COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

TÉCNICAS DE VIRTUALIZAÇÃO E AUTOCONFIGURAÇÃO PARA O PROJETO DE REDES DE NOVA GERAÇÃO

Natalia Castro Fernandes

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Otto Carlos Muniz Bandeira Duarte

Rio de Janeiro
Setembro de 2011

TÉCNICAS DE VIRTUALIZAÇÃO E AUTOCONFIGURAÇÃO PARA O PROJETO DE REDES DE NOVA GERAÇÃO

Natalia Castro Fernandes

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Otto Carlos Muniz Bandeira Duarte, Dr. Ing.

Prof. Maurício Ferreira Magalhães, Dr. Ing.

Prof. Julius Cesar Barreto Leite, Ph. D.

Prof. Edmundo Roberto Mauro Madeira, D. Sc.

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2011

Fernandes, Natalia Castro

Técnicas de Virtualização e Autoconfiguração para o Projeto de Redes de Nova Geração/Natalia Castro Fernandes. – Rio de Janeiro: UFRJ/COPPE, 2011.

XXIII, 150 p.: il.; 29, 7cm.

Orientador: Otto Carlos Muniz Bandeira Duarte

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2011.

Referências Bibliográficas: p. 139 – 150.

1. Internet do Futuro. 2. Virtualização. 3. Isolamento. 4. Qualidade de Serviço. 5. Autoconfiguração de endereços. 6. Redes Ad Hoc. I. Duarte, Otto Carlos Muniz Bandeira. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha família.

Agradecimentos

Agradeço, inicialmente, a Deus pela oportunidade de estar defendendo a minha tese de doutorado. Agradeço também à minha família, em especial meus pais, Aristides e Luiza, e minhas irmãs, Ana Luiza e Ana Carolina, pelo incentivo e apoio em todas as horas. Agradeço também ao meu querido João Kleber, que me ajudou de tantas maneiras durante todo esse processo e foi essencial nas minhas conquistas até aqui. À minha avó Dulce e aos meus tios e tias, agradeço pela força e as orações durante todo o meu processo de formação.

Aos amigos, também fico grata pelas horas de diversão e pelas sugestões recebidas ao longo da tese. Agradeço, em especial, aos meus amigos do GTA pela companhia e pelas sugestões recebidas. Destaco entre esses, os amigos Marcelo, Lyno, Carlo, Diogo, Hugo, Pisa, e Rodrigo, além de Igor, Miguel e Pedro, hoje já professores. Também agradeço aos meus amigos Marília, Mariangela, Danilo e Kaio pelo estímulo para alcançar os meus objetivos.

Agradeço também a todos os professores da COPPE/UFRJ por todos os conhecimentos e orientações recebidos ao longo do doutorado, que me foram de grande utilidade para o desenvolvimento da tese. Em especial, agradeço ao professor Otto, meu orientador de iniciação científica, de mestrado e de doutorado, pela amizade, conselhos e lições aprendidas, além da oportunidade de ter entrado no GTA. Agradeço também ao professor Luís Henrique, pela amizade, atenção e ajudas durante o desenvolvimento da tese. Por participar da minha banca examinadora, agradeço novamente ao professor Luís Henrique e também aos professores Maurício Magalhães, Julius Leite e Edmundo Madeira. Também agradeço ao professor Daniel Figueiredo pelas sugestões para a tese.

Agradeço aos órgãos de fomento à pesquisa FAPERJ, CAPES, FINEP e CNPq, pelos recursos financeiros recebidos, que permitiram o curso do doutorado com bolsa e a participação em congressos da área. Também aos funcionários do Programa de Engenharia Elétrica da COPPE/UFRJ, pela presteza no atendimento na secretaria do Programa. Por fim, um obrigado especial a todos que, embora não estejam citados aqui, me incentivaram, contribuindo de forma direta ou indireta, para a minha formação acadêmica e profissional.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

TÉCNICAS DE VIRTUALIZAÇÃO E AUTOCONFIGURAÇÃO PARA O PROJETO DE REDES DE NOVA GERAÇÃO

Natalia Castro Fernandes

Setembro/2011

Orientador: Otto Carlos Muniz Bandeira Duarte

Programa: Engenharia Elétrica

Esta tese aborda dois aspectos relevantes para a Internet do Futuro: a autoconfiguração de redes sem fio colaborativas e o provimento de isolamento na arquitetura virtualizada para redes de nova geração.

As redes ad hoc são redes que se beneficiam da cooperação entre os nós para obter um maior alcance. A atribuição de endereços em redes ad hoc requer um procedimento distribuído que solucione todas as colisões de endereço em uma rede dinâmica, com frequentes partições e com nós entrando e saindo. Assim, propõe-se um protocolo de autoconfiguração de endereços para redes ad hoc baseado em filtros que é leve e também robusto à perda de pacotes. As funcionalidades dos filtros são analisadas em diversos cenários e os resultados da simulação mostram que a proposta reduz em até 21 vezes a carga de controle quando comparada a outras propostas.

Outro aspecto da Internet do Futuro analisado é o uso da virtualização para desenvolver um núcleo de rede flexível e programável. Um sistema para a construção de ambientes virtuais isolados e com qualidade de serviço, chamado VIPER, é proposto. O VIPER, cujos componentes principais são o gerenciador de recursos e o controlador de admissão, garante um controle eficiente dos recursos físicos compartilhados entre as redes virtuais. O gerenciador de recursos garante o isolamento através da adaptação dinâmica dos seus parâmetros de acordo com as demandas de cada rede, enquanto que o controlador de admissão controla o acesso de novas redes virtuais ao substrato físico. Foi desenvolvido um protótipo cuja análise mostra que o VIPER, quando comparado a outras propostas: (i) garante a provisão dos recursos contratados , (ii) provê um controle de admissão eficiente de redes virtuais e (iii) garante primitivas básicas de QoS.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

VIRTUALIZATION AND AUTO CONFIGURATION TECHNIQUES FOR THE PROJECT OF NEW GENERATION NETWORKS

Natalia Castro Fernandes

September/2011

Advisor: Otto Carlos Muniz Bandeira Duarte

Department: Electrical Engineering

This thesis addresses two main issues for the Future Internet: the auto configuration of collaborative wireless networks and the provision of isolation in the virtualized architecture for next generation networks.

Ad hoc networks are networks that benefit from cooperation between nodes to achieve a greater range. The address assignment in ad hoc networks requires a distributed procedure that resolves all address collisions in a dynamic network, with frequent partitions and joining/leaving nodes. Thus, we propose a filter-based addressing protocol for auto configuration of ad hoc networks that is lightweight and also robust to packet loss. We analyze the functionalities provided by the filters for different scenarios, and simulation results show a control load reduction of 21 times when compared to other proposals.

We also analyze the provision of a flexible and programmable network core that is an important feature for the Future Internet. We propose VIPER, a system to build isolated virtual environments with QoS provision support. VIPER, whose main components are the resource sharing manager and the virtual network admission controller, guarantees a fine sharing of physical resources among virtual networks. The resource sharing manager achieves isolation by dynamically adapting itself to the resource demands of each virtual network and the admission controller controls the access of new virtual networks to the physical substrate. We developed a prototype whose evaluation reveals that VIPER, when compared to the other proposals: (i) enforces contracted agreements, (ii) provides an efficient admission control of virtual networks, and (iii) guarantees basic QoS primitives.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

A	Conjunto de elementos a serem inseridos no filtro de Bloom, p. 15
A_i	Conjunto de endereços alocados na partição i, p. 32
A_s	Tamanho do endereço, p. 19
C	Número de colisões de endereço, p. 34
CP	Chave pública, p. 96
C_1	Custo para fazer uma transferência entre máquinas virtuais, p. 98
C_2	Custo para fazer uma transferência entre uma máquina virtual e uma máquina externa, p. 98
C_3	Custo para fazer uma transferência entre uma máquina externa e máquina virtual, p. 98
C_4	Custo para fazer uma transferência entre o Dom0 e uma máquina virtual, p. 98
C_5	Custo para fazer uma transferência entre uma máquina virtual e o Dom0, p. 98
C_6	Custo para fazer uma transferência entre máquinas externas, p. 98
C_a	Constante de ajuste, p. 100
C_f	Conformidade da rede i, p. 124
C_t	Conjunto de tuplas (x_1, x_2) cuja soma $x_1 + x_2$ excede a capacidade do recurso C, p. 113
Cp	Chave privada, p. 96

$Crip_{ass}$	Criptografia assimétrica, p. 96
$Crip_{sim}$	Criptografia simétrica, p. 96
D	Vetor das demandas das redes virtuais, p. 104
E_{adm}	Limiar especificado pelo administrador de infraestrutura para e_{max}, p. 111
E_c	Erro a curto prazo da rede i, p. 124
E_l	Erro a longo prazo da rede i, p. 124
F_i	Número de mensagens inundadas em um processo de união de partições i, p. 34
G_r	Gasto de processamento de uma rede virtual r, p. 98
$H_{sft}(\ )$	Número de medidas no intervalo I do histograma do substrato deslocado de Δ, p. 113
$H_{sub}(\ )$	Número de medidas no intervalo I do histograma do substrato, p. 113
I	Intervalo do histograma, p. 112
I_c	Intervalo de tempo curto, p. 104
I_l	Intervalo de tempo longo, p. 104
K	Chave criptográfica, p. 96
K_s	Chave de sessão, p. 96
K_{adm}	Constante arbitrada pelo administrador de infraestrutura que determina quantos intervalos longos formam um intervalo de monitoramento, p. 111
K_{rand}	Número de intervalos longos que não serão avaliados após K_{adm} intervalos longos, p. 111
K_{si}	Número aleatório escolhido pelo domínio i, p. 96
L	Nível crítico de uso dos recursos no gerenciador com alocação de recursos ociosos, p. 100
L_s	Limite superior do intervalo I, p. 112

M	Mensagem, p. 96
$Mconf$	MANETconf, p. 29
N	Número de nós, p. 34
N_F	Número de transmissões de mensagens de inundação no FAP, p. 23
N_H	Número de redes virtuais hospedadas em um nó físico, p. 105
N_N	Número de vizinhos do novo nó, p. 34
N_T	Número de tentativas antes de concluir que o nó está só, p. 34
N_{MI}	Número de tentativas para alocar um endereço para um novo nó , p. 40
N_{MP}	Limite de perdas de mensagens para determinar que o nó com a identificação da partição está ausente, p. 40
N_{MR}	Número de tentativas de se conectar com o iniciador , p. 40
N_{MT}	Número de busca por vizinhos antes de concluir que o nó está só, p. 40
N_{int}	Número de intervalos no histograma do substrato, p. 113
N_{r_i}	Número de pacotes do tipo i, $i \in [1, 6]$, transitados pela rede virtual r, p. 98
$P(E_c)$	Probabilidade de existir pelo menos uma colisão de endereços em um conjunto com n nós, p. 10
$P(c_i \geq 2^b)$	Probabilidade de transbordamento de contadores no Filtro de Bloom, p. 16
PMF_{hist}	Função de massa de probabilidade para o histograma do substrato deslocado, p. 113
PMF_{new}	Função de massa de probabilidade para a nova rede virtual, p. 113
P_0	Probabilidade que um bit seja 0 após a inserção de n elementos, p. 16
P_1	Probabilidade de um bit ser 1 após a inserção de n elementos no filtro, p. 33

P_A Probabilidade de dois nós escolherem AREQs iguais, ou seja, escolherem o mesmo endereço e o mesmo identificador, p. 30

P_B Probabilidade de bloqueio de recurso demandado, p. 113

P_L Probabilidade de bloqueio de recursos aceita pelo administrador da infraestrutura, p. 114

$P_{BF_{col}}$ Probabilidade de que dois filtros de Bloom possuam os mesmos bits em 1 e os mesmos bits em 0, p. 32

P_{BF} Probabilidade de duas partições diferentes possuírem o mesmo filtro de endereços, utilizando-se o filtro de Bloom, p. 32

P_{SF} Probabilidade de duas partições diferentes possuírem o mesmo filtro de endereços, utilizando-se o filtro de sequência, p. 32

P_{fp} Probabilidade de falsos positivos em um filtro de Bloom, após a inserção de n elementos, p. 16

$Perc_i$ Percentual de recursos atribuídos como reserva mínima para a rede i, p. 99

R Conjunto dos endereços, p. 32

$R(t)$ Total de recursos físicos disponíveis, p. 100

R_t Total de recursos físicos disponibilizados para as redes virtuais, p. 105

R_{avg} Uso médio de recursos de cada rede virtual, p. 112

R_c Vetor com a taxa de recursos reservados a curto prazo para as redes virtuais, p. 104

R_e Vetor com a taxa de recursos reservados a curto prazo exclusivamente para cada rede virtual, p. 104

R_l Vetor com a taxa média de recursos reservados a longo prazo para as redes virtuais, p. 104

R_{max} Vetor com a taxa de recursos máxima que pode ser provida em um intervalo longo de tempo I_l, caso exista demanda, p. 104

$R_{nr}(t)$ Total de recursos não-reservados no gerenciador com alocação de recursos ociosos, p. 100

T Intervalo de monitoração do gerenciador com alocação de recursos ociosos, p. 99

T_C Tempo de espera após detecção de colisão no FAP, p. 23

T_L Tempo de escuta do meio no mecanismo de entrada na rede do FAP, p. 21

T_M Intervalo mínimo entre renovações de filtros, p. 28

T_P Intervalo mínimo entre partições, p. 25

T_S Tempo de armazenamento de uma nova assinatura, p. 26

T_W Tempo de espera por outros AREQs, p. 23

T_H Intervalo entre mensagens *Hello*, p. 40

T_L Tempo máximo escutando o meio, p. 40

T_{MAP} Limiar do *Allocation Pending Timer*, p. 40

T_{MA} Limiar do *Address Allocation Timer*, p. 40

T_{MN} Limiar do *Neighbor Reply Timer*, p. 40

T_{MP} Limiar do *Partition Timer*, p. 40

T_{MR} Limiar do *Request Reply Timer*, p. 40

T_M Intervalo mínimo entre renovações de filtros, p. 40

T_P Intervalo mínimo entre uniões de partições, p. 40

T_R Intervalo entre replicações de mensagens inundadas, p. 40

$T_{S'}$ Tempo armazenando as assinaturas da outra partição no FAP, p. 27

T_S Tempo de armazenamento de assinaturas de filtro geradas pelo nó, p. 40

T_W Tempo de espera por mensagens AREQ e AREP , p. 40

$U(t)$ Uso total de recursos no Dom0 em um intervalo T, p. 99

U_i Número de mensagens transmitidas em *unicast* em um processo de união de partições i, p. 34

$U_i(t)$	Uso total de recursos no Dom0 pelo roteador virtual i em um intervalo T, p. 99
U_{o_i}	Indicativo relativo ao uso de recursos ociosos pela rede virtual i, p. 103
V_l	Vetor com o volume de recursos que as redes virtuais podem usar dentro de I_l, p. 104
W_i	Peso de rede i no gerenciador com alocação de recursos ociosos, p. 99
Δ	Deslocamento do histograma do substrato, p. 112
δ	distância entre $a_{i_{suf}}$ e $a_{0_{suf}}$, p. 18
$\mathbb{Z}$	Conjunto dos números inteiros, p. 99
a_0	Elemento inicial da sequência de endereços no filtro de Sequência, p. 18
$a_{i_{suf}}$	Sufixo de endereço do elemento a_i, p. 18
b	Número de bits em cada contador do filtro de Bloom, p. 16
b_{suf}	Número de bits no sufixo do endereço, p. 18
c_i	Contador i do filtro de Bloom com contadores, p. 16
e	Número de Euler, p. 17
e_m	Maior erro no eixo y entre os dois histogramas normalizados, p. 111
e_s	Tamanho do espaço de eventos, p. 30
$h_1, h_2, \ldots, h_k$	Conjunto de funções *hash*, p. 15
id	Identificação do nó fonte da mensagem, p. 96
k	Número de funções *hash*, p. 15
m	Número de bits no filtro de Bloom, p. 15
n	Número de nós na rede, p. 10
r	Tamanho do espaço de endereçamento, p. 10
s	Tamanho do filtro de endereços, p. 19

$P_{ui}(t+T)$	Punição da rede i no intervalo $t+T$, p. 100
XenVM-1	Configuração de testes na qual as máquinas virtuais funcionam como roteadores Xen completos e o Dom0 e a máquina virtual compartilham o mesmo núcleo de CPU, p. 83
XenVM-2	Configuração de testes na qual as máquinas virtuais funcionam como roteadores Xen completos e o Dom0 e a máquina virtual possuem núcleos de CPU separados, p. 83
Point-To-Point	Configuração de testes na qual as máquinas de origem e destino são ligadas diretamente, p. 81

Lista de Abreviaturas

AF	*Address Filter*, p. 23
AODV	*Ad hoc On-Demand Distance Vector Routing*, p. 39
AREP	*Address Reply message*, p. 12
AREQ	*Address Request message*, p. 12
CABO	*Concurrent Architectures are Better than One*, p. 63
CPU	*Central Processing Unit*, p. 6, 83
CP	*Control Plane*, p. 61
DAD-PD	*DAD with partition detection*, p. 29
DAD	*Duplicate Address Detection*, p. 12
DAP	*Dynamic Address assignment Protocol*, p. 14
DHCP	*Dynamic Host Configuration Protocol*, p. 4
DMA	*Direct Memory Access*, p. 83
DP	*Data Plane*, p. 61
E/S	Entrada e Saída, p. 59
EGRE	*Ethernet-over-GRE tunnel*, p. 88
FAP	*Filter-based Addressing Protocol*, p. 4
GER	Gerenciador com alocação exata de recursos, p. 122
GRE	*Generic Routing Encapsulation*, p. 88
GRO	Gerenciador com alocação de recursos ociosos, p. 118
HAL	*Hardware Abstraction Layer*, p. 65

IETF	*Internet Engineering Task Force*, p. 12
IP	*Internet Protocol*, p. 4
IPv6	*Internet Protocol version* 6, p. 2
ISA	*Instruction Set Architecture*, p. 65
ISP	*Internet Service Providers*, p. 2
JVM	*Java Virtual Machine*, p. 66
MAC	*Medium Access Control*, p. 12
MANETconf	*Mobile Ad Hoc Network Configuration of Hosts*, p. 13
NS	*Network Simulator*, p. 39
OMNI	*OpenFlow MaNagement Infrastructure*, p. 85
PC	Plano de Controle, p. 69
PD	Plano de Dados, p. 69
PMF	*Probability mass function*, p. 113
PRESS	*PRedictive Elastic reSource Scaling*, p. 86
SLA	*Service Level Agreement*, p. 6
SLO	*Service Level Objectives - SLO*, p. 86
TCP/IP	*Transmission Control Protocol/Internet Protocol*, p. 1
VIPER	*Virtual network Isolation, Policy Enforcement, and Resource sharing system*, p. 6
VLAN ID	*Virtual Local Area Network Identification*, p. 73
VLAN	*Virtual Local Area Network*, p. 59
VMM	*Virtual Machine Monitor*, p. 59
VM	*Virtual Machine*, p. 59
VNET	*PlanetLab Virtualized Network Access*, p. 66
VNEXT	*Virtual NEtwork management for Xen-based Testbeds*, p. 85
VPN	*Virtual Private Networks*, p. 58
XTC	*Xen Throughput Control*, p. 88

Capítulo 1

Introdução

A Internet é parte essencial da vida na sociedade moderna. Seu uso se estende desde a vida privada de um individuo até os meios governamentais e econômicos globais. Seu sucesso é inegável, já tendo atingido a marca de dois bilhões de usuários. Seu uso tende a ser cada vez maior e mais diversificado, seja pelo acesso sem fio com dispositivos móveis como celulares e carros [1], redes domésticas inteligentes [2], nas quais os aparelhos domésticos passam a ser nós da rede para oferecer mais serviços e aumentar o conforto do usuário, redes de sensores [3], que monitoram o ambiente por razões de segurança ou comodidade, entre diversas outros. Contudo, o sucesso da Internet não implica que o conhecimento sobre redes já seja maduro [4]. De fato, pesquisadores apontam que a Internet cresce e muda mais rapidamente do que a compreensão sobre como projetar, construir e operar redes de grande porte.Essa evolução leva a questionamentos sobre o futuro da rede. A Internet atual e as redes de computadores, de forma geral, ainda apresentam diversos desafios que dificultam e, em alguns casos, impedem a criação de novas aplicações nas redes. Como exemplo, atualmente não é possível construir uma aplicação com qualidade de serviço fim-a-fim. Da mesma forma, não é possível construir um mecanismo de controle de acesso à rede que impeça a entrada de usuários mal intencionados na Internet. Os desafios abrangem áreas como a segurança, o acesso móvel, o roteamento, o endereçamento e a gerência. As soluções propostas, no entanto, muitas vezes não são compatíveis com o modelo TCP/IP ou não são suficientes para resolver completamente o problema [5].

A maioria dos problemas que a Internet tem hoje se deve a forma como ela foi criada e como foi realizada a sua evolução. A Internet foi projetada na década de 70 para fazer a comunicação entre universidades americanas. Eram poucos usuários, todos especializados na área e confiáveis. Hoje em dia, a Internet atende a um grande número de pessoas com formações diferentes, distribuídas por todo o planeta, em um ambiente diversificado e cheio de conflitos [6]. Além disso, as limitações atuais da rede também se devem ao "engessamento" da Internet, uma vez que grandes

modificações no núcleo da rede não são simples, porque dependeriam da aceitação dos provedores de serviço (*Internet Service Providers* - ISP). Os ISPs, por sua vez, possuem requisitos comerciais que precisam ser atendidos, de forma que a adoção de novos mecanismos que modificam o funcionamento de *softwares* bem conhecidos e testados não é bem recebida. Assim, a opção pela estabilidade acaba dificultando a modernização do núcleo da rede. Um exemplo desse engessamento é o uso dos endereços IPv6, com 128 bits, criados em 1998 [7]. Embora os endereços de 32 bits estejam ficando escassos e o IPv6 traga outras vantagens em termos de mobilidade e segurança, além da expansão da faixa de endereços, ele não foi adotado amplamente no núcleo da rede devido a sua incompatibilidade com o IPv4. Mesmo nos países aonde existe um uso maior do IPv6, em 2008, esse uso não chegava a 1% [8].

Outro fator que dificulta a criação de novos mecanismos para solucionar problemas é que novos tipos de rede e de aplicações surgem com uma frequência alta, trazendo novos desafios. As redes de nova geração, como as redes ad hoc [9, 10], as redes de sensores [3], as redes tolerantes a atrasos e desconexões [11, 12] e as redes veiculares [1] trazem novos desafios tanto na integração com o núcleo da Internet quanto no funcionamento próprio de cada rede. Outras aplicações também são responsáveis por grandes mudanças, como o uso de redes par-a-par [13] ou de computação em nuvem [14]. Uma nova Internet, ou a Internet do Futuro, deve ser flexível o suficiente para simplificar a criação, o teste e o uso de novas aplicações, ao mesmo tempo em que deve dar suporte a todas as redes de nova geração, tanto as que já são conhecidas quanto as que ainda estão por vir. Além disso, essa nova arquitetura da Internet deve considerar outros aspectos da sociedade contemporânea, como a preocupação com o baixo consumo de energia [15]. Com isso, os desafios nas redes de computadores são diversificados e devem ser solucionados ao longo das próximas décadas.

1.1 Motivação e objetivos

Essa tese aborda dois temas principais correlacionados com a Internet do Futuro. O primeiro tema tratado diz respeito às redes ad hoc [16–20], que são redes sem fio de nova geração com a característica de que todos os nós da rede funcionam como roteadores para obter um maior alcance. Assim, a rede funciona de forma colaborativa e pode ser formada sob demanda sem a necessidade de nenhum tipo de infraestrutura. Embora esse tipo de rede dê suporte a um grande número de aplicações com um baixo custo de implementação, essas redes apresentam diversos desafios no que diz respeito à autoconfiguração e à segurança. Em redes tradicionais, o uso de servidores é amplamente estimulado para configurar nós e para garantir a segurança para todos os usuários da rede. Nas redes ad hoc, não há como contar

com a presença de um servidor disponível a todo o momento para todos os nós para realizar funções básicas para os serviços da rede. Assim, os nós das redes ad hoc devem ser preparados para reagir autonomamente e se configurar independente da infraestrutura de rede disponível.

O segundo tema tratado nessa tese é a virtualização de redes, que ganhou maior ênfase com a introdução do conceito de pluralismo dentro da Internet. O conceito de pluralismo define a coexistência de diversas redes sobre o mesmo substrato físico, de forma a simplificar a inovação na rede. Dessa forma, diversas propostas para novas redes poderiam coexistir sobre a mesma infraestrutura, mesmo que fossem incompatíveis. Como exemplo, seria possível construir uma rede com suporte a IPv6, outra com suporte a IPv4, outra com amplo suporte ao *multicast*, todas virtualizadas e funcionando sobre os mesmos roteadores físicos da Internet, mas garantindo-se que uma não interfere nas demais. Assim, uma falha no *software* de uma determinada rede só teria influência no seu próprio ambiente, sem interferir no funcionamento das demais redes. Outro exemplo interessante seria a construção de pilhas de protocolos diferentes para cada tipo de rede de nova geração, como as redes ad hoc ou as redes tolerantes a atrasos e desconexões, permitindo que dispositivos com restrições de *hardware* e técnicas de encaminhamento diferentes do que é utilizado na Internet coexistam e possam se interconectar com nós em locais diferentes. Assim, a virtualização daria liberdade para a inovação através da integração de diferentes redes virtuais sobre a mesma rede física.

A idéia da virtualização dá amplo suporte a criação do ambiente pluralista, reforçando a idéia do "partindo do zero" (*clean-slate*) [21], no qual novas propostas para redes seriam feitas sem a necessidade de ter compatibilidade com os protocolos atuais. Outro ponto que favorece o desenvolvimento de técnicas de virtualização de rede, também correlacionado ao tema Internet do Futuro, é o desenvolvimento de redes de teste que permitam a experimentação em larga escala de novas propostas. Nesse sentido, diversos esforços internacionais foram propostos e vem sendo desenvolvidos para criar grandes redes para testes, que devem ser disponibilizadas para os pesquisadores em todo o mundo. Para permitir um uso mais eficiente dos equipamentos e um maior acesso dos pesquisadores, conta-se com a técnica de virtualização, de forma a permitir que cada pesquisador configure seu próprio conjunto de máquinas virtuais com seus experimentos, além de permitir que experimentos possam ser executados em paralelo, mas ainda sim, em um ambiente controlado [22, 23].

Um dos principais desafios relacionado à virtualização de redes é o desenvolvimento de ambientes virtuais isolados, ou seja, que uma rede virtual não interfira no funcionamento das demais [24–30]. O principal desafio no provimento do isolamento é o tratamento das operações de entrada e saída, que são muito frequentes em redes virtuais, mas não são adequadamente tratadas por hipervisores como o do Xen [31].

A seguir, são explicadas em detalhes as propostas apresentadas nessa tese, em cada um dos temas mencionados.

1.2 Redes ad hoc

As redes ad hoc são projetadas para ter topologias dinâmicas com múltiplos saltos compostas por enlaces sem fio com restrições de banda. Essas redes devem funcionar independentes de infraestrutura fixa ou administração centralizada, o que as faz atrativas para muitas aplicações, como o sensoriamento, o acesso à Internet em comunidades carentes, e recuperação de desastres. Por outro lado, os protocolos para redes ad hoc devem lidar com recursos limitados, como banda e energia, o meio compartilhado e as altas taxas de erro devido a propriedades do canal sem fio [32]. Além disso, partições são também prováveis nessas redes, pois a topologia da rede muda com frequência devido à mobilidade dos nós e às variações no enlace.

A natureza auto-organizável das redes ad hoc traz novos desafios para a atribuição de endereços IP para os nós da rede. Se um nó não possui endereço, ele se torna não operacional, pois a conectividade nas redes ad hoc atuais depende do uso do IP como identificador do nó. Para permitir uma formação espontânea da rede, a atribuição de endereços deve ser feita de forma autônoma. Mecanismos como o *Dynamic Host Configuration Protocol* (DHCP) entram em conflito com os conceitos das redes ad hoc, pois essas redes têm como objetivo configurar toda a rede automaticamente sem servidores e tratando a formação e a união de partições. Portanto, as redes ad hoc requerem mecanismos de autoconfiguração para alocar endereços.

Nessa tese, é descrita a proposta para uma abordagem eficiente para endereçamento em redes ad hoc chamada de *Filter-based Addressing Protocol* (FAP) [33, 34].[1] Os principais objetivos do FAP são:

- **autoconfigurar endereços** com base no conjunto atual de endereços alocados na rede, garantindo uma alocação única para cada nó;
- **utilizar estruturas compactas**, chamadas de filtro de endereços, para armazenar os endereços em uso na rede, e manter esses filtros sempre atualizados;
- **tratar a formação e a união de partições na rede**, garantindo a resolução de todos os conflitos de endereço resultantes da junção de dois conjuntos de endereço criados de forma independente ou não;
- **garantir uma baixa sobrecarga** de mensagens de controle;

[1] Esse tema foi inicialmente abordado, entre outros temas relacionados às redes ad hoc, na dissertação de mestrado [35]. O desenvolvimento da proposta e suas publicações foram realizados durante a tese de doutorado. Atualmente, a proposta se encontra em avaliação para publicação em revista.

- **garantir pequenos atrasos** na configuração e na resolução de conflitos de endereço.

Os filtros de endereço permitem que qualquer nó possa verificar se um endereço está disponível antes de tentar alocá-lo. Além disso, é proposto uso do *hash* do filtro, chamado de assinatura, como identificador de uma partição de rede. Esta idéia-chave se mostrou bem eficaz porque o identificador da rede fica atrelado ao conjunto de endereços alocados, ao invés de ser escolhido arbitrariamente. Se um nó entra na rede, o identificador é automaticamente modificado, o que facilita sobremaneira a detecção de uniões de partições. Se dois nós vizinhos possuem assinaturas de filtro diferentes, isso indica uma união de partições, que deve ser tratada para evitar a colisão de endereços. Por essas razões, os filtros de endereço reduzem a carga de controle e permitem uma acurada detecção de partições. Isso é importante, pois quanto maior a sobrecarga de controle, menor o tempo de vida da bateria dos nós móveis e menor a banda disponível para o envio de dados.

Os procedimentos propostos para o FAP garantem uma correta atualização dos estados em cada nó de forma distribuída e sem colisões de endereço. A análise matemática e as simulações realizadas mostram que o FAP apresenta baixa sobrecarga de comunicação e baixa latência, quando comparado a outras propostas da literatura, solucionando todas as colisões de endereço rapidamente.

1.3 Isolamento e qualidade de serviço em redes virtuais

A tecnologia de virtualização permite que a rede física seja dividida em fatias, chamadas de redes virtuais, cada qual com sua própria pilha de protocolos e esquema de endereçamento [36]. A virtualização, portanto, introduz mais flexibilidade no núcleo da rede, dando suporte à inovação. A implementação de redes virtuais requer, contudo, três principais características que não são providas pelas propostas que são focadas no controle do compartilhamento de recursos entre máquinas virtuais em *data centers* [29, 37]. Essas características são:

- **isolamento**, para garantir que redes virtuais hospedadas em um mesmo *hardware* físico não interfiram umas com as outras;
- **alto desempenho no encaminhamento de pacotes**, garantindo que roteadores virtuais são rentáveis, apresentando uma eficiência similar à dos roteadores físicos;

- **qualidade de serviço**, para compensar as restrições da camada de virtualização e dar incentivo ao desenvolvimento de novas aplicações com requisitos de banda e atraso [38, 39].

O provimento do isolamento entre redes virtuais depende de um compartilhamento de recursos justo entre os roteadores virtuais. As demandas por recursos como CPU, memória e banda dos roteadores virtuais, contudo, varia com o tempo, dificultando o provimento de requisitos estabelecidos nos acordos de nível de serviço (*Service Level Agreements* - SLAs). Isso impõe dois requisitos. Primeiramente, é necessário que as fatias de rede virtual se adaptem automaticamente às demandas de acordo com os SLAs. Em segundo, é necessário controlar o número de roteadores virtuais hospedados em uma mesma máquina física para limitar a probabilidade de que picos simultâneos por recursos violem os contratos. O provimento de uma divisão adaptativa eficiente dos recursos da rede e de um controle de admissão para roteadores virtuais ainda é um desafio, devido à dificuldade de fazer previsões de demanda em operações de entrada e saída e aos requisitos para uma utilização inteligente dos recursos físicos [29, 31, 37].

Para prover essas funcionalidades desafiadoras, essa tese descreve a proposta chamada *Virtual network Isolation, Policy Enforcement, and Resource sharing system* (VIPER) [24, 25, 40]. O VIPER provê meios para compartilhamento de recursos físicos baseado em um mecanismo adaptativo de divisão de recursos eficiente. Esses mecanismos garantem um forte isolamento entre as redes virtuais, respeitando os acordos de nível de serviço. O VIPER também dá suporte às opções para aumentar o desempenho no encaminhamento de pacotes, através da utilização de um plano de dados compartilhado, e estabelecendo primitivas de QoS para diferenciar o tráfego dentro das redes virtuais e entre as redes virtuais. A arquitetura do VIPER é constituída de dois blocos principais:

- **Gerenciador do Compartilhamento de Recursos -** monitora o tráfego de cada rede virtual, ajusta automaticamente os parâmetros de alocação de recursos e pune redes virtuais que violam SLAs. Assim, o gerenciador garante uma divisão correta dos recursos baseado na demanda de cada rede virtual. São propostos dois modelos para gerenciador de compartilhamento de recursos, sendo um para disponibilizar recursos sempre que houver demanda e recursos disponíveis, mesmo que a rede virtual esteja ultrapassando os limites estabelecidos nos SLAs, e outro para garantir o provimento de recursos exatamente como especificado nos SLAs, ainda que existam recursos ociosos.

- **Controlador de Acesso de Redes Virtuais -** verifica se existem recursos físicos suficientes para hospedar uma nova rede virtual sem restringir o desempenho de redes virtuais bem-comportadas hospedadas no mesmo *hardware*

físico. O algoritmo de controle de admissão provê uma acurada predição das demandas a longo prazo, o que permite evitar a sobrecarga do equipamento físico.

Foi desenvolvido um protótipo do VIPER na plataforma de virtualização Xen [31, 41]. Os resultados mostram que o VIPER garante alta conformidade entre o tráfego encaminhado e as características especificadas nos SLAs. Além disso, a proposta garante um uso mais eficiente do enlace quando comparada a soluções como a ferramenta *Traffic Control* (TC) [42], amplamente utilizada para esse tipo de operação. Os testes para verificar o isolamento e o provimento dos SLAs mostram que o VIPER é capaz de reduzir em até 18 vezes o atraso no encaminhamento de tráfego com requisitos de prioridade quando comparado a sistemas sem suporte a QoS no provedor de infraestrutura. Além disso, o sistema tem um desempenho até cinco vezes melhor que outras ferramentas no provimento de SLAs. Também foram realizadas simulações com diferentes padrões de tráfego para analisar o controlador de admissão de novas redes virtuais. Devido às técnicas introduzidas para estimar os aumentos de demanda, o VIPER é capaz de aumentar a eficiência na utilização dos recursos físicos quando comparado a outras técnicas de controle de acesso [37, 43].

De forma resumida, essa tese traz as seguintes contribuições chaves em redes virtuais:

1. a proposta e o desenvolvimento de um gerenciador de recursos eficiente que garante o isolamento e a provisão de SLAs para redes virtuais;

2. a introdução de novas técnicas para estimar o uso de recursos a longo prazo que garantem um acurado controle de admissão de redes virtuais;

3. o provimento de primitivas de QoS em ambientes virtualizados.

1.4 Organização do trabalho

A tese está organizada da seguinte forma. No Capítulo 2 são discutidos os principais aspectos relacionados a propostas de endereçamento em redes ad hoc. Neste capítulo, o FAP é descrito em detalhes. Em seguida, no Capítulo 3, é feita uma análise sobre algumas das principais propostas de endereçamento em redes ad hoc, incluindo o FAP. São apresentadas análises matemáticas sobre falhas e sobrecarga, além de resultados de simulação para verificar sobrecarga, atrasos e eficiência na detecção e resolução de conflitos de endereço. No final deste capítulo, é apresentada uma discussão sobre os principais aspectos observados com relação à proposta e à análise realizada.

No Capítulo 4 é apresentado mais em detalhes o conceito de virtualização de rede e são discutidas as principais técnicas de virtualização. Em seguida, são descritas as principais propostas para o controle do compartilhamento de recursos em redes virtualizadas. No Capítulo 5 são descritos os objetivos e a arquitetura detalhada da proposta VIPER, explicando o seu uso no Xen. Nesse capítulo, são apresentadas as técnicas de monitoramento utilizadas para fazer o controle do compartilhamento de recursos. Além disso, também são apresentados os dois modelos para o gerenciador de compartilhamento de recursos, além do controlador de admissão de novos roteadores virtuais. Em seguida, o Capítulo 6 apresenta os principais resultados obtidos com os gerenciadores e o controlador propostos, além de algumas considerações relativas aos resultados obtidos com as propostas para controle do compartilhamento de recursos em redes virtualizadas. Esse capítulo também apresenta algumas considerações para a adaptação do VIPER para o uso em outras plataformas de virtualização.

Por fim, o Capítulo 7 conclui a tese, ressaltando os temas tratados e os objetivos alcançados com as propostas. As principais vantagens e desvantagens dos mecanismos propostos são discutidas, assim como alguns trabalhos futuros relacionados a esses temas.

Parte I

Autoconfiguração de endereços em redes ad hoc

Capítulo 2

Propostas de endereçamento distribuído para redes ad hoc

A ausência de servidores impede o uso de esquemas de endereçamento centralizados em redes ad hoc. Em esquemas distribuídos simples, contudo, é difícil evitar endereços duplicados, pois uma escolha aleatória de endereços resultaria em uma alta probabilidade de colisão, como demonstrado pelo paradoxo do aniversário [44]. Supondo uma escolha de endereço aleatória, a probabilidade de existir pelo menos uma colisão de endereços em um conjunto com n nós, $P(E_c)$, é determinada pelo espaço de endereçamento disponível, r, e pelo número de nós disputando os endereços pertencentes a este espaço, n. A probabilidade de não ocorrer colisão é dada por

$$P_n(E_c) = \frac{r-1}{r} \cdot \frac{r-2}{r} \cdots \frac{r-n+1}{r} = \frac{r!}{r^n \cdot (r-n)!}. \tag{2.1}$$

Portanto, a probabilidade de existir pelo menos uma colisão de endereços é igual a

$$P(E_c) = 1 - P_n(E_c) = 1 - \frac{r!}{r^n \cdot (r-n)!}. \tag{2.2}$$

Usando a expansão da série de Taylor, dada por

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots, \tag{2.3}$$

então, pode-se concluir que

$$P(E_c) > 1 - e^{\frac{-(n(n-1))}{2r}}. \tag{2.4}$$

A Figura 2.1 apresenta o limite inferior para $P(E_c)$. Uma vez que a probabilidade de colisão é alta, fica claro que a escolha aleatória não é suficiente para garantir uma

alocação única de endereço para cada nó.

