *ActivityControl*, por sua vez, controla a duração de cada atividade individual, fatia a fatia, através da atividade temporizada *NewCycle*.

Os outros submodelos Regular (Figura 4), FlashMob (Figura 3) e BoT ou Intenso (Figura 5) possuem modelagem similar[3].

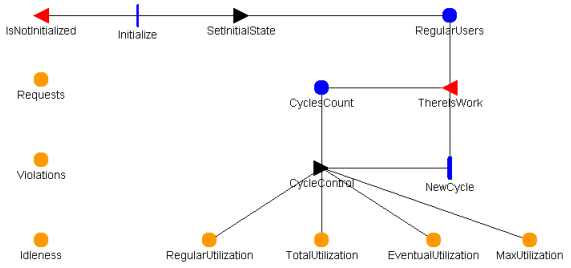


Figure 4. O modelo atômico (SAN) de um usuário do perfil FlashMob

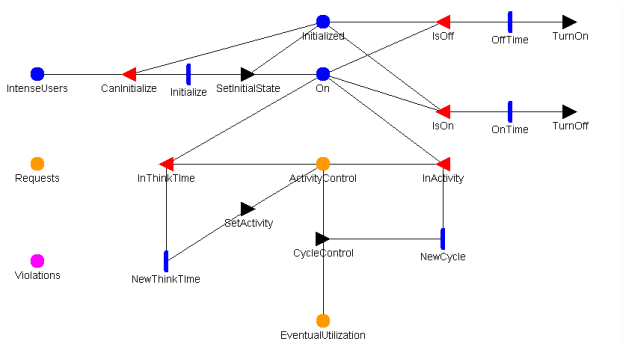


Figure 5. O modelo atômico (SAN) de um usuário do perfil BoT (Intenso)

Através da configuração dos parâmetros do sistema, modelados como variáveis globais, foi usado o simulador do Möbius para executar o modelo proposto e obter as medidas de interesse desejadas ao longo do tempo, incluindo a quantidade de recursos usados, o número de solicitações e o número de violações cometidas.

5.2. Parâmetros do Sistema

Para atribuição dos parâmetros do sistema foram usadas duas estratégias: projeto de experimento (DoE, do inglês *Design of Experiment*) e varredura de parâmetros. A parte dos parâmetros relacionada com a geração da carga sintética e associada com as distribuições descritas na Seção 4 foi tratada através de um DoE do tipo 2^k fatorial [Jain 1991]. Através do DoE foi possível analisar o efeito dos parâmetros das variáveis aleatórias $\tilde{o}$ (duração da sessão), $\tilde{i}$ (intervalo entre sessões), $\tilde{s}$ (duração da atividade), $\tilde{t}$ (*think time*) e também do valor de L sobre uma das variáveis de resposta do sistema: a utilização máxima do sistema em um dado intervalo $(max(\alpha(t)) \ \forall \ t, 1 \leq$

[3]O modelo Möbius completo usado nos experimentos de simulação usados nesta análise pode ser encontrado no sítio http://www.lsd.ufcg.edu.br/~rostand/IaaSModel.zip

Fator	Baixo	Alto	Efeito Estimado	Soma dos Quadrados	% Cont.
A: Limite superior u_o (em fatias) para $\tilde{o}$	36	108	$0,0624$	$0,0312$	$6,53$
B: Limite inferior k_i (em fatias) para $\tilde{i}$	120	360	$-0,0315$	$0,0080$	$1,66$
C: Média λ_d (em fatias) para $\tilde{d}$	$0,0625$	$0,1875$	$0,0726$	$0,0421$	$8,83$
D: Média λ_t (em fatias) para $\tilde{t}$	$0,125$	$0,375$	$-0,0220$	$0,0039$	$0,81$
E: L (em quantidade de recursos)	20	100	$0,2143$	$0,3673$	$77,05$

Table 1. Fatores, níveis e efeitos para DoE 2^k fatorial ($k = 5$)

$t \leq \Delta T$). Os níveis atribuídos para o experimento fatorial são apresentados na Tabela 1.

Foram conduzidas várias repetições dos 32 experimentos para obter médias com 95% de intervalo de confiança com erro máximo de 5% para mais ou para menos. A contribuição de cada fator está exibida na Tabela 1, com destaque para o fator predominante L que teve participação de $77,05\%$. A única interação relevante (acima de $0,5\%$) foi BC que apresentou uma contribuição de $2,53\%$. Como resultado da análise dos efeitos através de ANOVA [9], o F-Value de $158,6521$ implica que o modelo é significativo. O R^2 ajustado indica que o modelo explica $96,83\%$ da variação observada e o R^2 de predição está dentro de $0,20$ do R^2 ajustado, representando uma boa capacidade de predição do modelo. Maiores detalhes sobre este estudo, incluindo os gráficos de diagnóstico, cubo e interação, podem ser encontrados no mesmo endereço citado na Seção 5.1. De acordo com os resultados, a variação dos quatro primeiros fatores não afetou o comportamento da variável de resposta que ocorreu em função da variação de L.

Para a realização das simulações, os valores destes quatro parâmetros foram ajustados para os valores medianos entre os níveis "Alto" e "Baixo" usados no DoE e para os parâmetros *Percentual de Atividade Eventual*, *Percentual de Usuários com Perfil BoT*, *Número de Usuários Ativos* e L foi aplicada uma varredura de parâmetros. A Tabela 2 mostra como o sistema foi configurado para os experimentos.

6. RESULTADOS E ANÁLISE

No primeiro experimento, o objetivo foi observar como a lucratividade do provedor era impactada com o aumento do limite imposto pelo provedor (L). Nesse experimento nós consideramos uma situação em que a disponibilidade do provedor deve ser mantida em 100%. Para tal, a estratégia de alocação (A) simulada é tal, que para qualquer fatia t, sempre é possível alocar recursos para um usuário u que tenha uma demanda positiva ($d(u,t) > 0$) e, portanto, $a(u,t) = d(u,t) \forall u \in U \wedge 1 \leq t \leq \Delta t$. Dessa forma, considerando a Equação 2, como as penalidades serão nulas e a receita líquida da execução de uma mesma carga de trabalho é constante, o lucro do provedor é afetado apenas pela capacidade que precisa ser mantida para atender o nível de disponibilidade desejado. Para garantir condições similares de carga do sistema, o número de usuários ativos foi mantido constante para este experimento em 5.000 usuários. Entretanto, foi feita uma varredura dos parâmetros *Percentual de Atividade Eventual* e *Percentual de Usuários com Perfil BoT* para simular diferentes cenários de atividade regular e eventual e diferentes participações dos usuários com perfil *BoT*. Esta classe de usuários é especialmente interessante para esta análise porque possuem cargas de trabalho de alto volume e sensíveis ao tempo e tendem a consumir todo o limite máximo de alocação de recursos permitido.

Para cobrir todas as combinações dos parâmetros de entrada foram realizadas 288 simulações - repetidas até que as médias obtidas tivessem 95% de intervalo de confiança e erro máximo de 5% para mais ou para menos. A resposta de interesse observada foi a utilização máxima ($max(\alpha(t))$) observada em todas as fatias de tempo de cada configuração do sistema simulado, já que esta define a capacidade mínima necessária para garantir 100% de disponibilidade. Parte dos resultados obtidos estão exibidos graficamente na Figura 6.

Como pode ser observado, mesmo assumindo uma população de tamanho constante, a capacidade mínima necessária aumenta à medida que o limite é incrementado. Esta demanda por maior capacidade instalada associada com o aumento da alocação máxima de recursos por usuário já está presente mesmo em cenários onde a atividade regular é dominante (25% de usuários eventuais e 10% com perfil BoT). Onde a atividade eventual é mais preponderante com 95%, dos quais 25% dos usuários possuem perfil BoT, o aumento necessário da capacidade instalada chega a representar quase o dobro.

O segundo experimento teve como finalidade observar como o incremento na capacidade instalada afeta o nível de utilização do sistema. Usando os valores de $max(\alpha(t))$ obtidos no experimento anterior como a capacidade instalada do provedor (K), os mesmos cenários foram novamente executados (288 simulações, médias com 95% IC e erro máximo de 5%) para obter a ociosidade apresentada pelo sistema. A ociosidade é representada pela razão entre a utilização total observada em

Parâmetro	Valor
Duração da Sessão ($\tilde{o}$)	$l_o = 1$ hora e $u_o = 72$ horas
Intervalo entre Sessões ($\tilde{i}$)	$k_i = 240$ horas e $s_i = 2$
Duração da Atividade ($\tilde{d}$)	$\lambda_d = 0.125$ (8 horas)
Espera entre Atividades ou *think time* ($\tilde{t}$)	$\lambda_t = 0.25$ (4 horas)
ΔT	8.760 horas (1 ano)
Número de Usuários Ativos ($\|U_a\|$)	{ 625; 1.250; 2.500; 5.000 }
Percentual de Atividade Eventual	{ 25%; 35%; 45%; 55%; 65%; 75%; 85%; 95% }
Percentual de Usuários com Perfil *FlashMob*	1%
Percentual de Usuários com Perfil *BoT*	{ 10%; 15%; 20%; 25% }
Limite (L)	{ 20; 30; 40; 50; 60; 70; 80; 90; 100 }
Ticket Médio (τ)	2 recursos

Table 2. Parâmetros Usados na Simulação

ΔT e a capacidade total disponível no mesmo período: $(\frac{\sum_{t=1}^{\Delta T} \alpha(t)}{K \times \Delta T})$. Os resultados obtidos indicaram uma variação da ociosidade proporcional à variação do limite, com amplitude variando consistentemente de 20% a 65% de capacidade ociosa em todas as combinações de atividade eventual e perfil BoT simuladas. A Figura 7 ilustra a ociosidade encontrada em três cenários de atividade eventual, todos com 10% de usuários com perfil BoT.