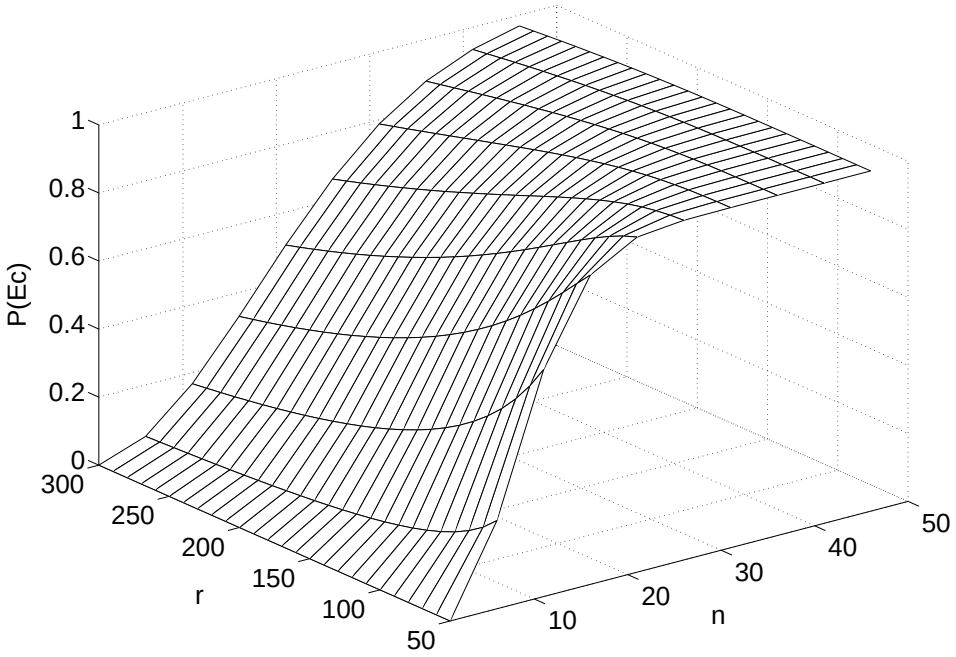


Figura 2.1: Limite inferior para a probabilidade de colisão de endereços.

O grupo de trabalho Zeroconf do *Internet Engineering Task Force* (IETF) propõe um esquema de endereçamento baseado em *hardware* [45], o qual atribui um endereço de rede IPv6 a cada nó baseado no endereço MAC (*Medium Access Control*) do dispositivo. Esse esquema, no entanto, apresenta problemas de privacidade. O uso do endereço MAC como o sufixo do endereço de rede implica em um identificador de interface único, o qual permite a identificação e a correlação de diferentes atividades de um usuário por um *sniffer* posicionado estrategicamente na rede [46]. Assim, a captura de um pacote indica quem é o usuário, em que rede ele está conectado e qual serviço ele está usando, pois o sufixo do IP identifica unicamente o usuário e o prefixo identifica a rede de acesso, e consequentemente, a posição do usuário. Se um usuário acessa, por exemplo, o seu e-mail diversas vezes por dia, o provedor do serviço estaria apto a identificar a trajetória do usuário ao longo do dia. Outro problema é que o esquema de endereçamento baseado em MAC é estático e não pode ser empregado quando o número de dispositivos físicos que pode acessar a rede é maior que o número de endereços disponível para alocação, ou seja, se o número de bits no sufixo do IP for menor que o número de bits do endereço MAC. Apenas a alocação dinâmica de endereços poderia resolver esse problema. Outro problema correlato é que o endereço MAC é maior que o endereço IPv4. Assim, o uso de IPv4 com esse tipo de proposta de endereçamento requer que o endereço MAC seja

resumido por meio de uma função *hash*, o que é equivalente a uma escolha aleatória em termos de probabilidade de colisão de endereços. Além disso, outro problema que surge com a identificação baseada em MAC é que não é possível garantir que todas as interfaces de rede de todos os dispositivos possuem um identificador MAC único e é muito simples para um usuário trocar o endereço de MAC da sua interface. Ambas as situações poderiam implicar em colisões de endereço na rede.

Propostas para autoconfigurar endereços em redes ad hoc sem armazenar a lista de endereços alocados são normalmente baseadas em um protocolo distribuído chamado de *Duplicate Address Detection* (DAD) [47]. Neste protocolo, cada novo nó escolhe aleatoriamente um endereço e inunda a rede certo número de vezes com uma mensagem de solicitação de endereço (*Address Request message* - AREQ) para garantir que todos os nós irão receber o novo endereço alocado. Se outro nó já está usando o mesmo endereço, ele envia uma mensagem de resposta de endereço (*Address Reply message* - AREP) para o novo nó. Quando o nó de ingresso recebe um AREP, ele escolhe aleatoriamente outro endereço e repete o processo de inundação. Caso contrário, ele aloca o endereço escolhido. A garantia da unicidade de endereço se apóia na premissa que todos os nós da rede são informados, através da inundação, da alocação de um determinado endereço. Esta proposta, portanto, não leva em conta as partições da rede e, como consequência, não se encaixa bem para redes ad hoc. Algumas extensões para o DAD utilizam mensagens *Hello* e identificadores de partição para lidar com as partições de rede [48, 49]. Esses identificadores são números aleatórios que identificam cada partição da rede. Um grupo de nós muda o seu identificador de partição sempre que identifica uma partição ou quando duas ou mais partições se unem. Fan e Subramani [48] propuseram um protocolo baseado no DAD para resolver colisões de endereço na presença de eventos de união de partições de redes. Este protocolo considera que duas partições estão se unindo quando um nó recebe uma mensagem *Hello* com um identificador de partição diferente do seu próprio identificador, ou quando o conjunto de vizinhos do nó sofre alguma alteração [48]. Fazio *et al.* [49] também propuseram um protocolo com base em identificadores de partição, mas que funciona de forma reativa. Em vez de enviar AREQs e AREPs a cada vez que um nó se junta à rede ou quando uma partição é identificada, o protocolo identifica as colisões apenas quando uma troca de dados é requisitada. Apesar de esse esquema reduzir o número de mensagens de controle periódicas, o uso de mecanismos reativos causa um atraso na transmissão de dados. Além disso, este protocolo só pode ser aplicado quando se utiliza um protocolo de roteamento reativo, porque, nos protocolos pró-ativos, as rotas são calculadas antes de serem necessárias e uma colisão de endereço poderia causar uma má escolha de rota.

Outras propostas também utilizam as informações de roteamento para solucionar

o problema do endereçamento [50, 51]. O *Weak* DAD [52], por exemplo, consegue rotear pacotes corretamente mesmo se houver uma colisão endereço. Neste protocolo, cada nó é identificado por seu endereço e uma chave. O DAD é executado na vizinhança de um salto de cada nó e outras colisões são identificadas com base em informações de roteamento. Cabe observar que, se alguns nós escolherem o mesmo endereço e a mesma chave, a colisão não é detectada. Além disso, o *Weak* DAD depende de modificações nos protocolos de roteamento para funcionar corretamente.

Outros protocolos mais complexos foram propostos para melhorar o desempenho da rede na detecção de uniões de partições e na realocação de endereços [53, 54]. Nestes protocolos, os nós armazenam estruturas de dados adicionais para executar o protocolo de endereçamento. O MANETconf é um protocolo com estados baseado nos conceitos de exclusão mútua do algoritmo de Ricart-Agrawala [54]. Nesse protocolo, os nós armazenam duas listas de endereços: a lista de endereços atribuídos (*Allocated list*) e a lista de endereços alocados pendentes (*Allocated Pending list*). No MANETconf, um novo nó requisita um endereço para algum vizinho, o qual se torna um líder no procedimento de alocação de endereço. O líder, então, seleciona um endereço disponível, o armazena na lista de endereços alocados pendentes e inunda a rede com essa informação. Cada nó recebe essa mensagem inundada e verifica se o endereço requisitado pertence a alguma lista. Se o endereço está na lista de endereços atribuídos, o nó recusa a alocação do endereço. Se o endereço está na lista de endereços alocados pendentes, então existem dois procedimentos concorrentes tentando alocar o mesmo endereço. Se o endereço do líder atual é maior que o endereço do outro líder que está tentando alocar o mesmo endereço, o nó responde positivamente ao pedido de alocação. Caso contrário, o pedido é negado. No caso de o endereço estar fora das duas listas, o nó aceita a requisição, envia uma resposta positiva para o líder e insere na lista de endereços alocados pendentes o novo endereço e o endereço do líder. Se todos os nós aceitarem o pedido de alocação e responderem positivamente ao líder, então o líder informa o endereço alocado ao novo nó, move o endereço da lista de endereços alocados pendentes para a lista de endereços atribuídos e inunda a rede novamente para confirmar a alocação do endereço. Após receber essa mensagem, cada nó move o endereço da lista de endereços alocados pendentes para a lista de endereços atribuídos. O MANETconf trata a realocação de endereços após a saída de nós, mas a detecção de partição depende de inundação periódica. Dessa forma, esse protocolo implica uma alta sobrecarga de mensagens.

Outro protocolo de endereçamento para redes ad hoc baseado no armazenamento de estados é o *Dynamic Address assignment Protocol* (DAP) [53], que é baseado na atribuição de conjuntos de endereços disponíveis para cada nó, em mensagens *Hello*, e em identificadores de partição. No DAP, um nó divide o seu conjunto de endereços

disponíveis com um novo nó sempre que o novo nó lhe pedir um endereço. Quando um nó possui um conjunto de endereço vazio, pois já atribuiu todos os endereços disponíveis para novos nós, ele pede uma redistribuição dos endereços disponíveis. Com isso, todos os nós devem reorganizar os seus conjuntos de endereços disponíveis, de forma que todos os nós possam ter endereços disponíveis e aceitar novos nós na rede. Esta realocação e a detecção que um determinado endereço não está mais sendo usado devido à saída de um nó podem causar uma alta sobrecarga de controle na rede, dependendo da forma como os endereços estão distribuídos entre os nós. Apesar dessas questões, pode-se dizer que o principal problema desta proposta é a união de partições, que é baseada no uso de identificadores de partição escolhidos como números aleatórios. Estes identificadores não dão nenhuma informação sobre o conjunto atual de nós em cada partição e, portanto, não são uma boa forma para identificar as partições. Por essa razão o DAP requer o uso do DAD em cada evento de união de partições, não só para os endereços alocados, mas também para a lista de endereços disponíveis armazenada em cada nó, o que aumenta muito a carga de controle. Outro problema do DAP é que essas listas de endereço disponíveis não são sempre seqüenciais, podendo exigir uma capacidade de armazenamento alta.

O *Prophet* é um protocolo que aloca endereços baseado em uma função pseudo-aleatória com alta entropia [55]. O primeiro nó da rede, chamado de profeta, escolhe uma semente para uma sequência aleatória e distribui endereços para os novos nós que o contatam. Por sua vez, cada novo nó que entra na rede também pode receber outros nós e alocar endereços. Para tanto, esse novo nó deve começar a distribuir os novos endereços a partir de um ponto diferente da sequência aleatória, construindo uma árvore de distribuição de endereços. A principal vantagem do *Prophet* é a baixa sobrecarga de controle, uma vez que o protocolo não depende de inundações na rede. Como desvantagens, esse protocolo depende de um espaço de endereçamento muito maior do que os outros protocolos apresentados para dar suporte ao mesmo número de nós na rede. Além disso, o desempenho do *Prophet* para evitar colisões depende da qualidade do gerador pseudo-aleatório. Para evitar esse problema, algum mecanismo como o DAD deveria ser usado para detectar a duplicação de endereços, o que também aumentaria a complexidade do protocolo e eliminaria a vantagem de possuir uma baixa sobrecarga de controle.

A proposta apresentada nessa tese, chamada de *Filter-based Addressing Protocol* (FAP), tem por objetivo reduzir a carga e controle e melhorar a detecção de eventos de união de partições sem necessitar de uma alta capacidade de armazenamento. Esses objetivos são alcançados através do uso de pequenos filtros e de um mecanismo distribuído acurado para atualizar os estados nos nós. São utilizadas assinaturas de filtros, ou seja, o *hash* do filtro, como identificadores de partição ao invés de números aleatórios. Enquanto o número aleatório serve apenas para o propósito de identificar

a partição, a assinatura do filtro tem o duplo propósito de identificar a partição ao mesmo tempo em que representa os nós que compõem a partição. Dessa forma, se o conjunto de endereços alocados muda, a assinatura do filtro também muda. Assim, uso de assinaturas de filtros aumenta a capacidade de se detectar e unir partições corretamente em redes ad hoc, o que não ocorre com os identificadores aleatórios.

2.1 *Filter-based Addressing Protocol* (FAP)

O protocolo proposto tem por objetivo autoconfigurar dinamicamente os endereços de rede, solucionando colisões com uma baixa carga de controle, mesmo em situações de entradas de nós e uniões de partições na rede. Para alcançar esses objetivos, o FAP utiliza um filtro compacto distribuído para representar todos os endereços alocados. Cada nó possui um filtro para simplificar os frequentes eventos de entrada de nós e para reduzir a sobrecarga de controle ao resolver as colisões de endereços que são inerentes à escolha aleatória de um endereço. Além disso, o uso do *hash* do filtro, também chamado de assinatura do filtro, é proposto para simplificar a detecção de partições.

No FAP, é proposto o uso de dois tipos de filtro, a serem escolhidos de acordo com o cenário: o filtro de Bloom, o qual é baseado em funções *hash*, e o filtro de sequência, proposto com o FAP, o qual comprime os dados com base na sequência de endereços.

2.1.1 Filtros de Bloom

O filtro de Bloom é uma estrutura de dados compacta usada em aplicações distribuídas como o rastreamento de IP [56] e *cache* na *Web* [57]. O filtro de Bloom é composto por um vetor de m bits que representa um conjunto $A = \{a_1, a_2, a_3, \ldots, a_n\}$ composto por n elementos. Os elementos são inseridos no filtro através de um conjunto de funções *hash* independentes, h_1, h_2, $\ldots$, h_k, cuja saída está uniformemente distribuída sobre m bits.

Para inserir elementos no filtro, primeiramente, todos os bits do vetor são zerados. Em seguida, cada elemento $a_i \in A$ é aplicado em cada uma das k funções *hashes*, cujas saídas representam uma posição a ser setada como 1 no vetor de m bits, como mostrado na Figura 2.2. Para verificar se um elemento a_j pertence à A, é necessário verificar se os bits do vetor correspondente às posições $h_1(a_j), h_2(a_j), \ldots, h_k(a_j)$ estão todos setados como 1. Se pelo menos um desses bits estiver setado como 0, então o elemento a_j não pertence ao filtro. Caso contrário, é assumido que o elemento pertence à A. Existe, contudo, uma probabilidade de falso positivo, na qual um elemento $a_j \notin A$ é reconhecido como parte de A. Isso pode ocorrer se os

bits na posição $h_1(a_j), h_2(a_j), \ldots, h_k(a_j)$ tiverem sido setados como 1 por elementos inseridos previamente.

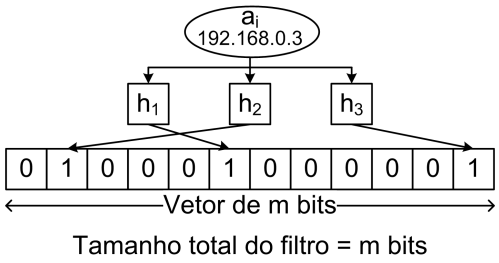


Figura 2.2: Procedimento de inserção de um elemento de endereço a_i em um filtro de Bloom com $k = 3$ funções *hash* e tamanho $m = 12$ bits.

Uma vez que a probabilidade que um bit seja 0 após a inserção de n elementos, P_0, é expressa por

$$P_0 = \left(1 - \frac{1}{m}\right)^{kn}, \tag{2.5}$$

a probabilidade de falsos positivos, P_{fp}, vale então

$$P_{fp} = (1 - P_0)^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k. \tag{2.6}$$

Assim, a Equação 2.6 mostra que os falsos positivos do filtro de Bloom decrescem se o número de elementos, n, do conjunto A é reduzido ou se o tamanho do filtro, m, é aumentado. Além disso, igualando a derivada da Equação 2.6 a zero, obtém-se o valor de k que minimiza a probabilidade de falsos positivos, o qual é dado por

$$k = \left\lceil \frac{m \cdot \ln 2}{n} \right\rceil. \tag{2.7}$$

Quando é necessário remover elementos do filtro, é necessário fazer uma modificação na estrutura do filtro. Assim, cada bit do filtro é trocado por um contador, como mostrado na Figura 2.3. Cada contador, c_i, indica o número de elementos a_i que selecionaram aquele bit para ser setado como 1. A probabilidade que um contador com tamanho de b bits sofra um transbordamento, ou seja, que o contador chegue a um valor superior a 2^b, é estimada pela probabilidade de um contador ser setado de 2^b até kn vezes após a inserção de n elementos. Assim, a probabilidade de transbordamento respeita a inequação

$$P(c_i \geq 2^b) < \sum_{i=2^b}^{kn} \binom{kn}{i} \left(\frac{1}{m}\right)^i \left(1 - \frac{1}{m}\right)^{kn-i}. \tag{2.8}$$

Usando a aproximação da binomial para uma Poisson e assumindo que o primeiro

termo da equação ($i = 2^b$) é predominante, então

$$P(c_i \geq 2^b) < \frac{e^{\frac{-kn}{m}}(\frac{kn}{m})^{2^b}}{2^b!}. \tag{2.9}$$

Para evitar os falsos positivos, é preciso garantir que os contadores não sofrerão um transbordamento. Assumindo um contador com 4 bits, ou seja, o valor máximo do contador é dado por $2^b = 2^4 = 16$, e o uso do valor ideal de k, dado pela Equação 2.7, então a probabilidade de transbordamento não depende do número de células de contadores no filtro, m, nem do número de elementos no filtro, n, e é expressa por

$$P(c_i \geq 16) < (e^{-\ln 2} \cdot (\ln 2)^{2^b})/((2^b)!) \approx 6.8 \cdot 10^{-17}. \tag{2.10}$$

Dessa forma, a probabilidade de transbordamento é desprezível, assumindo contadores com 4 bits.

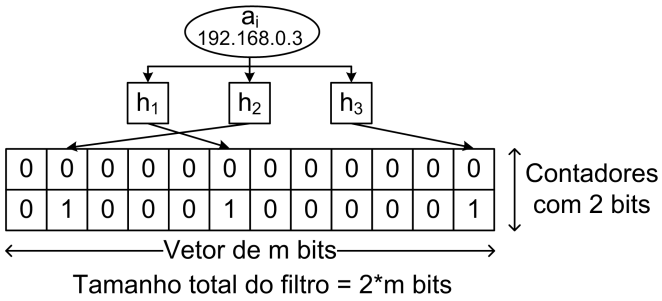


Figura 2.3: Procedimento de inserção de um endereço a_i em um filtro de Bloom com contadores de 2 bits, assumindo o uso de $k = 3$ funções *hash*.

2.1.2 Filtros de sequência

Além do uso de filtros de Bloom, o FAP também dá suporte ao uso de uma estrutura proposta chamada de filtro de sequência, a qual armazena de forma compacta uma sequência de endereços. Esse filtro é criado pela concatenação do primeiro endereço da sequência de endereços, chamado de elemento inicial (a_0), com um vetor de r bits, onde r é o número de endereços disponíveis para serem alocados. Dessa forma, sendo b_{suf} o número de bits no sufixo do endereço, então $r = 2^{b_{suf}}$. Nesse filtro, cada sufixo de endereço é representado por um bit, indexado por δ, que indica a distância entre o sufixo do elemento, $a_{i_{suf}}$, e o sufixo do elemento inicial, $a_{0_{suf}}$, ou seja,

$$\delta = a_{i_{suf}} - a_{0_{suf}}. \tag{2.11}$$

Se um bit do filtro vale 1, então o endereço relacionado a esse sufixo está inserido no filtro. Se o bit vale 0, então o elemento não pertence ao filtro. Portanto, não há falsos positivos ou falsos negativos no filtro de sequência, porque cada endereço disponível é representado de forma determinística no filtro. O filtro de sequência e o procedimento para inserir elementos no filtro estão ilustrados na Figura 2.4.

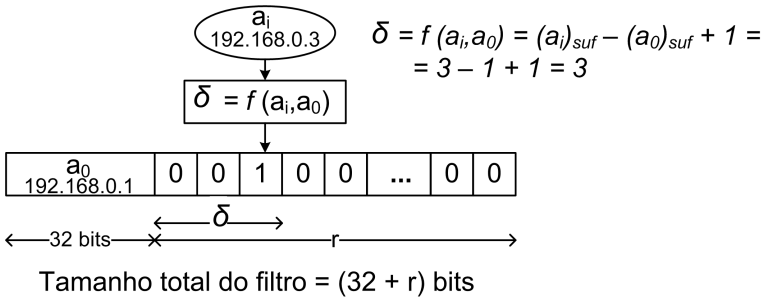


Figura 2.4: Procedimento de inserção do endereço a_i = 192.168.0.3 no filtro de sequência, assumindo uma faixa de endereços que vai de a_0 = 192.168.0.1 até a_{r-1} = 192.168.0.254.

2.1.3 Filtro de Bloom × filtro de sequência

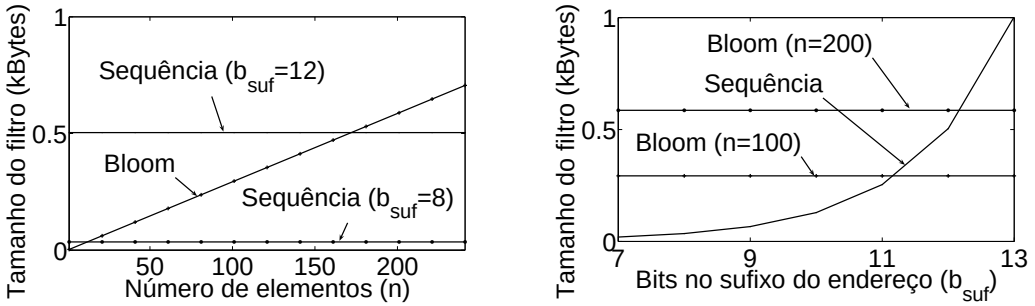
Os filtros de Bloom não possuem falsos negativos, o que significa que uma verificação de presença no filtro de um elemento que foi inserido no filtro sempre será positiva[1]. Esses filtros, contudo, apresentam uma probabilidade de falso positivo. Assim, uma verificação de presença no filtro de certo elemento pode ser positiva mesmo que o elemento não tenha sido inserido no filtro. Com isso, ao se utilizar filtros de Bloom, é preciso escolher um limiar superior para os falsos positivos levando em consideração o número máximo de elementos que podem ser inseridos no filtro. Por exemplo, caso seja escolhida uma probabilidade de falsos positivos máxima de $\approx 0,06$, assumindo que $m \gg k$ e que o valor máximo do contador é 15 para evitar transbordamentos, então a razão entre o número de células no filtro, m, e o número máximo de elementos a serem inseridos, n, vale $m/n = 6$ e o número ideal de funções *hash* vale 4, de acordo com as Equações 2.6 e 2.7. Assim, o tamanho s de um filtro de Bloom com contadores de quatro bits ($b = 4$) e probabilidade de falsos positivos máxima de aproximadamente 6% é $s = (m/n) \cdot n \cdot b = 6 \cdot n \cdot 4$. Para manter uma taxa de falsos positivos fixa, quanto maior for o número de elementos a serem inseridos no filtro, n, maior deve ser o número de células, m, no filtro[2]. Dessa forma, o tamanho do filtro de Bloom não é determinado pelo espaço de endereços, mas pelo número máximo de elementos a serem inseridos no filtro, o qual é uma estimativa do limite

[1]Assumindo-se uma escolha de parâmetros para os contadores que garanta que não acontecerá transbordamento quando filtros de Bloom com contadores forem utilizados.

[2]Uma vez que o tamanho do filtro não é variável no FAP, deve ser feita uma estimativa do número máximo de nós que deve estar na rede ao mesmo tempo a priori.

superior do número de nós ativos na rede. Por outro lado, o filtro de sequência é determinístico e, como consequência, não existem falsos negativos ou falsos positivos. Além disso, é mais fácil verificar se um elemento já foi inserido ou não no filtro de sequência do que no filtro de Bloom. Para verificar se um elemento pertence a um filtro de Bloom, é necessário calcular k funções *hash*, enquanto que a mesma operação no filtro de sequência implica em apenas uma subtração. O tamanho do filtro de sequência depende apenas do número de endereços disponíveis para serem alocados, r, e do tamanho do endereço, A_s. Assim, o número de bits no filtro de sequência é dado por $s = A_s + r$.

A Figura 2.5 mostra ambos os filtros, assumindo endereços de 32 bits, uma probabilidade de falsos positivos de $\approx 0,06$ para o filtro de Bloom e o uso do número de funções *hash*, k, ideal, o que significa que $m/n = 6$ e $k = 4$. A figura mostra que o tamanho do filtro de Bloom é constante independente do tamanho do espaço de endereços e cresce se o número de elementos a serem inseridos aumentar. Já o filtro de sequência funciona de maneira contrária, pois o filtro não depende do número de elementos e aumenta se o espaço de endereços aumentar. Como resultado, o filtro de Bloom é mais adequado para ambientes onde o espaço de endereços é muito grande, mas o número de nós é pequeno, enquanto que o filtro de sequência é adequado para o uso em redes onde o número de endereços é pequeno, mas o número de nós ativos é alto.



(a) De acordo com o número de nós.

(b) De acordo com o número de bits no sufixo do endereço.

Figura 2.5: Tamanho mínimo para o filtro, de acordo com os endereços disponíveis e a estimativa do número máximo de nós ativos na rede.

2.1.4 Procedimentos do FAP

A seguir, são descritos alguns dos principais procedimentos do FAP e é mostrado como os filtros são utilizados pelo protocolo.

Quando um nó entra na rede, ele requisita um filtro de endereços a um vizinho, escolhe aleatoriamente um endereço e verifica se esse endereço está alocado no filtro

recebido. Caso esteja alocado, o novo nó escolhe aleatoriamente um novo endereço. Caso contrário, ele aloca o endereço escolhido para si próprio e inunda a rede com essa informação. Enquanto nos protocolos baseados no protocolo DAD [47] um nó deve inundar a rede toda vez que recebe uma mensagem *Address Reply* indicando uma colisão, no protocolo proposto um novo nó inunda a rede apenas uma vez, pois o endereço escolhido não será uma colisão devido à utilização do filtro.

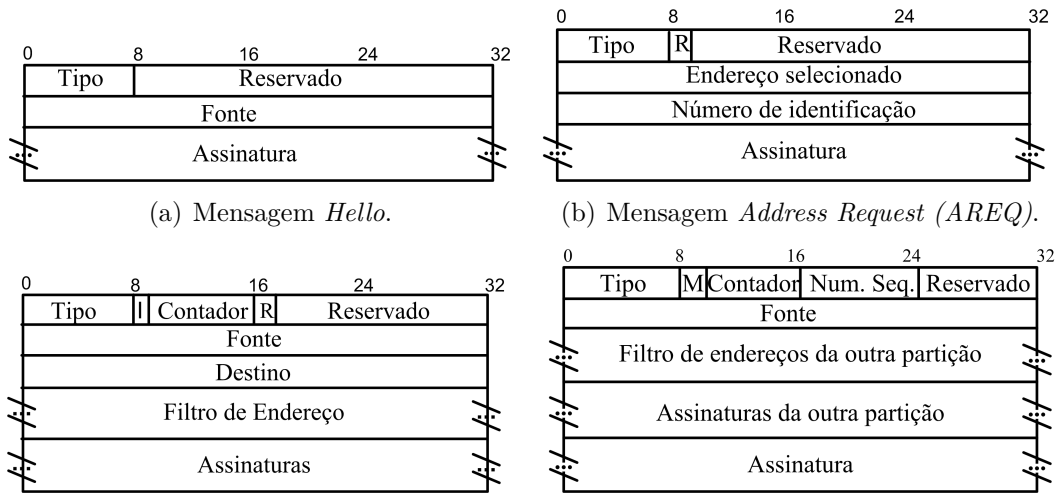
O uso da assinatura do filtro no FAP simplifica a detecção de partições, porque assinaturas de filtro diferentes indicam conjuntos diferentes de nós e, consequentemente, diferentes partições da rede. Além disso, em procedimentos de união de partições, os nós podem estimar o tamanho de cada partição com os filtros e apenas os nós com endereços colididos que estão na menor partição devem trocar os seus endereços. Nos protocolos baseados no DAD, contudo, todos os nós precisam verificar se o seu endereço está colidido após uma união de partições.

Consequentemente, o FAP aumenta a eficiência da alocação de endereços, reduz a carga de controle e simplifica a detecção de partições na rede. A seguir, os procedimentos de inicialização da rede, entrada de nós e união de partições são descritos com mais detalhes.

Inicialização da rede

O procedimento de inicialização da rede trata de como o conjunto inicial de nós deve se autoconfigurar. Dois cenários diferentes podem acontecer durante a inicialização: os novos nós chegam um após o outro, com um longo intervalo entre eles, o que é chamado de inicialização gradual da rede; no outro cenário, vários nós entram na rede aproximadamente ao mesmo tempo, o que é chamado de inicialização abrupta. A maioria dos protocolos assume apenas a existência do cenário gradual com um intervalo longo entre a chegada do primeiro e do segundo nó. Por exemplo, o protocolo proposto por Fan e Subramani [48] assume que o primeiro nó está só e pode escolher o identificador de partição da rede. Em seguida, os demais nós entram na rede com o auxílio do primeiro nó e dos demais nós que já entraram. Contudo, os protocolos de endereçamento deveriam funcionar independentemente de como os nós entram na rede.

O protocolo proposto nesta tese funciona com eficiência tanto na inicialização gradual quanto na inicialização abrupta, utilizando mensagens *Hello* e *Address Request* (AREQ), mostradas nas Figuras 2.6(a) e 2.6(b). A mensagem *Hello* é usada por um nó para anunciar o seu estado de associação e o seu identificador de partição. A mensagem AREQ é usada para anunciar que um endereço que estava livre está sendo alocado. Cada AREQ possui um número de identificação escolhido aleatoriamente, o qual é usado para diferenciar mensagens AREQ geradas por nós diferentes, mas com o mesmo endereço.



(a) Mensagem *Hello*.

(b) Mensagem *Address Request (AREQ)*.

(c) Mensagem *Address Filter* (AF).

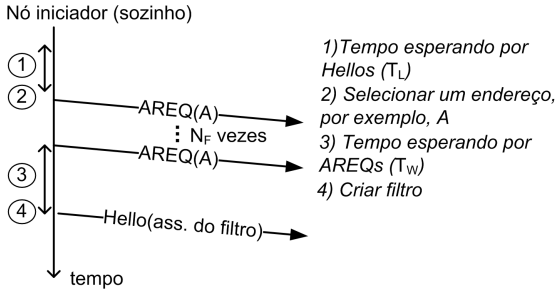
(d) Mensagem *Partition*.

Figura 2.6: Mensagens do FAP.

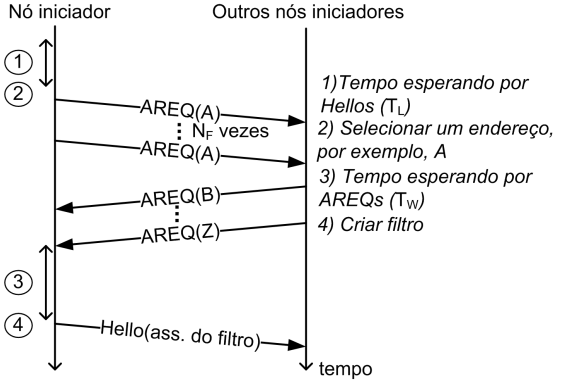
No FAP, um nó que deseja entrar na rede primeiro ouve o meio por um determinado período de tempo T_L. Se o nó não receber nenhuma mensagem *Hello* nesse período, então ele conclui que a rede não está formada e inicia o procedimento de inicialização da rede, tornando-se um nó iniciador. Um iniciador pode começar a rede sozinho, como descrito na Figura 2.7(a), ou com outros nós iniciadores, como descrito na Figura 2.7(b) e 2.7(c). Caso o nó receba uma mensagem *Hello*, então a rede já está formada, e o nó passa a agir como um novo nó, como será mostrado mais adiante.

O estabelecimento de um prefixo de rede é um problema em aberto para todas as propostas de endereçamento para as redes ad hoc que podem acessar a Internet ou alguma outra rede. Uma vez que não é possível garantir que o *gateway* está disponível para todos os nós em todos os momentos, o *gateway* não pode ser usado como um servidor de autoconfiguração e o prefixo da rede deve ser anunciado de alguma outra forma. De fato, existem duas possibilidades principais para configurar um prefixo. Na primeira, todos os nós conhecem o prefixo a priori. Na segunda, o primeiro nó da rede escolheria um prefixo e anunciaria essa informação para todos os outros nós que entrassem na rede. No caso de múltiplos nós entrarem na rede ao mesmo tempo, ou se duas partições se formam de maneira independente e se unem posteriormente, então seria necessário algum mecanismo para garantir que todos os nós chegassem a um mesmo acordo sobre o prefixo do endereço da rede através de algum mecanismo específico. Daqui em diante, assume-se que os nós estão aptos a decidir por um único prefixo de rede.

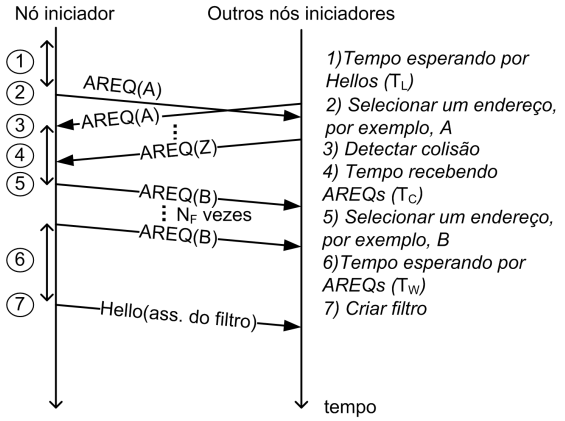
O nó iniciador cria um filtro de endereços em branco e inicia a fase de inicialização da rede. Nessa fase, o nó inunda a rede N_F vezes com mensagens AREQ,



(a) Inicialização gradual.



(b) Inicialização abrupta, considerando que não acontecem colisões.



(c) Inicialização abrupta, considerando que acontecem colisões.

Figura 2.7: Inicialização da rede com um nó (inicialização gradual) e com vários nós (inicialização abrupta) no FAP.

para aumentar a probabilidade de que todos os nós iniciadores recebam a mensagem, como indicado na Figura 2.7(a). Assim, ainda que ocorram colisões em uma das inundações, as demais inundações funcionam como redundância, aumentando a probabilidade de entrega da mensagem a todos os nós da rede. Se existem outros nós iniciadores, como na Figura 2.7(b), eles também devem enviar a mensagem AREQ N_F vezes, anunciando o endereço que foi escolhido aleatoriamente. Após esperar um período T_W sem escutar nenhum AREQ de outros nós iniciadores, caso eles existam, o nó iniciador sai da fase de inicialização e insere o seu endereço e todos os endereços recebidos em AREQs no filtro de endereços. Em seguida, o nó começa a enviar mensagens *Hello* com a assinatura do filtro de endereços, que corresponde ao *hash* do filtro. A assinatura do filtro identifica a rede e é usada para detectar partições caso elas ocorram. Se um nó iniciador recebe algum AREQ com o mesmo endereço que ele escolheu, como mostrado na Figura 2.7(c), mas com um número de identificação diferente daquele que foi escolhido pelo nó, ele detecta que o endereço escolhido foi alocado por algum outro nó iniciador, criando uma colisão. Nesse caso, o nó espera por um período T_C e, então, escolhe outro endereço e envia um novo AREQ. Durante o período T_C, o nó recebe mais AREQs com outros endereços já

alocados. Com isso, após T_C, o nó iniciador conhece uma lista mais completa dos endereços já alocados na rede, o que reduz a probabilidade desse nó escolher um novo endereço que também já tenha sido alocado. Assim, o período T_C reduz a probabilidade de colisão e, consequentemente, reduz a carga de controle na rede.

Após a fase de inicialização do FAP, todos os nós iniciadores possuem um endereço único, devido à escolha aleatória e a validação com AREQs com números de identificação. Além disso, após essa fase, todos os nós têm uma alta probabilidade de conhecer todos os endereços alocados, devido às N_F inundações da rede. Consequentemente, cada nó também cria um filtro de endereços contendo todos os endereços alocados.

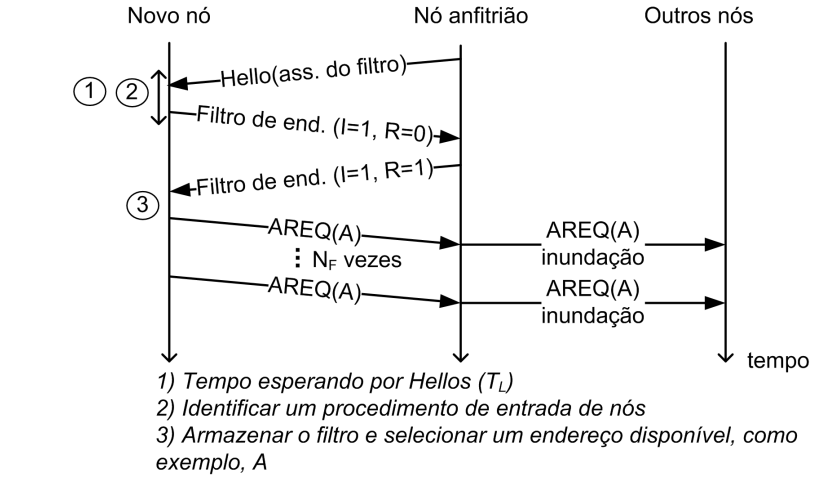
Entrada de nós na rede e união de partições

Após a inicialização, cada nó começa a enviar em *broadcast* mensagens *Hello* periódicas contendo a assinatura do filtro de endereços. Após a recepção de uma mensagem *Hello*, os vizinhos analisam se a assinatura contida na mensagem é igual a sua própria assinatura de filtro para detectar eventos de união de partições. Apenas nós que já entraram na rede estão aptos para enviar mensagens *Hello*, para receber um pedido de um novo nó para entrar na rede e para detectar uniões de partições.

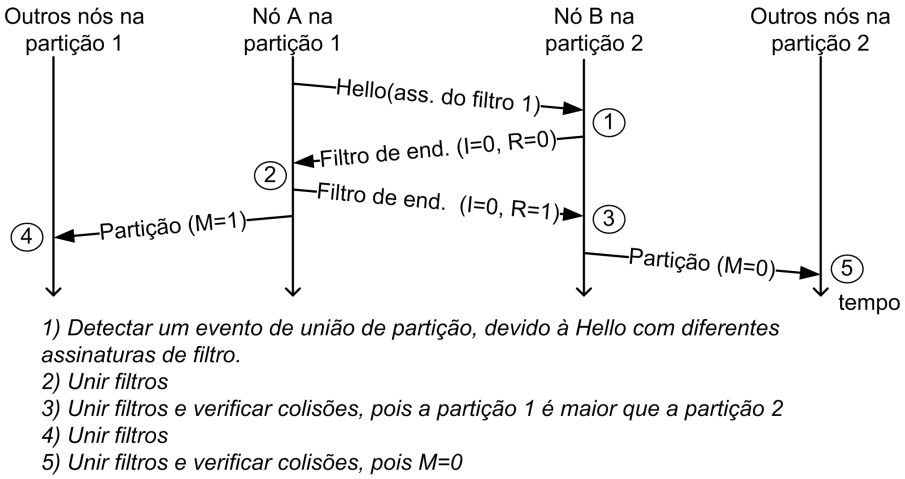
O procedimento de entrada de nós está descrito na Figura 2.8(a). Quando um nó é ligado, ele escuta o meio por um período T_L. Se o nó escuta uma mensagem *Hello*, então já existe pelo menos um nó com filtro de endereços e a rede já está formada. Dessa forma, o nó sabe que é um novo nó ao invés de ser um nó iniciador. Em seguida, o novo nó pede para o nó fonte do primeiro *Hello* enviado, doravante chamado de nó anfitrião para facilitar a explicação, para enviar o filtro de endereços da rede através da mensagem *Address Filter* (AF), mostrada na Figura 2.6(c). Quando o nó anfitrião recebe uma AF, ele verifica se o bit I, o qual indica se a mensagem está sendo usada para um procedimento de entrada de novo nó ou união de partições. Se $I = 1$, a mensagem foi enviada por um novo nó. Então, o nó anfitrião responde ao pedido com outra AF com o bit R setado como 1, indicando que a AF é uma resposta a um pedido de filtro. Quando o novo nó recebe a mensagem AF de resposta, ele armazena o filtro de endereços recebido, escolhe um endereço disponível aleatoriamente e inunda a rede com a mensagem AREQ para alocar o endereço escolhido. Quando os demais nós recebem a mensagem AREQ, eles inserem o novo endereço no filtro e atualizam a sua assinatura de filtro.

Eventos de união de partições também são detectados com base em mensagens *Hello* e AF, como descrito na Figura 2.8(b). Nós em partições diferentes podem selecionar o mesmo endereço, o que pode causar colisões após a união das partições.

No FAP, quando um nó recebe uma mensagem *Hello*, ele verifica se a assinatura do filtro contida na mensagem é diferente da sua assinatura atual. Caso seja, o nó



(a) Entrada de nós.



(b) Detecção e união de partições.

Figura 2.8: Esquemas do FAP para entrada de nós e união de partições.

sabe que ele e o nó fonte da mensagem possuem conjuntos de endereços alocados diferentes. Se já se passaram mais do que T_P segundos desde o último evento de união de partições, um novo procedimento de união de partições é iniciado. Nesse procedimento, ambos os nós trocam mensagens AF para disseminar os filtros das duas partições. Primeiramente, cada nó verifica se o seu endereço é maior do que o do outro nó. Apenas o nó com o maior endereço inicia o processo para evitar que ambos os nós enviem AFs simultaneamente com o bit $R = 0$, o qual indica que a AF demanda uma resposta. Se a verificação for positiva, o nó com o maior endereço envia uma mensagem AF com o seu filtro atual para o outro nó, o qual armazena o filtro recebido e envia um AF como resposta contendo o seu próprio filtro. Em seguida, ambos os nós inundam suas partições com uma mensagem *Partition*, mostrada na Figura 2.6(d), para que todos os nós da rede possam atualizar os seus filtros com os dados da outra partição. Após a recepção da mensagem *Partition*,

cada nó deve examinar o bit M na mensagem *Partition* para verificar se ele está na partição com menor prioridade. A partição com menor prioridade ($M = 0$) é selecionada como a menor partição ou, se as duas partições possuem o mesmo tamanho, como a partição do nó que iniciou o processo de união de partições. Para verificar qual a menor partição, os nós que trocaram as mensagens de AF devem observar os filtros de endereço. O número de nós no filtro de sequência é calculado pelo número de bits em 1 nesse filtro. No filtro de Bloom com contadores, o número de nós pode ser estimado pela soma de todos os contadores dividida pelo número de funções *hash* sendo utilizadas (k), pois cada inserção de elementos irá fazer com que provavelmente k contadores sejam incrementados. Esse valor é uma estimativa, pois duas funções hash diferentes podem indicar uma mesma célula do filtro para um mesmo elemento, mas a célula só é incrementada apenas uma vez. Com o filtro de Bloom convencional, o número de nós também é estimado pela soma de todos os bits em 1 dividida pelo número de funções *hash*, k. Essa medida também é uma estimativa, pois dois elementos diferentes podem ser representados pela mesma célula, que será contada apenas uma vez. Ainda assim, se um filtro de Bloom tem mais células em 1 do que outro filtro de Bloom, é provável que o primeiro filtro tenha mais elementos inseridos do que o segundo.

Cada nó na partição com menor prioridade deve verificar se o seu endereço pertence ao filtro da outra partição para verificar se existe uma colisão. Caso exista, o nó na partição menos prioritária escolhe um endereço disponível em ambos os filtros e, então, inunda a rede com um AREQ para alocar o novo endereço. Se o nó recebe um AREQ com o mesmo endereço que ele escolheu, mas com um número de sequência diferente, ele escolhe outro endereço, pois algum outro nó que também estava com endereço colidido selecionou o mesmo novo endereço. Por fim, todos os nós fazem uma união do filtro recebido da outra partição com o seu próprio filtro, inserem os endereços recebidos nos AREQs e atualizam a assinatura do filtro.

Atualização da assinatura do filtro

O FAP necessita de um mecanismo para sincronizar a atualização da assinatura de filtro após uma entrada de nó ou evento de união de partições para evitar falsos positivos na detecção de uniões de partições. Por exemplo, pode-se observar o cenário descrito na Figura 2.9, na qual o nó A perde a primeira e a segunda transmissão de AREQ devido a colisões, mas um dos vizinhos do nó A conseguiu receber a mensagem. Se esse vizinho atualizar o filtro imediatamente após ter recebido a mensagem, o nó A poderá receber uma mensagem Hello com uma assinatura de filtro diferente da sua própria assinatura. Assim, o nó A iria detectar erroneamente uma união de partições, o que caracteriza um falso positivo, gerando uma alta sobrecarga de mensagens.

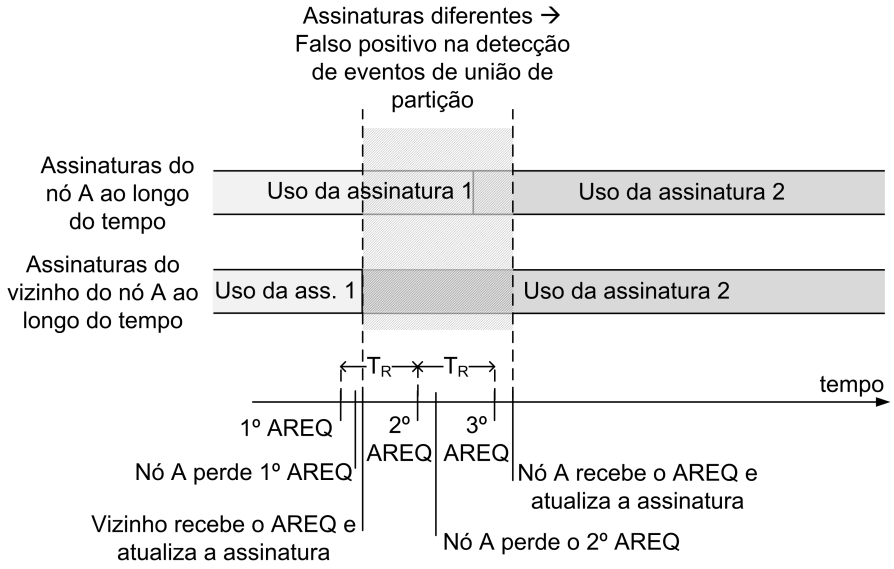


Figura 2.9: Atualização da assinatura do filtro sem o controle proposto: falsos positivos na detecção de união de partições.

O mecanismo proposto para atualizar a assinatura dos filtros, descrito na Figura 2.10, evita esse problema e é baseado no armazenamento por um período curto de assinaturas extras. Com o mecanismo, o nó A perde a primeira e a segunda mensagem Hello, enquanto que o vizinho não perde nenhuma mensagem. Ao invés de atualizar imediatamente a assinatura, o vizinho gera uma nova assinatura e a armazena por um período T_S. Esse período deve ser longo o suficiente para garantir que todas as retransmissões de um AREQ foram inundadas na rede. Portanto, antes do fim de T_S, o nó A acaba recebendo o AREQ, gerando a nova assinatura e armazenando-a. Com isso, nenhuma partição será detectada pelo nó A. Após T_S, o nó vizinho armazena a assinatura desatualizada por mais T_S segundos e começa a utilizar a nova assinatura. Para evitar os falsos positivos na detecção de partições, todas as assinaturas armazenadas são consideradas válidas pelos nós. Assim, quando o vizinho atualizar a assinatura, o nó A perceberá que a nova assinatura usada pelo vizinho está armazenada e, então, é considerada válida, assim como o vizinho aceitará a assinatura desatualizada nos *Hellos* do nó A, pois essa assinatura também está armazenada. Portanto, a nova assinatura é usada após T_S e a assinatura antiga só é descartada após $2 \cdot T_S$. Assim, garante-se que nenhum nó irá atualizar sua assinatura até que todos os nós tenham sido notificados sobre o evento. Além disso, também fica garantido que nenhum nó irá descartar a assinatura antiga enquanto todos os nós não tiverem feito a atualização para a assinatura nova. Com esse procedimento, cada nó pode validar corretamente o seu filtro durante o processo de troca de conteúdo do filtro.

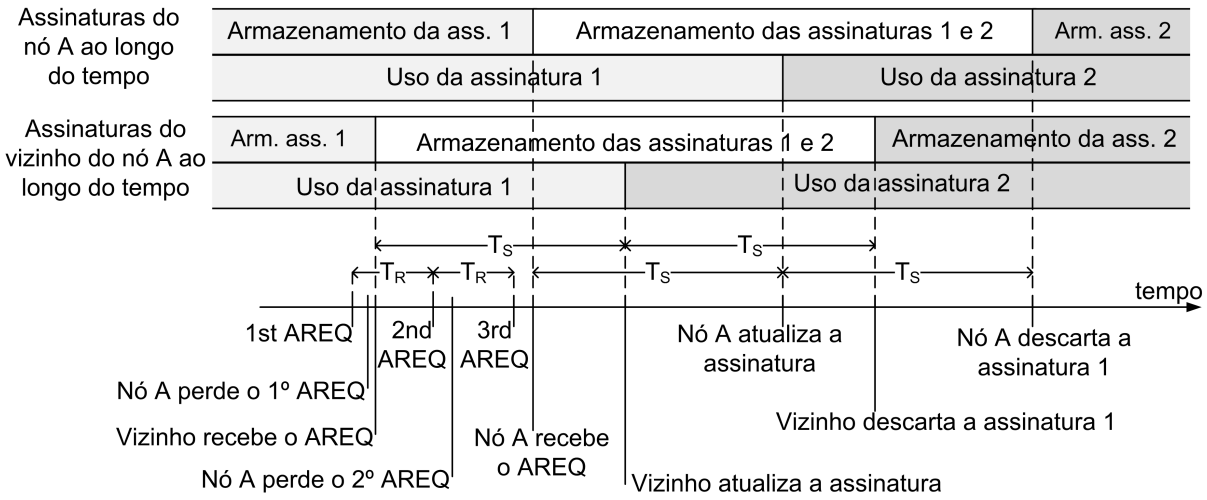


Figura 2.10: Atualização da assinatura do filtro armazenando assinaturas: nós aceitam assinaturas diferentes por um período $2T_S$.

A mensagem AF possui um campo especial chamado de *Signatures*, o qual contém todas as assinaturas de filtro armazenadas pelo nó fonte da mensagem. Após receber a AF, o nó armazena as assinaturas recebidas por um período $T_{S'}$, para manter compatibilidade com os nós na outra partição.

Saída de nós

O processo de realocação de um endereço é importante para provimento de disponibilidade da rede. Assim, quando um nó deixa a rede, o seu endereço deve se tornar disponível para que outros nós entrem na rede com este endereço ou para que o endereço seja atribuído a outro nó da rede por ocasião de uma união de partição. Se o nó for desligado corretamente, ele inunda a rede com uma notificação, que faz com que todos os nós retirem aquele endereço do filtro. Se o nó sair sem notificar a rede, o endereço fica alocado no filtro, o que pode causar escassez de endereços após um tempo. Isto pode ser identificado pela fração de bits em 1 no filtro de Bloom e no filtro de sequência e pela fração de contadores maior do que um no filtro de Bloom com Contador. O limiar para considerar que um filtro está cheio é diferente para o filtro de Bloom e o filtro de sequência, pois, no filtro de Bloom, o número de bits em 1 não indica o número de elementos no filtro, como acontece no filtro de sequência. Com isso, cada nó verifica essa fração no seu filtro de endereços cada vez que o filtro é atualizado. Se essa fração alcançar um limiar que indica que o filtro está cheio ou quase cheio, todos os nós reiniciam o filtro de endereços e retornam a fase de inicialização. Ao invés de escolher um novo endereço, o nó usa o seu endereço atual, o qual não é uma colisão, para reduzir a sobrecarga e evitar a quebra das conexões

de dados ativas. As mensagens de AREQ usadas durante a reinicialização do filtro apresentam o bit $R = 1$ para indicar que todos os nós precisam reiniciar o filtro.

Apesar da fase de inicialização parecer sobrecarregar a rede, ela é equivalente, em termos de carga de controle, a uma união de partições em protocolos de auto-configuração de endereços baseados em DAD, como o protocolo proposto por Fan e Subramani [48]. Para evitar renovações de filtro frequentes em redes com alta ocupação do espaço de endereços, existe um período mínimo entre renovações de filtros, definido por T_M.

Capítulo 3

Análise dos protocolos de endereçamento

Nesse capítulo é apresentada uma análise do protocolo proposto, *F*ilter-based Addressing Protocol (FAP), do DAD [47], da proposta de Fan e Subramani [48], doravante chamada de *DAD with Partition Detection* (DAD-PD) e do MANETconf [54], indicado como Mconf nos gráficos. A comparação com o protocolo DAD é para criar uma base de referência, pois esse é um protocolo sem estados e bem simples, muito embora não trate a formação de partições na rede. O FAP também é comparado ao DAD-PD, que estende o DAD para tratar uniões de partições. Por fim, o FAP é comparado ao MANETconf, um protocolo que também armazena os endereços alocados, mas que apresenta uma lógica diferente para evitar colisões de endereço.

Primeiramente, é apresentada uma análise matemática com a probabilidade de colisão de endereços no FAP. Em seguida, estima-se o número de mensagens enviadas por protocolo em cada um dos procedimentos. Por fim, é apresentada uma análise com simulação que valida o modelo do número de mensagens e analisa os protocolos sobre outros aspectos, como atraso e número de colisões.

3.1 Probabilidade de colisões de endereço no FAP

O FAP foi analisado para avaliar a probabilidade de que o esquema proposto cause colisões de endereços. Uma colisão de endereços pode ocorrer em três situações. Uma colisão ocorre quando dois nós diferentes geram a mesma mensagem AREQ, ou seja, quando dois nós escolhem o mesmo endereço e o mesmo número de identificação. Nesse caso, nenhum dos nós é capaz de perceber que a mensagem recebida foi enviada por outro nó ao invés de ser uma réplica gerada pela inundação da sua própria mensagem. Outra possibilidade é a perda de todas as N_F transmissões da mensagem AREQ dos dois nós com endereço colidido. Nesse caso, nenhum dos

dois nós receberia um AREQ do outro nó indicando a colisão e a colisão, portanto, não seria solucionada. Uma terceira possibilidade para a ocorrência de colisão é que duas partições disjuntas possuam exatamente o mesmo filtro e tentem se unir. Nesse caso, o processo de união de partições não seria iniciado, pois as assinaturas em todas as mensagens *Hello* são iguais e, consequentemente, a rede teria uma colisão para cada endereço alocado. Excluindo-se esses casos, os nós que entram na rede nunca poderiam causar uma colisão de endereço. Um novo nó sempre envia uma mensagem AREQ e, quando um nó que já anunciou o seu endereço recebe um AREQ de outro nó com o mesmo endereço, ele deve trocar de endereço, independente do seu estado atual. Assumindo que não existe comportamento malicioso na rede, essa situação ocorre apenas na inicialização ou quando dois nós se unem à rede aproximadamente ao mesmo tempo, pois ambos os novos nós poderiam escolher o mesmo endereço disponível no filtro. Assim, se dois novos nós recebem o filtro de endereço aproximadamente ao mesmo tempo e escolhem o mesmo endereço, ambos os nós trocarão de endereço, porque cada nó irá receber o AREQ do outro nó. No caso do AREQ de um dos novos nós ser perdido devido a perdas no enlace e a colisões, apenas um nó trocará de endereço e a colisão é resolvida. O caso no qual os dois AREQs são perdidos causaria uma colisão, mas a probabilidade desse evento é desprezível, principalmente devido ao uso de N_F transmissões da mesma mensagem inundada. Outros eventos de perda de mensagens também não causam colisão no FAP, pois essas mensagens perdidas são compensadas por eventos falsos de união de partições. Se um nó não receber nenhuma transmissão de uma mensagem AREQ, seu filtro será diferente do filtro de outros nós. Isso causará um procedimento de união de partições até que todos os nós possuam a mesma informação em seus filtros.

3.1.1 Probabilidade de colisão de AREQs

A probabilidade que dois nós escolham o mesmo endereço e o mesmo número de identificação para um AREQ, P_A, causando uma colisão, pode ser derivada considerando o paradoxo do aniversário [44], discutido no Capítulo 2, com o tamanho do espaço de eventos, e_s, sendo dado pela concatenação do endereço com o número de identificação, e o número de eventos, n, sendo o número de nós tentando acessar a rede aproximadamente ao mesmo tempo, o que significa um conjunto de nós iniciadores com n nós ou um conjunto de novos nós tentando acessar a rede aproximadamente ao mesmo tempo com tamanho n. Então,

$$P_A = 1 - \left[\frac{e_s - 1}{e_s} \cdot \frac{e_s - 2}{e_s} \cdots \frac{e_s - n + 1}{e_s}\right]. \tag{3.1}$$

A Figura 3.1 mostra a probabilidade de colisão, P_A, considerando um espaço de endereços com 256 entradas. Observa-se que existe uma alta probabilidade de colisão

de endereços, mas a probabilidade de colisão de AREQs é desprezível, mesmo para um número de nós maior do que o espaço de endereços disponível, devido ao uso dos números de identificação escolhidos aleatoriamente na mensagem AREQ. Os números de identificação possuem um tamanho fixo de 32 bits, como mostrado descrição da mensagem AREQ na Fig. 2.6(b).

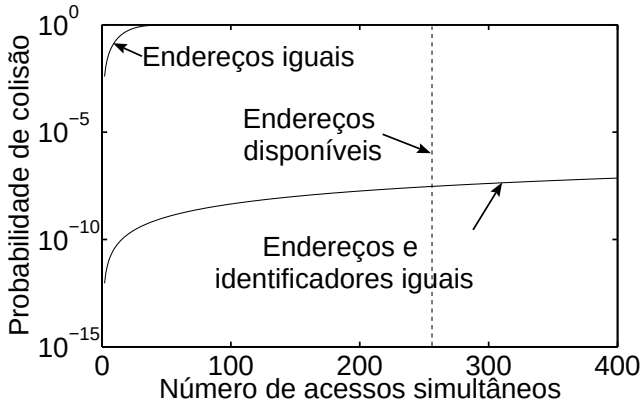


Figura 3.1: Probabilidade de colisões de AREQ com n acessos simultâneos a rede, assumindo que existem 256 endereços disponíveis para alocação.

3.1.2 Probabilidade de colisão de filtros

A probabilidade de que duas partições diferentes possuam o mesmo filtro de endereços é derivada para ambos o filtro de Bloom (P_{BF}) e o filtro de sequência (P_{SF}). Seja A_1 o conjunto de endereços da primeira partição, A_2 o conjunto de endereços na segunda partição e R o conjunto de endereços disponíveis. Então $A_1 \subset R$, $A_2 \subset R$ e $|A_1| + |A_2| \leq |R|$ para garantir que existem endereços suficientes para todos os nós da rede. Se essa condição não for atendida, então nenhum protocolo de endereçamento é capaz de evitar colisões. Além disso, assume-se que todos os pares $(|A_1|, |A_2|)$ são igualmente prováveis, uma vez que o tamanho de cada partição depende apenas da topologia da rede.

Para o filtro de sequência, P_{SF} é obtido pela divisão do número de casos nos quais os dois filtros possuem os mesmos endereços inseridos dividido por todas as possíveis configurações de dois filtros. O número de casos de dois filtros com os mesmos elementos inseridos é obtido pela contagem de todas as possíveis combinações de conjuntos com até $\lfloor \frac{r}{2} \rfloor$ elementos, onde $|R| = r$ é o número de endereços disponíveis para alocação. O número total de pares de conjuntos é computado por todas as possíveis combinações de elementos no primeiro conjunto multiplicadas por todas as possíveis combinações de elementos no segundo conjunto, com a restrição de que a soma dos tamanhos de dois conjuntos é igual ou menor a r. Assim, P_{SF} é dada por

$$P_{SF} = \frac{\sum_{i=1}^{\lfloor \frac{r}{2} \rfloor} \binom{r}{i}}{\sum_{j=1}^{r-1} \left(\binom{r}{j} \cdot \sum_{g=1}^{r-j} \binom{r}{g} \right)}. \tag{3.2}$$

Para o filtro de Bloom, a probabilidade de colisão de filtro também depende dos falsos positivos, porque a representação de um conjunto é igual à representação de um conjunto diferente que contém os mesmos elementos mais os falsos positivos. A probabilidade de colisão de filtros é maior para um filtro de Bloom convencional do que para um filtro de Bloom com contador. Algumas colisões de filtro causadas pelos falsos positivos no filtro de Bloom não irão ocorrer nos filtros de Bloom com contadores, porque os contadores com valores diferentes diferenciam os filtros. Por exemplo, se um filtro de Bloom contém os elementos 'a', 'b' e 'c' e também aponta o elemento 'd' como inserido no filtro, então outro filtro de Bloom contendo 'a', 'b', 'c' e 'd' terá a mesma representação que o primeiro filtro. No caso dos filtros de Bloom com contadores, a inserção do elemento 'd' causará mudanças nos contadores do segundo filtro, de forma que a assinatura do primeiro e do segundo filtro não serão mais iguais. Com isso, ambos os filtros apontariam 'd' como presente, mas os filtros não seriam iguais. Portanto, a probabilidade de existirem duas partições com o mesmo filtro de Bloom, P_{BF}, é um limite superior para a probabilidade de existirem dois filtros de Bloom com contadores iguais.

A probabilidade P_{BF} é obtida pela divisão de todos os possíveis casos de colisão de filtro, ponderados pela probabilidade daquela configuração de filtro acontecer, pelo número de possíveis configurações de partições, o que significa todos os possíveis pares de partições ($|A_1|$,$|A_2|$). Então, utilizando-se a probabilidade de que dois filtros de Bloom possuam os mesmos bits em 1 e em 0, que é dada por

$$P_{BF_{col}} = \sum_{i=0}^{m} \binom{m}{i} \left[P_0^i P_1^{(m-i)} \right] \left[{P'}_0^i {P'}_1^{(m-i)} \right], \tag{3.3}$$

assumindo que o tamanho da primeira partição é $|A_1| = n$ e o tamanho da segunda é $|A_2| = n'$, e ainda considerando que

$$P_{total} = \left[\sum_{j=0}^{m} \binom{m}{j} P_0^j P_1^{m-j} \right] \left[\sum_{g=0}^{m} \binom{m}{g} {P'}_0^g {P'}_1^{m-g} \right] = 1, \tag{3.4}$$

é possível obter a probabilidade de colisão de filtros para partições de qualquer tamanho, que é dada por

$$P_{BF} = \frac{\sum_{n=1}^{r-1} \sum_{n'=1}^{r-n} P_{BF_{col}}}{\sum_{n=1}^{r-1} \sum_{n'=1}^{r-n} P_{total}}, \tag{3.5}$$

onde P_0 e P_0' são, respectivamente, a probabilidade de um bit ser igual a 0 após n e n' inserções no filtro, o que é dado pela Equação 2.5, $P_1 = 1 - P_0$, $P_1' = 1 - P_0'$, r é o número de endereços disponíveis para alocação, m é o tamanho do filtro e k é o número de funções *hash*.

A Figura 3.2 mostra as curvas expressas pelas Equações 3.2 e 3.5, de P_{SF} e P_{BF} respectivamente, e mostra que essas probabilidades são desprezíveis quando o número de endereços disponível para alocação, r, atende a condição $40 \leq r \leq 100$. A probabilidade P_{SF} diminui quando o espaço de endereços aumenta, porque quanto maior o espaço de endereços, maior o tamanho do filtro. A probabilidade P_{BF} aumenta com o valor de r, porque foi assumido o uso de um filtro de Bloom com tamanho fixo e, consequentemente, quando se inserem mais elementos, o filtro fica sobrecarregado com uma alta taxa de falsos positivos. Contudo, mesmo assumindo $r = 100$, o que significa que o número de elementos em ambos os filtros varia de 2 a 100, em um filtro de Bloom com tamanho 120, a probabilidade de colisão obtida é menor do que 10^{-10}. De fato, quanto maior o m, menor a probabilidade de que ocorra uma colisão de filtros quando se utiliza filtros de Bloom ou filtros de Bloom com contadores, porque quanto maior o m, menor é a probabilidade de falsos positivos no filtro.

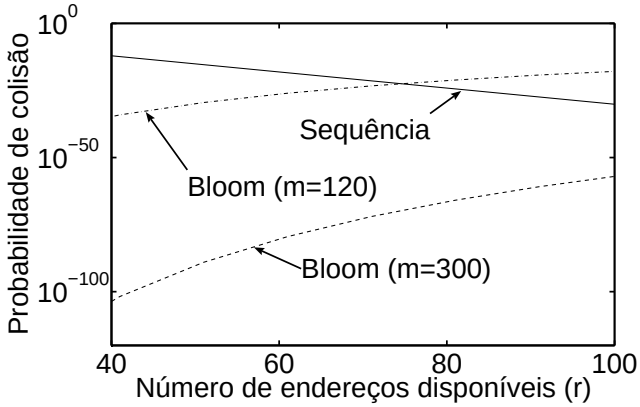


Figura 3.2: Probabilidade de colisão de filtro de endereços durante eventos de união de partições.

Portanto, o FAP não é propenso a erros devido a colisões de AREQs ou a colisões de filtros de endereço na inicialização, na entrada de novos nós ou na união de partições.

3.2 Estimativa da sobrecarga de controle

Os principais procedimentos em protocolos de endereçamento são a inicialização da rede, os nós entrando e saindo e a união de partições. Usualmente, esses proce-

dimentos, assim como outras operações comuns do protocolo, geram sobrecarga de controle, reduzindo a banda disponível para enviar dados dos usuários. Assim, os protocolos devem garantir que colisões de endereço não aconteçam e também devem assegurar uma sobrecarga baixa de mensagens de controle. Dessa forma, para avaliar a sobrecarga de controle de cada protocolo, foi estimado o número de mensagens de controle enviadas em todos os procedimentos.

A análise mostra uma estimativa do número de mensagens enviadas por protocolo no caso ótimo de novos nós, união de partições e inicialização. Como caso ótimo, entende-se o caso no qual não existem perdas de mensagens e apenas um procedimento é executado de cada vez.

A Tabela 3.1 mostra as variáveis usadas nessa análise.[1] Assume-se que todos os valores indicados na Tabela 3.1 são conhecidos no cenário escolhido. O número de colisões de endereço, C, pode ser usado como um valor médio, para estimar o número esperado de mensagens, ou como o valor determinístico, após um experimento, para verificar o impacto das perdas de mensagens e da execução de mais de um mecanismo simultaneamente.

Tabela 3.1: Notação utilizada para a estimativa da sobrecarga.

Variável	**Descrição**
N_F	Número de transmissões de mensagens de inundação por um nó fonte em um evento de inundação
C	Número de colisões de endereço
N	Número de nós na rede
N_P	Total de nós em ambas as partições
$F_j(i)$	Número de mensagens inundadas em um processo de união de partições i no protocolo j
$U_j(i)$	Número de mensagens transmitidas em *unicast* em um processo de união de partições i no protocolo j
$B_j(i)$	Número de mensagens transmitidas em *broadcast* em um processo de união de partições i no protocolo j
N_T	Número de tentativas antes de concluir que o nó está só
N_N	Número de vizinhos do novo nó

3.2.1 Entrada de nós

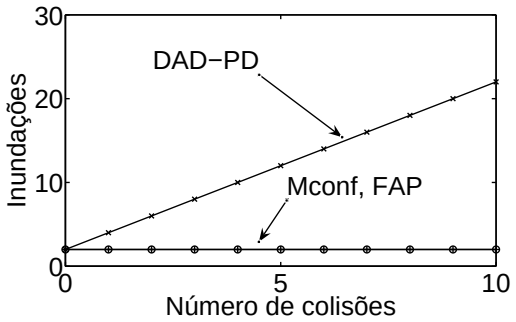
A Tabela 3.2 mostra a estimativa do número de inundações, *unicasts* e *broadcasts* durante a entrada de novos nós no FAP, no DAD-PD e no MANETconf. A Figura 3.3

[1]Todas as N_F mensagens possuem o mesmo conteúdo, exceto pelo número de identificação, que é escolhido aleatoriamente para cada uma das transmissões da mensagem.

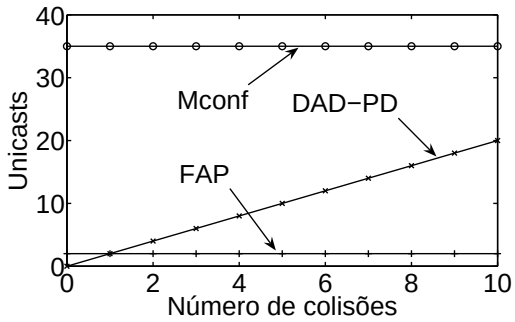
mostra os valores para números de mensagens nos três protocolos de acordo com essas estimativas, assumindo que $N_F = 2$, $N_N = 4$ e $N = 36$, os quais são valores típicos para uma rede ad hoc de médio porte.

Tabela 3.2: Número de mensagens de controle na entrada de nós.

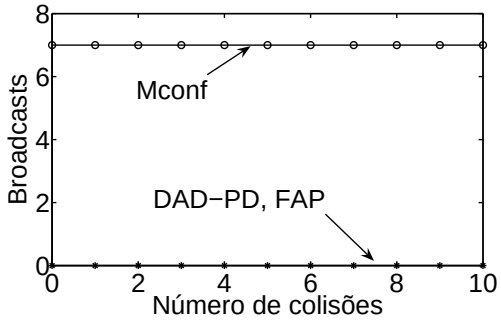
Protocolo	**Inundação**	***Unicast***	***Broadcast***
FAP	N_F	2	0
DAD-PD	$N_F(1+C)$	$N_F \cdot C$	0
Mconf	2	$N-1$	$3+N_N$



(a) Número de mensagens enviadas em inundação.



(b) Número de mensagens enviadas em *unicast*.



(c) Número de mensagens enviadas em *broacast*.

Figura 3.3: Estimativa da sobrecarga de controle na entrada de novos nós nos protocolos de endereçamento, assumindo $N_F = 2$, $N_N = 4$ e $N = 36$.

A sobrecarga do DAD-PD em um procedimento de entrada de novos nós depende do número de colisões de endereço, C. De fato, esse protocolo não armazena a lista de endereços usados, o que significa que o novo nó escolhe um endereço aleatoriamente que pode já estar alocado para outro nó na rede. O FAP tem um desempenho superior ao DAD-PD quando uma ou mais colisões de endereço acontecem, $C \geq 1$, pois isso aumenta o número de inundações apenas no DAD-PD e as inundações são as operações com mais alto custo. O valor real de C depende da razão entre o número de nós na rede, N, e o número de endereços disponíveis, r. Quando $N/r \to 1$, a

probabilidade de colisão de endereços aumenta e, consequentemente, o desempenho do DAD-PD é reduzido.

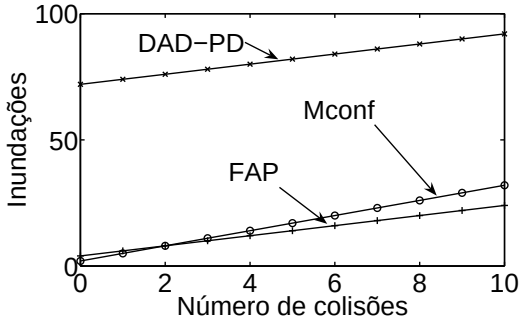
O MANETconf, assumindo os parâmetros escolhidos, possui uma sobrecarga maior que o FAP, pois o MANETconf é baseado na premissa que todos os nós devem concordar antes que um endereço seja alocado, o que demanda muitas mensagens de controle enviadas em *unicast*. Dessa forma, o novo nó troca três mensagens através de *broadcasts* com o nó que irá alocar o endereço, o qual é chamado de iniciador no MANETconf. O iniciador, então, inunda a rede perguntando se todos os nós concordam com a alocação do endereço escolhido. Se todos os nós concordarem, então o iniciador inunda a rede novamente para anunciar a alocação do endereço. Além da sobrecarga causada por todos os eventos de inundação, dependendo do protocolo de roteamento, cada fluxo de mensagens transmitido em *unicast* pode implicar em uma inundação para buscar a rota entre o nó fonte e o nó destino. Por exemplo, ao se usar o protocolo *Ad hoc On-Demand Distance Vector Routing* (AODV), cada requisição por um destino ainda não conhecido implica na inundação de uma mensagem de requisição de rota. Então, o uso intensivo de *unicasts* para destinos diferentes, como acontece no MANETconf, pode causar uma alta sobrecarga de controle.

3.2.2 União de partições

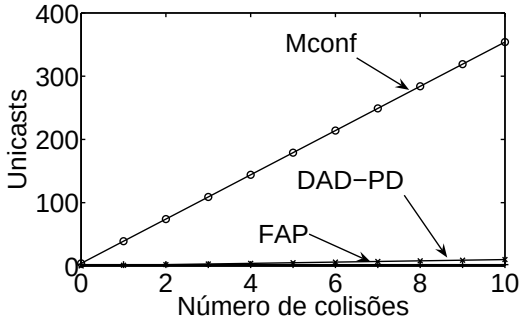
A Tabela 3.3, cujas equações estão representadas na Figura 3.4, assumindo que $N_F = 2$, $N_N = 4$ e $N = 36$, mostra que o FAP possui um baixo custo para realizar as uniões de partições, porque os filtros no FAP identificam os endereços colididos com os filtros de endereço. No FAP, os dois nós que detectaram a união trocam seus filtros e então disseminam o filtro da outra partição para os demais nós através de inundação. Em seguida, apenas uma inundação é requisitada por colisão, para anunciar o novo endereço alocado. No MANETconf, os nós que detectaram a partição também trocam e inundam as listas de endereço como no FAP, mas as colisões são tratadas de forma similar ao mecanismo de novos nós, o qual implica em uma alta sobrecarga de mensagens em *unicast*. No DAD-PD, todos os nós inundam a rede após uma detecção de união de partições e cada colisão envolve o envio em *unicast* de um AREP e a inundação de um AREQ por nó envolvido em alguma colisão.

Tabela 3.3: Número de mensagens de controle na união de partições.

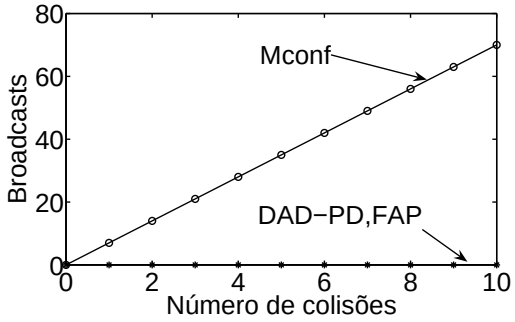
Protocolo	**Inundação** (F_j)	***Unicast*** (U_j)	***Broadcast*** (B_j)
FAP	$N_F(2+C)$	2	0
DAD-PD	$N_F(N_P+C)$	C	0
Mconf	$2+3C$	$4+C(N_P-1)$	$C(3+N_N)$



(a) Número de mensagens enviadas em inundação.



(b) Número de mensagens enviadas em *unicast*.



(c) Número de mensagens enviadas em *broacast*.