Este aumento proporcional da ociosidade com o aumento do limite ganha um impacto significativo nos custos do provedor. Considerando o preço cobrado pelo principal provedor de IaaS e usando a fórmula para cálculo do lucro (Equação 2), foi realizada uma terceira análise. Foram aplicadas diferentes margens de lucro aos valores obtidos nos experimentos anteriores para identificar a partir de que ponto a operação do provedor se torna equilibrada, ou seja, sem lucro nem prejuízo, em cada configuração. Foi observado que à medida que o limite é incrementado o ponto de equilíbrio da operação só é alcançado quando a margem de lucro também é aumentada, com reflexos diretos na competitividade do provedor. Na Figura 8, que traz o estudo referente a variação de 25 a 75% de atividade eventual e com 10% de usuários com perfil BoT, pode ser visto que a margem de lucro necessária para empatar receitas e despesas chega a 300% no maior valor considerado para o limite.

Os mesmos experimentos foram repetidos para quantidades diferentes de usuários ativos para observar se havia variação do comportamento em escalas diferentes. Como pode ser observado na Figura 9, as curvas são bastante similares para os valores entre 625 e 5.000 usuários quando são mantidas as mesmas condições de limite e perfis de atividade[4].

[4] 5.000 foi o máximo de instâncias do modelo que a ferramenta suportou simular.

7. CONCLUSÃO

A partir da modelagem de um provedor de IaaS e da geração de uma carga de trabalho sintética, nós simulamos diversos cenários de utilização. A análise dos resultados aponta que não só a atribuição de um limite para alocação de recursos é necessário como o valor atribuído tem impacto relevante nos investimentos necessários em infraestrutura para garantir um nível adequado de disponibilidade do provedor. Também foi observado que usuários com utilização eventual e intensa pressionam a capacidade mínima necessária e aumentam a ociosidade do sistema, aumentando os custos operacionais do provedor. Mantido o mesmo perfil da população e o mesmo valor de limite, a dinâmica do sistema independe da quantidade de usuários não sendo, portanto, um contexto onde a economia de escala possa significar uma melhoria direta.

O uso de modelo mais próximo da realidade, mesmo com o impacto na sua tratabilidade, nos pareceu a opção mais adequada para este estudo. Para mitigar a complexidade do modelo e a inexistência de dados de campo, usamos técnicas como o design de experimento, para identificar as variáveis independentes mais importantes, e a varredura de parâmetros para instanciação de um amplo espectro de cenários. Obtivemos resultados consistentes em todos os cenários simulados.

Os resultados ajudam a entender a necessidade do uso de um limite e como o seu impacto na lucratividade do provedor está diretamente relacionado com o padrão de utilização da população de usuários, nos fazendo concluir que algumas categorias de usuários/aplicações que se beneficiariam de uma elasticidade mais ampla, continuarão sendo mal servidas se o modelo atual de provisionamento de recursos for mantido.

Os próximos passos da nossa pesquisa incluem a investigação de alternativas para minimizar os custos envolvidos com a disponibilidade de capacidade pelos provedores. Estes custos são um dos principais obstáculos para a oferta de elasticidade em condições mais flexíveis, mesmo que ainda limitada, mas que permitam que classes

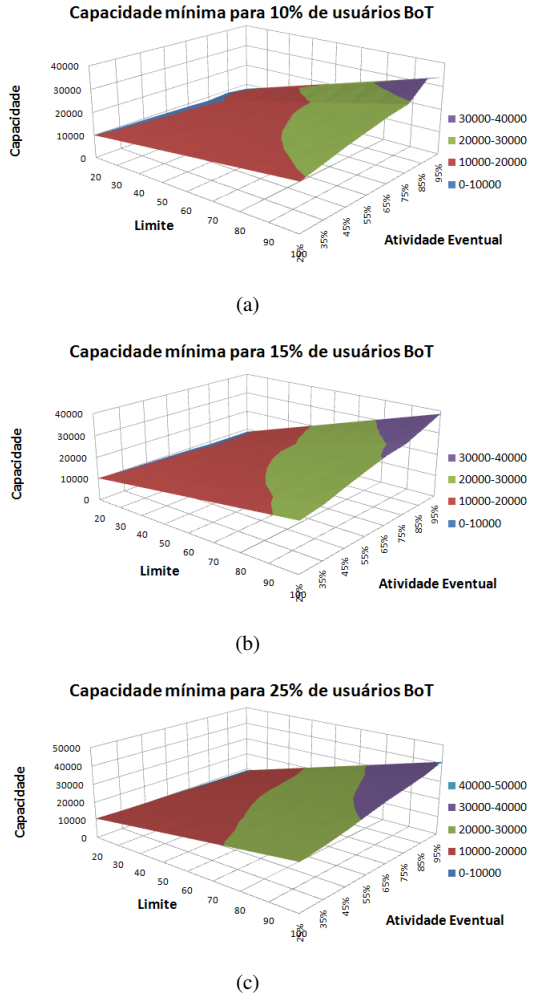


Figure 6. Capacidade mínima para atender 100% dos pedidos dos usuários em diferentes cenários de limite (L), Percentual de Atividade Eventual e Percentual de Usuários com Perfil BoT

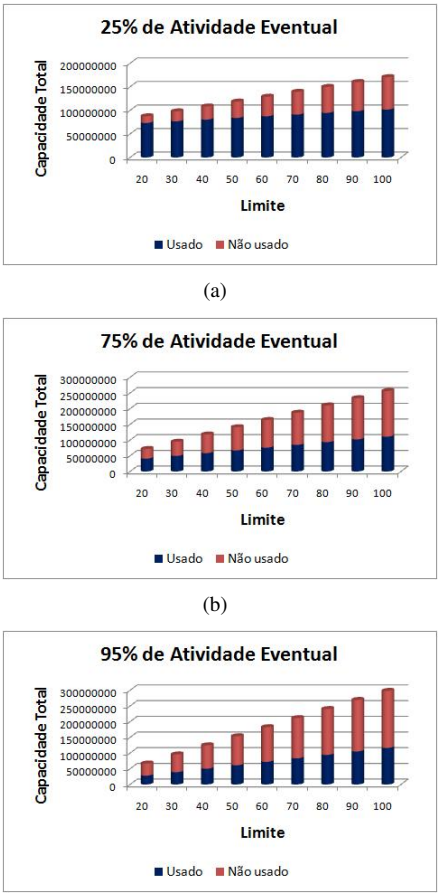


Figure 7. Ociosidade observada com a variação de limite para diferentes cenários de Atividade Eventual e com 10% de usuários com perfil BoT

de aplicações de uso intenso possam se beneficiar das vantagens do modelo de computação na nuvem. A descoberta, federação e revenda de recursos já amortizados em outros contextos podem representar um caminho promissor, pois se baseiam na existência de capacidade ociosa em contextos onde os custos de disponibilidade já foram absorvidos por outros negócios ou finalidades. Dentre estes contextos com capacidade de processamento ociosa e amortizada, podemos citar data centers privados, grades P2P e recursos computacionais não convencionais, como dispositivos móveis, telefones celulares, receptores de TV Digital etc.

O maior desafio é dotar os provedores comerciais de computação na nuvem com mecanismos, tecnológicos e comerciais, que permitam a montagem dinâmica de infraestruturas computacionais sobre estes recursos de forma a atender aplicações com diferentes requisitos de QoS e com diferentes expectativas de custos.

Agradecimentos

Os autores gostariam de agradecer a Jacques Sauvé pelas várias sugestões sobre o modelo de simulação e o gerador de cargas de trabalho sintéticas e a equipe do Möbius pelo prestativo suporte no uso da ferramenta. Francisco Brasileiro gostaria de agradecer o apoio financeiro do CNPq.

References

[1] Amazon. Amazon Web Services, 2010.

[2] Arun Anandasivam, Stefan Buschek, and Rajkumar Buyya. A Heuristic Approach for Capacity Control in Clouds. In *2009 IEEE Conference on Commerce and Enterprise Computing*, July 2009.

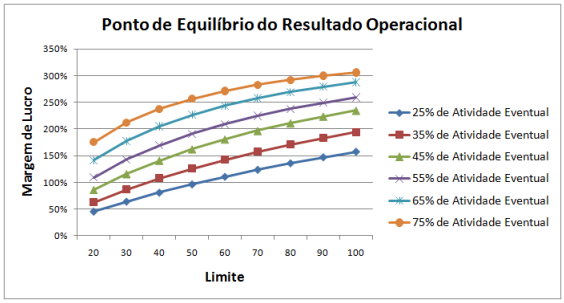


Figure 8. Equilíbrio do resultado operacional com diferentes valores de limite

[3] Luiz André Barroso and Urs Hölzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, December 2007.

[4] W. Cirne, D. Paranhos, L. Costa, E. Santos-Neto, and F. et al Brasileiro. *Running Bag-of-Tasks applications on computational grids: the MyGrid approach*. IEEE, 2003.