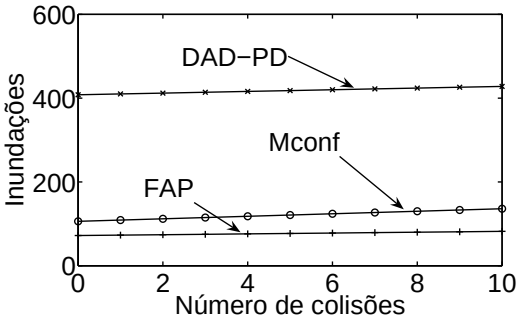
Figura 3.4: Estimativa da sobrecarga de controle na união de partições nos protocolos de endereçamento, assumindo $N_F = 2$, $N_N = 4$ e $N = 36$.

3.2.3 Inicialização abrupta da rede

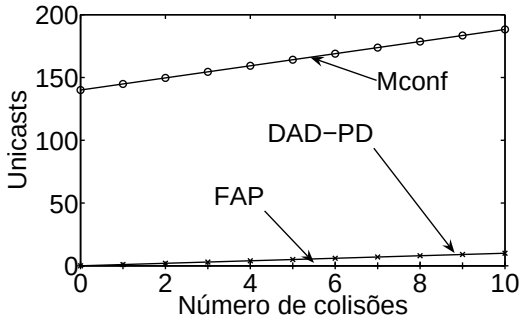
As estimativas na Tabela 3.4 são relativas à inicialização da rede e estão demonstradas na Figura 3.5, assumindo que $N_F = 2$, $N_N = 4$ e $N = 36$. O FAP apresenta a menor sobrecarga, porque ele possui um mecanismo similar ao proposto pelo DAD para inicializar a rede. O DAD-PD e o MANETconf geram uma alta sobrecarga de controle, porque eles especificam apenas um mecanismo de inicialização gradual. Se diversos nós entrarem na rede aproximadamente ao mesmo tempo, cada nó acreditará que está sozinho, irá escolher aleatoriamente um endereço e um identificador de partição. Em seguida, os nós percebem a presença de vizinhos com identificadores de partição diferentes e iniciam procedimentos de união de partições. Uma vez que todos os nós precisam unir suas partições para criar uma única partição representando a rede inteira, eles executarão pelo menos $N - 1$ procedimentos de união de partições. É importante mencionar que cada inundação durante a inicialização é menos impactante do que a inundação quando a rede já está formada, porque as mensagens em uma inundação são encaminhadas apenas pelos nós com o mesmo identificador de partição. Assim, cada inundação alcança um número de nós menor do que N durante essa fase de inicialização.

Tabela 3.4: Número de mensagens de controle na inicialização.

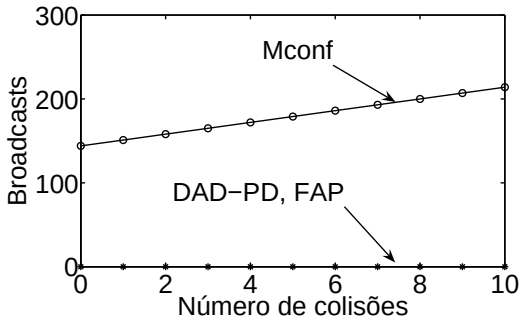
Protocolo	**Inundação**	***Unicast***	***Broadcast***
FAP	$(N_F \cdot N) + C$	0	0
DAD-PD	$\sum_{i=1}^{N-1} F_{DAD-PD}(i)$	$\sum_{i=1}^{N-1} U_{DAD-PD}(i)$	0
Mconf	$N + \sum_{i=1}^{N-1} F_{Mconf}(i)$	$\sum_{i=1}^{N-1} U_{Mconf}(i)$	$N_T \cdot N + \sum_{i=1}^{N-1} B_{Mconf}(i)$



(a) Número de mensagens enviadas em inundação.



(b) Número de mensagens enviadas em *unicast*.



(c) Número de mensagens enviadas em *broacast*.

Figura 3.5: Estimativa da sobrecarga de controle na inicialização abrupta nos protocolos de endereçamento, assumindo $N_F = 2$, $N_N = 4$ e $N = 36$.

3.2.4 Operação normal da rede

Após a inicialização, cada nó inicia a operação normal do protocolo. Cada nó no FAP, DAD-PD e MANETconf enviam periodicamente mensagens *Hello* em *broadcast*. Além disso, o MANETconf também é baseado em inundações periódicas para manter os identificadores de partição. Portanto, o MANETconf é o protocolo com maior sobrecarga durante a operação normal.

3.2.5 Saída de nós

O procedimento de saída de nós é necessário apenas para o FAP e o MANETconf, porque esses protocolos mantêm o estado sobre os endereços alocados. No procedi-

mento de entrada de nós do MANETconf, o nó iniciador, que escolhe o endereço para o novo nó, contata todos os outros nós da rede para garantir que cada nó concorda com o novo endereço que deve ser alocado. Após N_{MR} vezes tentando contatar um nó específico sem receber resposta, o iniciador conclui que o nó está ausente e inunda a rede com essa informação. No FAP, o endereço alocado para os nós que já deixaram a rede são liberados apenas quando o filtro de endereços está quase cheio. Nesse caso, o filtro é zerado e um procedimento de inicialização é iniciado. Dessa forma, o procedimento de saída de nós do FAP é custoso, mas raramente é chamado, o que significa que esse procedimento possui um baixo impacto sobre a sobrecarga total do FAP.

3.3 Resultados da simulação

Os protocolos de endereçamento foram implementados no *Network Simulator-2* (NS-2), considerando o modelo *Shadowing* para propagação de rádio e o modelo do IEEE 802.11 do NS para controle de acesso ao meio. Esses modelos têm como objetivo criar um cenário similar ao de uma rede comunitária real, e para tanto se utilizou parâmetros de equipamentos comerciais. Assim, os parâmetros usados na simulação são: um alcance médio de transmissão de $18,5$ m, um alcance de *carrier sense* máximo de 108 m, e uma densidade de 0.0121 $nós/m^2$ [58]. Foram medidos o tráfego de controle gerado por todos os nós, os atrasos para a alocação de endereços e o número de colisões de endereço, considerando um intervalo de confiança de 95% nos resultados. Utilizou-se o protocolo de roteamento *Ad hoc On-Demand Distance Vector Routing* (AODV), pois esse protocolo apresenta uma implementação estável no NS-2.

O principal objetivo ao comparar o FAP com o DAD-PD é observar o impacto no desempenho do protocolo do uso das assinaturas de filtro ao invés de identificadores de partição arbitrários para detectar eventos de união de partições. É importante observar, contudo, que no DAD-PD original, o novo identificador de partição após a união de partições era dado pela soma dos identificadores de partição, o que causava instabilidade no protocolo. Assim, para melhorar o desempenho do protocolo, foi proposta uma modificação no DAD-PD, de forma que o novo identificador de partição passa a ser o maior identificador de partição ente as partições que estão se unindo, ao invés de ser a soma dos identificadores. Com isso, o número de falsas detecções de partições foi reduzido e o desempenho do protocolo melhorou. Dessa forma, nas comparações a seguir, foi utilizado o DAD-PD modificado ao invés de se utilizar o DAD-PD original. O MANETconf e o DAD foram implementados de acordo com as suas descrições originais.

Os parâmetros dos protocolos estão nas Tabelas 3.5 e 3.6. Esses parâmetros

foram escolhidos com base em experimentos para aumentar o desempenho de todos os protocolos e também nas recomendações dos autores de cada protocolo[2]. Tanto o FAP quanto o DAD-PD usam *Hellos* com tamanhos iguais, pois foi assumido que tanto o identificador de partição do DAD-PD quanto a assinatura do FAP são compostos por 4 bytes. Foi assumido um espaço de endereçamento com 150 endereços e uma ocupação média do espaço de endereços de 1/3, o que significa que são utilizados aproximadamente 50 nós na rede. Com essa escolha de parâmetros, é possível garantir que o espaço de endereços não é uma restrição para nenhum protocolo e não causará nenhum tipo de instabilidade. De acordo com esses parâmetros e o cálculo para o tamanho do filtro de endereços apresentado na Seção 2.1.3, são utilizados nos experimentos com o FAP filtros de Sequência com 23 bytes.

Tabela 3.5: Parâmetros de tempo do FAP (F), DAD-PD (PD), DAD (D) e MANETconf (M).

Variável	**Descrição**	**Valor**	**Protocolo**
T_L	Tempo máximo escutando o meio	1,0 s	F, PD
T_P	Intervalo mínimo entre uniões de partições	3,0 s	F, PD
T_W	Tempo de espera por mensagens AREQ e AREP	1,2 s	F, PD, D
T_R	Intervalo entre replicações de mensagens inundadas	0,3 s	F, PD, D
T_H	Intervalo entre mensagens *Hello*	1,0 s	F, PD, M
T_C	Tempo de espera antes de trocar de endereço na inicialização	0,3 s	F
T_S	Tempo de armazenamento de assinaturas de filtro geradas pelo nó	0,5 s	F
$T_{S'}$	Tempo de armazenamento de assinaturas de filtro recebidas	3,0 s	F
T_M	Intervalo mínimo entre renovações de filtros	5,0 s	F
T_F	Tempo máximo esperando por um filtro de endereços	0,5 s	F
T_{MA}	Limiar do *Address Allocation Timer*	1,0 s	M
T_{MR}	Limiar do *Request Reply Timer*	0,3 s	M
T_{MN}	Limiar do *Neighbor Reply Timer*	0,35 s	M
T_{MAP}	Limiar do *Allocation Pending Timer*	1,0 s	M
T_{MP}	Limiar do *Partition Timer*	2,0 s	M

[2]Apesar de maiores explicações sobre cada parâmetro não terem sido providas, os parâmetros foram listados com seus respectivos valores para garantir a completude e a reprodutibilidade do experimento.

Tabela 3.6: Parâmetros de limiares do FAP (F), DAD-PD (PD), DAD (D) e MANETconf (M).

Variável	**Descrição**	**Valor**	**Protocolo**
N_F	Número de transmissões de cada mensagem inundada	2	F, PD, D
N_T	Número de busca por vizinhos antes de concluir que o nó está só	4	M
N_{MR}	Número de tentativas de se conectar com o iniciador	2	M
N_{MI}	Número de tentativas para alocar um endereço para um novo nó	2	M
N_{MP}	Limite de perdas de mensagens para determinar que o nó com a identificação da partição está ausente	3	M

3.3.1 Verificação do número de mensagens enviadas

Na Seção 3.2 foi realizado um estudo do número de mensagens enviadas em cada protocolo em um cenário ideal, ou seja, um cenário no qual procedimentos não ocorrem simultaneamente e não existem perdas de mensagens. Essas estimativas foram utilizadas para validar as implementações feitas e para avaliar o impacto das perdas de mensagens sobre cada um dos protocolos. Assim, calculou-se o erro entre a estimativa pelas equações e o resultado das simulações, de acordo com

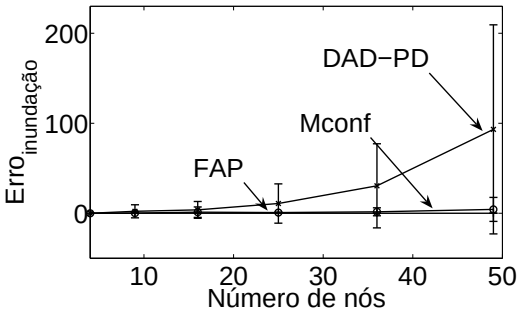
$$Erro = V_{sim} - V_{est}, \tag{3.6}$$

onde V_{sim} é o valor obtido com a simulação e V_{est} é o valor obtido com a estimativa das equações.

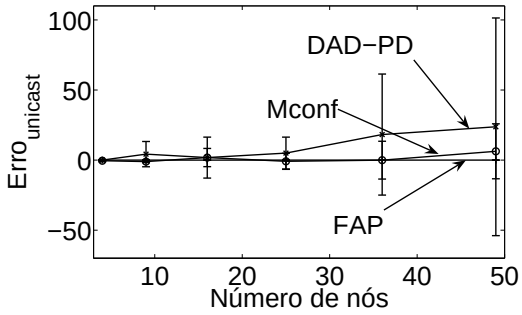
Entrada de nós

A Figura 3.6 mostra o Erro (Eq. 3.6) quando um nó entra em uma rede com N nós dispostos em grade. O novo nó pode estar em qualquer posição dentro da grade. Para essa análise, assumiu-se que $N_F = 2$ para o DAD-PD e o FAP.

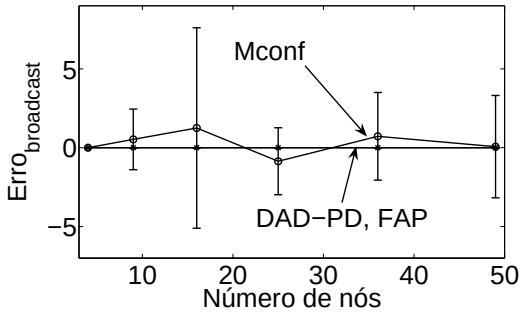
A Figura 3.6(a) mostra o erro de cada protocolo com relação às mensagens de inundação. O FAP não apresentou erros independente do número de nós na rede, de forma que o caso ideal modela o caso real sem erros. O MANETconf apresentou um pequeno erro, que se deve principalmente às perdas de mensagens para unir listas após uma união de partição, para trocar endereços após se detectar colisões, e para excluir nós que não respondem a um pedido de alocação de endereço. De fato,



(a) Erro no número de mensagens enviadas em inundação.



(b) Erro no número de mensagens enviadas em *unicast*.



(c) Erro no número de mensagens enviadas em *broacast*.

Figura 3.6: Erro na estimativa da sobrecarga de controle na entrada de novos nós nos protocolos de endereçamento, assumindo $N_F = 2$.

esse protocolo pode gerar inconsistências nas listas de endereço devido ao processo de união de partições. Quando dois nós se encontram pela primeira vez, devem trocar mensagens de ID com a identificação da partição, dada pelo nó com menor endereço na partição e um número aleatório. Ao receber uma mensagem de ID com dados diferentes da sua própria partição, o nó deve responder com uma mensagem contendo a lista de endereços da sua rede. Por fim, após receber uma mensagem com a lista, o nó deve gerar uma mensagem em inundação para anunciar essa nova lista para os outros nós da sua partição. Se uma mensagem de ID ou uma mensagem de lista é perdida, então pode acontecer dos dois nós ficarem com o mesmo identificador de partição, porém com listas diferentes. O mesmo pode ocorrer se algum nó não receber a mensagem de partição. Como a inicialização da rede se deu de forma abrupta, através de diversos processos de união de partição, então é comum que os nós possuam listas de endereço diferentes. O acontecimento desse evento é mais frequente nas redes que possuem mais nós, pois isso implica em mais processos de união de partição durante a inicialização da rede. Assim, em redes com mais nós, o erro com relação ao caso ideal aumenta tanto para as inundações quanto para os *unicasts* no MANETconf. O erro com relação às mensagens de *broadcast* do

MANETconf é pequeno e se deve às variações no número de vizinhos de um novo nó.[3]

O DAD-PD apresentou um erro que cresce exponencialmente com o número de nós da rede na inundação. Isso ocorre devido a falsas detecções de partição no protocolo. Um nó considera que uma partição foi formada e que o identificador deve ser trocado toda vez que perde um vizinho. Quando um nó muda de endereço após já ter enviado pelo menos uma mensagem de Hello, os seus nós vizinhos consideram que perderam um vizinho, que utilizava o endereço antigo do nó. Por essa razão, um processo para modificação de identificador de partição é iniciado, através de um mecanismo de partição, sobrecarregando a rede com mensagens de *Address Request* (AREQ). O erro no *unicast* se deve ao envio de cópias de mensagens de *Address Reply* (AREP), quando a segunda transmissão de uma mensagem AREQ é feita antes da chegada do AREP da primeira cópia.

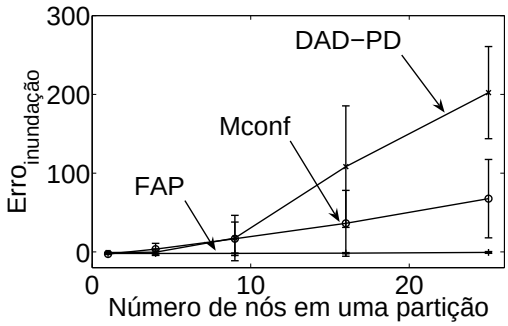
União de partições

O erro na estimativa do número de mensagens enviadas durante uma união de partições é mostrado na Figura 3.7. O cenário utilizado para a simulação considera a união de duas partições de tamanho igual por um nó que interconecta as duas. Esse nó é ligado após a inicialização das duas partições, e, assim, realiza um processo de entrada de novo nó com uma das partições e um processo de união de partições com a outra. O erro apresentado considera a soma da estimativa desses dois eventos.

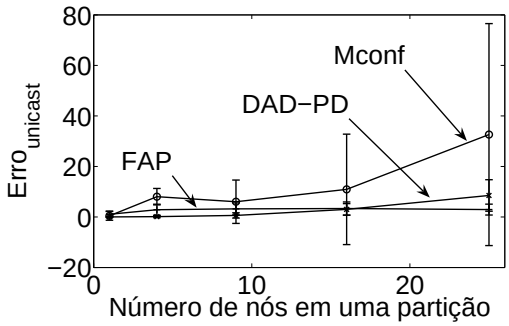
Observa-se que o FAP não apresentou erros com relação à estimativa de inundações durante um processo de partição. Já o MANETconf e o DAD-PD apresentaram um erro que aumenta de acordo com o número de nós na rede. As causas para essa variação, e também da variação na estimativa do número de mensagens em *unicast*, são semelhantes às encontradas na análise de entrada de novos nós. Contudo, esses erros são acentuados no processo de união de partições devido à existência de mais colisões, que são mais prováveis durante uma união de partições do que em um processo de entrada de novos nós.

Os erros na estimativa do número de mensagens de *unicast* do FAP se deve ao mecanismo para impedir que várias uniões de partições sejam iniciadas pelo mesmo nó simultaneamente. Assim, após o nó entrar na rede, ele aguarda um período T_P antes de aceitar um pedido de união de partições de algum nó. O nó vizinho do novo nó na outra partição pode tentar iniciar a união das partições com mensagens Filtro durante esse período. Contudo, o novo nó ignora essas mensagens durante T_P, gerando o erro observado na estimativa.

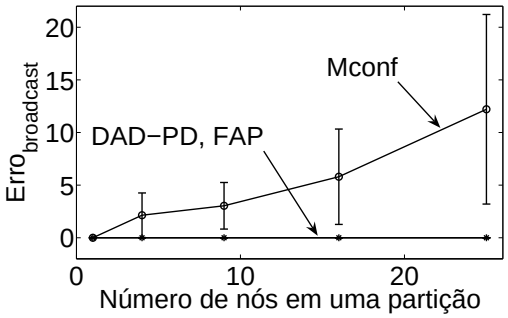
[3]Nessa análise, foi usado para o cálculo das estimativas o número médio de vizinhos em cada cenário.



(a) Erro no número de mensagens enviadas em inundação.



(b) Erro no número de mensagens enviadas em *unicast*.



(c) Erro no número de mensagens enviadas em *broacast*.

Figura 3.7: Erro na estimativa da sobrecarga de controle na união de partições nos protocolos de endereçamento, assumindo $N_F = 2$.

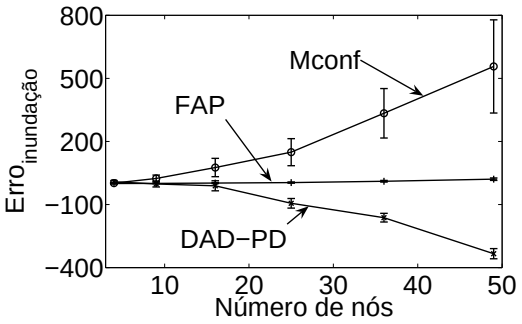
Inicialização abrupta da rede

Na Figura 3.7, é apresentado o resultado do erro entre a simulação e a estimativa do número de mensagens relativo ao procedimento de inicialização abrupta da rede. Nessa análise, o FAP seguiu a estimativa de mensagens com um erro muito baixo, enquanto que o DAD-PD e o MANETconf apresentaram variações mais significativas e que aumentam com o número de nós na rede.

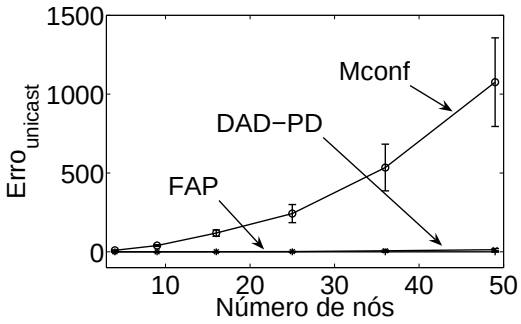
Os erros observados no MANETconf acontecem devido às detecções falsas de partições. Essas detecções ocorrem devido à perda de mensagens, que acabam gerando a liberação de endereços em uso e a união incorreta de listas de endereços. Quanto mais uniões de partições são realizadas, mais prováveis são esses erros. Assim, se dois nós vizinhos que possuem listas diferentes, mas com uma grande intercessão, acreditam estar em um processo de união de partições, todo o conjunto de nós que estão nas duas listas irá verificar, através de inundações, se a colisão de endereço realmente existe, sobrecarregando a rede. O erro no *unicast* é gerado pelas falsas detecções de partição e também pelas perdas de mensagens durante o processo de resolução das colisões entre as duas partições. O erro no número de mensagens de *broadcast* ocorre devido ao erro na estimativa do número de vizinhos de cada nó,

que varia ao longo do processo de inicialização da rede.

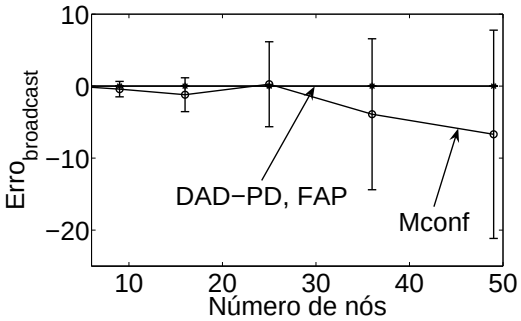
O DAD-PD apresentou um comportamento distinto do esperado, reduzindo o número de mensagens estimadas. Isso ocorre porque as estimativas assumem que cada processo de união de partições ocorre separadamente, mas na prática vários processos ocorrem simultaneamente e se misturam, reduzindo o número final de uniões de partições.



(a) Erro no número de mensagens enviadas em inundação.



(b) Erro no número de mensagens enviadas em *unicast*.



(c) Erro no número de mensagens enviadas em *broacast*.

Figura 3.8: Erro na estimativa da sobrecarga de controle na inicialização abrupta da rede nos protocolos de endereçamento, assumindo $N_F = 2$.

Discussão

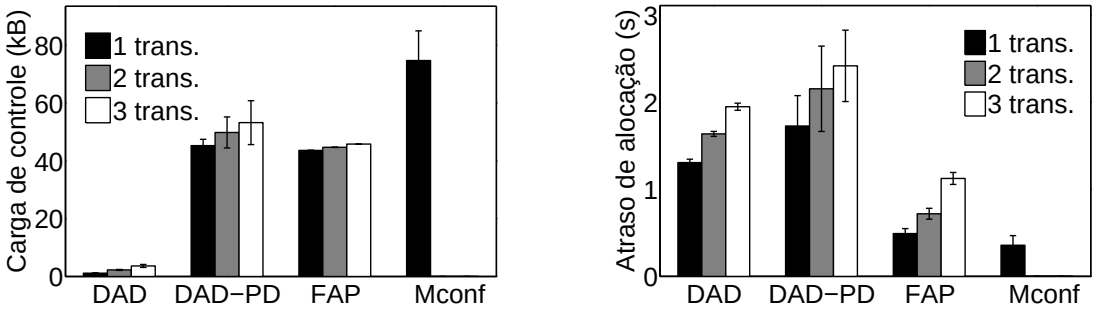
O FAP apresentou as menores sobrecargas nas estimativas para entrada de novos nós, união de partições e inicialização da rede. Ao observar o funcionamento dos protocolos em um ambiente sujeito a falhas e no qual diversos procedimentos podem ocorrer simultaneamente, o FAP se manteve com o menor erro com relação às estimativas na maior parte dos casos, confirmando a tendência deste protocolo a gerar uma sobrecarga menor que os demais.

Essa análise também permite observar o comportamento de cada protocolo e qual o impacto da perda de mensagens em cada um deles. O MANETconf é o mais afetado pelas perdas de mensagem, pois esses eventos causam, com frequência, in-

consistências nas listas dos endereços alocados, causando um aumento da sobrecarga de controle e também propiciando o protocolo a não identificar colisões de endereço. A seguir, é avaliado cada um dos protocolos com relação ao número efetivo de mensagens transmitidas por todos os nós da rede. Assim, o impacto das inundações, dos *broadcasts* e dos *unicasts* fica mais claro com relação à sobrecarga de controle total de cada protocolo. São avaliados também os atrasos e as colisões em cada um dos protocolos.

3.3.2 Avaliação da entrada de nós

Nessa seção, é analisado o impacto da entrada de um nó na rede. Para essa análise, considera-se uma área retangular com 15 nós distribuídos em grade[4]. A carga de controle é medida após o décimo sexto nó ser ligado, assim como o atraso até esse nó obter o seu endereço, como mostrado na Figura 3.9. A carga de controle foi medida como o volume de bits enviados por todos os nós, considerando-se apenas as mensagens de controle geradas pelo protocolo de endereçamento. Foi analisado o impacto do uso de uma, duas ou três transmissões de cada mensagem inundada para aumentar a confiabilidade do FAP, DAD-PD e DAD. Para o MANETconf, foi considerada apenas uma transmissão de cada mensagem inundada, pois esse protocolo já possui os seus próprios mecanismos para detectar falhas na entrega de mensagens e não necessita de nenhum outro mecanismo para garantir confiabilidade na entrega de mensagens.



(a) Sobrecarga de controle.

(b) Atraso para alocação dos endereços.

Figura 3.9: Sobrecarga de controle e atraso na alocação de endereço em um procedimento de entrada de nó em uma rede com 15 nós, assumindo N_F=1,2, e 3.

Nos resultados desse experimento, foi observado que o FAP apresenta uma sobrecarga superior ao DAD, porque o FAP utiliza mensagens de *Hello* para detectar partições. O protocolo DAD-PD apresentou a maior sobrecarga de controle, porque

[4]Embora um ambiente real não se organize exatamente como uma grade, essa é uma boa aproximação que permite um controle mais preciso dos eventos que ocorrem durante o experimento.

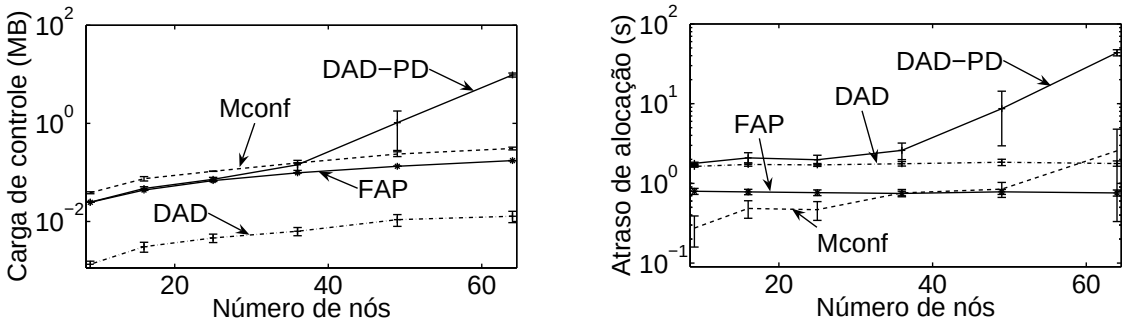
esse protocolo inicia diversos procedimentos de união de partições devido a falhas na detecção de uniões de partição após a entrada de novos nós. De fato, quando um nó entra na rede, o identificador de partição deve ser trocado para representar o novo conjunto de endereços alocados. A atualização desse valor, contudo, pode causar detecções falsas de união de partições, o que aumenta a sobrecarga de controle. O protocolo MANETconf, como esperado, também apresentou uma carga de controle superior a do FAP, como descrito na Seção 3.2.

Apesar de serem protocolos mais complexos, tanto o FAP quanto o MANETconf apresentaram atrasos inferiores ao do DAD, que é um protocolo simples. O FAP apresentou um baixo atraso para uma única transmissão de cada mensagem inundada, pois o novo nó não espera por um período fixo antes de alocar o endereço, como acontece no DAD e no DAD-PD, que esperam mensagens de AREP. Assim, o FAP apresenta baixa sobrecarga e um pequeno atraso na entrada de novos nós, sem causar colisões de endereços.

Uma vez que os protocolos são baseados em inundação, também foi avaliado o impacto do número de nós na rede na sobrecarga de controle e no atraso da entrada do último nó na rede, como mostrado na Figura 3.10. Observa-se, como esperado, que um número maior de nós implica em um aumento da carga de controle. As diferenças entre as curvas da sobrecarga de controle são explicadas pelas mesmas razões apresentadas no experimento com 15 nós. O atraso médio para se obter um endereço, contudo, possui um comportamento diferente. O número de nós na rede quase não influencia no atraso do FAP. Nesse protocolo, o novo nó obtém o filtro de endereços com um vizinho, escolhe um endereço e envia uma mensagem em inundação. Já o DAD e o DAD-PD são influenciados pelo número de nós na rede, porque o novo nó escolhe aleatoriamente um novo endereço. Quanto mais nós na rede, maior a probabilidade de ocorrer uma colisão que forçará o nó a repetir o processo de inundação, atrasando a alocação do endereço. O atraso do MANETconf também sofre com o aumento do número de nós na rede, pois a alocação do endereço depende da confirmação de todos os nós. Portanto, o FAP apresenta uma baixa sobrecarga de controle e um atraso baixo e aproximadamente constante, o que mostra que o FAP é mais eficiente do que tanto o DAD-PD quanto o MANETconf no procedimento de entrada de nós na rede.

3.3.3 Avaliação da inicialização abrupta

A seguir, é analisado o impacto do número de transmissões de mensagens inundadas, da densidade e do número de nós na inicialização abrupta da rede. Para as análises de número de transmissões e densidade, assumiu-se uma rede com 49 nós distribuídos em grade.

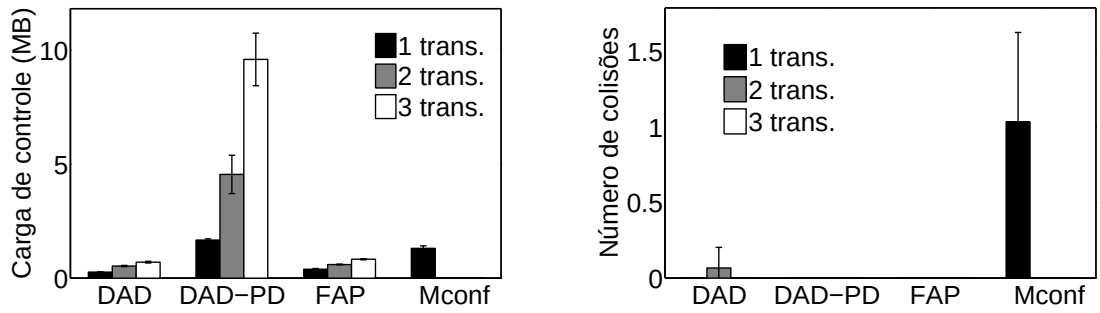


(a) Carga de controle após a entrada no novo nó.

(b) Atraso para a alocação do endereço.

Figura 3.10: Impacto do procedimento de entrada de nós de acordo com o número de nós na rede e assumindo que $N_F = 2$.

Na Figura 3.11, é mostrado o impacto do uso de uma, duas ou três transmissões de cada mensagem de inundação durante a inicialização da rede. Todos os nós entram na rede aproximadamente ao mesmo tempo e selecionam aleatoriamente um endereço.



(a) Carga de controle.

(b) Número de colisões de endereço.

Figura 3.11: Impacto do número de transmissões de cada mensagem inundada na inicialização abrupta em uma rede com 49 nós.

O protocolo proposto, FAP, possui um procedimento específico para a inicialização da rede que reduz a carga de controle em até $72,44\%$, quando comparado com o MANETconf, e em até $91,35\%$ quando comparado com o DAD-PD.

Apesar de o DAD enviar poucas mensagens, ele não é robusto a perda de mensagens. Assim, o DAD apresentou uma pequena probabilidade de colisão nesse cenário. O MANETconf apresenta ainda mais colisões que o DAD, pois a inicialização desse protocolo ocorre através de diversos procedimentos de união de partições e o protocolo pode ficar instável devido à perda de mensagens quando muitos procedimentos de união de partições são iniciados simultaneamente. De fato, o MANETconf foi projetado assumindo que cada nó entra na rede em um momento diferente e existe um intervalo não desprezível entre duas entradas de nós. O FAP é robusto a perda

de mensagens, pois se um nó não recebe uma informação de endereço, seu filtro de endereços fica diferente dos demais filtros de endereço, o que causa procedimentos de união de partições falsos até que todos os nós possuam o mesmo filtro de endereços. Com apenas uma transmissão de cada mensagem, o FAP é capaz de compensar perdas de mensagens, embora o DAD e o DAD-PD não sejam.

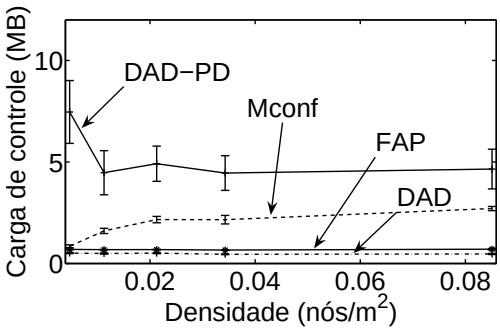
O DAD-PD apresenta o pior resultado com relação à sobrecarga de controle, pois ele não armazena a lista de endereços alocados, usa mensagens *Hello* e não foi projetado para a inicialização com ingresso simultâneo dos nós. Quando vários nós entram na rede ao mesmo tempo com o DAD-PD, cada nó escolhe o seu próprio identificador de partição e trata os seus vizinhos como nós em partições diferentes. Com isso, é possível concluir que o FAP apresenta o melhor resultado, pois não foram detectadas colisões com o uso desse protocolo e a carga de controle se manteve baixa em todos os experimentos.

Em seguida, é avaliado o impacto da densidade e do tamanho da rede quando todos os nós entram na rede aproximadamente ao mesmo tempo. A densidade foi variada de ≈ 0.064 $nós/m^2$, representando cenários urbanos com alta densidade, até ≈ 0.0035 $nós/m^2$, representando áreas rurais com baixíssima densidade [58]. Para a análise de densidade, foi assumido o uso de uma rede com 49 nós.

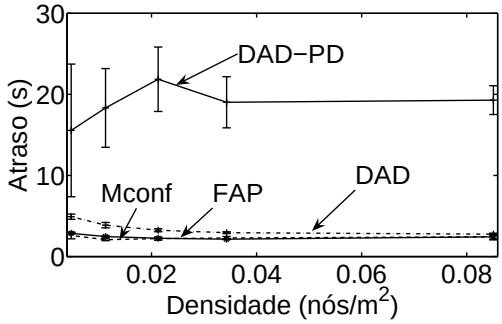
As Figuras 3.12(a) e 3.12(b) mostram que a densidade tem um impacto pequeno sobre a carga de controle e o atraso do FAP, do DAD, e do MANETconf. A carga de controle do DAD-PD diminui com a densidade, pois a inicialização é baseada em diversos eventos de união de partição. Com mais vizinhos, o processo de união de partições é simplificado, pois existe uma probabilidade maior de que diversos processos sejam iniciados ao mesmo tempo, envolvendo o mesmo conjunto de nós, reduzindo a carga de controle do protocolo. Apesar de o MANETconf apresentar baixa carga de controle e baixo atraso, esse protocolo não pode solucionar todas as colisões de endereço, como mostrado na Figura 3.12(c), devido a perdas de mensagens.

A Figura 3.13(a) mostra que o aumento do tamanho da rede tem impacto sobre todos os protocolos durante a inicialização abrupta, devido ao uso de inundações. O DAD-PD, contudo, sofre uma influência maior do número de nós do que os outros protocolos, pois ele inunda a rede mais vezes devido aos falsos positivos nas detecções de união de partições. O FAP apresenta uma carga de controle menor que o MANETconf e ambos os protocolos apresentam uma carga menor que o DAD-PD.

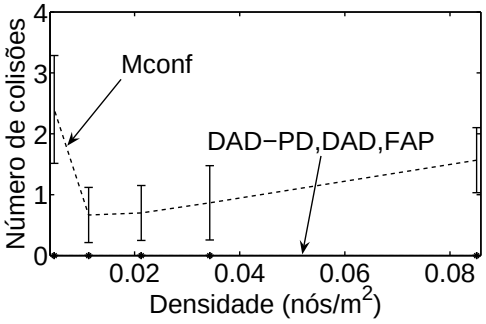
Pela Figura 3.13(b), pode-se concluir que o DAD-PD possui o maior atraso médio de convergência e se torna instável quando a rede tem mais do que 49 nós. Também se observa que o FAP apresenta um atraso de convergência menor que o do DAD e do MANETconf em redes com número alto de nós. O FAP reduz em até 24,18% o atraso do MANETconf e em até 49,73% o atraso do DAD. Após detectar uma



(a) Sobrecarga de controle de acordo com a densidade.



(b) Atraso médio para a alocação de um endereço de acordo com a densidade da rede.



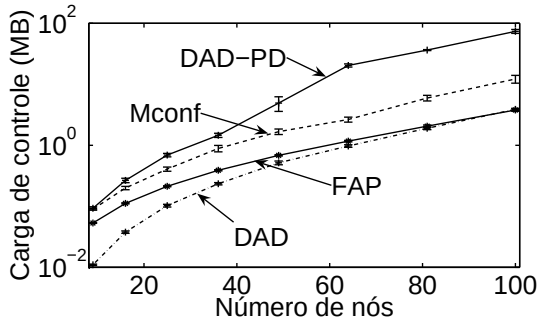
(c) Número de colisões de endereço de acordo com a densidade da rede.

Figura 3.12: Impacto da densidade sobre a rede.

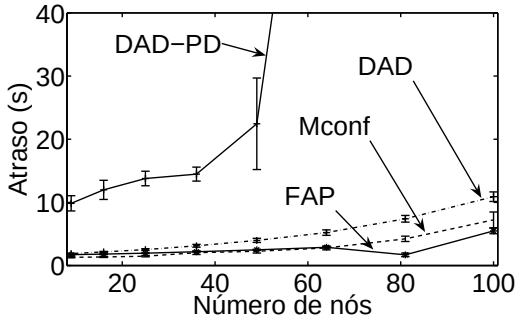
colisão de endereço durante a inicialização, ao invés de imediatamente escolher um novo endereço, o FAP armazena por um período curto de tempo o endereço que está sendo alocado por outros nós para reduzir as chances de selecionar um novo endereço que já tenha sido alocado. Evitando mais colisões, o FAP também reduz os atrasos.

A Figura 3.13(c) mostra que o MANETconf e o DAD são mais propensos a colisões de endereço quando se aumenta o número de nós. Esse efeito ocorre com o MANETconf devido às uniões de partições simultâneas e no DAD devido às perdas de mensagens, mais frequentes com o aumento do tamanho da rede.

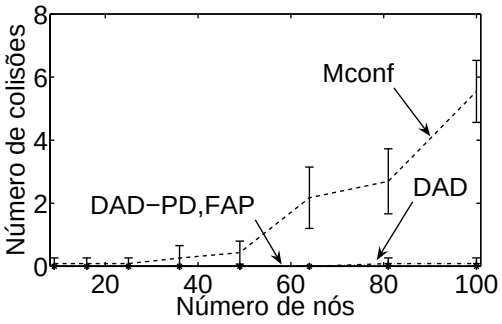
Por fim, a Figura 3.14(a) mostra o impacto da mobilidade sobre todos os protocolos durante a inicialização abrupta. Foi utilizado o modelo de mobilidade de nós *random waypoint* com tempo de pausa de 1 s e velocidade de nós variável. Observa-se que diferentes velocidades dos nós não causam grande impacto sobre a carga de controle dos protocolos, as quais seguem o mesmo comportamento observado para os outros cenários de inicialização abrupta. Pela Figura 3.14(b), pode-se concluir que o DAD-PD possui o maior atraso médio de convergência, enquanto que o MANETconf apresentou o menor atraso. O FAP apresentou um atraso próximo do MANETconf, apresentado um bom desempenho. Cabe observar que, de acordo



(a) Sobrecarga de controle de acordo com o número de nós na rede.



(b) Atraso médio de acordo com o número de nós na rede.



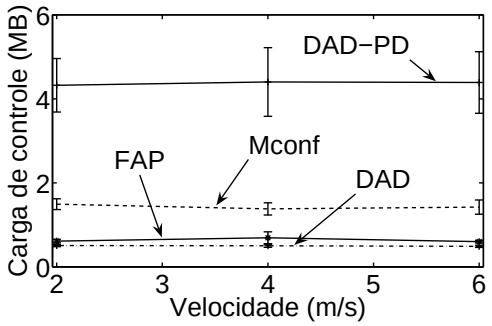
(c) Número de colisões de acordo com o número de nós na rede.

Figura 3.13: Impacto do número de nós sobre a rede.

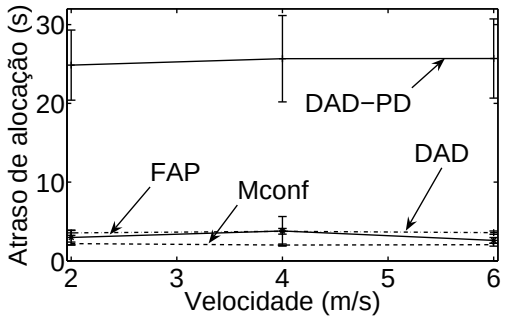
com a Figura 3.14(c), o MANETconf e o DAD apresentam uma probabilidade de colisão não nula, o que os desqualifica como protocolos de alocação de endereços.

3.3.4 Avaliação da união de partições

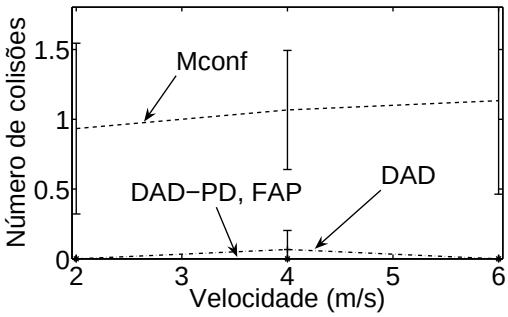
A última análise é relativa ao desempenho dos protocolos em uniões de partições. Foi variado o número de eventos de união de partições em um cenário estático composto por 50 nós através da ativação/desativação de alguns nós responsáveis por conectar partições disjuntas. Em todos os protocolos, os eventos de união de partições foram iniciados apenas após a inicialização da rede ter sido finalizada. As nuvens que formam cada partição possuem uma densidade de aproximadamente $0,03$ nós/m^2 para garantir que a densidade de toda a rede é de aproximadamente $0,01$ nós/m^2, simulando um cenário típico de rede comunitária. Os nós em cada partição são ativados simultaneamente e os nós que interconectam as partições são ativados em momentos específicos. A Figura 3.15(a) mostra a carga de controle após a detecção do primeiro evento de união de partições. O DAD praticamente não apresenta nenhuma carga de controle, pois ele não detecta eventos de união de partições. A carga de controle do DAD é composta apenas por AREQs enviados pelos nós que conectam as partições quando eles são ativados. O DAD-PD apre-



(a) Sobrecarga de controle de acordo com a velocidade dos nós na rede.



(b) Atraso médio de acordo com a velocidade dos nós na rede.

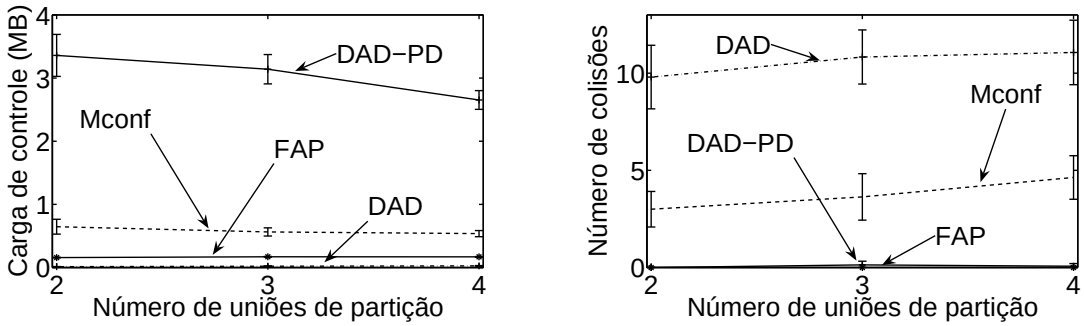


(c) Número de colisões de acordo com a velocidade dos nós na rede.

Figura 3.14: Impacto da mobilidade dos nós sobre os protocolos de endereçamento, assumindo o modelo *random waypoint*.

senta uma carga de controle até $21,35$ vezes maior que o FAP, pois o FAP reduz o número de AREQs durante o procedimento de união de partições. No FAP, apenas metade dos nós com endereços colididos escolhem um novo endereço e enviam uma mensagem AREQ, enquanto que, no DAD-PD, todos os nós devem enviar uma mensagem AREQ, e cada colisão implica em pelo menos mais uma inundação na rede. O aumento do número de uniões de partições não causa impactos no FAP. O número de mensagens *Address Filter* aumenta com o número de partições, mas essas mensagens são enviadas em *unicast* entre dois vizinhos e não causam impacto na carga de controle. Contudo, o número de mensagens *Partition* e AREQ, que são mensagens enviadas por inundação para anunciar o filtro da outra partição e para resolver colisões, respectivamente, decresce. Isso ocorre porque o número de nós em cada partição é menor, o que reduz o impacto causado pela inundação dessas mensagens. Isso também explica a redução da carga de controle no MANETconf, a qual é até $4,10$ vezes maior do que a carga do FAP.

A Figura 3.15(b) mostra o número de colisões de endereço após os eventos de união de partições. O DAD, como era esperado, não solucionou nenhuma das colisões, pois ele não foi projetado com esse fim. O MANETconf apresenta menos colisões de endereço do que o DAD, porque esse protocolo possui mecanismos para



(a) Carga de controle.

(b) Número de colisões de endereço.

Figura 3.15: Impacto dos eventos de união de partições em uma rede com 50 nós.

tratar a união de partições, embora esses mecanismos não sejam robustos. O FAP não apresentou nenhuma colisão de endereço após a simulação, mas o DAD-PD apresentou uma pequena probabilidade de colisão causada por perda de mensagens. O FAP evita esse tipo de colisão de endereços, por causa do mecanismo que detecta a perda de mensagens e inicia procedimentos de união de partições falsos. Como resultado, o FAP solucionou todas as colisões de endereço e também apresentou uma baixa sobrecarga de controle.

3.4 Considerações sobre endereçamento em redes ad hoc

O endereçamento é um grande desafio para todas as redes de nova geração que não possuem uma infraestrutura fixa e acessível para todos os nós em todos os momentos. Nessa tese, é apresentada uma proposta de protocolo para endereçamento em redes ad hoc chamada FAP, além de uma análise sobre este e outros protocolos da literatura.

O protocolo proposto é baseado na idéia de um mecanismo distribuído e autogerenciável para o endereçamento de nós em redes ad hoc. Esse protocolo se mostrou adequado para o uso em redes ad hoc, pois é robusto mesmo quando a rede é dinâmica, existem problemas no canal de transmissão, acontecem partições na rede e os nós entram e saem da rede com frequência. A idéia chave que gerou todos esses ganhos foi o uso de filtros de endereço e suas assinaturas, que evitam colisões de endereço, reduzem a carga de controle e diminuem o atraso para alocar um endereço para um nó. A utilização de assinaturas de filtros permitiu a criação de um mecanismo de detecção de partições e de incoerências de estado na rede preciso e com baixa sobrecarga. De fato, as assinaturas de filtro ainda trazem outras vantagens, como um mapeamento adequado entre o identificador de partição e o

conjunto de endereços que fazem parte da partição. Enquanto no FAP a atualização do identificador de partição é automática a cada mudança de membros no conjunto de endereços alocados, nos demais protocolos é necessária alguma forma de detectar modificações no conjunto e fazer essa atualização em toda a rede.

Na análise realizada, o FAP se mostrou eficiente para a resolução de todas as colisões de endereço, mesmo durante os procedimentos de união de partições. Isso foi alcançado porque o FAP é capaz de detectar todos os eventos de união de partições, além de ser robusto a perdas de mensagens. O procedimento de inicialização do FAP é simples e eficiente, requerendo uma carga de controle similar à carga de controle do DAD, o qual é um protocolo com baixa sobrecarga, mas que não trata uniões de partições. Além disso, os outros resultados da simulação mostraram que o FAP sofre menos com o aumento do número de nós na rede do que outras propostas, como o DAD-PD e o MANETconf. No processo de união de partições, o FAP apresentou uma carga de controle até $21,35$ vezes menor que o DAD-PD e até $4,10$ vezes menor do que a do MANETconf. Ambos os resultados são explicados pela detecção acurada do FAP de uniões de partições e do baixo número de eventos de inundação que o FAP requer. Com isso, pode-se afirmar que o FAP é um protocolo robusto e eficiente para o uso em redes ad hoc.

Parte II

Virtualização de redes

Capítulo 4

Conceitos e propostas para a virtualização de redes*

O modelo da Internet é baseado em dois pilares principais: o serviço de transferência de dados fim-a-fim e a pilha TCP/IP [59]. Esses dois pilares garantem que o núcleo da rede é simples e transparente, enquanto toda a inteligência fica nas extremidades. Essa escolha arquitetural torna mais simples o suporte a novas aplicações, pois não existe necessidade de se modificar o núcleo da rede para permitir que uma aplicação de rede seja executada. Por outro lado, esse modelo engessa a Internet, tornando difícil solucionar problemas estruturais como a escalabilidade, o gerenciamento, a mobilidade e a segurança [60]. Atualmente, existe uma corrente forte que acredita que remendos à rede não são suficientes para atender os requisitos atuais e futuros da rede. Assim, a Internet precisaria ser reformulada para prover uma infraestrutura flexível que dê suporte à inovação na rede, o que é chamado de Internet do Futuro [60, 61].

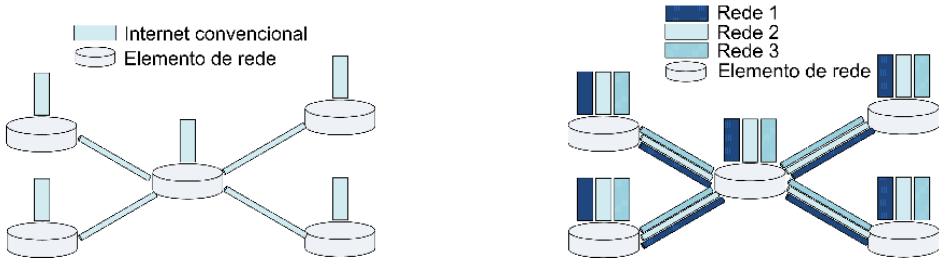
Para se definir aspectos relacionados à Internet do Futuro, começaram a surgir pesquisas para tentar tornar a Internet evolutiva e para pensar sobre o futuro da rede. Alguns projetos americanos, tais como o *New Arch Project* [60] e o *Clean Slate* [62], tinham como objetivo levantar diretrizes para a construção de uma nova arquitetura para a Internet. Esses projetos colocaram a seguinte questão em discussão: "se fosse possível desenvolver um novo projeto da Internet a partir do zero, mas tendo o conhecimento que se tem hoje em dia, como seriam feitas as decisões de projeto mais básicas?" Tal filosofia começou a ser amplamente discutida, pois a idéia de reinventar a Internet permitiria ultrapassar algumas limitações estruturais da rede, tais como a ausência de suporte para segurança e mobilidade; permitir a criação de novas classes de serviços e aplicações, como a disseminação de conteúdo, o sensoriamento do mundo físico, entre outros; permitir que a Internet continue a

*Esse capítulo é parcialmente baseado em [5, 38].

ser uma plataforma para inovações e, assim, uma engrenagem para o crescimento econômico e para a prosperidade da sociedade [62]. A filosofia de "partir do zero" (*clean-slate*) traz a vantagem de permitir a inovação sem limitações com questões como a compatibilidade com outros mecanismos existentes, o que possibilitaria deixar de fazer propostas incrementais para fazer propostas inovadoras. A idéia por trás desse pensamento é que, olhando a longo prazo, problemas pontuais e suas soluções a curto prazo não deixam de ser resultados, mas se os pesquisadores se restringirem a pensar apenas a curto prazo, então jamais será desenvolvida uma solução consistente em uma direção a longo prazo. As soluções a curto prazo solucionam pequenos problemas, mas nem sempre são compatíveis umas com as outras, podendo levar a rede a inconsistências e tensões que ficam mais evidentes com o passar do tempo, tornando cada vez mais difícil fazer a rede progredir [60].

Novas propostas começaram a surgir para tentar tornar a Internet mais flexível e evolutiva. Duas novas correntes surgiram, uma chamada de purista [4, 63], descrita na Figura 4.1(a), e a outra de pluralista [36, 60, 64], descrita na Figura 4.1(b). Na corrente purista, acredita-se que a solução para a rede será em um modelo semelhante ao que existe hoje em dia, com uma arquitetura com uma única pilha de protocolos capaz de solucionar todos os problemas que surgirem. Em geral, os defensores da corrente purista acham que a Internet é capaz de evoluir e se transformar para poder dar suporte aos novos requisitos, apesar de todas as dificuldades já conhecidas. Com isso, a rede deveria ter uma arquitetura monolítica, mas flexível o suficiente para dar suporte às novas aplicações [38]. A corrente pluralista acredita em um núcleo de rede que dê suporte a diversos tipos de rede funcionando em paralelo. Dessa forma, cada uma das redes poderia atender a um requisito diferente, de forma mais especializada, garantindo uma maior eficiência. Esse modelo também permitiria uma maior inovação no núcleo, pois o teste de novos mecanismos poderia ser feito sobre o mesmo substrato físico, sem que os testes interferissem com o funcionamento da rede de produção e vice-versa. Uma vantagem intrínseca do modelo pluralista é que ele provê compatibilidade natural com a Internet atual, que poderia ser uma entre as diversas pilhas de protocolo funcionando em paralelo. Além disso, o modelo pluralista é uma abordagem mais simples para desenvolver uma rede que possa solucionar todos os problemas conhecidos, assim como todos os problemas ainda não conhecidos/criados. Para tanto, bastaria criar uma nova rede virtual que fosse especializada na solução do problema em questão.

Todas as propostas pluralistas são baseadas na mesma idéia de que múltiplas redes devem funcionar sobre o mesmo substrato físico [36], mesmo que tenham formatos de pacotes, esquemas de endereçamento e protocolos diferentes. A tecnologia de virtualização é essencial para que múltiplas redes operem sobre um mesmo substrato físico. Surge o conceito de virtualização de redes, que permite colocar diversas



(a) Modelo de arquitetura purista atual: apenas uma pilha de protocolos sendo executada no substrato físico.

(b) Modelo da arquitetura pluralista: diferentes pilhas de protocolo sendo executadas ao mesmo tempo sobre o mesmo substrato físico.

Figura 4.1: Modelos purista e pluralista para as arquiteturas de rede.

redes em paralelo, cada uma com sua própria pilha de protocolos, em paralelo, sem que uma interfira na outra. Se a arquitetura da Internet fosse virtualizada, seria possível, por exemplo, já ter criado uma rede com suporte a IPv6 ou com suporte a *multicast* sem precisar mudar em nada o núcleo da Internet como conhecemos, permitindo que as soluções para os requisitos que vão surgindo sejam aplicadas sem que uma nova solução interfira na outra. Com isso, a arquitetura da rede como um todo passa a ser evolutiva e as barreiras para a inovação são derrubadas.

Embora a virtualização de redes seja um conceito interessante e que apresenta diversas vantagens, ele ainda apresenta muitos desafios. Um dos primeiros desafios é como criar as redes em paralelo sem que uma rede interfira na outra. Iniciativas como as *Virtual Private Networks* (VPN) e as redes *overlay* permitem a coexistência de ambientes com características diferentes em paralelo, mas não apresentam todas as características desejáveis para uma virtualização de redes isolada. Entre essas características, deseja-se que as redes virtuais possam operar com qualquer tipo de mecanismos como se fossem redes não virtualizadas, ou seja, a rede virtual não deveria sofrer restrições por ser uma rede virtual e não deveria ficar ciente da existência de outras redes virtuais.

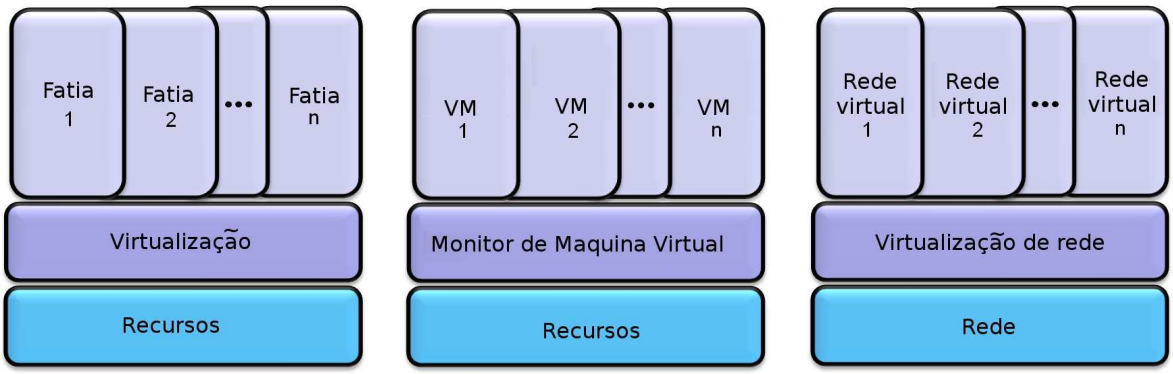
Boa parte dos avanços na área de virtualização vem do desenvolvimento de novas plataformas de experimentação. Muitas das iniciativas atuais para testar novos mecanismos para a Internet em larga escala vêm sendo criadas com o auxílio da virtualização de redes, pois isso permite que a rede de teste seja utilizada por diversos pesquisadores em paralelo, sem que o experimento de um pesquisador interfira nos experimentos de outros pesquisadores.

4.1 O conceito de virtualização de redes

A virtualização é uma abstração do recurso que permite o compartilhamento no tempo do recurso sem que o usuário possa perceber que o recurso está sendo compartilhado. Assim, os usuários do recurso são representados por diversas fatias, como mostrado na Figura 4.2. Essa abstração é frequentemente implementada como uma camada de *software* que provê "interfaces virtuais fatiadas" muito semelhantes à interface real. A coexistência de diversas fatias virtuais sobre o mesmo recurso é possível porque a camada de virtualização quebra o acoplamento entre os recursos reais e a camada superior. A Figura 4.2 mostra dois exemplos de virtualização: a virtualização de computadores e a virtualização de redes. A abstração da virtualização de computadores é implementada pelo *Virtual Machine Monitor* (VMM), o qual provê para as máquinas virtuais (*Virtual Machines* - VMs) uma interface muito semelhante à interface de *hardware* do computador pessoal, a qual inclui o processador, a memória, os dispositivos de entrada e saída (E/S) etc. Assim, cada máquina virtual (VM) acredita estar funcionando diretamente sobre o *hardware* físico, mas de fato o *hardware* físico é compartilhado entre diversas VMs. Esse tipo de compartilhamento de recursos é chamado de fatiamento, porque as máquinas virtuais ficam isoladas: uma VM não pode interferir com as demais.

A virtualização de computadores é amplamente utilizada em *data centers* para permitir que diversos servidores operem em uma única máquina física. Essa técnica reduz o consumo de energia e reduz custos com manutenção, embora a característica mais importante dos ambientes virtualizados seja a flexibilidade. De fato, cada máquina virtual pode ter o seu próprio sistema operacional, aplicações, regras de configuração e procedimentos administrativos. A flexibilidade de se executar o que for desejado em uma fatia virtual, como pilhas de protocolos diferentes e customizáveis, é a maior motivação para a aplicação da idéia da virtualização no ambiente de redes [64]. Como é mostrado na Figura 4.2, a virtualização de redes é análoga a virtualização de computadores, mas, nesse caso, o recurso físico compartilhado é a rede. O conceito de virtualização de redes não é novo e vem sendo utilizado em tecnologias como as *virtual private networks* (VPNs) e *virtual local area networks* (VLANs). Atualmente, esse conceito foi estendido e as novas técnicas permitem que até mesmo o roteador seja completamente virtualizado.

O uso da virtualização de redes cria novas primitivas de gerenciamento para as redes virtuais. Com isso, agora, passa a ser possível instanciar, apagar e monitorar redes virtuais, migrar elementos de rede e realocar os parâmetros de uso de recursos de cada rede virtual, como uso de processamento, memória, banda passante, entre outros. Essas primitivas fazem com que a virtualização de redes possa alocar diversas redes virtuais diferentes em paralelo, o que permite a implementação do pluralismo



(a) Visão esquemática do conceito de fatias utilizando-se a virtualização.

(b) Fatias virtuais em um *hardware* físico de um computador.

(c) Fatias virtuais em uma rede.

Figura 4.2: Fatias de recursos em diferentes ambientes virtualizados.

de redes. De fato, a virtualização de redes garante as seguintes características [38]:

- **criação de múltiplas redes customizáveis -** Na arquitetura pluralista, múltiplas redes virtuais são executadas em paralelo. A primitiva de instanciação pode ser usada para instanciar elementos de redes virtuais, como roteadores virtuais, comutadores virtuais ou enlaces virtuais, permitindo que diversas redes possam ser instanciadas e possam funcionar em paralelo. Cada rede virtual pode ter a sua própria pilha de protocolos, topologia de rede, política de administração, entre outros. Isso permite a inovação na rede e a criação de novos modelos de negócio [36]. Com a virtualização da rede, um provedor de serviços pode alocar caminhos virtuais fim-a-fim com características específicas, como o provimento de qualidade de serviço. Assim, novos serviços podem ser facilmente desenvolvidos e novos serviços podem ser oferecidos na rede, aquecendo o mercado de serviços promovido pela Internet;

- **gerenciamento flexível -** A camada de virtualização de redes quebra o acoplamento entre a lógica usada para construir a tabela de encaminhamento e o *hardware* físico que encaminha os pacotes [65]. Assim, a primitiva de migração permite mover um elemento de rede virtual de um *hardware* para outro, sem mudar a topologia física da rede como um todo ou a topologia lógica da rede virtual. Além disso, a engenharia de tráfego e as técnicas de otimização podem ser utilizadas em parceria com a técnica de migração para otimizar o uso dos recursos da rede, seja para aumentar o desempenho no encaminhamento de pacotes, para garantir QoS para uma determinada rede, para reduzir o consumo de energia na rede, entre diversas outras funções objetivo [66];

- **controle em tempo real -** A arquitetura das redes virtuais também dá

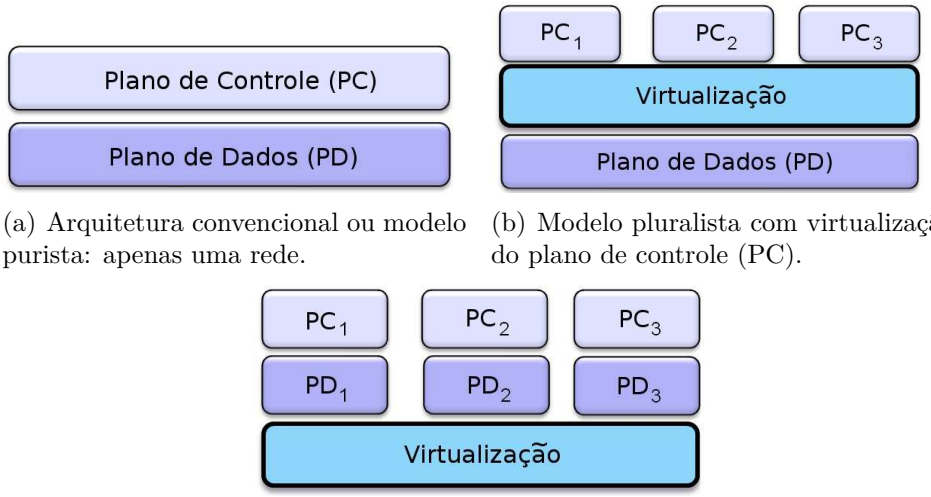
suporte ao controle em tempo real dos recursos utilizados em cada rede virtual, pois é possível ajustar a qualquer momento o volume de recursos físicos que é atribuído para cada rede, garantindo o cumprimento dos acordos de nível de serviço (*Service Level Agreements - SLAs*;

- **monitoramento -** A virtualização das redes também traz um conjunto de ferramentas de monitoração necessárias para medir variáveis de interesse, como banda disponível, uso de processador e memória, atrasos fim-a-fim etc. É importante notar que, embora essas variáveis dos ambientes virtuais sejam interessantes para tornar eficiente o funcionamento da rede, elas também tornam mais complexa a tarefa de gerenciamento, que passa a demandar mais inteligência e capacidade de lidar com um volume de dados maior.

4.2 Modelos conceituais de redes virtuais

A virtualização de redes pode ser feita de diferentes formas. Dois fatores são importantes para entender o modelo de virtualização de redes: o plano de controle e o plano de dados. O plano de controle é o conjunto de protocolos e mecanismos que definem como a rede deve ser controlada. O plano de controle é responsável por mecanismos como o roteamento e o gerenciamento, por exemplo. O plano de dados diz respeito ao encaminhamento de dados, o que inclui as regras de encaminhamento, filtragem e priorização, assumindo que as regras para todas essas tarefas já foram definidas pelo plano de controle. Com base nesses conceitos, a Figura 4.3 compara duas abordagens básicas para a virtualização de um elemento de rede.

A Figura 4.3(a) mostra a arquitetura de um elemento convencional de rede, com um único plano de controle e um único plano de dados. Virtualizar um elemento de rede significa que a camada de virtualização deve ser colocada em algum nível da arquitetura do elemento de rede para permitir a coexistência de múltiplos elementos de rede sobre um mesmo elemento de rede física. Assumindo a camada de virtualização colocada entre o plano de controle e o plano de dados, então apenas o plano de controle está sendo virtualizado, como mostrado na Figura 4.3(b). Nesse caso, o plano de dados é compartilhado por todas as redes virtuais e cada rede virtual pode executar o seu próprio *software* de controle. Comparado com a arquitetura de rede convencional, essa abordagem aumenta significantemente a programabilidade da rede, pois passa a ser possível executar múltiplas pilhas de protocolo personalizadas, ao invés de uma única pilha de protocolos fixa. Por exemplo, é possível programar uma pilha de protocolos na rede 1, que é diferente das pilhas nas redes 2 e 3, como mostrado na figura. Na segunda abordagem de virtualização de redes, ambos os planos de dados e de controle são virtualizados, como mostrado na



(a) Arquitetura convencional ou modelo purista: apenas uma rede.

(b) Modelo pluralista com virtualização do plano de controle (PC).

(c) Modelo pluralista com virtualização do PC e do plano de dados (PD).

Figura 4.3: Diferentes abordagens para virtualização de redes comparadas ao modelo tradicional sem virtualização.

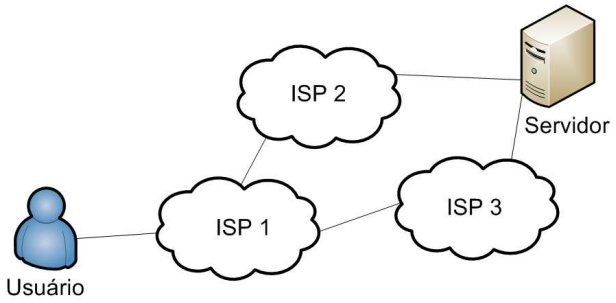
Figura 4.3(c). Nesse caso, cada elemento de rede virtual implementa o seu próprio plano de dados, além do plano de controle, aumentando ainda mais a programabilidade da rede. Essa abordagem permite customizar o plano de dados com o custo de perda de desempenho, pois o plano de dados não é mais dedicado à execução de uma tarefa comum.

É importante mencionar que a abordagem que virtualiza apenas o plano de controle pode ser subdividida em mais subcategorias dependendo do nível de isolamento no plano de dados. Se existe um isolamento forte, então cada rede virtual tem acesso apenas a sua parte do plano de dados e não pode interferir com outras redes virtuais. Por outro lado, se o plano de dados é realmente compartilhado entre os planos de controle, então é possível que um plano de controle interfira nos demais. Por exemplo, seria possível que um plano de controle esgotasse o espaço de memória com entradas para sua tabela de encaminhamento, causando o descarte de pacotes de outras redes virtuais que não puderam instalar suas próprias rotas na tabela de encaminhamento. A decisão entre um isolamento forte (divisão de recursos) e isolamento fraco (compartilhamento) é análogo à decisão entre comutação de circuito e de pacotes.

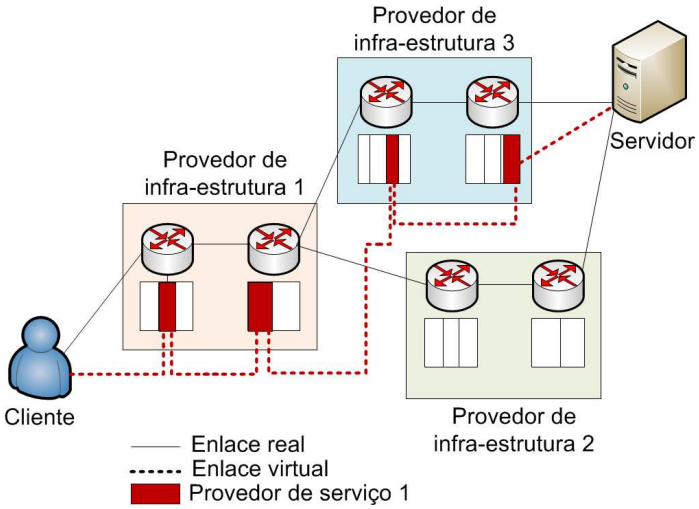
4.3 Modelo de entidades de redes virtualizadas

O projeto *Concurrent Architectures are Better than One* (CABO) é um exemplo de iniciativa internacional para a construção de um modelo econômico para a utilização da rede virtualizada, estando entre as primeiras iniciativas para a construção

de um núcleo pluralista na rede. O objetivo do CABO [36] é reestruturar a forma como o serviço é prestado pelos ISPs na Internet. Atualmente, os ISPs têm duas funções principais: o gerenciamento da infraestrutura e o provimento de serviços. A arquitetura do CABO visa reestruturar a Internet de forma que esses dois papéis sejam separados para permitir o desenvolvimento de novos protocolos e serviços. Para tornar isso possível, o CABO defende a utilização de roteadores virtualizados, que permitam que cada máquina virtual seja isolada das demais máquinas. Com isso, cada máquina virtual poderia implementar os protocolos de rede com parâmetros diferentes ou ainda uma arquitetura completamente diferente da atual para prover diferenciação de serviço ou segurança, sem interferir no tráfego das outras máquinas virtuais.



(a) Encaminhamento de dados através de múltiplos ISPs.



(b) Encaminhamento de dados utilizando o CABO com um único provedor de serviço e diversos provedores de infraestrutura.

Figura 4.4: Diferença entre Internet atual e CABO.

Atualmente, um ISP não pode diferenciar o seu serviço, pois não possui controle sobre todos os ASes entre as duas extremidades de uma comunicação, como pode ser visto na Figura 4.4(a). Uma vez que alguns serviços, como o QoS e a segurança, só podem ser prestados se houver um controle fim-a-fim na comunicação, os ISPs não tem como prover esse tipo de serviço, mesmo existindo um apelo comercial.

Com a proposta do CABO, os provedores de serviço contratariam os serviços dos provedores de infraestrutura durante uma comunicação fim-a-fim, podendo controlar todos os parâmetros e diferenciar o serviço em todo o caminho entre os dois nós, como mostrado na Figura 4.4(b).

Dessa forma, os provedores de serviço poderiam prestar serviços fim-a-fim diferenciados, modificando os roteadores dos provedores de infraestrutura que estejam entre dois nós em uma comunicação. A desvantagem dessa proposta é que, para o seu sucesso, é necessário que todos os roteadores da Internet utilizem o CABO. Além disso, o sucesso do modelo econômico proposto depende da separação das funções do ISP, mas não há como garantir apenas com o CABO que essa separação irá ocorrer.

Seguindo o modelo proposto no CABO, o projeto europeu *4ward* propôs um modelo mais completo de como virtualizar a Internet [67]. No cenário atual da Internet, os provedores de serviço (*Internet Service Providers* - ISP) oferecem o acesso à rede através de sua infraestrutura para usuários finais ou para outros ISPs. Ao considerar o uso de redes virtuais, onde diversos provedores de infraestrutura podem existir e competir, uma nova camada de abstração deve ser acrescentada, adicionando duas novas entidades ao modelo da Internet do Futuro: os provedores de redes virtuais e os operadores de redes virtuais [67, 68]. O esquema desse novo modelo está na Figura 4.5.

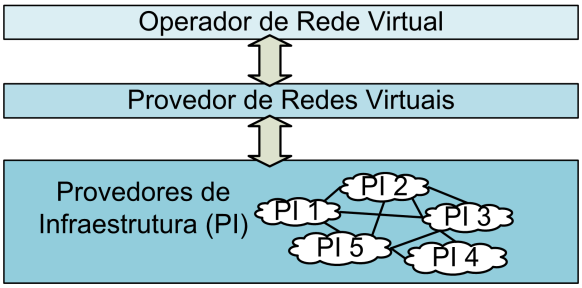


Figura 4.5: Esquema do modelo de serviços para redes virtualizadas.

Nesse cenário, o provedor de infraestrutura possui a infraestrutura física, gerenciando os recursos físicos e vendendo fatias de rede para o provedor de redes virtuais. O provedor de redes virtuais é responsável por unir diversos provedores de infraestrutura para formar uma topologia virtual de acordo com os requisitos do operador de rede virtual. Assim, o provedor de redes virtuais faz o mapeamento da rede virtual sobre a rede física, alocando os recursos em cada provedor de infraestrutura conforme o requisitado pelo cliente. A união das fatias de rede físicas é, então, oferecida ao operador de redes virtuais. O operador de rede virtual é responsável por instalar e operar a rede virtual, oferecendo serviços na rede ou oferecendo a sua rede virtual para alguma outra entidade.

4.4 Plataformas de virtualização

Nessa seção, os conceitos e modelos para virtualização de redes são exemplificados através de plataformas reais de virtualização. Uma vez que um roteador pode ser construído a partir de um computador pessoal, plataformas de virtualização de computadores podem ser usadas para a construção de roteadores virtuais. A seguir, são descritos os tipos de virtualização de computadores e algumas das principais plataformas de virtualização que são usadas para a criação de redes virtuais.

4.4.1 Virtualização de computadores

A virtualização é amplamente utilizada para a consolidação de servidores em *data centers*, para reduzir custos com manutenção, espaço e energia. Com isso, diversas plataformas foram desenvolvidas, cada uma com suas próprias peculiaridades. O elemento fundamental na virtualização de computadores é o hipervisor, também chamado de monitor de máquinas virtuais. O hipervisor aloca recursos do sistema para cada ambiente virtual. Ele pode ser instalado como uma aplicação ou como parte do sistema operacional. Em geral, o hipervisor "engana" cada ambiente virtual através da emulação de dispositivos, instruções, entre outros, fazendo o ambiente virtual acreditar que tem acesso exclusivo sem compartilhar o substrato físico com outros ambientes virtuais [69]. A virtualização de computadores pode ser feita em diferentes níveis [70]:

- Abstração da *Instruction Set Architecture* (ISA) - Baseada em emulação, esse tipo de virtualização é implementada através da execução do sistema virtual através de um mapeamento das instruções para o conjunto de instruções nativas. Uma consequência do uso desse tipo de virtualização é a perda de desempenho na execução das tarefas do sistema virtual. O QEMU [71] é um exemplo de sistema que usa esse tipo de virtualização.

- Camada de abstração de *hardware* (*Hardware Abstraction Layer* - HAL) - o hipervisor da máquina virtual simula a arquitetura completa da máquina para o sistema virtual. O sistema virtual acredita estar sendo executado diretamente sobre um *hardware* físico, sem perceber que é virtualizado. Entre os sistemas classificados nesse tipo de virtualização estão o Xen [72], o VMware [73], o Virtual PC [74] e o VirtualBox [75].