[5] D.D. Deavours, G. Clark, T. Courtney, and D. et al Daly. The Mobius framework and its implementation. *IEEE Transactions on Software Engineering*, 28(10):956–969, October 2002.

[6] Dror G Feitelson. *Workload Modeling for Computer Systems Performance Evaluation*. Hebrew University of Jerusalem (Online Book), 0.30 edition, 2009.

[7] Albert Greenberg, James Hamilton, David a. Maltz, and Parveen Patel. The cost of a cloud. *ACM SIGCOMM Computer Communication Review*, 39(1):68, December 2008.

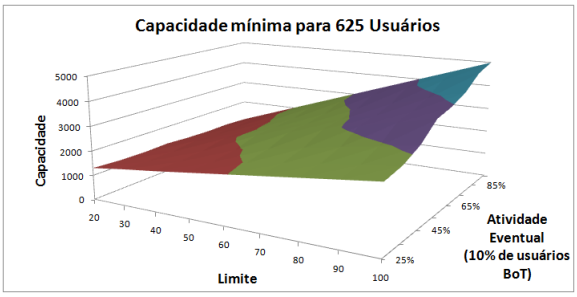
[8] A J G Hey and A E Trefethen. *The Data Deluge: An e-Science Perspective*, chapter 36, pages 809–824. Wiley and Sons, 2003.

[9] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.

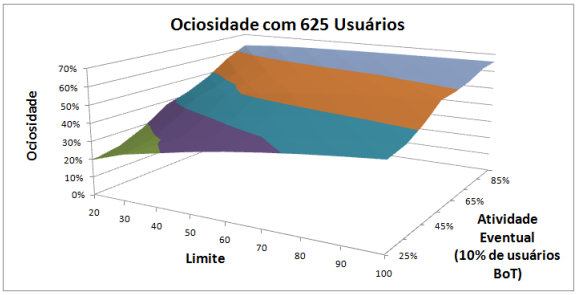
[10] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. *Flash crowds and denial of service attacks*. ACM Press, New York, New York, USA, 2002.

[11] Xinhui Li, Ying Li, Tiancheng Liu, Jie Qiu, and Fengchun Wang. The Method and Tool of Cost Analysis for Cloud Computing. *2009 IEEE International Conference on Cloud Computing*, September 2009.

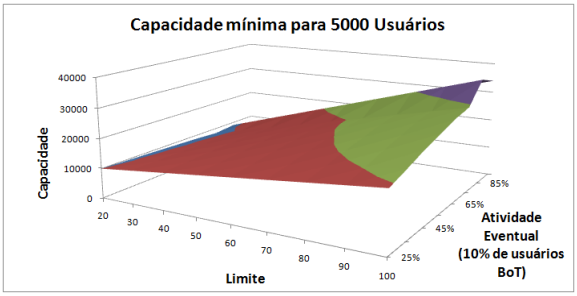
[12] Daniel A Menascé and Paul Ngo. Understanding Cloud Computing: Experimentation and Capacity Planning. In *2009 Computer Measurement Group Conference*, 2009.



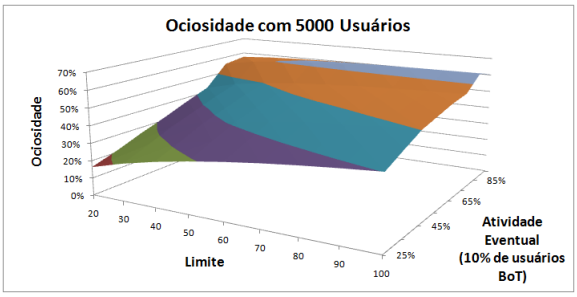
(a)



(b)



(c)



(d)

Figure 9. Resultados de Capacidade Mínima e Ociosidade para 625 e 5.000 usuários

[13] L. Mieritz and B Kirwin. Defining Gartner Total Cost of Ownership, 2005.

[14] Martin Sevior, Tom Fifield, and Nobuhiko Katayama. Belle monte-carlo production on the Amazon EC2 cloud. *Journal of Physics: Conference Series*, 219(1):012003, April 2010.

[15] Katarina Stanoevska-Slabeva and Thomas Wozniak. Cloud Basics: An Introduction to Cloud Computing. *Grid and Cloud Computing*, pages 47–61, 2010.

[16] David Talby. *User Modeling of Parallel Workloads by User Modeling of Parallel Workloads*. Hebrew University of Jerusalem (PhD Thesis), 2006.

RouteFlow: Roteamento Commodity Sobre Redes Programáveis

Marcelo Nascimento[1], **Christian Rothenberg**[1],
Rodrigo Denicol[1], **Marcos Salvador**[1], **Maurício Magalhães**[2]

[1]CPqD - Centro de Pesquisa e Desenvolvimento em Telecomunicações
marcelon@cpqd.com.br

[2]Unicamp - Universidade Estadual de Campinas
mauricio@dca.fee.unicamp.br

Abstract

Today's networking gear follows the model of computer mainframes, where closed source software runs on proprietary hardware. This approach results in expensive solutions and prevents equipment owners to put new ideas into practice. In contrast, recent alternatives of highly flexible software-based routers promise low cost and programmability at the expense of low performance. Motivated by the availability of an open API to control packet forwarding engines (i.e., OpenFlow), we propose RouteFlow, a commodity IP routing architecture that combines the line-rate performance of commercial hardware with the flexibility of open source routing stacks (remotely) running on general-purpose computers. The challenge is to ensure reliability, scalability and performance to a network running a remote and centralized control plane architecture that allows a flexible mapping between the control and forwarding elements. The outcome is a novel point in the design space of cost-effective IP routing solutions with far-reaching implications. The initial experimental evaluation of our prototype implementation validates the feasibility of the design.

Keywords: OpenFlow, RouteFlow, routing, architecture

Resumo

Os roteadores atuais implementam uma arquitetura composta de uma camada de software e um hardware proprietários. Este modelo resulta em soluções de alto custo e inviabiliza a experimentação de novas ideias. Em contrapartida, existem alternativas de alta flexibilidade baseadas em software e, consequentemente, de baixo custo. Entretanto, essas soluções apresentam baixo desempenho. Motivado pela disponibilidade de uma API aberta para programação do plano de encaminhamento (i.e., OpenFlow), este trabalho apresenta o projeto RouteFlow. Trata-se de uma arquitetura de roteamento IP que procura combinar o alto desempenho de hardwares de prateleira (commodities) com a flexibilidade de uma pilha de roteamento executada remotamente em computadores de uso geral. O grande desafio é garantir confiabilidade, escalabilidade e desempenho à rede, a partir de um controle remoto e centralizado, cuja arquitetura permita maior flexibilidade no mapeamento entre os elementos de controle e encaminhamento. O resultado corresponde a uma nova solução de roteamento IP com perspectivas promissoras do ponto de vista do custo e da flexibilidade. A avaliação do protótipo apresentada no artigo comprova a viabilidade da arquitetura proposta.

Palavras-chave: OpenFlow, RouteFlow, roteamento, arquitetura

1. INTRODUÇÃO

A infraestrutura das redes de pacotes é normalmente composta por equipamentos proprietários, fechados e de alto custo, cujas arquiteturas básicas são concebidas a partir da combinação de um processador dedicado ao processamento de pacotes e responsável por garantir o alto desempenho. A arquitetura é implementada por uma camada de software de controle constituída por uma com-

plexa pilha de protocolos com o objetivo de atender adequadamente os requisitos das redes de computadores [10]. No entanto, torna-se evidente a necessidade de especialização da lógica de controle de acordo com cada tipo e objetivo de rede, em especial, no caso das redes corporativas. No âmbito da pesquisa, essa exigência cresce com a necessidade de experimentações de novos protocolos [1]. A ausência de flexibilidade e o alto custo da infraestrutura vigente são barreiras que dificultam o avanço das redes e a inovação [18].

Com o objetivo de suprir tais necessidades, existem alternativas como *software routers* (ex: RouteBricks [6], [22]), que são soluções que utilizam "hardware de prateleira" com grande capacidade de processamento (ex.: x86) e onde todo processamento de pacotes é realizado no nível de software. Essa opção torna a proposta bastante flexível mas acarreta, por outro lado, um grande prejuízo de desempenho [2]. Alternativas baseadas em FPGAs apresentam bom desempenho, entretanto, são de alto custo e o tempo de desenvolvimento é bastante elevado. Este último fator é ainda mais crítico nas alternativas baseadas em *network processors* (NP) [20]. Propostas recentes [11] têm explorado também o uso de *graphics processing units* (GPUs) para acelerar o processamento de pacotes nos *software routers*.

A meta desse trabalho é abordar, simultaneamente, as questões de desempenho, flexibilidade e custo, conforme apresentado em [15]. A estratégia baseia-se no uso da recente tecnologia de *switches* programáveis [13], o que possibilita mover o plano de controle, antes embarcado no equipamento, para um dispositivo externo [9]. A finalidade, no caso, é permitir a programação remota do plano de encaminhamento. O resultado consiste em uma solução flexível de alto desempenho e comercialmente competitiva, denominada **RouteFlow**, a partir da combinação de recursos disponíveis, tais como: (a) *switches* programáveis de baixo custo e software embarcado reduzido; (b) pilha de protocolos de roteamento *open source*; e (c) servidor de prateleira de alto poder de processamento e também de baixo custo.