- Nível de sistema operacional - Nesse tipo de virtualização, conjuntos de processos são executados de forma isolada sobre o mesmo sistema operacional. A cada ambiente virtual é possível atribuir uma fatia dos recursos físicos e programar qualquer tipo de aplicação ou serviço. A restrição principal desse tipo de virtualização é que todos os ambientes devem utilizar o mesmo sistema operacional. Entre os exemplos desse tipo de virtualização estão o *FreeBSD Jail* [76] e o OpenVZ [77].

- Nível de Aplicação - A virtualização no nível de aplicação consiste na abstração da camada de execução. Essa solução é utilizada para fazer com que um determinado *software* funcione sobre qualquer sistema operacional. O principal exemplo desse tipo de virtualização é o *Java Virtual Machine* (JVM).

4.4.2 Plataformas de virtualização de redes

A virtualização de redes possui características que a tornam diferente da virtualização de computadores, muito embora algumas das soluções de virtualização de computadores possam ser usadas para criar um ambiente de rede virtualizado. Em alguns modelos de redes virtuais, plataformas de virtualização de PC são utilizadas, como o Xen, o OpenVZ e o Linux VServer [78].

O Linux VServer, que é um tipo de virtualização no nível de sistema operacional, é utilizado como plataforma de virtualização do PlanetLab, pois ele é capaz de criar isolamento de espaço de nomes e de desempenho entre ambientes virtuais (*slivers*) em uma mesma máquina física [79]. O Linux VServer, no entanto, é apenas usado para compartilhar a máquina física entre ambientes virtuais, mas não é suficiente para criar a rede virtual. Assim, a virtualização de rede é feita utilizando-se o *PlanetLab Virtualized Network Access* (VNET) [80]. O VNET permite a criação de *sockets* e garante o isolamento entre tráfegos de ambientes virtuais. O VNET foi desenvolvido de forma a ser o mais transparente possível, de forma que os usuários da rede não precisem saber que o VNET está virtualizando o seu acesso.

O OpenVZ [77] é outra plataforma baseada no tipo de virtualização de computadores no nível de sistema operacional. Também muito utilizada no contexto de redes virtualizadas, o OpenVZ se destaca pelo amplo suporte a primitivas como a migração de redes [66]. Assim como o Linux VServer, o OpenVZ tem como restrições de flexibilidade a imposição de que todos os ambientes virtuais, ou *containers*, como são chamados no OpenVZ, devem funcionar sobre um mesmo sistema operacional.

Uma plataforma também utilizada para a virtualização de redes, mas que dá maior flexibilidade na programação dos ambientes virtuais é o Xen [72]. Por ser baseado em virtualização com camada de abstração de hardware (*H*ardware Abstraction Layer - HAL), o Xen permite que cada ambiente virtual possa funcionar sobre qualquer sistema operacional, tendo total controle de todas as camadas da arquitetura da rede. O Xen possui dois modos de operação, sendo um baseado em virtualização total e o outro baseado em paravirtualização. Na virtualização total, o *hardware* é totalmente emulado para a máquina virtual e o sistema operacional visitante executa ações protegidas. Uma vez que a máquina virtual é executada no espaço de aplicação do sistema operacional hospedeiro, essas ações privilegiadas não poderiam ser executadas. Para evitar erros, o hipervisor intercepta essas ações, as

executa e retorna o resultado para a máquina virtual. Com isso, a máquina virtual perde desempenho. Na paravirtualização, o sistema operacional da máquina virtual é levemente modificado para chamar o hipervisor sempre que uma ação privilegiada é necessária. Com isso, o hipervisor pode ser projetado de forma mais simples e o desempenho do ambiente virtual melhora. [1]

Embora diferentes plataformas de virtualização possam ser utilizadas para a construção de redes virtualizadas, existem ainda outras propostas de virtualização de rede que são baseadas em conceitos diferentes. Esse é o caso do OpenFlow [81]. O OpenFlow é baseado no conceito de que os elementos da rede devem ser simples e facilmente programáveis. Nesse caso, não existe um hipervisor dentro de cada elemento da rede, mas sim um hipervisor que controla o acesso dos planos de controle de cada rede virtual aos elementos da rede.

Novas abordagens para virtualização de rede também são criadas com base na junção de diferentes tecnologias de virtualização. O XenFlow [82] é uma plataforma baseada em Xen e OpenFlow, cujo objetivo é criar uma rede virtualizada com amplo suporte à migração e com controle distribuído. A ideia é juntar a principal vantagem do Xen, que é o amplo suporte à inovação, uma vez que o ambiente virtual é uma máquina com sistema operacional próprio, com a principal vantagem do OpenFlow, que é a facilidade de migrar fluxos, remapeando as topologias virtuais sobre a topologia física.

Uma vez que o Xen e o OpenFlow são alvos de muitas pesquisas sobre virtualização de redes e representam duas formas distintas de virtualizar a rede, essas duas plataformas são discutidas com mais detalhes a seguir.

4.5 Virtualização de redes com a plataforma Xen

O Xen é uma das ferramentas de virtualização mais utilizadas. A arquitetura do Xen (Figura 4.6) é composta pelo hipervisor, pelas máquinas virtuais, chamadas de domínios sem privilégios (DomU), e por uma máquina virtual privilegiada, chamada de Domínio 0 (Dom0). O hipervisor controla o acesso aos recursos físicos e trata as operações de entrada e saída (E/S) realizadas pelos domínios, isolando os ambientes virtualizados. O Dom0 é um domínio de *drivers* e um domínio administrativo do sistema Xen, que possui acesso direto ao *hardware*. Por ser um domínio de *drivers*, o Dom0 possui todos os *drivers* reais dos dispositivos físicos em uso. Dessa forma, o Dom0 é responsável por fazer o interfaceamento entre os *drivers* virtuais, localizados nos DomUs, e os dispositivos físicos. Além disso, o Dom0 também é a interface de

[1] Uma vez que as aplicações de rede demandam um alto desempenho do ambiente virtual, no restante do texto, todas as referências ao Xen se referem à utilização dessa plataforma com paravirtualização.

gerenciamento entre o hipervisor e o usuário. Assim, é através do Dom0 que o usuário pode criar máquinas, modificar parâmetros e gerenciar o funcionamento do Xen.

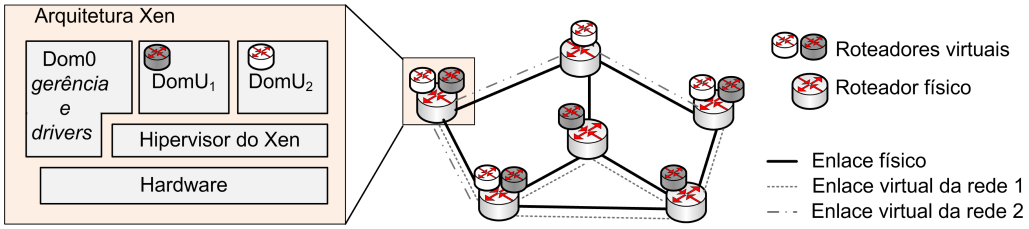


Figura 4.6: Arquitetura do Xen com duas redes virtuais. Cada máquina virtual é um roteador virtual.

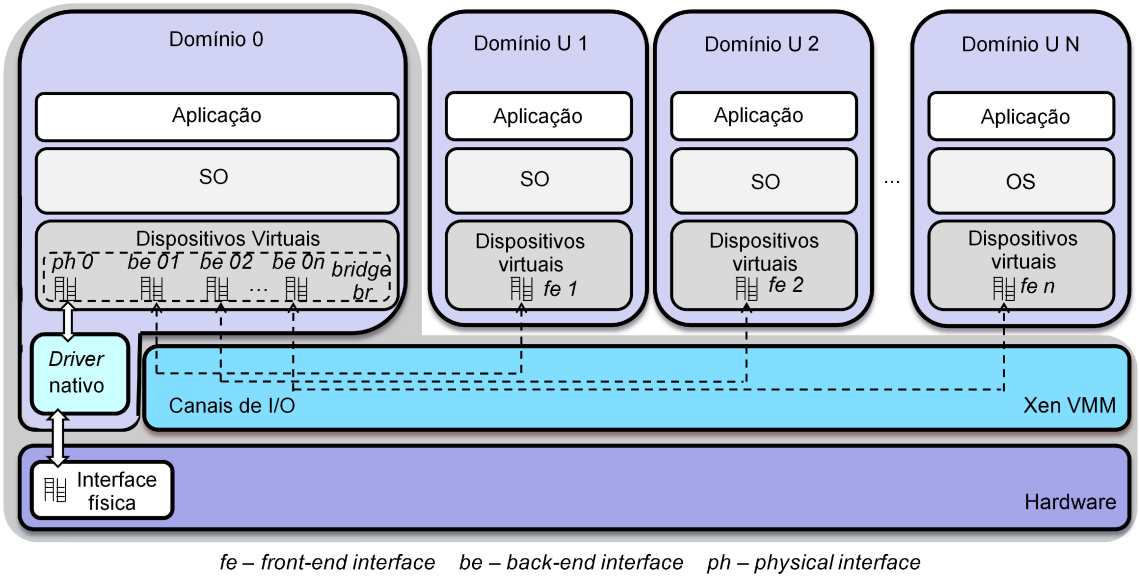


Figura 4.7: A arquitetura do Xen e o modo *b*ridge.

O Xen virtualiza uma única interface de rede física através da demultiplexação de pacotes de entrada provenientes da interface física para os DomUs e, no sentido inverso, através da multiplexação dos pacotes de saída gerados por estes DomUs. Esse procedimento é chamado de virtualização de E/S de rede e é feito no Domínio 0, que acessa diretamente os dispositivos de E/S usando seus *drivers* de dispositivo nativos, encaminhando pacotes entre as interfaces físicas e as interfaces virtuais dos DomUs. Os DomUs, por sua vez, utilizam dispositivos de E/S virtuais, controlados por *drivers* virtuais, para solicitar ao Dom0 o acesso ao dispositivo físico [83], conforme ilustrado na Figura 4.7. Cada DomU possui interfaces de rede virtuais, chamadas de interfaces *front end*, utilizadas para realizar todas as comunicações da sua rede. As interfaces *back end* são criadas no Domínio 0 e são correspondentes a cada interface *front end* em um DomU, funcionando como um *proxy* para as interfaces físicas

no Dom0. As interfaces de *front end* e *back end* são conectadas umas às outras através de um canal de E/S, que emprega um mecanismo sem cópias de memória para remapear a página contendo o pacote no domínio de destino do pacote [83]. Vale ressaltar que, para o sistema operacional de cada DomU, as interfaces *front end* funcionam como interfaces físicas, de forma que o DomU não sabe da existência da interface de *back end*. Todas as interfaces de *back end* no Dom0 estão ligadas às interfaces físicas e também umas às outras através de uma ponte virtual. Esta é a arquitetura padrão usada pelo Xen e é chamada de modo *bridge*.

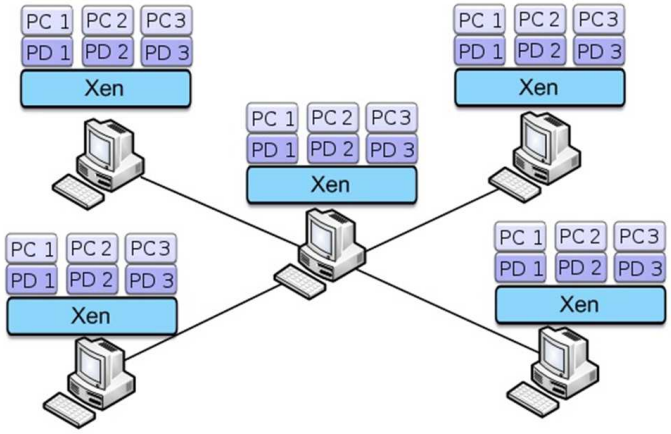
Diferentes elementos de rede virtual podem ser implementados no Xen, uma vez que essa plataforma permite que múltiplas máquinas virtuais executem simultaneamente no mesmo *hardware* [31], como mostrado na Figura 4.8(a). Nesse caso, cada máquina virtual funciona como um roteador virtual. Como a camada de virtualização está em um nível baixo, cada roteador virtual pode ter seu próprio controle e seu próprio plano de dados.

4.5.1 Isolamento de ambiente virtuais

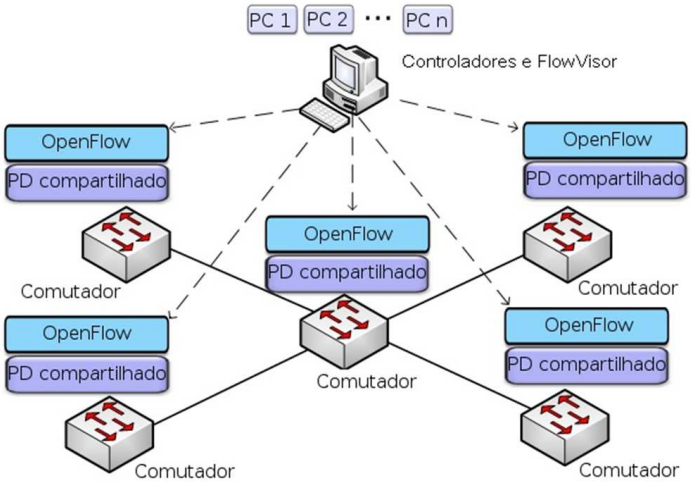
O envio e a recepção de pacotes são operações de E/S, o que demanda o uso de *drivers*, e, portanto, passam pelo Dom0. Dessa forma, todas as operações de rede dos DomUs geram uma sobrecarga tanto em memória quanto em CPU para o Dom0, tornando os recursos do Dom0 um possível ponto de estrangulamento quando sobrecarregado em uso de memória ou de CPU. Isso resulta em uma importante vulnerabilidade do Xen, pois, caso os recursos do Dom0 fiquem escassos, então todas as máquinas virtuais terão seu desempenho reduzido, uma vez que o Dom0 é um domínio compartilhado por todas as máquinas virtuais sem que exista algum mecanismo de isolamento eficiente. Essa falta de isolamento é uma falha grave de segurança, que pode ser explorada por redes (DomUs) maliciosas buscando afetar o desempenho de outras redes ao exaurir os recursos do Dom0.

A Tabela 4.1 mostra o consumo de CPU do Dom0 medido com a ferramenta Top ao se realizar operações de rede na plataforma Xen[2], assumindo um intervalo de confiança (IC) de 95% nas medidas. Observa-se que as operações de transferência de dados entre dois DomUs e também entre o DomU e o Dom0 são as mais custosas para o Dom0. Os resultados comprovam que uma ação maliciosa em um DomU pode facilmente exaurir os recursos do Dom0 e assim comprometer o desempenho de todos os outros domínios. Isso ocorre porque os recursos que são compartilhados no Dom0 por todas as redes virtuais não são controlados pelo escalonador do hipervisor. O

[2]Os testes são relativos a uma máquina com processador Intel Core 2 Quad com 4GB de memória RAM e Xen 3.4-amd64. As duas máquinas virtuais são configuradas com uma CPU virtual cada e 128 MB de memória, enquanto o Dom0 é configurado com uma CPU virtual e sem restrições de memória. Além disso, cada CPU virtual é associada a uma única CPU física de uso exclusivo. Os testes foram feitos utilizando-se a ferramenta Iperf.



(a) Xen: um plano de dados (PD) e um plano de controle (PC) por roteador virtual.



(b) OpenFlow: um plano de dados (PD) compartilhado e planos de controle (PC) nos controladores.

Figura 4.8: Redes virtuais usando Xen e OpenFlow.

hipervisor apenas controla o acesso de cada máquina virtual aos recursos físicos. O Dom0 é uma máquina virtual cujos recursos são utilizados por outras máquinas virtuais, mas o que ocorre dentro do Dom0 não é controlado pelo hipervisor. O hipervisor apenas dita quando o Dom0 tem acesso ao *hardware*.

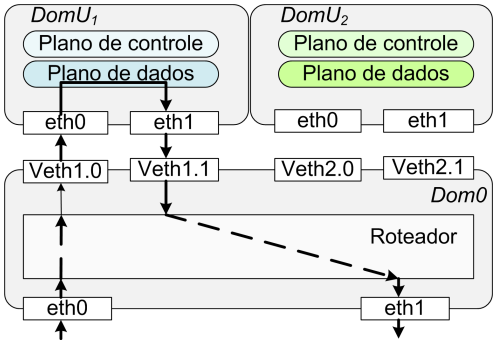
4.5.2 Desempenho no encaminhamento de pacotes

Uma alternativa para reduzir o volume de tráfego que passa pelas máquinas virtuais e para aumentar o desempenho no encaminhamento é o paradigma da separação de planos, que desacopla o plano de dados, responsável pelo encaminhamento, do plano de controle. A separação de planos é feita transferindo-se o encaminhamento de pacotes da máquina virtual para o Dom0, que possui acesso direto ao *hardware*. Sem a separação de planos, quando um pacote chega ao Dom0, ele é encaminhado para o DomU, o qual consulta a sua tabela de encaminhamento e re-

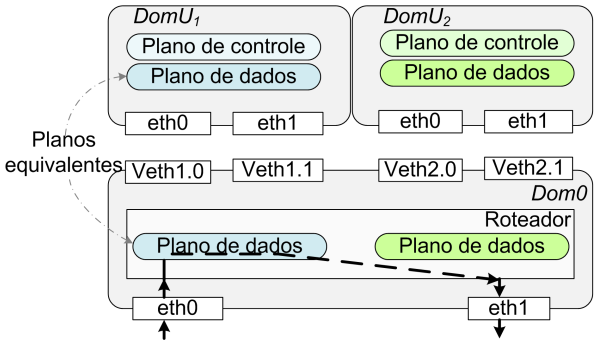
Tabela 4.1: Consumo de CPU no Dom0 devido a operações nas máquinas virtuais

Uso de CPU (%)	**IC (95%)**	**Descrição**
0,71	0,60	Consumo básico de CPU do *Dom*0
66,43	8,93	Transferência de dados com TCP de *DomU* para *Dom*0
85,49	5,91	Transferência de dados com TCP de $DomU_1$ para $DomU_2$
2,52	0,85	Transferência de dados com TCP de máq. externa para *Dom*0
1,79	1,01	Transferência de dados com TCP de máq. externa para *DomU*
1,20	0,65	Transferência de dados com TCP de *DomU* para máq. externa

torna o pacote para o Dom0, que o encaminha para fora da máquina física. Com a separação de planos, o pacote não precisa ser encaminhado até o DomU, pois uma cópia atualizada da tabela de encaminhamento do DomU está no Dom0, como mostra a Figura 4.9(b). É importante observar que apenas os pacotes de dados são encaminhados pela tabela do Dom0. Os pacotes de controle devem ser sempre encaminhados para o DomU, para que o plano de controle possa ser atualizado. O Xen, nativamente, não disponibiliza a separação de planos.



(a) Sem a separação de planos: todos os pacotes passam pela máquina virtual e pelo Dom0.



(b) Com separação de planos: os pacotes passam apenas pelo Dom0, reduzindo o processamento e o atraso por pacote.

Figura 4.9: Modos de encaminhamento de um pacote no Xen assumindo que o fluxo de dados pertence à rede do roteador virtual em $DomU_1$.

4.5.3 Qualidade de serviço

Por fim, o Xen também não disponibiliza nenhum esquema para o provimento de QoS para as redes virtuais. Assim, um esquema para o controle de políticas de tráfego precisa ser construído para garantir o QoS dentro de cada rede e também entre as redes virtuais.

4.6 Virtualização de redes com a plataforma OpenFlow

O OpenFlow [84] permite que a infraestrutura de rede das universidades seja usada não somente para a rede de produção, mas também para as redes experimentais. O projeto OpenFlow, proposto pela Universidade de Stanford, visa criar ambientes virtuais para a experimentação de inovações em paralelo com a rede de produção. Para tanto, a plataforma OpenFlow permite o uso compartilhado de elementos de rede como comutadores, roteadores, pontos de acesso e computadores pessoais.

Apresentando uma nova arquitetura para criação de ambientes de rede virtual, o OpenFlow traz como idéia-chave a separação física da função de encaminhamento na rede, realizada pelo plano de dados, da função de controle da rede. Assim, os planos de dados e de controle são executados por elementos de rede diferentes. A virtualização dos elementos de encaminhamento, também chamados de comutadores OpenFlow, é realizada por uma tabela de fluxo compartilhada, o que representa o plano de dados, e todos os planos de controle são centralizados em um ou mais elementos de rede, chamados de controladores, que executam aplicativos que controlam cada rede virtual. Um exemplo de rede usando OpenFlow está na Figura 4.8(b). Para dar um maior isolamento à rede e permitir o uso de múltiplos controladores, é necessário usar um elemento extra na rede, chamado de FlowVisor [85]. O FlowVisor funciona como um *proxy* entre os controladores e os elementos encaminhadores e é o elemento responsável por definir qual tráfego pertence a qual rede, assim como por determinar quanto dos recursos físicos podem ser utilizados por rede virtual. Na ausência do FlowVisor, um único controlador pode atuar em nome de várias redes virtuais, onde cada plano de controle é definido como um conjunto de aplicações.

4.6.1 O protocolo OpenFlow

O protocolo OpenFlow define a comunicação entre os comutadores OpenFlow e o controlador de cada rede. Esse protocolo é baseado na criação de um canal seguro entre cada comutador OpenFlow e o controlador, que utiliza este canal para monitorar e configurar os comutadores OpenFlow. Basicamente, o OpenFlow define um fluxo como uma seqüência de pacotes com características em comum e realiza o encaminhamento baseado em fluxos. Toda vez que o primeiro pacote de um fluxo ainda não classificado chega a um comutador OpenFlow, ele é enviado ao controlador. Então, o controlador define um caminho para os pacotes desse fluxo, definindo regras de encaminhamento em cada comutador OpenFlow que pertence ao caminho escolhido. O controlador também pode definir um conjunto de fluxos

que deve ser encaminhado de forma convencional na camada dois e na camada três como se o OpenFlow não estivesse sendo usado. Essa característica é a razão pela qual o OpenFlow pode ser usado em paralelo com a rede de produção sem afetar o tráfego de produção.

4.6.2 Plano de dados compartilhado

O plano de dados no OpenFlow é uma tabela de fluxos definida por descritores de cabeçalho, contadores e ações. Os descritores de cabeçalho são uma tupla de doze campos que descrevem o cabeçalho do pacote, como mostrado na Figura 4.10. Esses campos especificam um fluxo definindo um valor para cada campo ou usando uma máscara para definir apenas um subconjunto dos campos. A tabela de fluxo também suporta o uso de máscaras de rede, se o *hardware* em uso também oferece suporte a esse tipo de operação [86]. Essa tupla dá uma grande flexibilidade para a função de encaminhamento no OpenFlow, porque o fluxo pode ser encaminhado com base não apenas no IP de destino, como no sistema convencional de rede TCP/IP, mas também na porta TCP, no endereço MAC, etc. Com isso, um comutador OpenFlow traz muito mais funcionalidades do que um comutador comum, pois é capaz de encaminhar pacotes com base em dados de todas as camadas do modelo TCP/IP. Após os campos de cabeçalho, a descrição do fluxo é seguida pelos contadores, que são usados para monitorar os comutadores OpenFlow. Contadores contabilizam dados como a duração do fluxo e a quantidade de bytes que foram transmitidos por um elemento. O último campo na descrição do fluxo são as ações, que são um conjunto de instruções que devem ser executadas em cada pacote de um fluxo específico pelos comutadores OpenFlow. Essas ações incluem não só encaminhar um pacote para uma porta, mas também a mudar os campos de cabeçalho como o VLAN ID, a prioridade e os endereços de origem/destino. Com isso, os comutadores OpenFlow podem funcionar como qualquer elemento de rede [87].

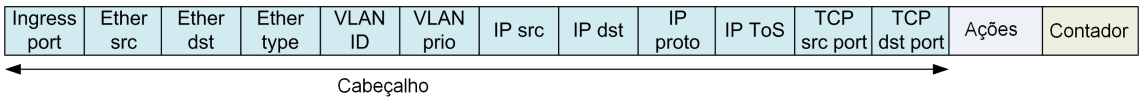


Figura 4.10: Definição de um fluxo em um comutador OpenFlow.

4.6.3 O controlador da rede

O controlador é um elemento central na rede, que se comunica com todos os nós para configurar as tabelas de fluxo. O controlador executa um sistema operacional de rede, que fornece as funções básicas de configuração de rede para os aplicativos

que gerenciam as redes virtuais. Assim, o controlador no OpenFlow funciona como uma interface entre as aplicações de rede e os comutadores OpenFlow, fornecendo as funções básicas para acessar o primeiro pacote de cada fluxo e para monitorar os comutadores OpenFlow. O OpenFlow funciona com qualquer controlador compatível com o protocolo OpenFlow, como o Nox [88]. A Figura 4.11 mostra uma rede virtual no OpenFlow, que é definida por seu plano de controle e pelos fluxos que estão sendo controlados por este plano de controle. Nesse caso, cada conjunto de aplicações pode ser executado por certa rede virtual, definindo formas de controle diversas, ainda que sobre o mesmo controlador[3].

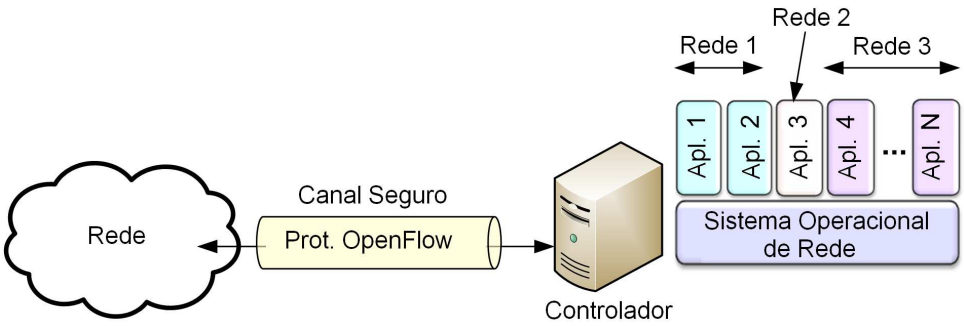


Figura 4.11: O modelo de controlador do OpenFlow.

O OpenFlow fornece uma infra-estrutura flexível, baseada na idéia de elementos distribuídos de encaminhamento, que fornecem as funções básicas para o funcionamento de uma rede, e planos de controle centralizados. Utilizando essa infra-estrutura, é possível dividir a rede física em várias redes virtuais. No OpenFlow, a instanciação de uma rede é apenas a criação de conjuntos de aplicativos em um controlador. Os fluxos da nova rede serão criados sob demanda, de acordo com os pacotes que chegam à rede. Além disso, o OpenFlow também oferece uma infra-estrutura flexível para a realocação de recursos na rede. De fato, realocar redes virtuais no OpenFlow corresponde a apenas remapear fluxos nas tabelas de encaminhamento, o que é uma operação simples para o controlador, que conhece a topologia física da rede.

[3]Para um maior isolamento das redes virtuais, cada conjunto de aplicações deve rodar em um controlador diferente e o FlowVisor deve ser usado para controlar o acesso de cada controlador a sua fatia de rede.

4.7 Comparação entre as plataformas Xen e OpenFlow na virtualização de redes

Tanto o Xen [31] quanto o OpenFlow [84] são plataformas que permitem que o substrato físico seja compartilhado entre diferentes redes virtuais, mas com características que trazem vantagens e desvantagens específicas de cada uma dessas tecnologias. Dessa forma, a análise dessas duas plataformas permite extrair algumas características sobre o funcionamento das redes virtuais de acordo com o modelo de virtualização adotado.

O Xen e o OpenFlow têm conceitos diferentes de virtualização. O Xen cria redes virtuais através do "fatiamento" dos elementos físicos da rede em roteadores virtuais concorrentes. Consequentemente, uma rede virtual é vista como um conjunto de roteadores virtuais interligados por enlaces virtuais mapeados sobre a infraestrutura física. Diferentemente, o OpenFlow cria redes virtuais através do fatiamento do plano de controle, aonde cada fatia cria uma tabela de encaminhamento própria nos elementos encaminhadores. Embora seja compartilhado, o elemento encaminhador no OpenFlow não tem conhecimento da existência ou não de virtualização na rede. Assim, no OpenFlow, uma rede virtual é um conjunto de fluxos com características comuns, que são controlados por um controlador ou por um conjunto de aplicativos de um controlador. As diferenças entre os modelos de virtualização do Xen e do OpenFlow influenciam questões como a escalabilidade da rede, a capacidade de programação e o processamento/encaminhamento dos dados.

4.7.1 Processamento de dados na rede e programabilidade

Uma das principais vantagens do modelo pluralista é apoiar a inovação. Como conseqüência, a rede deve ser suficientemente flexível para prover caminhos fim-a-fim sobre a infraestrutura física disponível, garantindo ao administrador o controle de toda a rede, que inclui a escolha da pilha de protocolos, as regras de encaminhamento, o processamento de dados etc.

Uma vez que a camada de virtualização Xen fica diretamente sobre o *hardware*, cada roteador virtual tem acesso a todos os componentes do computador, como memória, processador e dispositivos de E/S. O administrador da rede é livre para escolher tudo o que é executado sobre a camada de virtualização do Xen. Assim, diferentes sistemas operacionais, tabelas de encaminhamento, regras de encaminhamento e assim por diante, podem ser definidas para cada rede virtual. Além disso, tanto o plano de dados quanto o de controle podem ser totalmente virtualizados, como mostrado na Figura 4.8(b). Portanto, o Xen oferece uma plataforma poderosa e flexível para o controle e o gerenciamento da rede, permitindo o processamento

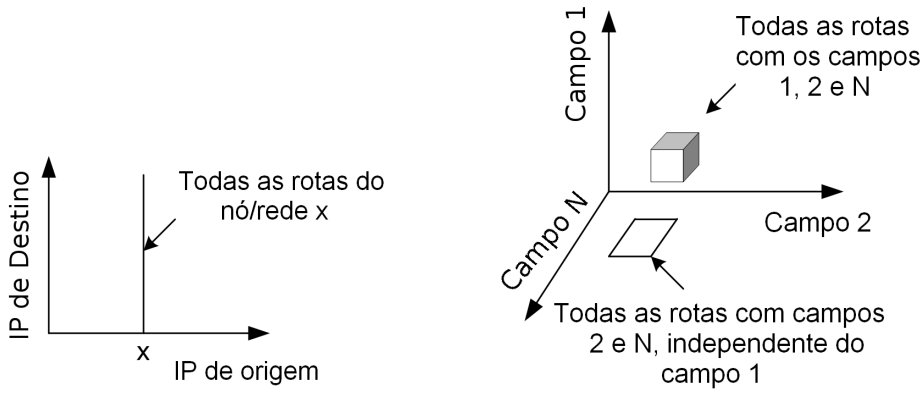
salto-a-salto de pacotes. Desta forma, as redes virtuais com novas funcionalidades podem ser facilmente implementadas. Por exemplo, uma rede virtual com suporte para a assinatura dos pacotes poderia ser instanciada para garantir o controle de acesso e a autenticação. Esta funcionalidade resolveria vários problemas de segurança da Internet atual, mas não pode ser implementada devido ao "engessamento da rede" [60]. Até mesmo modelos de rede com suporte à disrupção e atrasos podem ser implementados no Xen devido à flexibilidade no processamento de pacotes.

O modelo de virtualização do OpenFlow é diferente do modelo do Xen, porque a fatia virtual é um conjunto de fluxos e, como conseqüência, as ações na rede são direcionadas a fluxos ao invés de pacotes. O OpenFlow provê um esquema simples de transmissão de pacotes no qual o comutador OpenFlow procura uma entrada na tabela de fluxo para encaminhar cada pacote que chega. Se não houver nenhuma entrada, o pacote é encaminhado para o controlador que estabelece uma regra de encaminhamento em cada nó da rota escolhida para encaminhar o pacote. Assim, a principal desvantagem do OpenFlow é que todas as redes virtuais têm as mesmas primitivas de encaminhamento (tipo de pesquisa na tabela de fluxo, uso de máscaras, e as ações por fluxo), porque o plano de dados é compartilhado por todas as redes virtuais em cada nó da rede. Isso, no entanto, não implica em um processamento de pacotes totalmente inflexível. Na verdade, a versão 1 do protocolo OpenFlow especifica que o controlador pode definir ações para os fluxos que definem que um campo do cabeçalho pode ser modificado antes da transmissão de pacotes. Assim, OpenFlow fornece uma tabela de encaminhamento que pode definir fluxos de forma completa, muito mais flexível do que a tabela de encaminhamento atual do TCP/IP. Por exemplo, o comutador OpenFlow poderia mudar o endereço de destino de um pacote para encaminhá-lo para um *middle box* antes de o pacote atingir o próximo salto na rede. Por outro lado, funcionalidades em nível de pacote, tais como a verificação de assinaturas no pacote salto-a-salto, não são facilmente implementados no OpenFlow, porque tais funcionalidades não podem ser implementadas diretamente no comutador OpenFlow e devem ser executadas pelo controlador ou por uma *middle box*, o que provocaria uma grande perda no desempenho da rede.

Contrapondo-se ao modelo de plano de dados compartilhado do OpenFlow, o Xen oferece planos de dados independentes para as diferentes redes virtuais[4]. Contudo, o Xen, nativamente, ainda é baseado na tabela de encaminhamento do TCP/IP, que provê um roteamento baseado em IP. Isso significa que o plano de encaminhamento é baseado apenas no endereço IP de destino. No OpenFlow, a definição do espaço de fluxo é composta por n dimensões, onde n é o número de campos do cabeçalho que podem ser usados para especificar um fluxo, como mostrado na Figura 4.12. Por isso, um fluxo é definido com base em todas as dimensões ou com base em

[4]Desde que não esteja utilizando o paradigma da separação de planos.

uma máscara que define quais os campos do cabeçalho devem ser considerados para o encaminhamento dos pacotes do fluxo [84]. A consequência do uso deste tipo de tabela de encaminhamento é que os pacotes são encaminhados com base não apenas no IP de destino, mas também em outros parâmetros, tais como o tipo de aplicativo que está em uso. Este tipo de tabela de encaminhamento é compatível com o Xen, mas ainda não está disponível, precisando ser construída com o auxílio de ferramentas como o Click [89].



(a) Definição do espaço de fluxos no modelo TCP/IP.

(b) Definição do espaço de fluxos no modelo OpenFlow.

Figura 4.12: Modelos de espaço de fluxos para definir a tabela de encaminhamento nas redes baseadas em TCP/IP e em OpenFlow.

Outra diferença fundamental entre o Xen e o OpenFlow no que diz respeito à programação é o modelo de plano de controle. No Xen, cada nó da rede virtual tem o plano de dados bem como o plano de controle e, conseqüentemente, o controle da rede é descentralizado. No OpenFlow, cada nó da rede possui apenas o plano de dados. O plano de controle é centralizado no controlador, que é um nó especial na rede. O uso de um plano de controle centralizado facilita o desenvolvimento de novos algoritmos para o controle da rede, quando comparado ao uso de uma abordagem descentralizada. Por outro lado, o uso de um controlador centralizado dificulta o uso de *softwares* legados de controle de rede, pois esses são, em sua maioria, baseados em controle distribuído. Além disso, um controle centralizado cria a necessidade de um servidor extra na rede e também cria um ponto único de falha na rede.

4.7.2 Desempenho no encaminhamento de dados na rede

Uma importante característica de um ambiente de redes virtuais é o desempenho no encaminhamento de pacotes. A eficiência no encaminhamento não depende exclusivamente do *hardware*, mas também da lógica apresentada em cada tecnologia. Nesta seção, é assumido que tanto o Xen quanto o OpenFlow são executados no

mesmo *hardware* para avaliar as perdas em desempenho que cada tecnologia impõe para a transmissão de dados.

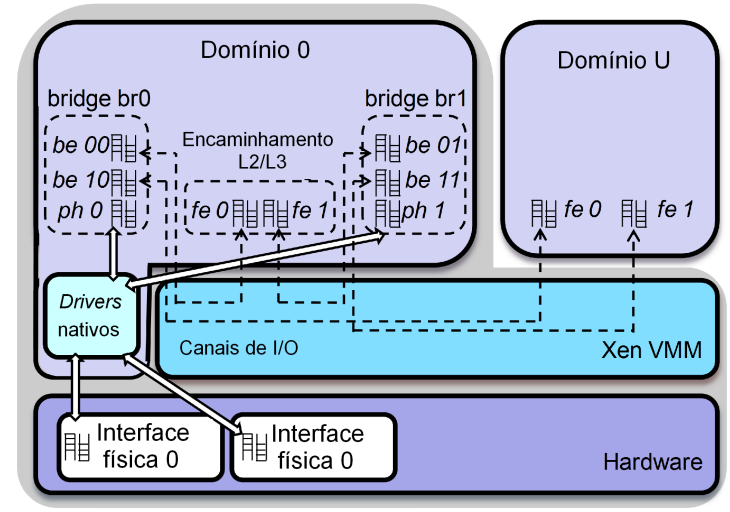


Figura 4.13: Caminho de envio/recebimento de pacote no modo *bridge*.

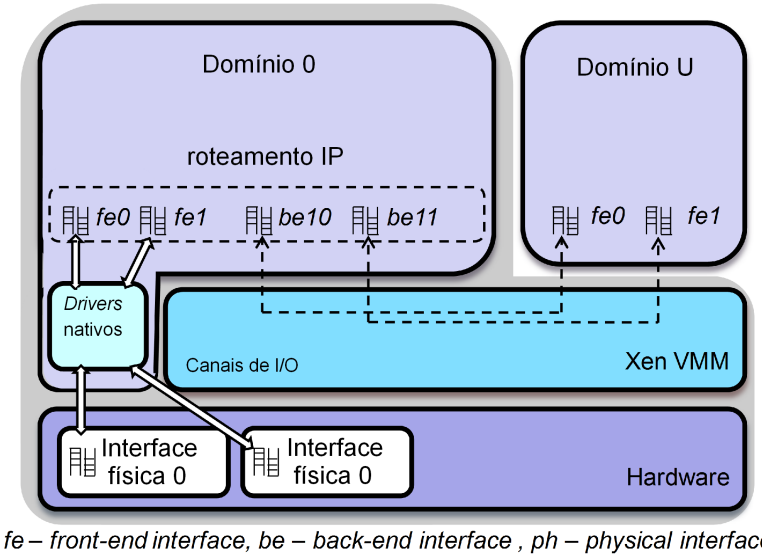


Figura 4.14: Caminho de envio/recebimento de pacote no modo *router*.

O desempenho do Xen depende do local onde é realizado o encaminhamento de pacotes. Para cada roteador virtual, o encaminhamento de pacotes pode ser executado pelo sistema operacional em execução no DomU correspondente ao roteador virtual ou pelo Domínio 0. No caso de execução no DomU, existe mais flexibilidade no processamento de pacotes, mas os custos associados com a veiculação dos pacotes entre o Dom0 e o DomU para realizar o encaminhamento introduzem uma

sobrecarga de controle alta que reduz consideravelmente o desempenho do Xen. No caso de execução no Domínio0, os pacotes de/para todos os roteadores virtuais são encaminhados pelo Dom0, que controla múltiplas tabelas de encaminhamento simultaneamente. O desempenho do Xen no encaminhamento de pacotes também depende de qual dos dois modos possíveis para transferir pacotes entre as máquinas virtuais é utilizado [31]: o modo *bridge* e o modo *router*. O modo *bridge* é usado como arquitetura padrão pelo Xen e é descrito na Figura 4.7. No entanto, essa arquitetura não se aplica a um roteador, porque são necessárias mais do que uma interface física em cada dispositivo. A Figura 4.13 mostra um exemplo do modo *bridge* com duas interfaces físicas. Nesse caso, existem duas *bridges* no Dom0, sendo uma por interface física, conectando as interfaces do *back-end* com as interfaces físicas. O encaminhamento de pacotes pode ser realizado no Dom0 usando a camada 2, enlace, ou a camada 3, rede. Seja p um pacote que chega à interface física ph_0 e que deve ser enviado para a interface física ph_1. Em primeiro lugar, p é tratado pelo *driver* da placa de rede no dom0. Neste momento, p está na $ph0$, a qual está conectada a *bridge* br_0. Esta ponte demultiplexa o pacote p e o move para a interface de *back end* $be00$ com base no endereço MAC de destino do quadro. Depois disso, p é movido a partir de be_{00} para a interface de *front end* $fe0$ usando o canal de E/S através do hipervisor. O pacote p é, então, enviado para a interface *front end* fe_1 e, depois, outro canal de E/S é usado para mover p para a interface de *back end* be_{01}. Esta interface é a mesma *bridge* br_1 da interface física ph_1. Assim, p atinge sua interface de saída. Vale ressaltar que o hipervisor é chamado duas vezes para encaminhar um pacote.

No modo *router*, ilustrado pela Figura 4.14, cada interface física no Dom0 possui um endereço IP associado. Como conseqüência, o modo *router* não precisa de *bridges* ligando cada interface física com canais de E/S, ou seja, o encaminhamento de pacotes entre interfaces físicas no Dom0 ocorre assim como no Linux nativo, através das ferramentas de roteamento do kernel. Neste caso, se o Dom0 é usado como plano de dados compartilhado (Figura 4.3(b)), não existem chamadas para o hipervisor. Com o modo *router*, o hipervisor é chamado somente quando cada roteador virtual implementa seu próprio plano de dados, conforme ilustrado na Figura 4.3(c). Neste caso, os pacotes são roteados para a interface *back end* associada ao DomU de destino e, em seguida, são movidos para a interface *front end*, utilizando o canal de E/S através do hipervisor. Em seguida, os pacotes são transferidos para a interface de *back end* e, finalmente, encaminhados para a interface de saída física. A fim de permitir que DomUs enviem e recebam pacotes, os endereços IP também são atribuídos a interfaces de *back end* em contraste com o modo *bridge*.

O OpenFlow não assume o uso de um plano de dados virtualizado sobre os elementos de encaminhamento e, consequentemente, segue o modelo de um único plano

de dados compartilhado para todas as redes. Assim, é esperado que o desempenho OpenFlow seja semelhante ao encaminhamento nativo de pacotes. O OpenFlow, entretanto, apresenta uma desvantagem quando o fluxo não está configurado na tabela de encaminhamento. Como foi explicado antes, quando um pacote chega a um comutador OpenFlow, se o fluxo não está configurado na tabela, o pacote é enviado através da rede para o controlador. Esse mecanismo introduz um atraso maior para encaminhar o primeiro pacote de cada fluxo, devido à troca de pacotes e os atrasos de processamento do controlador. Se o tráfego é majoritariamente formado por pequenos fluxos, então esse tipo de processamento pode implicar em uma redução de desempenho do OpenFlow, dependendo de fatores como a capacidade de processamento do controlador, a posição do controlador na rede, os enlaces de entrada e saída do controlador com a rede, entre outros.

4.7.3 Escalabilidade com relação às redes virtuais

A escalabilidade com relação ao número de redes virtuais rodando sobre o mesmo substrato físico é um fator importante para a definição do tipo de aplicações que podem ser feitas com o ambiente virtualizado. Os requisitos da nova Internet ainda são uma questão em aberto e a nova arquitetura não deveria restringir o número de redes que operam sobre a infraestrutura física disponível. A abordagem do Xen é menos flexível nesse sentido, porque o elemento de rede virtual é uma máquina virtual, a qual exige mais recursos de *hardware*, como poder de processamento e espaço de memória, do que o processamento de um simples conjunto de fluxos em um comutador OpenFlow. Por esta razão, OpenFlow suporta milhares de redes virtuais rodando em paralelo, enquanto o Xen é restrito ao número de máquinas virtuais que podem ser multiplexadas sobre o mesmo hardware. Vale ressaltar que a escalabilidade Xen pode ser melhorada se o Domínio 0 é usado como um plano de dados compartilhado.

4.7.4 Resultados experimentais

A Tabela 4.2 resume as questões que foram discutidas anteriormente. A seguir, é apresentada uma análise prática sobre as plataformas de virtualização de redes do Xen e do OpenFlow. Os primeiros experimentos medem a taxa de encaminhamento alcançada em cada uma das configurações descritas. Os quadros são configurados com tamanhos entre 64 e 1512 bytes. Os quadros pequenos são usados para gerar altas taxas de pacotes, sobrecarregando o processamento da máquina encaminhadora, enquanto que os quadros grandes são usados para saturar o enlace físico de 1 Gb/s.

As Figuras 4.15(a) e 4.15(b) mostram as taxas de transmissão obtidas com o

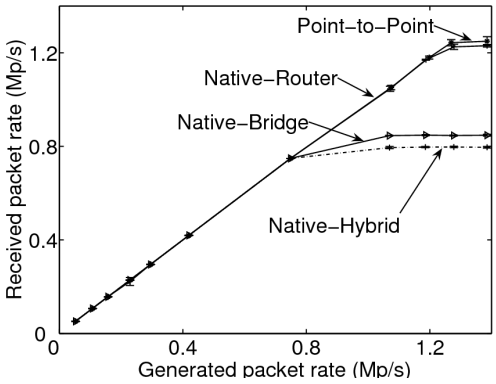
Tabela 4.2: Comparação entre as plataformas Xen e OpenFlow na virtualização de redes.

Característica	**Xen**	**OpenFlow**
Plano de dados	Distribuído	Distribuído
Plano de controle	Distribuído	Centralizado
Isolamento	Não é provido nativamente	FlowVisor
Desempenho no encaminhamento	Baixo, pela máquina virtual e semelhante ao nativo pelo Dom0	Semelhante ao nativo
Roteamento	Por pacote, com base em endereço de destino	Por fluxo, com base em qualquer campo do cabeçalho
Programabilidade	Processamento por pacote no roteador virtual	Uso de middle boxes
Escalabilidade com relação ao número de redes virtuais	Limitado pelo número máximo de máquinas virtuais	Limitado pelo número de fluxos ativos

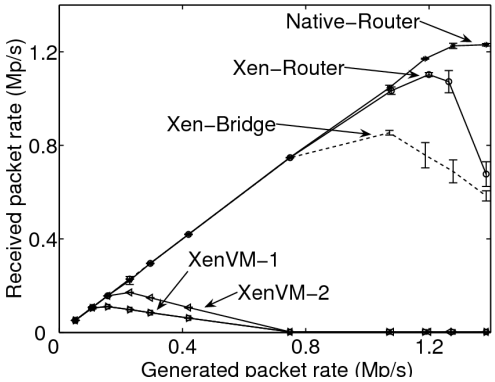
Linux nativo, as quais são a base para avaliar o impacto da virtualização da rede no Xen e no OpenFlow. Também é mostrada no gráfico a taxa de transmissão ponto-a-ponto, obtida quando a máquina receptora e a máquina geradora estão diretamente conectadas, sem o intermédio da máquina encaminhadora. Na configuração ponto-a-ponto, chamada de *Point-To-Point* nos gráficos, é possível observar a taxa máxima entre a máquina geradora e a receptora, a qual é um limite superior para todas as curvas. Os resultados mostram que o Linux Nativo no modo *router* tem um desempenho semelhante ao cenário ponto-a-ponto. Isso é explicado pela baixa complexidade do mecanismo de roteamento do *kernel*. O modo *Native-Bridge*, no entanto, apresenta um desempenho inferior ao do modo *Native-Router*. Segundo Mateo [90], este resultado se deve ao módulo *bridge* do Linux não ser otimizado para suportar altas taxas de pacote. Por fim, observa-se, como esperado, que o modo *Native-Hybrid* apresenta o pior desempenho devido às limitações mencionadas anteriormente do modo *bridge* e aos custos incrementais necessários para transmitir pacotes entre as *bridges* criadas.

Os resultados para taxa de encaminhamento no Xen são mostrados na Figura 4.15(b). Primeiramente, é analisado um cenário onde o Dom0 encaminha os pacotes[5]. Neste experimento, testamos as configurações *Xen-Bridge* e *Xen-Router*.

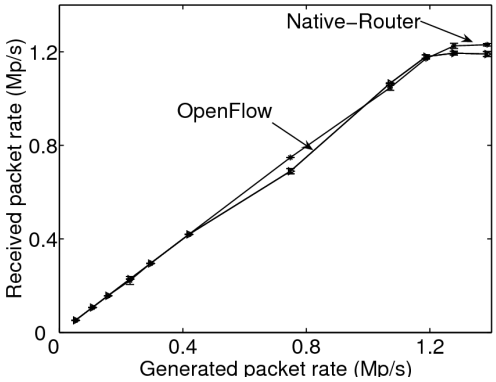
[5]Neste cenário nenhuma máquina virtual está executando, pois foi observado que os mesmos resultados são obtidos quando existem máquinas virtuais, mas elas não encaminham pacotes, ficando ociosas. Egi *et al.* também obtiveram o mesmo resultado com relação à existência das máquinas virtuais nesse tipo de teste [31].



(a) Linux nativo.



(b) Xen.



(c) OpenFlow.

Figura 4.15: Taxa de transferência de pacotes para configurações diferentes do elemento encaminhador, usando quadros de 64 bytes.

Observa-se que a *Xen-Bridge* sofre as mesmas limitações que a *Native-Bridge*, uma vez que ambas as configurações dependem do módulo de comutador por *software* do Linux. Além disso, a configuração *Xen-Bridge* encaminha pacotes do comutador por *software* para a camada de roteamento, como na configuração *Native-Hybrid*, o que combinado com as chamadas ao hipervisor, resultam no desempenho mostrado na figura. Como esperado, a *Xen-Bridge* possui um desempenho inferior a todas as configurações do Linux nativo. A configuração *Xen-Router* possui um desempenho superior a configuração *Xen-Bridge*, pois o comutador por *software* do Linux não é utilizado e o hipervisor do Xen não é chamado para o encaminhamento de pacotes pelo Dom0. Contudo, a *Xen-Router* possui desempenho inferior ao da *Native-Router*. A taxa de encaminhamento diminui rapidamente depois de cerca de 1,2 Mpct/s. Esse comportamento também é observado na *Xen-Bridge* e nos experimentos passando pela máquina virtual. Esta perda de desempenho está relacionada à implementação do Xen do tratamento de interrupções e precisa ser mais investigada.

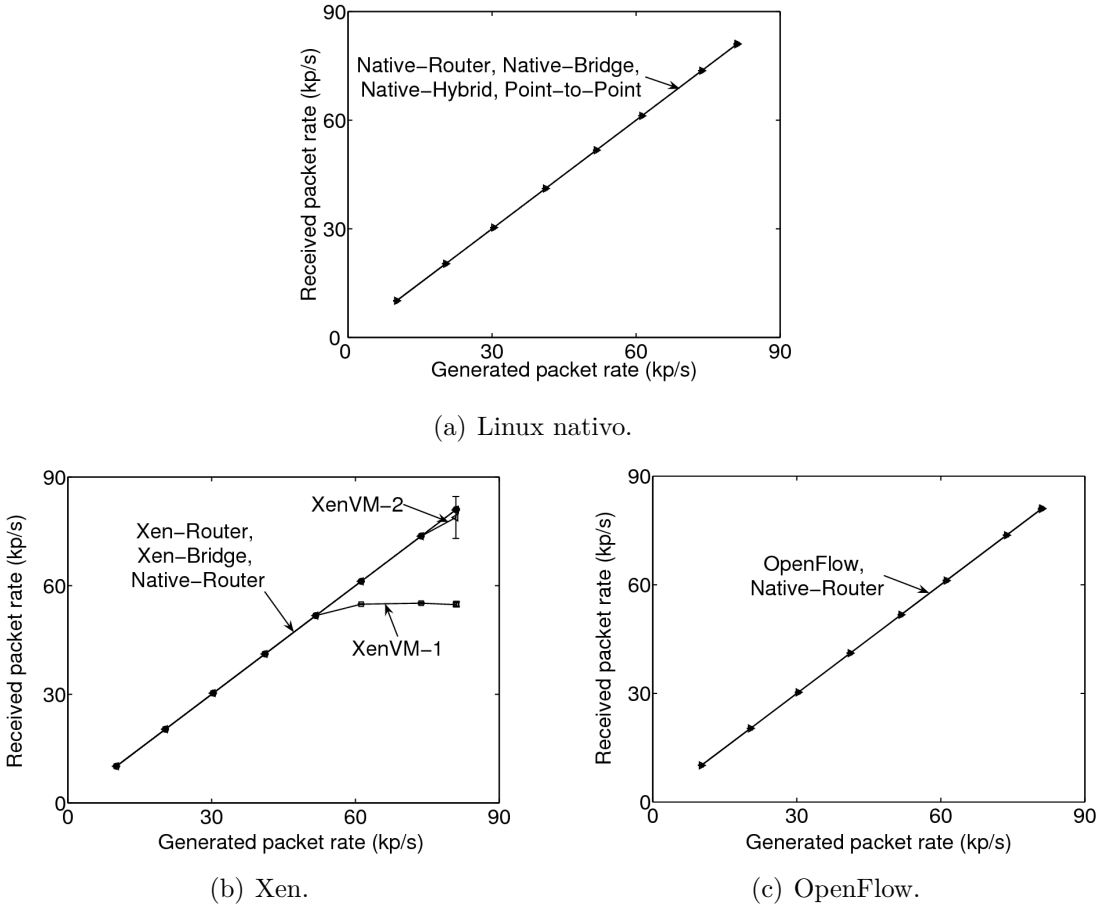
Em seguida, foi analisado o encaminhamento pelas máquinas virtuais usando

o modo *bridge* do Xen, o qual é a configuração padrão do Xen. Na configuração *XenVM-1*, tanto a máquina virtual quanto o Dom0 compartilham o mesmo núcleo da CPU. Esse resultado mostra uma redução significativa no desempenho em comparação com os resultados anteriores do Xen, nos quais o Dom0 encaminhava os pacotes. À primeira vista, este desempenho ruim poderia ser causado por uma alta contenção de recursos da CPU, devido ao fato de que um único núcleo da CPU é compartilhado entre os domínios. Para eliminar a disputa por recursos da CPU, foi utilizada a configuração *XenVM-2*, cujos resultados estão na Figura 4.15(b). Nessa configuração, cada domínio possui o seu próprio núcleo exclusivo. O desempenho da *XenVM-2* é superior ao da *XenVM-1*, mas ainda é pior do que os resultados com encaminhamento pelo Dom0. Isso pode ser explicado devido à alta complexidade envolvendo o encaminhamento de pacotes em máquinas virtuais. Quando o tráfego é encaminhado através de máquinas virtuais, deve passar por um caminho mais complexo antes de chegar à máquina receptora. De fato, após a recepção do pacote, ele é transferido via *Direct Memory Access* (DMA) para a memória do Dom0. O Dom0 demultiplexa o pacote para seu destino, obtém uma página de memória livre associado à máquina virtual receptora, troca a página livre pela página que contém o pacote e, em seguida, notifica a máquina virtual. Para uma máquina virtual enviar um pacote, ela deve apresentar um pedido de transmissão, juntamente com uma referência para a área de memória onde o pacote está no anel de E/S do Xen. O Dom0, então, verifica o anel de E/S e quando ele recebe o pedido de transmissão, mapeia a referência em um endereço de página física e, em seguida, envia para a interface de rede [91]. Esse aumento da complexidade é parcialmente responsável pela baixa taxa de pacotes obtida nas duas curvas em que as máquinas virtuais são usadas para encaminhar pacotes.

A Figura 4.15(c) mostra que o OpenFlow tem um desempenho similar ao Linux nativo no modo *router*. Além disso, a comparação entre os resultados da configuração *OpenFlow* com a *XenVM* mostra o equilíbrio entre flexibilidade e desempenho. Na *XenVM* temos mais flexibilidade, porque os planos de dados e de controle estão sob o controle total de cada administrador de rede virtual. No OpenFlow, por outro lado, a flexibilidade é menor porque o plano de dados é compartilhado por todas as redes virtuais. Entretanto, devido à menor sobrecarga de processamento, o desempenho da configuração *OpenFlow* é melhor do que a da configuração *XenVM* em nosso cenário. O desempenho do Xen pode ser aumentado se o plano de dados for movido para o Dom0, como podemos ver nos resultados das configurações *Xen-Router* e *Xen-Bridge*. Neste caso, entretanto, a flexibilidade dos planos de dados customizados é reduzida.

Também foram realizados experimentos de encaminhamento com quadros de 1.470 bytes, como mostrado na Figura 4.16. Com pacotes grandes, todas as confi-

gurações, exceto a *XenVM-1*, têm o mesmo comportamento da configuração *Native-Router*. Isso significa que não há perda de pacotes na máquina encaminhadora e o gargalo, neste caso, é o enlace de 1 Gb/s. Na configuração *XenVM-1*, na qual todos os domínios, incluindo o Dom0, compartilham uma única CPU, a taxa de pacotes alcançada é menor. Nos experimentos com a *XenVM-2*, na qual cada domínio possui um núcleo exclusivo, o comportamento obtido é semelhante à configuração *Native-Router*. Com isso, concluímos que a queda de desempenho no resultado da *XenVM-1* é causada pela alta contenção de recursos de CPU entre domínios.



(a) Linux nativo.

(b) Xen.

(c) OpenFlow.

Figura 4.16: Taxa de transferência de pacotes para configurações diferentes do elemento encaminhador, usando quadros de 1512 bytes.

4.8 Estado da Arte no Controle de Recursos Virtualizados

Nas seções anteriores, foram discutidos os modelos de virtualização, as entidades relacionadas a esses modelos e algumas das principais tecnologias para a criação de redes virtuais. Com base nessa análise, pode-se concluir que o isolamento é uma

característica primordial para o provimento da virtualização de redes.

O provimento do isolamento, assim como de qualidade de serviço, de ambientes virtuais depende, entre outros, de um refinado controle do uso dos recursos compartilhados pelos ambientes virtualizados. Em geral, os sistemas de controle contam com ferramentas que provêem as primitivas básicas para o controle da plataforma de virtualização. O *Virtual NEtwork management for Xen-based Testbeds* (VNEXT) [92, 93] provê primitivas básicas da plataforma de virtualização Xen como instanciação e deleção de nós, visualização das topologias física e virtuais, migração, monitoramento, entre outros. Uma ferramenta similar para a plataforma OpenFlow é o *OpenFlow MaNagement Infrastructure* (OMNI) [94, 95], que provê primitivas semelhantes ao VNEXT. É importante observar que o OMNI disponibiliza primitivas de controle importantes para o gerenciamento de uma rede virtual, mas a criação de redes virtuais no OpenFlow é feita através do FlowVisor [85]. O FlowVisor funciona como um *proxy* entre os elementos encaminhadores e os planos de controle, permitindo a configuração de parâmetros de redes virtuais no OpenFlow, tais como a divisão dos espaços de rede, ou seja, quais características definem cada rede virtual, a definição do mapeamento entre a topologia física e a virtual e a configuração do valor de banda mínima disponibilizada para cada rede virtual. Assim, sistemas como o VNEXT, o OMNI e o FlowVisor são essenciais para as ferramentas de controle, que fazem uso das primitivas básicas de cada plataforma de virtualização. Outra ferramenta importante para o controle de ambientes virtualizados é a Libvirt [96]. A Libvirt é uma biblioteca de gerenciamento de máquinas virtuais que provê uma interface para diferentes hipervisores. Essa biblioteca provê primitivas de controle como ligar e desligar um nó, migrar e monitorar. Embora essa abordagem disponibilize primitivas de controle para vários hipervisores, ela não lida com funções de virtualização de rede diretamente, como a criação de uma topologia virtual.

As propostas para controle de recursos podem ser separadas entre controle para servidores e para roteadores virtuais. Outro ponto importante é a capacidade local ou global de controle, ou seja, se a proposta faz o controle de recursos em cada nó da rede ou se faz um controle central que tenta otimizar o uso de recursos globalmente. Cabe observar que propostas de controle globais dependem da existência de um controle por nó eficiente para garantir o isolamento. Assim, o controle global lida com operações como o mapeamento de topologias virtuais e a migração de nós, enquanto que o controle local monitora os recursos de cada nó físico que são atribuídos a cada rede virtual, tratando o isolamento entre redes. A seguir, são listadas algumas das principais propostas relativas a cada uma dessas vertentes.

4.9 Controle de recursos em *data centers*

A maior parte dos trabalhos relacionados ao gerenciamento de recursos virtuais é focada em *data centers* e sistemas em nuvens. Apesar dessas propostas de controle visarem o provimento de recursos para máquinas virtuais, elas não são projetados para o provimento de recursos para roteadores virtuais. Os roteadores virtuais, diferentemente de servidores, têm sua demanda determinada basicamente por operações de entrada e saída (E/S), o que implica em diferentes perfis de consumo de recursos. De fato, muitas plataformas de virtualização apresentam falhas de isolamento quando expostas às altas cargas de operações de E/S [24, 29], mas esses fatores não são tratados, em geral, pelos sistemas de controle de recursos para *data centers*.

Gong *et al.* propuseram um sistema chamado *PRedictive Elastic reSource Scaling* (PRESS), o qual prevê as demandas por recursos de aplicações utilizando cadeias de Markov e transformadas de Fourier rápidas [97]. As predições são utilizadas para ajustar automaticamente a alocação de recursos, de acordo com os objetivos de nível de serviço (*Service Level Objectives - SLO*). O sistema foi implementado utilizando a plataforma de virtualização Xen e foi testado sob diferentes cargas de aplicações de servidores.

A alocação do recurso CPU entre os ambientes virtualizados também representa um desafio. Menasce e Bennani propuseram um sistema para otimizar a função de utilização dos ambientes virtuais através da realocação dos recursos de processamento de acordo com a variação da carga de trabalho dos servidores [98]. Os autores utilizam um controlador do tipo caixa branca que permite que seja dada prioridade a uma carga de trabalho específica. Payer *et al.* modificam o escalonador do Xen para modificar a forma como a CPU é alocada para as máquinas virtuais, com o objetivo de reduzir a latência no provimento de serviços [99].

O Xen apresenta problemas de isolamento relacionados não apenas com as operações de entrada e saída, mas também a outros aspectos [100, 101]. Jin *et al.* propuseram um mecanismo para garantir justiça no uso das memórias caches L2 e L3 no Xen, cujo uso também não é contemplado pelos mecanismos de isolamento do hipervisor do Xen [101]. A proposta modifica o algoritmo de alocação de páginas de memória do hipervisor utilizando a técnica de coloração de páginas.

Xu *et al.* propuseram um sistema de gerenciamento de recursos compartilhados por ambientes virtuais em *data centers* que se organiza em dois níveis [102]. O primeiro nível cuida do controle local, através do uso de técnicas de lógica *fuzzy*, enquanto que o segundo nível cuida do controle global. O controle global realiza a alocação de recursos baseado em um modelo de lucros, que tenta maximizar a utilização dos recursos no *data center*.

Outra técnica para compartilhamento de recursos é a utilização de migrações

para evitar a sobrecarga de servidores. Essa técnica é utilizada em uma proposta chamada de Sandpiper [37]. O Sandpiper é um sistema de controle centralizado que detecta *hotspots* e, para eliminar ou restringir os efeitos nocivos do *hotspot*, determina um novo mapeamento para as redes virtuais sobre a rede física, migrando máquinas virtuais. O sistema é avaliado quando em uso com sistemas de monitoração que tem acesso à máquina virtual e com sistemas de monitoração que só tem acesso a dados do recurso físico compartilhado. A proposta é implementada e avaliada na plataforma Xen.

Meng *et al.* buscam otimizar o uso dos recursos físicos através da migração de máquinas virtuais em *data centers* [103]. Dessa forma, máquinas virtuais são realocadas adequando a distância de comunicação entre elas de acordo com os padrões de tráfego observados. Assim, máquinas virtuais que apresentam uma grande interação, demandando muita banda, são realocadas com uma menor distância entre elas, de forma a reduzir o uso dos recursos físicos do *data center*.

4.10 Controle de recursos em redes virtuais

4.10.1 Sistemas de controle local

Egi *et al.* investigaram a viabilidade do uso do Xen como plataforma de virtualização de redes. Os autores concluem que o Xen associado ao conceito de separação de planos é uma alternativa viável para a construção de redes virtuais, pois garante um alto desempenho no encaminhamento de pacotes [31]. Para tratar o isolamento em redes virtuais, Egi *et al.* também investigaram a construção de uma plataforma de roteadores virtuais usando o Xen e o Click [89], avaliando o provimento de isolamento e justiça entre as redes [41]. Os autores verificam a utilização de diferentes planos de dados, assumindo o roteamento pelo Domínio 0 e pela máquina virtual, e avaliam a capacidade de se dividir os recursos entre as redes virtuais. Para fazer a monitoração dos recursos, os autores estenderam o escalonador de CPU do Click para levar em consideração os gastos de CPU com o encaminhamento de pacotes de cada plano de dados. Esse trabalho, no entanto, não apresenta mecanismos de gerenciamento que permitam definir recursos por redes autonomamente e diferenciar ou priorizar tráfegos para garantir o QoS das redes virtuais.

Outra abordagem baseada no Linux e no Click para a criação de um plano de dados compartilhado é proposta por Keller e Green [104]. Nessa proposta, cada rede virtual pode criar o seu próprio plano de dados, baseadas em primitivas genéricas de encaminhamento de pacotes definidas no Click. Apesar de disponibilizar ferramentas para a criação de um plano de dados genérico, os autores não tratam da divisão justa dos recursos compartilhados entre os planos de controle.

O Trellis [29] é um sistema para permitir o encaminhamento de pacotes com alto desempenho que garante o isolamento de enlaces virtuais em plataformas de virtualização em nível de sistema operacional. O Trellis implementa uma *bridge* personalizada de alto desempenho, além de implementar enlaces virtuais usando um novo tipo de túnel proposto, chamado de *Ethernet-over-GRE (EGRE) tunnels*. O principal objetivo do Trellis é garantir que a variação do atraso e a banda utilizada por uma rede não interfiram nesses mesmos parâmetros nas outras redes virtuais hospedadas no mesmo substrato físico. Contudo, o Trellis não trata os problemas de isolamento das operações de E/S durante o encaminhamento de pacotes.

O *Xen Throughput Control* (XTC) é um sistema para o controle da vazão nos roteadores virtuais criados com a plataforma Xen [27, 105]. O XTC utiliza um controle realimentado para ajustar o valor do *cap* das máquinas virtuais. O *cap* é um parâmetro do escalonador do Xen que determina um limite para a utilização de CPU física por uma máquina virtual e, assim, representa a porcentagem máxima do tempo de CPU dado para cada máquina virtual. A idéia básica é que se uma rede contratou um SLA, mas o seu processamento não é suficiente para garantir a taxa contratada, então o sistema de controle aumenta o percentual de CPU dessa máquina, para que ela consiga aumentar sua vazão. O XTC não impõe limites superiores para o consumo de recursos nos roteadores virtuais, mas apresenta uma forma de dar garantias mínimas de banda em roteadores virtuais criados sem separação de planos no Xen.

Bourguiba *et al.* [106] propuseram um estudo para determinar os parâmetros do escalonador de recursos do Xen de forma a aumentar a vazão dos roteadores virtuais. Essa proposta foca na configuração do parâmetro peso do escalonador do Xen para cada máquina virtual para obter uma maior vazão, balanceando os valores de peso entre as máquinas virtuais e o Domínio 0. Outra análise realizada foi a priorização do tráfego de uma rede virtual, através da inserção de uma fila para recepção de pacotes para a rede privilegiada, enquanto os pacotes para os outros roteadores virtuais são tratados por um escalonador com a política *Round Robin*. Os autores mostram quais configurações asseguram uma maior vazão de um roteador virtual sem separação de planos no Xen, mas não propõem mecanismos para ajustar esses parâmetros automaticamente conforme as demandas das redes virtuais.

Gupta *et al.* estudaram o isolamento de CPU em máquinas virtuais Xen, considerando o consumo de recursos de CPU no domínio de *drivers* por máquina virtual durante operações de entrada e saída [107]. Os autores propõem três ferramentas para monitoramento do uso de recursos, sendo elas o XenMon, que mede o uso de recursos por máquina virtual, incluindo os recursos utilizados em operações de entrada e saída; um escalonador de máquinas virtuais que considera os dados obtidos com o XenMon; e um mecanismo de controle para o Domínio 0 que tem por objetivo

limitar o tempo de uso que cada domínio tem em operações de rede. Esse mecanismo de controle assume que o uso de recursos é modelado como um limiar máximo e toda a vez que alguma máquina virtual excede esse limiar, todo o tráfego gerado com origem ou destino para a máquina virtual é bloqueado durante um período de tempo proporcional ao volume de recursos que foi usado a mais.

McIlroy e Sventek propuseram um controle local baseado em Xen para a Internet atual no qual cada fluxo que precisa de QoS é alocado para uma máquina virtual, chamada de QoS *routelet* [108]. Cada máquina virtual aplica, então, as suas políticas de QoS em todo o tráfego recebido. O protótipo é implementado no Xen, no qual o tráfego sem QoS é encaminhado pelo domínio privilegiado e os demais tráfegos são encaminhados pelas máquinas virtuais. Os autores observam que não é possível garantir QoS no sentido mais estrito com esse modelo, uma vez que o escalonador do Xen não é apropriado para essa tarefa. Outros problemas da proposta são a escalabilidade, pois é preciso uma máquina virtual por fluxo com QoS, e o baixo desempenho no encaminhamento de pacotes.

4.10.2 Sistemas de controle global

Schaffrath *et al.* propuseram uma arquitetura para controle global das redes virtualizadas [68]. A proposta foi implementada com o Xen e assume uso de um controle centralizado responsável por criar fatias da rede, através da instanciação de máquinas e enlaces virtuais. Assim, ao receber um pedido para alocar uma nova rede, o sistema contata cada um dos nós físicos selecionados e instancia as máquinas virtuais necessárias para construir a rede, assim como os enlaces entre elas, através de tunelamento IP. Outras abordagens semelhantes para o controle global são encontradas nos *testbeds* baseados em virtualização, como o GENI [109]. O controle de acesso dos pesquisadores à rede de teste é feito através de uma entidade central chamada de câmara de compensação (*Clearing House*). A câmara de compensação é um arcabouço de gerenciamento que monitora quais nós físicos e serviços estão disponíveis em cada um dos *testbeds* federados, quem está autorizado a usá-los e que fatias estão agendadas para cada pesquisador.

As entidades com controle global também podem realizar outras funções, como a migração. Houid *et al.* propuseram um sistema de controle global baseado em multi-agentes para alocação dinâmica de recursos para as redes virtuais através do uso de migração [110]. O sistema monitora os recursos disponíveis em cada máquina física assim como monitora as mudanças de demanda das redes virtuais. Ao notar que os recursos estão escassos, o agente no nó físico busca um nó físico semelhante para receber uma ou mais de suas máquinas virtuais.

O Genesis é um *kernel* para criação de redes virtuais com diferentes arquitetu-

ras [111]. Baseado nos conceitos de hierarquia e herança, o Genesis propõe que as diferentes redes virtuais "filhas" devem ser criadas com base em uma rede "raiz", da qual as filhas podem herdar características comuns. Assim como o Trellis, o Genesis é baseado na premissa de que todos os planos de controle funcionam sobre o mesmo sistema operacional. Por outro lado, ele permite a criação de diferentes políticas e a atribuição de mecanismos de QoS para cada rede virtual. No entanto, por ser implementado em nível de usuário e inserir uma camada de virtualização, o Genesis possui um desempenho no encaminhamento muito baixo.

4.11 Controle de admissão de elementos virtuais

Outro ponto importante do gerenciamento dos recursos compartilhados em redes virtuais é o controle de admissão de novos roteadores virtuais em um roteador físico.

Fajjari *et al.* propuseram um algoritmo de mapeamento de redes virtuais sobre o substrato físico baseado na meta-heurística da colônia de formigas [43]. Essa proposta aumenta o número de redes virtuais aceitas no substrato físico através do remapeamento das redes virtuais, de tal modo a criar uma fatia de rede com recursos suficientes para hospedar a nova rede virtual. Essa proposta faz o controle de acesso por nó da rede considerando um provimento de recursos estático, ou seja, cada rede virtual tem um limite superior de recursos que podem ser usados. Se o somatório do limite superior de recursos de todas as redes virtuais hospedadas em conjunto com a nova rede for menor ou igual ao somatório total de recursos, então a nova rede pode ser hospedada.

Lischka e Karl propuseram um algoritmo de mapeamento de redes virtuais baseado em isomorfismo que determina a posição de nós e de enlaces em um único estágio. Essa proposta apresenta um algoritmo que é capaz de mapear um enlace virtual em vários enlaces físicos e oferece um controle de admissão de redes virtuais [112]. Assim como Fajjari *et al.*, os autores dessa proposta realizam o controle de admissão supondo um limite superior de uso de recursos para cada nó virtual.

O Sandpiper também propõe uma heurística para controle de acesso, mas com caráter probabilístico [37]. Nesse caso, assume-se que o sistema já conhece o perfil de uso de uma máquina virtual e vai avaliar se ela pode ser migrada ou não para uma máquina física diferente. No Sandpiper, perfis de uso de recurso são gerados para cada uma das máquinas virtuais hospedadas. Esse perfil é guardado como uma função de densidade de probabilidade. O sistema estima a demanda de pico da máquina virtual como o valor que representa uma probabilidade acumulada de 95%. Se o valor de pico estimado for menor ou igual ao volume de recursos ociosos na nova máquina física, então a máquina virtual pode ser migrada. Essa proposta não estima possíveis aumentos na demanda da rede, mas supõe que, se o consumo de recursos

de uma máquina física exceder certo limiar, então um conjunto de migrações deve ser disparado para reorganizar o mapeamento entre os recursos virtuais e os recursos físicos.

4.12 Observações

O gerenciamento do compartilhamento de recursos em roteadores virtuais tem várias vertentes a serem exploradas. Nos próximos capítulos, é discutida a proposta VIPER, cujo objetivo é criar um sistema de controle local eficiente para garantir um isolamento forte, um alto desempenho no encaminhamento de pacotes e a disponibilização das primitivas básicas de QoS dentro de um sistema de rede virtualizado.

Capítulo 5

VIPER: *Virtual network Isolation, Policy Enforcement, and Resource sharing system**

O VIPER é um sistema para gerenciar redes virtuais. Entre os principais objetivos do VIPER, destacam-se o isolamento das redes virtuais, através do controle do uso dos recursos compartilhados, como CPU, memória e banda passante, e o provimento de QoS. O VIPER garante um isolamento robusto e acurado entre as máquinas virtuais além de permitir uma diferenciação dos recursos físicos atribuídos a cada rede virtual em conformidade com os acordos de nível de serviço (*S*ervice Level Agreement – SLA). Outras funcionalidades do VIPER incluem o monitoramento de quebras de SLA, o controle de acesso de novas redes virtuais à máquina física e a modificação dinâmica dos parâmetros da rede.

O VIPER foi implementado na plataforma de virtualização Xen para avaliação do seu desempenho. Por essa razão, a descrição detalhada da proposta será apresentada com base nessa plataforma. A utilização da proposta em outras plataformas depende de pequenos ajustes na monitoração do uso dos recursos e na aplicação da punição, como será explicado na Seção 6.5.

5.1 Arquitetura do VIPER

A Figura 5.1 mostra a interação entre o VIPER e o Xen em alto nível. O VIPER disponibiliza uma interface entre os operadores de redes virtuais e os provedores de infraestrutura, garantindo que os SLAs requisitados para cada rede virtual sejam providos pelas máquinas físicas. O VIPER é constituído por um servidor localizado no Domínio 0, e, para os roteadores virtuais que utilizam separação de planos, um

*Esse capítulo é baseado no texto publicado em [24, 25, 40].

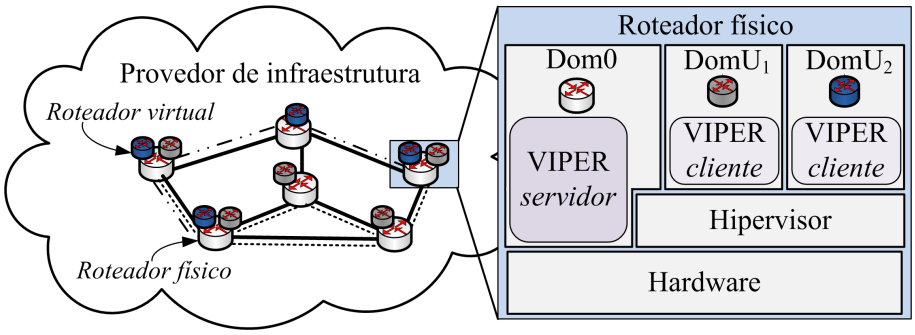


Figura 5.1: Visão geral da arquitetura do VIPER aplicada ao Xen, assumindo duas redes virtuais, onde cada DomU é um roteador virtual.

cliente na máquina virtual.