O RouteFlow armazena a lógica de controle dos *switches* programáveis utilizados na infraestrutura de rede através de uma rede virtual composta por máquinas virtuais (MVs), cada uma executando um código (*engine*) de roteamento de domínio público (*open source*). Essas MVs podem ser interconectadas de maneira a formar uma topologia lógica espelhando a topologia de uma rede física correspondente. O ambiente virtual é armazenado em um servidor externo, ou um conjunto deles, que se comunica com os equipamentos do plano de dados através de um elemento chamado de controlador, cuja responsabilidade é transportar para o plano de encaminhamento as decisões tomadas pelo plano de controle.

A arquitetura proposta procura atender os seguintes requisitos: (a) integridade e sincronização das informações de configuração trocadas entre o ambiente físico e virtual; (b) mecanismo eficiente para detecção de atualizações no plano de controle com o objetivo de minimizar o atraso da respectiva implantação no plano de dados; (c) isolamento máximo entre as instâncias de roteamento (MVs) de modo que os recursos sejam idealmente compartilhados; (d) gerenciamento dinâmico das conexões entre as MVs e, por último, (e) separação do tráfego que deve ser mantido no plano de dados daquele que necessita ser encaminhado para a topologia virtual.

Deve ser destacado que a arquitetura do RouteFlow promove a efetiva separação entre os planos de controle e de encaminhamento. Essa separação traz inúmeras vantagens com relação ao isolamento de falhas e, também, às questões relacionadas à segurança. Outros destaques da arquitetura proposta são a de flexibilidade de configuração e implementação das funções de controle (roteamento) nas máquinas virtuais e o desempenho, uma vez que o tráfego de dados é processado em hardware na taxa de transferência das interfaces (*line rate*). Podemos ressaltar ainda a utilização de equipamentos comerciais de prateleira, o que resulta em uma implementação de baixo custo. Esta abordagem privilegia, inicialmente, os requisitos das redes corporativas/campus. Extensões como, por exemplo, a hierarquização do plano de controle, poderá permitir, futuramente, a aplicação de modelos similares no caso das redes de núcleo *backbones*.

O restante deste artigo está organizado da seguinte forma: a Seção 2 discute as tecnologias e arquiteturas relacionadas à proposta apresentada neste artigo; a Seção 3 trata dos trabalhos relacionados; a Seção 4 apresenta a arquitetura e os componentes do RouteFlow; a Seção 5 descreve a elaboração e implementação de um protótipo aderente à arquitetura RouteFlow; a Seção 6 apresenta o ambiente de testes e a avaliação do protótipo implementado; a Seção 7 apresenta uma discussão dos resultados obtidos; a Seção 8 trata dos trabalhos futuros e, por último, a Seção 9 apresenta as conclusões.

2. TECNOLOGIAS E ARQUITETURAS RELACIONADAS

Uma análise da arquitetura de roteamento atual (lado esquerdo da Figura 1) permite observar que se trata de um modelo formado basicamente por duas camadas bem distintas: o software de controle e o hardware dedicado ao encaminhamento de pacotes. O primeiro, encarregado de tomar as decisões de roteamento, transfere essas decisões para o plano de encaminhamento através de uma API proprietária. A única interação da gerência com o dispositivo é através de interfaces de configuração (e.g., Web, SNMP, CLI), limitando o uso dos dispositivos às funcionalidades

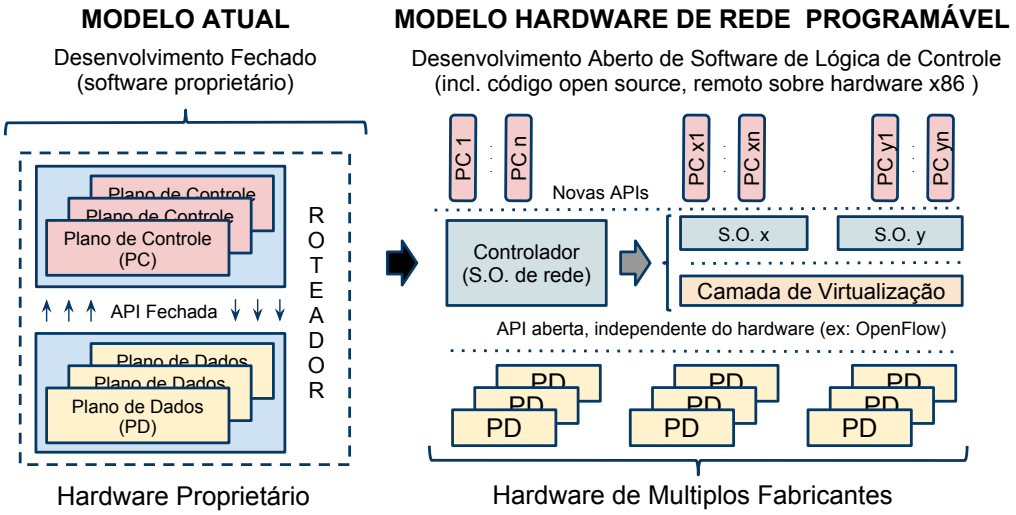


Figura 1. Arquiteturas de roteamento.

programadas pelo fabricante.

A conclusão de que a arquitetura atual é composta por duas camadas auto-contidas sugere que elas não precisariam encontrar-se acopladas em um mesmo equipamento. Para isso, basta que exista uma forma padrão de programar o dispositivo de rede remotamente, permitindo a camada de controle ser movida para um servidor dedicado e com alta capacidade de processamento. Deste modo, mantém-se o alto desempenho no encaminhamento de pacotes aliado à flexibilidade de inserir, remover e especializar aplicações utilizando uma API aberta para programação do roteador (lado direito da Figura 1). Com este propósito, surgiu o consórcio OpenFlow [17].

O OpenFlow define um protocolo padrão para determinar as ações de encaminhamento de pacotes em dispositivos de rede tais como *switches*, roteadores e pontos de acesso sem fio. A principal abstração utilizada na especificação do OpenFlow é o conceito de fluxo. Um fluxo é constituído pela combinação de campos do cabeçalho do pacote a ser processado pelo dispositivo. As tuplas podem ser formadas por campos das camadas de enlace, de rede ou de transporte, segundo o modelo TCP/IP. Deve ser enfatizado que a abstração de tabela de fluxos ainda está sujeita a refinamentos com o objetivo de expor melhor os recursos do hardware.

Pragmaticamente, a especificação OpenFlow procura reutilizar as funcionalidades dos hardwares existentes (ACL - *Access Control List*) através da definição de um conjunto simples de regras e das ações associadas (ex., encaminhar, descartar, enviar para o controlador, reescrever campos do cabeçalho do pacote, etc.). Deste modo, o OpenFlow pode ser suportado por dispositivos existentes através de uma atualização do *firmware*. Deve ser destacado que o OpenFlow tem atraído a atenção da indústria o que tem se traduzido na disponibilidade de equipamentos com suporte ao OpenFlow e protótipos de grandes empresas como Cisco, HP, NEC, Extreme e Juniper.

3. Trabalhos Relacionados

O RouteFlow alinha-se com a tendência de logicamente centralizar o controle da rede, unificando a informação do estado da rede e desacoplando-a (encaminhamento e configuração) dos elementos de hardware [5].

Com objetivos similares ao RouteFlow, o FIBIUM [19] é uma proposta que combina um roteamento em software, rodando em um computador *commodity*, com um hardware OpenFlow responsável pela maior parte do encaminhamento de pacotes. O foco do trabalho é a otimização do número de entradas instaladas em hardware baseada na observação da popularidade das mesmas. A nossa proposta diferencia-se, principalmente, pelo plano de controle virtualizado em servidores remotos cuja arquitetura permite o mapeamento flexível da infraestrutura de rede programável.

Outro trabalho que também visa paradigmas avançados e flexíveis sobre roteadores baseados em software é o DROP [3]. O DROP fundamenta-se nas diretrizes do padrão IETF ForCES e tem como foco principal a criação de nós lógicos de rede através da separação entre os elementos de controle e de encaminhamento.

4. Arquitetura RouteFlow

O RouteFlow é uma proposta de roteamento remoto centralizado que visa o desacoplamento entre o plano de encaminhamento e o plano de controle, tornando as re-

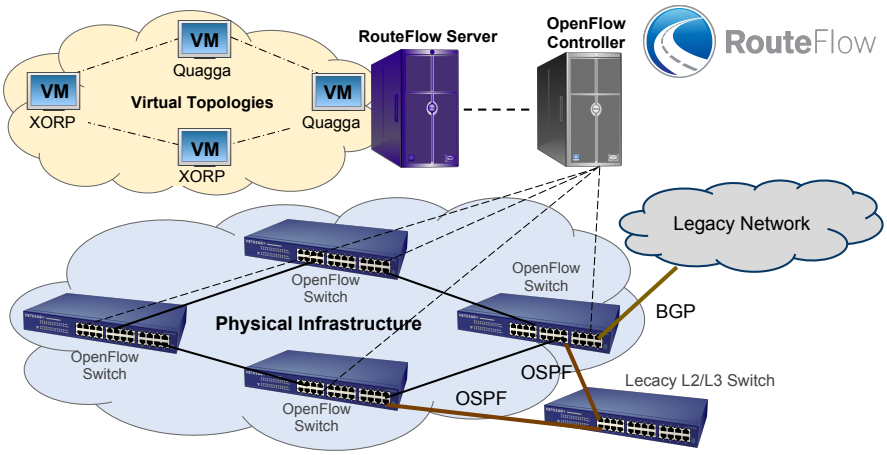


Figura 2. Visão geral do RouteFlow.

des IP mais flexíveis pela facilidade da adição, remoção e especialização de protocolos e algoritmos. No caso, o RouteFlow move a lógica de controle dos equipamentos de rede, até então embarcada, para um controlador de rede remoto cuja responsabilidade é tomar decisões de encaminhamento e programar os elementos do plano de encaminhamento para aplicar essas decisões. Portanto, a base da proposta é a existência de elementos de hardware do plano de encaminhamento que ofereçam interfaces de programação de aplicação (APIs), como por exemplo a proposta na especificação OpenFlow, que permitam tal programação.

Na arquitetura RouteFlow, o plano de controle é formado por protocolos de roteamento IP de código aberto. O plano de roteamento é constituído de máquinas virtuais (MVs) conectadas de forma a representar, através de uma topologia lógica de roteamento, a topologia física da rede controlada conforme mostrado na Figura 2.

Na topologia lógica virtual, cada MV roda uma *engine* de roteamento padrão, ou seja, o mesmo software de controle (protocolos) que estaria embarcado em um roteador. As interfaces das MVs representam as portas do roteador e se conectam a um software *switch* através do qual é possível configurar fluxos e promover as devidas conexões entre as MVs. Ainda por meio dos fluxos pode-se configurar quais pacotes devem permanecer na rede virtual e quais devem ir para o plano de encaminhamento.

Manter os pacotes de protocolos confinados na topologia virtual torna-se interessante para efetuar a separação entre o plano de controle e plano de dados. As decisões tomadas pelas *engines* de rotamento dentro das MVs são enviadas para o controlador, que as instala nos respectivos elementos físicos da rede. Além da instalação de rotas, o controlador faz o gerenciamento das máquinas virtuais bem como a amarração entre elas, isto é, a configuração dos fluxos do *switch* no qual as MVs estão conectadas também é de responsabilidade do controlador.

Apesar de o controle estar fisicamente centralizado, ele continua distribuído logicamente. Desta forma, não há necessidade de qualquer alteração dos protocolos de roteamento existentes. Além disso, a solução pode tornar-se mais escalável no futuro com o uso de vários servidores de alto desempenho.

4.1. Modos de Operação

A arquitetura RouteFlow promove a efetiva separação entre os planos de controle e encaminhamento. Esta característica combinada com os recursos da tecnologia de virtualização utilizada no plano de controle RouteFlow agregam interessantes funcionalidades ao sistema. Dentre elas pode-se destacar a flexibilidade no mapeamento entre os equipamentos físicos e as instâncias de controle virtuais (MVs), que resulta em três modos de operação com características bem definidas, como ilustrado na Figura 3.

A **separação lógica** reflete o mapeamento (1:1), que representa o espelhamento entre os planos de controle e dados. O modo de **multiplexação** (1:n) permite que múltiplas instâncias de roteamento operem sobre o mesmo equipamento físico, promovendo melhor aproveitamento de recursos (virtualização da rede). Por fim, no modo de **agregação** (m:1) é possível simplificar a engenharia de protocolos de rede através do agrupamento de *switches*, resultando em um único elemento lógico.

4.2. Componentes

Uma visão global dos componentes do RouteFlow pode ser observada na Figura 4. A seguir apresenta-se a descrição de cada um dos componentes.

RouteFlow-Slave: cada MV do plano de controle executa um processo (*daemon*) responsável pelas seguintes funções: i) registro da MV no RF-Server como recurso

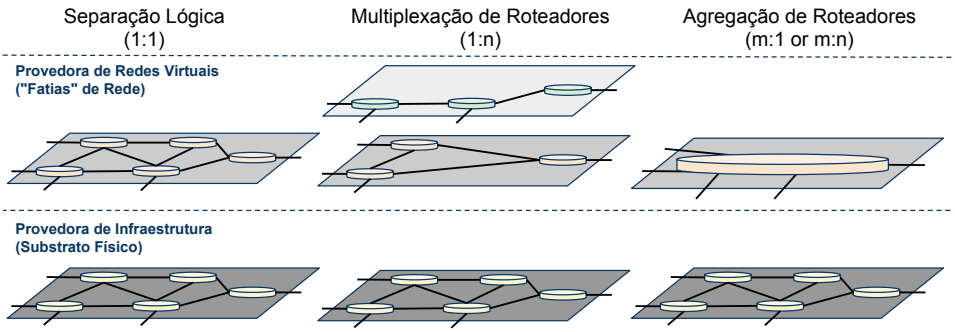


Figura 3. Diferentes variantes de virtualização de redes.

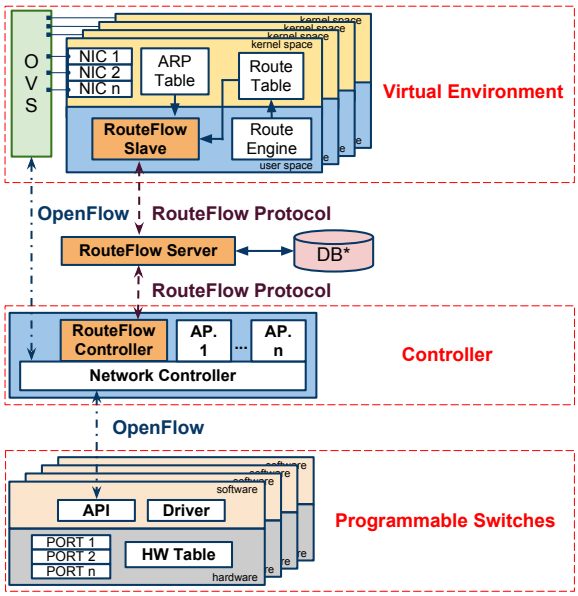


Figura 4. Componentes da solução RouteFlow.

da topologia virtual; ii) gerenciamento das interfaces de rede do sistema (portas do roteador); iii) detecção das atualizações das tabelas ARP e de roteamento do sistema e, iv) conversão de rotas em fluxos a serem instalados no plano de dados por meio da API do OpenFlow.

Pacotes de protocolos e outros, cujos destinos sejam o próprio roteador são encaminhados e recebidos pela MV para processamento (por exemplo, ARP, ICMP, Telnet, SSH, OSPF, RIP, etc.). Esses pacotes chegam às MVs pelas interfaces do sistema para que sejam devidamente tratados. Pacotes como ARP, ICMP são tratados pela pilha TCP/IP do Linux, enquanto os pacotes dos protocolos de controle são utilizados pela *engine* de roteamento para cálculo de rotas. O caminho inverso ocorre do mesmo modo, ou seja, os pacotes injetados pelo Linux, ou pela *engine* de roteamento, nas interfaces são recebidos pelo *software switch* responsável por encaminhar os pacotes para o controlador ou, dependendo da configuração, para outras MVs.

Concomitantemente ao processamento de pacotes da MV, um mecanismo executa a tarefa de verificar as tabelas ARP e ROUTE do sistema em busca de atualizações que devem ser reproduzidas no plano de dados. Quando atualizações são detectadas, as modificações correspondentes são convertidas na instalação ou remoção de entradas da tabela de fluxos atribuída à MV.

RouteFlow-Controller: aplicação que executa sobre o controlador de rede. Ele é a interface entre o componente central do sistema (RF-Server) e os elementos do plano de dados, atuando como um *proxy*. O objetivo principal é isolar o sistema do controlador de rede facilitando a adaptação do RouteFlow para operar com múltiplos controladores. As principais responsabilidades deste componente são: i) registro no controlador de rede para recebimento eventos de entrada de pacotes, conexão/desconexão de *switches* e estado dos enlaces; ii) encaminhamento dos eventos de rede ao RF-Server; iii) instalação/remoção de fluxos nos equipamentos do plano de dados e, iv) envio de pacotes aos planos de dados e de controle, este último através do *software switch*.

Os pacotes que necessitam subir do plano de dados até o plano de controle são recebidos pelo RF-Controller através do evento de entrada de pacotes (*packet-in*). Esse pacote chega com as informações do *switch* remetente e da porta de entrada. O pacote é então armazenado em um *buffer* e um evento de entrada de pacote é enviado ao RF-Server, cuja responsabilidade é resolver o mapeamento entre os elementos dos planos de dados e de controle e devolver ao RF-Controller a informação da porta do *software switch* na qual o pacote deve ser enviado para ser entregue à MV e porta correspondentes. O processo reverso ocorre da mesma forma.

RouteFlow-Server: componente central e de maior complexidade do RouteFlow, pois possui o conhecimento de todo o sistema sendo, portanto, o único elemento capaz de gerenciar a amarração entre o plano de dados e a topologia virtual. As principais responsabilidades do RF-Server são: i) conexão com a aplicação RF-Controller;

ii) processamento de eventos de rede recebidos do RF-Controller; iii) registro de recursos (MVs) disponíveis na topologia virtual, através da conexão com os componentes RF-Slave de cada MV; iv) configuração do *software switch* para construção da topologia lógica da rede; v) gerenciamento do acoplamento entre *switches programáveis* e MVs e, vi) processamento de mensagens de comandos recebidas do RF-Slave.

A amarração das MVs pode ser feita estaticamente através de um arquivo de configuração permitindo, dessa forma, a configuração de uma topologia virtual independente da topologia física. Alternativamente, a configuração pode ser dinâmica com base em um mecanismo de descoberta de topologia governado pelo controlador de rede. Neste último caso, a rede virtual será uma réplica da topologia física. Outra possibilidade consiste em agregar um conjunto de nós físicos como, por exemplo, *switch stacking/trunking*, em uma única MV no plano virtual, ou ter várias *engines* de roteamento atuando sobre um mesmo elemento físico, o que remete ao conceito de roteadores virtuais.

RouteFlow-Protocol: protocolo desenvolvido para a comunicação entre os componentes do RouteFlow, que opera com mensagens no modelo de TLV. Nele estão definidas as mensagens e os comandos básicos para conexão e configuração das MVs, gerenciamento das entradas de roteamento em hardware e, também, troca de informações de eventos de rede.

5. PROTÓTIPO

Com o objetivo de tornar a solução robusta e aplicável em qualquer ambiente, utilizou-se na implementação do protótipo o Quagga [8], uma conhecida *engine* de roteamento *open source* com suporte aos principais protocolos de roteamento (RIP, OSPF, BGP). Poderia-se também utilizar outras pilhas de roteamento como o XORP [21] sem alteração de código.

Existe uma variedade de virtualizadores que podem seguir paradigmas diferenciados. Nesta implementação foi utilizado o QEMU, que é um software livre e permite uma virtualização completa. Neste modelo de virtualização temos a vantagem de um isolamento forte, no entanto o consumo de recursos é mais elevado e apresenta um desempenho inferior quando comparado com um virtualizador no nível do sistema operacional.

Como plataforma para programabilidade remota dos equipamentos de rede utilizou-se a tecnologia OpenFlow na versão 1.0. Uma grande vantagem desse protocolo é a abordagem *multi-vendor*. Isto significa que independentemente do fabricante e do modelo do equipamento que compuser a rede, desde que haja suporte ao protocolo OpenFlow, não serão necessárias mudanças no controlador nem nas aplicações de rede que executam sobre ele.

A nossa infraestrutura do plano de dados é formada por NetFPGAs, que são hardware programáveis com quatro interfaces de rede Gigabit e com módulo OpenFlow. A escolha da NetFPGA se deu pela flexibilidade de programação do plano de encaminhamento e pela taxa de processamento de pacotes em Gigabits.

O componente RF-Slave, *daemon* que roda em cada máquina virtual, foi implementado em C++ e se utiliza de *bash scripts* e chamadas de sistemas para monitorar as atualizações das tabelas ARP e ROUTE do Linux através de um mecanismo de *polling* com intervalo de verificação ajustado em 100 milissegundos.[1] As rotas aprendidas pelo Quagga são convertidas em entradas de fluxos OpenFlow, onde o MAC da interface local e o IP da rede destino se traduzem na regra cuja ação correspondente é a reescrita do SRC_MAC (endereço MAC da interface local), do DST_MAC (endereço MAC do *nexthop*) e, por fim, no encaminhamento para a porta do roteador que se encontra conectada ao próximo nó.[2] Além da tabela de rotas IP, as entradas da tabela ARP também devem ser instaladas no plano de dados, pois é através delas que se torna possível alcançar os nós das redes diretamente conectadas ao roteador. A diferença básica entre uma rota e uma entrada ARP é a máscara, que nesta última será sempre 32 bits (denominado de *exact match*) e, no caso das rotas, dependerá da rede em questão.

A componente RF-Server foi integrado ao RF-Controller e também implementado em C++ como uma aplicação do controlador NOX [16]. Trata-se de um controlador OpenFlow *open source* cujo objetivo é prover uma plataforma simples para programação de aplicações de redes OpenFlow. Com o objetivo principal de avaliar a viabilidade da solução, esse módulo foi desenvolvido apenas com a funcionalidade de criação estática da rede virtual como uma réplica da topologia física. Nesta implementação, onde as conexões entre as MVs não ocorrem de forma dinâmica, em caso de quebra de enlace no plano de encaminhamento, a falha não será reproduzida na topologia virtual. Por essa razão, é necessário o envio do tráfego de controle para o plano de encaminhamento para que o protocolo detecte a queda do enlace e possa recalcular as rotas e se recuperar do problema. Portanto, o tráfego de controle dos protocolos de roteamento (ex. OSPF Hello, LSA) é sempre direcionado às NetFPGAs de modo que os protocolos detectem falhas automaticamente.

[1] Na versão atual, temos implementado um mecanismo orientado a eventos com a utilização da biblioteca RTNetLink do Linux.

[2] No OpenFlow 1.1 existem as ações de decremento do TTL e *checksum update*.

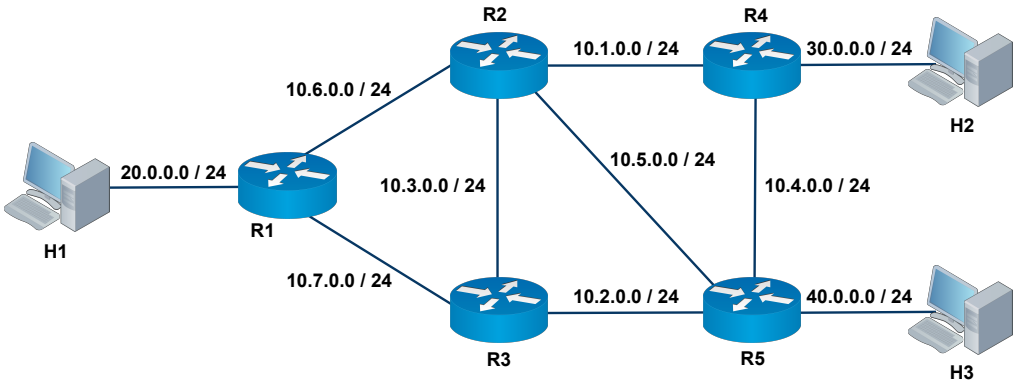


Figura 5. Rede de teste montada com NetFPGAs.

6. AVALIAÇÃO

A avaliação foi elaborada em cima de um ambiente de rede com cinco NetFPGAs como nós de encaminhamento com suporte ao OpenFlow, um servidor QuadCore para virtualização das cinco máquinas virtuais e ainda um servidor para o controlador NOX, conforme a configuração de rede na Figura 5.

Por se tratar de um ambiente real e não simulado e, consequentemente, devido à limitação do número de equipamentos disponíveis com suporte ao OpenFlow, a rede testada apresenta um número reduzido de nós. Entretanto, a quantidade de nós é suficiente para uma avaliação da viabilidade da proposta através da análise detalhada das trocas de mensagens e dos tempos relacionados à convergência da rede em caso de falha como, por exemplo, o tempo de processamento das mensagens RouteFlow e OpenFlow, que não existe em uma arquitetura clássica de roteador com pilha de protocolos embarcada.

6.1. TEMPO DE CONVERGÊNCIA COM TRÁFEGO *line rate*

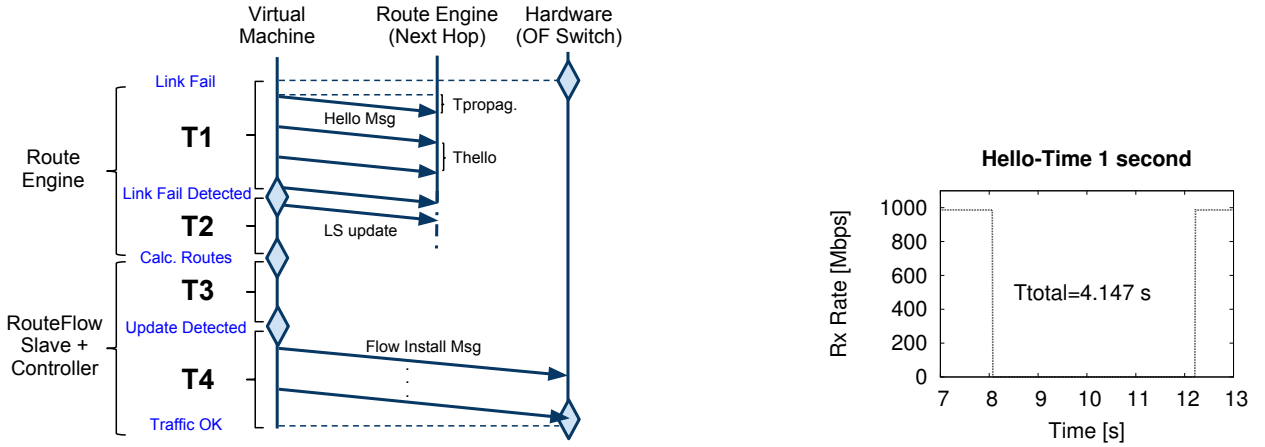
Para os testes de convergência foi utilizado um gerador e analisador de tráfego configurado para transmitir pacotes IP de 1500 bytes de tamanho em ambos os sentidos (*fullduplex*) a uma taxa de 1 Gbps. Desta forma garante-se que esta solução de roteamento remota é capaz de operar com tráfego na taxa de linha uma vez que os pacotes são processados em hardware.

Os tempos de convergência estão ligados ao parâmetro de configuração *hello-time* do OSPF, porém é necessário uma análise mais detalhada para identificarmos o impacto de utilizar uma pilha de protocolos remota ao equipamento. Desta forma podemos fragmentar o tempo de convergência em quatro partes (Figura 6 (a)), a saber: (T1) tempo que o protocolo leva para identificar a falha de *link*; (T2) tempo gasto pelo OSPF com as trocas de mensagens e o cálculo de novas rotas; (T3) tempo necessário para o RouteFlow-Slave detectar as modificações na tabela de roteamento no Linux; e (T4) tempo requerido para o RouteFlow-Slave encaminhar as mensagens de instalação de fluxos para o RouteFlow-Controller e este, através do controlador NOX, instalar os fluxos em hardware. De fato, os dois primeiros tempos (T1 e T2) são inerentes do protocolo de roteamento, portanto o impacto do RouteFlow está claramente presente nos tempos T3 e T4.

O gráfico da Figura 6 (b) mostra o tempo de convergência da rede, utilizando o protocolo OSPFv3 configurado com o *hello-interval* em 1s, após falha de um *link*. Este é o principal parâmetro do OSPF diretamente relacionado ao tempo de detecção de falhas. O *hello-interval* foi configurado para 1 segundo, 5 segundos e 10 segundos, correspondendo aos valores típicos da indústria para enlaces Ethernet [14]. O *dead-interval* usado foi o recomendado de quatro vezes o *hello-interval*.

Com o intuito de fragmentar os tempos mostrados no gráfico da Figura 6 (b), foram feitas análises das mensagens enviadas e recebidas pelo hardware OpenFlow para o controlador NOX. Nos resultados está incluso o tempo gasto pelas mensagens para transitar entre o dispositivo de encaminhamento, o controlador e a MV. O tempo T1 pode ser obtido a partir do intervalo entre o instante em que ocorre a interrupção do tráfego até a primeira mensagem de LS-Update gerada pela *engine* de roteamento ao detectar a falha, em seguida leva T2 + T3 para que o RF-Slave inicie o envio da primeira mensagem de alteração de fluxo e por último o T4 finaliza o processo com o restabelecimento do tráfego. Nos gráficos da Figura 7 e na Tabela 1 apresenta-se os tempos conforme a fragmentação proposta anteriormente, em que cada um deles possui o valor de tempo abaixo do qual se encontra metade dos resultados e o valor de tempo abaixo do qual se encontram 90% dos testes num total de 20 repetições para cada uma das três configurações de *hello-time*.

Conforme os dados apresentados na Tabela 1, o tempo total de convergência é bem próximo do tempo T1, ou seja, o tempo de detecção da falha. Portanto, o tempo



(a) Fluxo de eventos durante convergência.

(b) Tempo total de convergência.

Figura 6. Convergência do OSPF após falha.

Tabela 1. *Tempos de convergência após falha.*

Hello Time	T1 [s]		T2+T3 [s]		T4 [s]		Ttotal [s]	
OSPF	Tmed.	T90%	Tmed.	T90%	Tmed.	T90%	Tmed.	T90%
1 sec.	3.249	3.923	0.360	0.398	0.070	0.123	3.700	4.373
5 sec.	16.713	18.937	0.320	0.389	0.057	0.099	17.135	19.308
10 sec.	36.406	37.846	0.358	0.497	0.042	0.106	36.807	38.266

restante gasto com o cálculo de rotas, detecção de modificações da tabela de roteamento pelo RouteFlow-Slave e a instalação do fluxo têm pouco impacto no tempo de convergência. Apesar dos tempos T2 e T3 não terem sido analisados separadamente, sabe-se que o tempo de *polling* para verificar atualizações da tabela de roteamento e ARP do Linux é fixo em 100 ms, ou seja, da soma T2 + T3, o T3 representa até 100 milissegundos no pior caso. Vale ressaltar que foi utilizada a estratégia de *polling* por simplificação; no entanto, podem ser feitas otimizações para reduzir substancialmente o tempo T3. Uma opção seria fazer com que a aplicação se registrasse no Zebra (módulo central do Quagga) para receber notificações sobre a RIB, por consequência a informação chegaria ao RF-Slave de forma muito mais rápida e eficiente.

Dada a baixa representatividade dos tempos T3 e T4 sobre o tempo total, consideramos viável a solução RouteFlow dentro do contexto proposto.

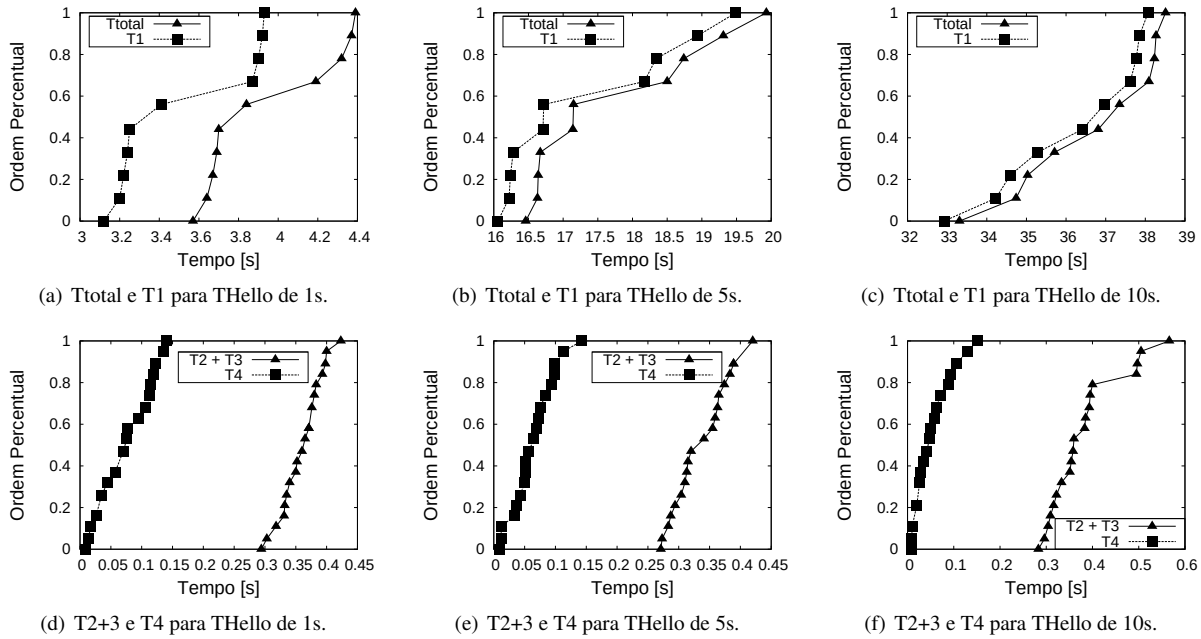
6.2. Encaminhamento em *Slow Path* e *Fast Path*

Outra forma de avaliar o impacto de uma pilha remota de protocolos de roteamento é comparar os tempos para os modos de encaminhamento por *slow path* e *fast path*. *Slow path* refere-se ao encaminhamento lento que ocorre no plano de controle, em software, por exemplo, quando o roteador precisa se comunicar com um nó de uma rede diretamente conectada a ele e ainda não ocorreu o aprendizado do endereço MAC do destino, que é necessário para o encaminhamento em hardware. Este cenário tipicamente ocorre para os nós que entraram na rede ou que ficaram sem comunicação por um longo período, por isso a relevância desta operação nos roteadores é baixa, embora necessária. O *fast path* refere-se ao encaminhamento rápido, em hardware, que ocorre após o aprendizado dos endereços dos nós vizinhos e o preenchimentos das tabelas em hardware. Portanto, o *slow path* ocorre, quando necessário, apenas para o primeiro pacote, a partir do qual será realizado o aprendizado e o restante dos pacotes serão processados na velocidade de linha.

Através deste teste é possível avaliar a diferença de tempo entre um encaminhamento por software e por hardware no caso de um roteador com protocolos embarcados e o RouteFlow, cujo plano de controle é remoto.

Com este objetivo foram realizados testes de PING (ICMP) entre dois *hosts* diretamente conectados a portas distintas de um mesmo roteador, configuradas em redes diferentes. O experimento parte do ponto no qual os *hosts* não têm conhecimento do endereço MAC dos seus *gateways* e o roteador também não contempla ambos *hosts* em suas tabelas. Após o aprendizado dos endereços, obtém-se os valores de *fast path*. Foram utilizados os seguintes equipamentos de camada 3 (*Layer 3 Switches*): CISCO 3560-e Catalyst, Extreme x450-e, CPqD Enterprise (protótipo) e o RouteFlow. Os resultados dos testes estão apresentados na Tabela 2.

Foi observado nos testes com o RouteFlow que ambos



(a) Ttotal e T1 para THello de 1s. (b) Ttotal e T1 para THello de 5s. (c) Ttotal e T1 para THello de 10s.

(d) T2+3 e T4 para THello de 1s. (e) T2+3 e T4 para THello de 5s. (f) T2+3 e T4 para THello de 10s.

Figura 7. Fragmentação do tempo de convergência do OSPF.

Tabela 2. *Tempo de resposta ICMP.*

Equipamento	Slow Path [ms]		Fast Path [ms]	
	Tmed.	T90%	Tmed.	T90%
CISCO 3560-e Catalyst	5.46	7.75	0.100	0.130
Extreme x450-e	11.30	14.00	0.106	0.141
CPqD Enterprise	14.20	17.30	0.101	0.147
RouteFlow	116.00	138.00	0.082	0.119

os pacotes ICMP *request* e ICMP *reply* são encaminhados por software. O motivo deste comportamento está relacionado principalmente ao tempo de *polling* de 100ms para que as atualizações nas tabelas do Linux sejam detectadas. No momento da resposta ICMP os fluxos ainda não estão instalados e o pacote precisa novamente ser direcionado ao controlador para ser encaminhado por software. Portanto, como exemplificado na Seção 6.1, é válido pensar em um tempo consideravelmente menor para este teste como uma otimização da forma com que o RFSlave passa a ter conhecimento das atualizações.

Outro ponto a ser destacado sobre o resultado do *slow path*, consideravelmente, superior quando comparado aos dispositivos com software embarcado, é o desempenho do controlador OpenFlow, NOX, utilizado nos testes, cuja proposta é a de prover uma plataforma simples de desenvolvimento de aplicações de rede sem o compromisso de manter o foco em desempenho. Neste sentido, já foram identificadas possíveis otimizações que devem trazer ganhos significativos no tempo de processamento das mensagens no controlador como, por exemplo, tornar o controlador multi-tarefa de forma a realizar o processamento paralelo das mensagens; outra opção consiste na implementação de mensagens que agreguem mais informação resultando em um menor número de chaveamentos de contextos da aplicação para processar cada pacote recebido.

Em contrapartida, pode-se observar o melhor desempenho de *fast path* do RouteFlow, que é a forma de encaminhamento mais relevante. Este fato pode ser explicado pelo rápido encaminhamento da tabela de fluxos quando comparado ao *longest prefix match*, que é utilizado no encaminhamento IP.

7. DISCUSSÃO

Nesta seção serão discutidos brevemente algumas considerações adicionais.

7.1. ISOLAMENTO

Durante os testes de avaliação foi possível notar variações de desempenho que de modo geral podem estar relacionadas à questão do isolamento entre as instâncias de roteamento virtualizadas. Problemas de isolamento entre MVs já foram identificados na literatura, principalmente na análise do processamento do sistema em qualquer solução montada em cima de ambientes virtuais. No entanto, tal fato pode ser compensado com a utilização de (um *cluster* de) servidores com grande poder de processamento.

7.2. Escalabilidade

O *testbed* esteve limitado ao número de dispositivos disponíveis, o que inviabilizou uma avaliação de escalabilidade do RouteFlow. É clara a necessidade de uma infraestrutura virtual com servidores distribuídos e alta capacidade de processamento para suportar o aumento do tamanho da rede sem qualquer prejuízo à solução. Porém, sabe-se que a capacidade de processamento disponível em soluções comerciais aumenta rapidamente com o passar dos anos (lei de Moore), além de uma redução progressiva dos custos envolvidos, contribuindo assim com a relação custo-eficiência e a escalabilidade do RouteFlow.

Outra questão pertinente é o gargalo observado nas implementações disponíveis do OpenFlow quanto ao tamanho da tabela de fluxos (entre 2 e 4 mil) e a capacidade de instalação de novas entradas por segundo (valor em torno de 100 fluxos/seg.). Entendemos que estas limitações são transitórias e serão contornadas com a maturidade da tecnologia, incluindo o acesso às tabelas L2/L3 de hardware a partir da versão 1.1 do OpenFlow.

7.3. Virtualização de redes

A partir de uma arquitetura baseada na separação entre os planos de controle e dados e no isolamento, tanto do tráfego na infraestrutura física quanto nas instâncias de controle, torna-se possível explorar o conceito de roteamento como serviço [12]. Desta forma podemos ter topologias lógicas independentes sobre uma mesma rede e que rodem protocolos distintos, resultando numa abordagem de virtualização com um aproveitamento melhor dos recursos da infraestrutura que agora pode ser compartilhada com diferentes propósitos, em alinhamento com as propostas de arquiteturas pluralistas [7].

8. Trabalhos Futuros

No site do projeto RouteFlow[3] encontra-se uma lista atualizada dos trabalhos em andamento. Recentemente foi integrado o uso do *software switch* na interconexão das MVs, dando suporte a ambientes dinâmicos e fisicamente distribuídos. Também foi implementado o mecanismo de detecção de atualizações da tabela de roteamento destacado na secão 6.1.

Dentre os próximos passos identificados daremos foco nos testes com o protocolo BGP, que tem sido um dos grandes desafios e causa de problemas nas arquiteturas de roteamento IP. Em paralelo iremos investir nos recursos da tecnologia de virtualização com foco na otimização do ambiente virtual [4]. A utilização de técnicas avançadas para a migração e o gerenciamento do estado (ex: *checkpointing*, *rollback*) das MVs também serão exploradas, assim como os modos de operação de agregação e multiplexação discutidos na Seção 4.1.

[3] http://go.cpqd.com.br/routeflow

9. Conclusões

A proposta deste trabalho representa uma abordagem *"scale-out"* para arquiteturas de redes de pacotes. Com o plano de controle externo ao dispositivo de rede, passa a existir maior independência entre este plano e o de encaminhamento, de forma que ambos escalem e evoluam separadamente. Neste sentido o RouteFlow possibilita o surgimento de redes mais baratas e flexíveis mantendo compatibilidade com redes legadas, apoiando uma evolução das redes onde a conectividade IP pode se tornar um *commodity* ofertado em um modelo de plataforma como serviço [12].

Os resultados alcançados na avaliação do protótipo sugerem o grande potencial do RouteFlow como solução de roteamento para redes de campus/corporativas com o ganho de flexibilidade e poder de inovação. As questões pertinentes ao desempenho não inviabilizam a proposta, uma vez que otimizações já identificadas e a maturidade do protocolo OpenFlow trarão significativas melhorias de desempenho.

Referências

[1] Muhammad Bilal Anwer, Murtaza Motiwala, Mukarram bin Tariq, and Nick Feamster. Switchblade: a platform for rapid deployment of network protocols on programmable hardware. SIGCOMM '10, pages 183–194, New York, NY, USA, 2010. ACM.

[2] Katerina Argyraki, Salman Baset, Byung-Gon Chun, Kevin Fall, Gianluca Iannaccone, Allan Knies, Eddie Kohler, Maziar Manesh, Sergiu Nedevschi, and Sylvia Ratnasamy. Can software routers scale? PRESTO '08, pages 21–26, New York, NY, USA, 2008. ACM.

[3] Raffaele Bolla, Roberto Bruschi, Guerino Lamanna, and Andrea Ranieri. Drop: An open-source project towards distributed sw router architectures. In *GLOBECOM*, pages 1–6. IEEE, 2009.

[4] Carlos N. A. Corrêa, Sidney Lucena, Christian E. Rothenberg, and Marcos R. Salvador. Desempenho de soluções de virtualização para plano de controle de roteamento de redes virtuais. In *SBRC 2011*, May 2011.

[5] Martin Casado, Teemu Koponen, Rajiv Ramanathan, and Scott Shenker. Virtualizing the network forwarding plane. In *Presto '10*, August 2010.

[6] Dobrescu and et al. Routebricks: exploiting parallelism to scale software routers. SOSP '09, pages 15–28, New York, NY, USA, 2009. ACM.

[7] Natalia Fernandes, Marcelo Moreira, Igor Moraes, Lyno Ferraz, Rodrigo Couto, Hugo Carvalho, Miguel Campista, Luís Costa, and Otto Duarte. Virtual networks: isolation, performance, and trends. *Annals of Telecommunications*, pages 1–17, October 2010.

[8] GNU Quagga Project. http://www.quagga.org.

[9] Gude and et al. NOX: towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110, 2008.

[10] James Hamilton. Networking: The last bastion of mainframe computing. http://perspectives.mvdirona.com/2009/12/19/NetworkingTheLastBastionOfMainframeComputing.aspx, Dec 2009.

[11] Sangjin Han, Keon Jang, KyoungSoo Park, and Sue Moon. Packetshader: a gpu-accelerated software router. SIGCOMM '10, pages 195–206, New York, NY, USA, 2010. ACM.

[12] Eric Keller and Jennifer Rexford. The 'Platform as a Service' model for networking. In *INM/WREN 10*, April 2010.

[13] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38:69–74, 2008.

[14] J. Moy. OSPF Version 2. RFC 2328, April 1998.

[15] Marcelo Ribeiro Nascimento, Christian Esteve Rothenberg, Marcos Rogério Salvador, and Maurício Ferreira Magalhães. QuagFlow: partnering Quagga with OpenFlow. *SIGCOMM Comput. Commun. Rev.*, 40:441–442, August 2010.

[16] NOX. An open-source openflow controller. http://noxrepo.org.

[17] OpenFlow Switch Consortium. Official website. http://www.openflowswitch.org.

[18] Kok-Kiong Yapy Guido Appenzeller Martin Casado Nick McKeowny Guru Parulkary Rob Sherwood, Glen Gibby. Can the production network be the testbed? OSDI'10, pages 1–14. USENIX Association, 2010.

[19] Nadi Sarrar, Anja Feldmann, Steve Uhlig, Rob Sherwood, and Xin Huang. FIBIUM - towards hardware accelerated software routers. In *EuroView 2010*, August 2010.

[20] Tammo Spalink, Scott Karlin, Larry Peterson, and Yitzchak Gottlieb. Building a robust software-based router using network processors. *SIGOPS*, 35:216–229, October 2001.

[21] The XORP Project. extensible open router platform. http://www.xorp.org.

[22] Vyatta. Series 2500. http://vyatta.com/downloads/datasheets/vyatta_2500_datasheet.pdf.

Revista Brasileira de Redes de Computadores e Sistemas Distribuídos

LARC - Laboratório Nacional de Redes de Computadores
SBC - Sociedade Brasileria de Computação