A arquitetura detalhada do VIPER é mostrada na Figura 5.2. O principal módulo do VIPER é o gerenciador de recursos compartilhados, o qual é responsável pelo monitoramento e punição de redes virtuais para garantir o isolamento entre as redes. O controlador de admissão de novas redes virtuais verifica os requisitos das redes virtuais que se candidatam a uma fatia dentro do nó físico, avaliando se é possível ou não receber a nova máquina virtual. Os demais módulos são funções de gerenciamento, dentre os quais se destacam os módulos de QoS para operador e provedor, que possibilitam a diferenciação de tráfego dentro das redes e entre redes. Além disso, o VIPER também garante uma comunicação segura entre o Domínio 0 e as máquinas virtuais para troca de dados de configuração e controle, assim como permite a opção por separação de planos, caso a rede necessite de alto desempenho, ou por roteamento pela máquina virtual, caso a rede precise de alta flexibilidade no plano de dados.

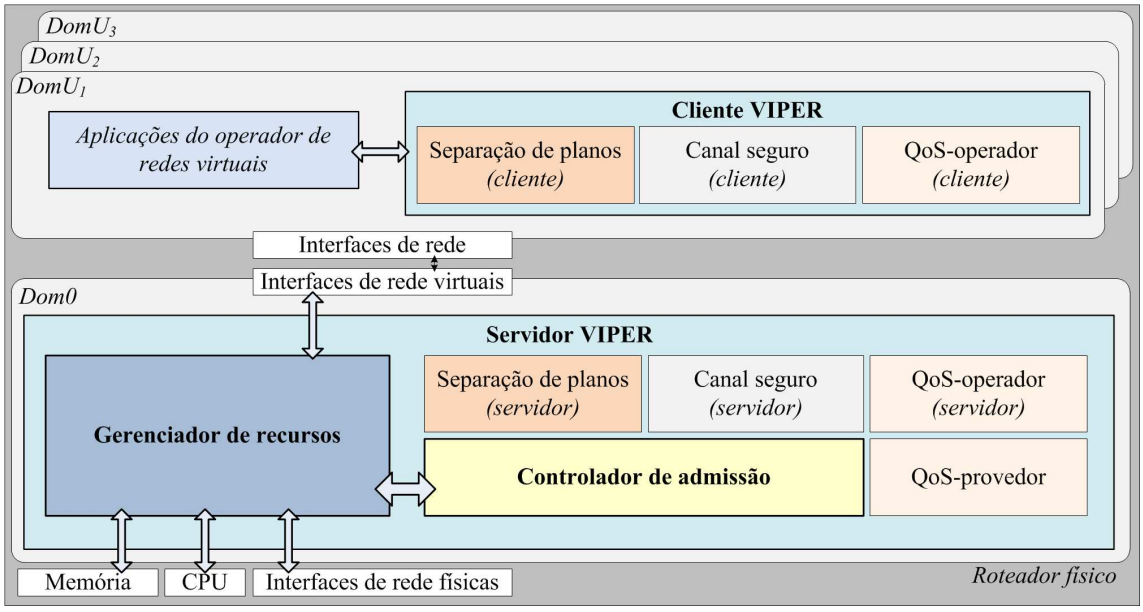


Figura 5.2: Arquitetura do VIPER.

O gerenciador de recursos compartilhados pode funcionar com diferentes políticas de controle de uso de recursos. Na Seção 5.2, dois tipos de política de controle são propostas e detalhadas. Além disso, essa seção também descreve o controlador de admissão de novas redes virtuais.

5.1.1 Módulos de separação de planos

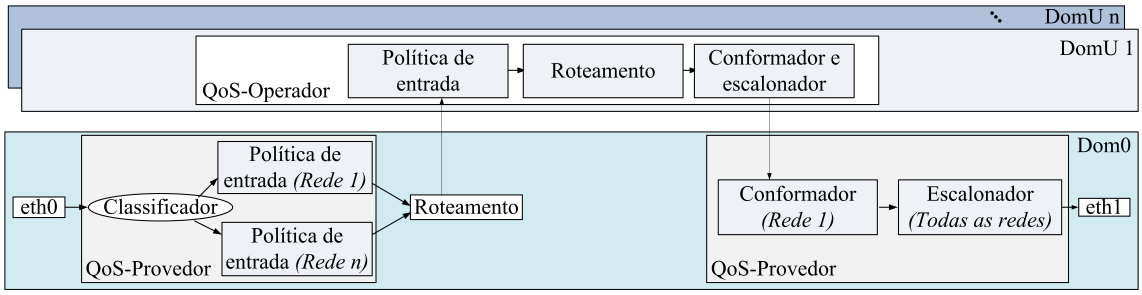
Os módulos de separação de planos têm como objetivo separar as funções de controle da rede da função de encaminhamento de dados. A principal função de controle da rede é criar a tabela de encaminhamento a partir das mensagens do protocolo de roteamento que cada roteador virtual recebe. Como mencionado anteriormente, para um maior desempenho no encaminhamento dos dados, esta função de encaminhamento deve ser executada no Dom0. No entanto, ao separar a função de encaminhamento da função de controle, os roteadores virtuais ficam impedidos de atualizarem suas tabelas de encaminhamento e seus filtros de pacote, porque eles não têm permissão de acesso à memória do Dom0. A solução no VIPER foi manter tabelas e filtros nos DomUs e fazer uma réplica no Dom0. Portanto, os clientes VIPER (nos DomUs) possuem um módulo que faz o monitoramento das modificações na tabela de encaminhamento e nos filtros de pacote, e o servidor (no Dom0) faz o mapeamento tanto da tabela de encaminhamento quanto do filtro de pacotes construídos em cada DomU em uma tabela de encaminhamento e um filtro de pacotes no Dom0.

Toda mudança na tabela de encaminhamento ou filtro de pacotes do roteador virtual (DomU) é atualizada imediatamente na sua réplica no Dom0. Por essa razão, o módulo de monitoração de modificações na tabela de encaminhamento e nos filtros observa a interface de entrada do DomU e verifica as mudanças quando uma nova mensagem de controle chega ao domínio. Em seguida, sempre que for encontrada uma diferença na tabela ou no filtro, apenas a diferença é transmitida para o Dom0, através do módulo de comunicação segura. A comunicação segura, explicada com mais detalhes na Seção 5.1.3, é essencial para que não exista nenhuma possibilidade de interferência de outra rede virtual com más intenções. Do lado do servidor do VIPER, o módulo de separação de planos, ao receber uma mensagem notificando uma modificação na tabela ou filtro, busca pelas definições da rede para descobrir em qual tabela do Dom0 aquelas modificações devem ser inseridas.

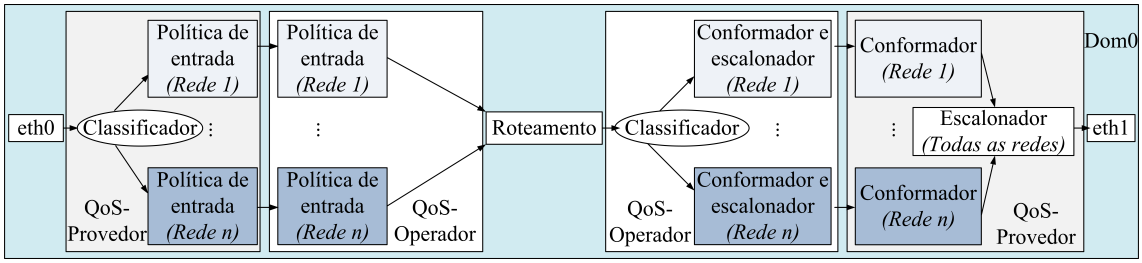
5.1.2 Provimento de QoS

Uma das principais vantagens do VIPER com relação a outras propostas é o suporte para inserção de QoS na infraestrutura de redes virtuais. Embora um roteador virtual que esteja contido dentro de uma máquina virtual possa configurar

um controle de tráfego para os pacotes que chegam até a máquina virtual, isso não é suficiente para garantir a qualidade de serviço uma vez que o controle é feito apenas no ambiente virtualizado. Uma máquina virtual não possui controle sobre o nível mais abaixo e, portanto, não controla o que acontece com o tráfego até a sua chegada na máquina virtual em um sentido e após deixar a máquina virtual no outro sentido, o que pode incluir atrasos e descarte de pacotes. Assim, é necessário incluir algumas regras para diferenciação de redes virtuais antes da chegada e após a saída do pacote, como mostrado na Figura 5.3(a). Nessa figura são apresentados os pontos aonde o provedor de infraestrutura e o operador de rede virtual podem inserir suas configurações de QoS. De acordo com esse esquema, um provedor poderia inserir prioridades para o acesso à rede física pelas redes virtuais, assim como poderia limitar algum tipo específico de tráfego de uma rede.



(a) Assumindo roteamento pela máquina virtual.



(b) Assumindo o uso de separação de planos, ou seja, o encaminhamento pelo Dom0.

Figura 5.3: Esquema para provimento de QoS no VIPER.

A utilização da separação de planos modifica o esquema para provimento de QoS tanto para o operador de redes virtuais quanto para o provedor de infraestrutura, pois a separação de planos implica na inserção do módulo de provimento de QoS do operador dentro do Dom0, como mostrado na Figura 5.3(b). Assim, o VIPER precisa dar suporte para que cada máquina virtual possa configurar o seu controle de tráfico no Dom0, garantindo que uma rede virtual não poderá interferir no controle de tráfego das demais redes virtuais. Por essa razão, a arquitetura do VIPER tem um módulo de servidor e outro de cliente no provimento de QoS para o operador de redes virtuais.

5.1.3 Módulo de estabelecimento de um canal seguro

O objetivo do módulo de estabelecimento de um canal seguro é permitir a comunicação entre o Dom0 e os DomUs utilizando autenticação mútua e privacidade na transferência dos dados. Esse deve ser um módulo leve, pois é usado com frequência na atualização do plano de dados no Dom0. A autenticação mútua é necessária para garantir que nenhuma máquina adversária tentará se passar por uma máquina bem comportada ou pelo Dom0 para gerar dados falsos no plano de dados no Dom0.

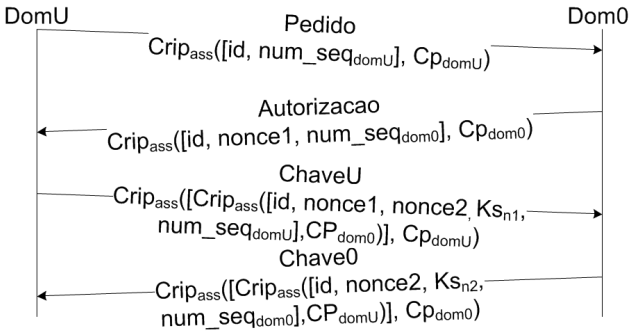
A comunicação segura é composta por dois protocolos: um para troca de chaves de sessão, descrito na Figura 5.4(a) e um para troca de dados entre as máquinas, descrito na Figura 5.4(b), nos quais se assume que Cp é a chave privada, CP é a chave pública, $crip([M], K)$ é a cifragem da mensagem M com a chave K e id é a identificação do nó fonte da mensagem. Uma vez que o custo computacional para o estabelecimento do canal seguro deve ser baixo, optou-se pela criação de um protocolo para autenticar e estabelecer uma chave de sessão e um protocolo para transmitir dados com segurança. Com isso, a autenticação pode ser feita utilizando-se criptografia assimétrica, enquanto que a comunicação é feita utilizando-se criptografia simétrica. Além disso, sempre que existe troca de mensagens de controle, é necessário impedir os ataques de *replay*, no qual o domínio adversário repete transmissões antigas de mensagens de controle de um domínio comum ou do Dom0. Um domínio adversário poderia, por exemplo, repetir uma mensagem de atualização antiga para adicionar rotas que acabaram de ser excluídas no plano de dados do domínio comum. Com isso, seria possível gerar inconsistências na tabela de encaminhamento no Dom0 com mensagens autenticadas. Para evitar ataques de *replay*, o Dom0 e as demais máquinas virtuais utilizam números de sequência e nonces, que são números aleatórios que devem ser usados apenas uma vez, nas mensagens de controle. Além disso, ao se chegar ao valor máximo do número de sequência, as máquinas virtuais devem trocar a sua chave de sessão, para garantir maior robustez contra *replays*.

Dessa forma, ao se garantir a autenticidade, a privacidade e a não-reprodutibilidade dos dados, é possível afirmar que a comunicação entre o Dom0 e os DomUs ocorre de forma segura.

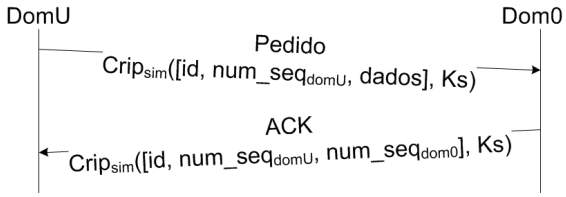
5.2 Algoritmos de controle do VIPER

Os algoritmos de controle do VIPER têm como base a monitoração dos recursos que estão sendo utilizados em cada máquina virtual. Com base nesses dados, o gerenciador de recursos pode calcular e aplicar a punição nas redes que apresentam um comportamento não desejado.

A seguir, são apresentadas todas as fases do controle proposto. Inicialmente, é



(a) Sequência de mensagens para estabelecimento de chaves de sessão $K_s = f(K_{s1}, K_{s2})$.



(b) Sequência de mensagens para sincronização de dados entre o Dom0 e o DomU.

Figura 5.4: Diagramas de tempo das mensagens para comunicação segura.

descrito o esquema de monitoramento utilizado, que é comum aos dois modelos de gerenciador de recursos propostos. O primeiro modelo é chamado de gerenciador de uso de recursos ociosos, pois reserva uma taxa mínima de recursos para cada rede virtual e divide os recursos ociosos pelas redes que têm uma demanda superior à reserva mínima. O outro gerenciador é chamado de gerenciador com alocação exata de recursos, pois restringe o uso dos recursos de cada rede ao que foi estabelecido nos SLAs. A aplicação da punição calculada com um dos dois algoritmos é apresentada na sequência e, por fim, é apresentado o controle de admissão.

5.2.1 Monitoração dos Recursos

Para cada rede virtual, o VIPER monitora o uso da banda nos enlaces de saída, o uso de CPU e uso de memória no Dom0 por rede virtual. A monitoração é realizada em cada período de tempo T e disponibiliza o consumo de recursos por rede virtual e o uso total de recursos no substrato físico.

A monitoração da banda é feita pela observação do tráfego em cada uma das interfaces de saída físicas e virtuais. A monitoração do gasto de CPU é feita com base na observação do número de pacotes que chegam a cada período T nas interfaces de entrada. Para identificar a qual rede pertence o tráfego, é preciso classificar os pacotes de cada rede na entrada e na saída dos enlaces físicos e dos enlaces entre máquinas virtuais.

Uma vez que o sistema operacional não fornece uma medida do gasto de CPU

no encaminhamento de pacotes por rede virtual, então esse valor é estimado com base no número de pacotes processados. Por simplicidade, assume-se que todos os pacotes, sejam de dados ou de controle, têm o mesmo impacto sobre o gasto de CPU no Domínio 0. O custo de processamento atribuído a cada pacote é diferenciado apenas pela origem e pelo destino do pacote, pois, como indicado na Tabela 4.1, o processamento de cada pacote tem um impacto diferente sobre a CPU do Dom0 dependendo de se o pacote vai para ou vem de um DomU, ou se vai para ou vem de uma máquina externa. Assim, a estimativa do uso de CPU é feita ponderando-se o número de pacotes pelo custo de processamento de cada pacote. Além disso, em transferências entre DomUs, o DomU de origem se responsabilizará por todos os custos de uso de CPU, para impedir que um domínio malicioso consiga exaurir os recursos de CPU de outro domínio através de uma transmissão de dados com alta taxa. Em transferências entre DomU e Dom0, os custos de uso de CPU são sempre contabilizados para o DomU. Assim, é preciso conhecer o custo para fazer a transferência entre máquinas virtuais (C_1), entre uma máquina virtual e uma externa (C_2) e vice-versa (C_3), entre o Dom0 e as máquinas virtuais (C_4) e vice versa (C_5), e entre máquinas externas (C_6). Com base nesses dados, o gasto de processamento (G_r) para uma rede virtual r é dado por:

$$G_r = \sum_{i=1}^{6} N_{r_i} \cdot C_i, \tag{5.1}$$

onde N_{r_i} é o número de pacotes do tipo i transitados pela rede virtual r.

É importante notar que o tamanho do pacote não importa para o cálculo de CPU, mas sim o número de pacotes, pois o processamento depende apenas do cabeçalho e não da carga útil do pacote [106, 107].

A memória é estimada apenas pelo volume de regras de filtragem e de encaminhamento de pacotes no Dom0 de cada rede virtual. De fato, a filtragem e o encaminhamento de pacotes implicam em um gasto de memória no Dom0, mas esse é pequeno, uma vez que o gargalo para o encaminhamento acontece primeiro na CPU do Dom0. Dessa forma, o VIPER contabiliza apenas o tamanho da tabela de encaminhamento e do filtro de pacotes que a rede virtual insere no Dom0. Assim, uma rede virtual que opte por fazer o encaminhamento de pacotes pela máquina virtual tem custo de memória zero para o VIPER, pois essas duas operações serão feitas dentro da máquina virtual. Já as redes que optarem pela separação de planos devem observar também um volume máximo de regras de filtragem e de tamanho de tabela de encaminhamento.

5.2.2 Gerenciadores de recursos compartilhados

Gerenciador com alocação de recursos ociosos

A idéia chave do gerenciador com alocação de recursos ociosos é prover uma reserva mínima para cada rede virtual e repartir a alocação dos recursos além dessa reserva mínima entre as redes. Os recursos não reservados são divididos entre as redes virtuais de acordo com o parâmetro peso, especificado pelo provedor de infraestrutura. Quanto maior o peso de uma rede, maior o seu acesso aos recursos não reservados. Portanto, no gerenciador com alocação de recursos ociosos todas as redes têm suas demandas atendidas desde que existam recursos ociosos suficientes.

O gerenciador monitora o uso e aloca os recursos do Dom0 para os DomUs a cada período de tempo T. Portanto, o controlador monitora o uso total dos recursos do Dom0, $U(t)$, assim como o uso de recursos por roteador virtual i, dado por $U_i(t)$, e pune os DomUs mal comportados.

A alocação de recursos do Dom0 às redes virtuais se processa de duas formas: por reserva fixa e por demanda. A alocação por reserva fixa é feita de acordo com os parâmetros configurados pelo administrador da máquina física. Assim, já na configuração, especifica-se e reserva-se uma quantidade fixa de cada recurso do Dom0 para cada DomU e, por consequência, garante-se uma qualidade mínima para cada rede virtual. A alocação por demanda garante uma melhora no desempenho das redes virtuais com demandas além da qualidade mínima, porque disponibiliza os recursos não utilizados evitando a ociosidade dos recursos do Dom0. Portanto, são disponibilizados para alocação por demanda os recursos que não foram reservados de forma fixa e também os recursos reservados que não foram usados.

Com base nesses conceitos, uma premissa do controlador é disponibilizar os recursos fixos para uma rede i, representados como um percentual, $Perc_i$, dos recursos totais do Dom0, $R(t)$, sempre que houver demanda. Outra premissa é que todos os recursos que não são usados são alocados por demanda aos roteadores virtuais segundo uma prioridade pré-estabelecida na configuração do gerenciador. A prioridade é dada por um parâmetro, chamado de peso, dado por $W_i \in \mathbb{Z}$, onde $1 < W_i \leq 1000$. Quanto maior o peso de uma rede virtual, maior o acesso que ela tem aos recursos ociosos. Assim, a alocação por demanda dos recursos com base nos pesos provê uma qualidade adicional diferenciada para cada rede virtual.

- **Cálculo da punição**

A punição dos domínios adversários (maliciosos ou de comportamento não-desejável) é feita pelo descarte de pacotes, caso se observe um uso excessivo de CPU ou de banda no Domínio 0. A punição da memória é feita através do descarte

de algumas entradas da tabela de roteamento. Com isso, os pacotes correspondentes às rotas excluídas são processados pela máquina virtual ao invés do Domínio 0.

O algoritmo proposto para o cálculo da punição busca um valor de probabilidade de descarte de pacotes/rotas que equilibre o uso dos recursos do Dom0 pelos roteadores virtuais. Os descartes são feitos apenas se o roteador virtual usou mais do que os recursos reservados para ele e se o uso dos recursos atingir um nível crítico, dado por um percentual L dos recursos totais $R(t)$, que possa impedir outros roteadores de usarem os seus recursos reservados. A taxa de descarte é atualizada a cada intervalo T, para que a punição possa acompanhar a taxa de chegada de pacotes de cada rede virtual.

O algoritmo proposto para o gerenciador com alocação de recursos ociosos depende do total de recursos não reservados, dado por

$$R_{NR}(t) = R(t) - \sum_{\forall i} Perc_i \cdot R(t), \tag{5.2}$$

e também de C_a, que é uma constante de ajuste escolhida pelo administrador. A atualização da probabilidade de descarte da rede virtual i no intervalo $t+T$, $P_{ui}(t+T)$, é calculada com base nas regras a seguir.

O cálculo da punição é feito de forma diferenciada, dependendo do volume de recursos ociosos, do valor anterior da punição da rede e do uso de recursos no intervalo de monitoração anterior. De forma resumida, as redes são classificadas em cada um dos tipos abaixo para o cálculo da punição.

- **Tipo 1 -** se a rede não excedeu a sua reserva fixa e não está sendo punida, então a sua punição é zero.

- **Tipo 2 -** se a rede usou menos recursos que a sua reserva fixa, mas está sendo punida, então a sua punição é reduzida rapidamente de acordo com a equação

$$P_{ui}(t+T) = max(P_{ui}(t) - C_a \cdot (1 - \frac{1}{W_i}) \cdot \frac{P_{ui}(t)}{3}, 0). \tag{5.3}$$

  Nessa equação, quanto maior o peso W_i, mais rapidamente é reduzida a punição da rede. Além disso, a redução da punição também é proporcional ao valor da punição anterior, para reduzir as oscilações até atingir a punição ideal para a rede.

- **Tipo 3 -** se a rede está sendo punida e usou mais que a sua reserva fixa, mas existem recursos ociosos, então a sua punição é reduzida lentamente de acordo com a equação

$$P_{ui}(t+T) = max(P_{ui}(t) - (1 + (1 - max(U_{oi}, 1)) \cdot (1 - \frac{1}{W_i}) \cdot \frac{P_{ui}(t)}{3 + \frac{1}{W_i}}, 0), \tag{5.4}$$

aonde U_{oi} é um valor derivado do uso dos recursos ociosos no último intervalo de monitoramento, dado por

$$U_{oi} = \frac{U_i(t) - Perc_i \cdot R(t)}{R_{NR}(t)}. \tag{5.5}$$

Novamente, quanto maior o peso, mais rápida a redução da punição. Além disso, se a rede usou proporcionalmente mais recursos não reservados que as outras redes, então a sua punição é reduzida mais lentamente.

- **Tipo 4 -** se os recursos ociosos estão quase se exaurindo e a rede usou recursos além da sua reserva fixa, então a sua punição passa a ser

$$P_{ui}(t+T) = P_{uinicial}, \tag{5.6}$$

se a rede não estava sendo punida no intervalo anterior, e

$$P_{ui}(t+T) = min(P_{ui}(t) + (1+U_{oi}) \cdot (1+\frac{1}{W_i}) \cdot \frac{P_{ui}(t)}{(3-\frac{1}{W_i})}), 1), \tag{5.7}$$

se a rede já estava sendo punida.

A idéia chave é que a punição da rede seja aumentada até que existam recursos ociosos na rede e seja reduzida toda vez que existem recursos ociosos.

Uma descrição mais detalhada da lógica do gerenciador é dada no Algoritmo 1. As entradas para esse algoritmo, descritas na linha 1, são a punição atual, o peso da rede, o total de recursos físicos para o recurso avaliado[1], o total de recursos utilizados, o total de recursos utilizado pela rede, o total de recursos que não está reservado, o percentual de recursos máximo que pode ser usado e a constante de ajuste para o recurso físico em questão. Como saída, é obtida apenas a punição da rede virtual no próximo intervalo T. Como mostrado na linha 2, o algoritmo só é executado se o consumo de recursos exceder o limiar máximo ou se a rede já possui alguma punição. Se o uso excedeu o limiar máximo, então alguma rede pode estar sendo prejudicada e a punição de cada rede precisa ser reavaliada. Se alguma rede possui alguma punição, essa deve ser sempre atualizada de acordo com a demanda de recursos daquela rede virtual e a disponibilidade de recursos no nó físico. Assim, enquanto a punição for diferente de zero, o algoritmo precisa ser executado para encontrar a punição mais adequada de acordo com as políticas estabelecidas para o nó virtual. Se não foi excedido o uso total de recursos e a rede não está sendo punida, então o algoritmo é finalizado e a punição da rede continua como zero, como mostrado na linha 22. Na linha 4, o valor U_{o_i} é calculado, representando um indicativo relativo ao

[1]O algoritmo deve ser executado para cada rede, em cada um dos recursos físicos.

uso de recursos ociosos pela rede virtual. Assim, é calculado o quanto a rede utilizou além da sua reserva e esse valor é dividido pelo total de recursos não-reservados. O valor de U_{o_i} pode ser maior do que um, pois $D(t)$ representa apenas os recursos não reservados, que podem ser inferiores ao total de recursos ociosos, ou seja, que não foram utilizados por nenhuma máquina virtual.

Algoritmo 1: Cálculo da punição para cada rede virtual.

```
1  Entrada: P_ui(t), W_i, Perc_i, R(t), U(t), U_i(t), R_NR(t), L, C_a
   Saída: P_ui(t+T)
2  se (Perc_i · R(t) < U_i(t)) ou (P_ui(t) > 0) então
3      % Calcula uso de recursos ociosos
4      U_oi = (U_i(t) − Perc_i · R(t))/R_NR(t)
5      se (Perc_i · R(t) < U_i(t)) então
6          se (L · R(t) ≤ U(t)) então
7              se (P_ui(t) > 0) então
8                  % Como não existem recursos ociosos, alguma rede pode estar sendo
                     prejudicada e a punição é aumentada
9                  P_ui(t+T) = min(P_ui(t) + (1 + U_oi) · (1 + 1/W_i) · P_ui(t)/(3 − 1/W_i), 1)
10             senão
11                 P_ui(t+T) = P_uinicial   % Seta um valor inicial de punição
12             fim
13         senão
14             % Como existem recursos ociosos que podem ser distribuídos, a punição é
                 reduzida
15             P_ui(t+T) = max(P_ui(t) − (1 + (1 − max(U_oi, 1)) · (1 − 1/W_i) · P_ui(t)/(3 + 1/W_i), 0)
16         fim
17     senão
18         % Como utilizou apenas os seus recursos fixos, a punição é diminuida rapidamente
19         P_ui(t+T) = max(P_ui(t) − C_a · (1 − 1/W_i) · P_ui(t)/3, 0)
20     fim
21 senão
22     P_ui(t+T) = 0
23 fim
```

Nas linhas seguintes, o algoritmo avalia se a punição do nó virtual deve ser aumentada, reduzida ou zerada, de acordo com as condições da rede e com o uso dos recursos pela rede virtual. Primeiramente, se avalia se os recursos utilizados pela rede virtual estão acima do valor reservado para essa rede. Caso não esteja, então a punição da rede deve ser reduzida, como indicado na linha 19. Quanto maior o peso da rede, mais rapidamente a punição deve ser reduzida. É importante observar que mesmo que o DomU consuma menos do que sua reserva fixa, a punição não é imediatamente zerada para evitar instabilidades. O algoritmo busca a punição mais adequada para o DomU de acordo com as políticas utilizadas e essa busca pode ter instantes nos quais a punição vai estar acima ou abaixo do valor correto até a estabilização do valor da probabilidade de punição. Em seguida, na linha 6, se a rede excedeu a sua reserva mínima, é preciso avaliar se esse excesso prejudicou a rede como um todo ou não. Assim, se o uso de recursos totais excedeu o limiar máximo, L, a punição da rede deve ser aumentada. Caso contrário, ela deve ser reduzida. O

aumento da punição, que segue a equação descrita na linha 9, é maior para as redes que possuem pesos menores, assim como é maior para as redes que possuem um U_{o_i} maior. A redução da punição segue a equação da linha 15 e quanto menor o U_{o_i} e quanto maior o peso, mais rapidamente a punição é reduzida.

Além do controle da punição pelo algoritmo, para impedir que as operações de E/S (comunicação de rede) geradas pelos roteadores virtuais interrompam os demais serviços do Dom0 por sobrecarga de CPU, uma punição residual pode ser aplicada constantemente nas interfaces de saída das máquinas virtuais. Tal punição deve ser pequena o suficiente para não impactar transmissões de baixo volume entre roteadores virtuais, mas deve impedir que o DomU consuma todos os recursos do Dom0, aumentando o seu tempo de resposta.

Gerenciador com alocação exata de recursos

O gerenciador com alocação exata de recursos tem um objetivo distinto do gerenciador com alocação de recursos ociosos. Nesse gerenciador, o controle é bem mais acurado e "à la carte", pois se assume que cada rede virtual possui uma especificação completa do seu acordo de nível de serviço, permitindo a modelagem de diversos tipos de padrões de tráfego. Além disso, o controle, nesse gerenciador, é feito através de observações a curto e a longo prazo.

O gerenciador assume a existência de um conjunto de características para especificar o uso de cada recurso físico por rede de acordo com o SLA contratado, sendo elas:

- **reserva de taxa a curto prazo ($R_c[i]$)** - taxa de recursos que deve ser garantida para a rede i sempre que houver demanda em um intervalo de tempo curto I_c;
- **reserva de taxa a curto prazo exclusiva ($R_e[i]$)** - taxa de recursos reservados a curto prazo que não deve ser disponibilizada a outras redes, mesmo que não exista demanda da rede i;
- **reserva de volume a longo prazo ($V_l[i]$)** - volume de recursos que deve ser garantido apenas em um intervalo longo de tempo I_l, caso exista demanda. Também pode ser representada pela taxa média $R_l[i]$, aonde $V_l[i] = R_l[i] \cdot I_l$;
- **taxa a longo prazo máxima ($R_{max}[i]$)** - taxa de recursos máxima que pode ser provida em um intervalo longo de tempo I_l, caso exista demanda;

Parâmetros para a construção de limitadores, conformadores e prioridades de tráfego também fazem parte do SLA, mas são tratados pelos módulos de QoS do operador e QoS do provedor.

Com base nesse conjunto de características, é possível especificar diferentes tipos de perfis de uso de recursos físicos. A escolha dos parâmetros de cada rede deve variar de acordo com a necessidade do operador da rede virtual e do preço que ele está disposto a pagar por sua fatia de rede.

- **Cálculo da punição**

O gerenciador com alocação exata de recursos proposto se baseia nas características especificadas para cada rede virtual i para determinar se a demanda da rede $D[i]$ está dentro ou fora dos SLAs determinados.

A idéia chave do algoritmo é que todas as redes que não excederam a sua reserva a longo prazo recebam uma fatia dos recursos que contenha a sua reserva de taxa a curto prazo e um valor proporcional ao peso calculado sobre os demais recursos, conforme a equação

$$proximo\_liberado[rede] = R_c[rede] + total\_disponivel \cdot \frac{peso[rede]}{peso\_total}, \qquad (5.8)$$

onde *total_disponível* é o total de recursos além das reservas de taxa a curto prazo disponível para alocação e *peso_total* é o somatório do peso de todas as redes. Após essa divisão inicial dos recursos, verificam-se recursivamente quais redes possuem uma demanda inferior aos recursos recebidos (*proximo_liberado*) e essa diferença é redistribuída pelas redes que possuem uma demanda superior aos recursos recebidos. Ao final do algoritmo, as punições são calculadas para as redes que possuem uma demanda acima do valor de recursos recebidos de acordo com

$$punicao[rede] = 1 - \frac{proximo\_liberado[rede]}{D[rede]} \qquad (5.9)$$

No gerenciador com alocação exata de recursos, o parâmetro peso não é configurado pelo administrador, mas calculado dinamicamente pelo gerenciador de acordo com o uso da reserva de volume a longo prazo. O peso é calculado de forma a dar mais prioridade aos recursos para as redes que possuem um maior volume de recursos reservados a longo prazo ainda não utilizados. Assim, o peso é atualizado a cada intervalo curto de acordo com o valor da reserva de volume a longo prazo e com o volume de recursos usado desde o início do intervalo longo.

Resumidamente, a relação entre as variáveis do gerenciador proposto se encontra na Figura 5.5. A vazão de uma rede virtual no próximo intervalo curto é determinada pela demanda da rede e pela punição arbitrada à rede. Com base na vazão obtida em cada intervalo curto, é possível calcular o volume de recursos usado por rede virtual desde o início do intervalo longo até o momento atual (utilizado). Esse valor, associado à reserva de volume a longo prazo da rede, é usado para atualizar o peso

atribuído à rede no intervalo curto atual. Com base nesses dados, o algoritmo pode calcular o quanto de recursos a rede poderá usar no próximo intervalo curto (próximo liberado) e a respectiva punição.

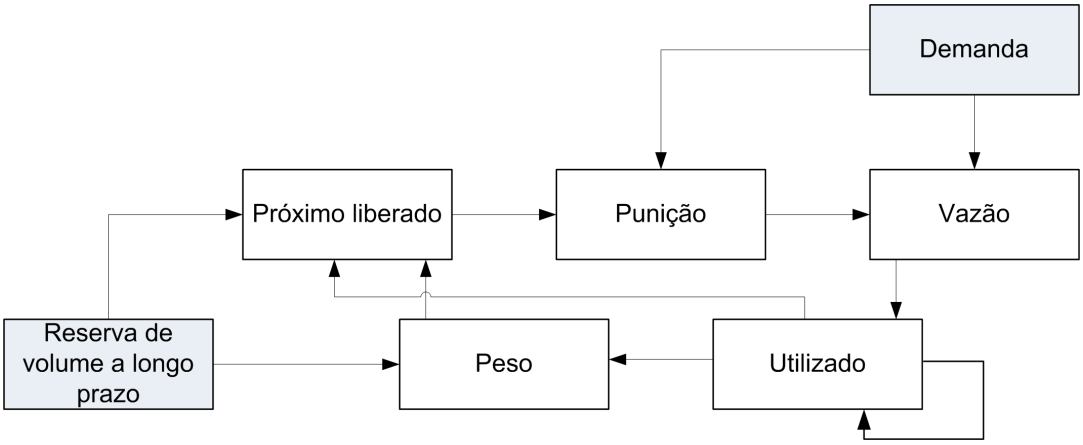


Figura 5.5: Relação entre as variáveis do gerenciador com alocação exata de recursos.

O Algoritmo 2 é uma versão detalhada do cálculo das punições de cada rede[2]. Esse algoritmo deve ser executado a cada I_c para cada um dos recursos monitorados do Dom0, ou seja, a banda de saída de cada enlace físico, a CPU e a memória. Esse algoritmo assume que a $D[i]$ da rede i no último I_c é conhecida e esse valor é utilizado como uma estimativa da demanda da rede no próximo I_c. Com base na estimativa de demanda de todas as redes, os recursos são divididos entre as redes virtuais. O Algoritmo 2 assume ainda que são conhecidos o número de redes virtuais hospedadas (N_H), o total de recursos físicos disponibilizados para as redes virtuais (R_t), o total de recursos utilizado por rede até o momento dentro do intervalo longo ($utilizado[i]$) e o peso atual de cada rede virtual i ($peso[i]$).

Primeiramente, o algoritmo zera todas as punições, calcula o total de recursos não reservados e o somatório dos pesos de todas as redes. Esses valores são utilizados para calcular o volume de recursos que devem ser liberados para cada rede no próximo I_c, o que é representado pela variável $proximo\_liberado[\ ]$. Assim, caso a rede virtual ainda não tenha gastado toda a sua reserva a longo prazo, ela recebe como $proximo\_liberado[\ ]$ a sua reserva a curto prazo mais uma fatia proporcional ao seu peso dos recursos não-reservados. Caso contrário, a política do gerenciador define que a vazão da rede deve ser limitada pela reserva a curto prazo desta rede. Com isso, todas as redes que possuem uma demanda superior ao $proximo\_liberado[\ ]$ recebem uma punição proporcional ao uso excedente. Para aperfeiçoar a distribuição dos recursos, o gerenciador verifica, após esses cálculos, se alguma rede recebeu uma fatia dos recursos superior à demanda estimada, o que é feito pela função $verifica(\ )$ na linha 10. Caso isso ocorra, as redes que tiveram

[2]A versão estendida desse algoritmo considera os vetores R_e e R_{max}.

Algoritmo 2: Cálculo da punição para cada rede virtual.

Entrada: R_t, N_H, I_c, I_l, $R_c[\,]$, $V_l[\,]$, $peso[\,]$, $utilizado[\,]$, $D[\,]$
Saída: $punicao[\,]$
$Zerar(punicao[\,])$;
$total\_disponivel = R_t - \sum_{i=1}^{N_H}(R_c[i])$;
$peso\_total = \sum_{i=1}^{N_H} peso[i]$;
para $rede = 1$ *até* N_H **faça**
    se $(utilizado[rede] < V_l[rede])$ **então**
    $proximo\_liberado[rede] = R_c[rede] + total\_disponivel \cdot peso[rede]/peso\_total$;
    senão $proximo\_liberado[rede] = R_c[rede]$;
    se $(D[rede] > proximo\_liberado[rede])$ **então**
    $punicao[rede] = 1 - proximo\_liberado[rede]/D[rede]$;
fim
enquanto $(verifica(proximo\_liberado[\,], D[\,], N_H) == 1)$ **faça**
    $peso\_total = 0$;
    para $rede = 1$ *até* N_H **faça**
        se $(proximo\_liberado[rede] > D[rede])$ **então**
        $proximo\_liberado[rede] = D[rede]$;
        se $((D[rede] > proximo\_liberado[rede])\&(utilizado[rede] < V_l[rede]))$ **então**
        $peso\_total += peso[rede]$;
    fim
    $sobra = R_t - \sum_{i=1}^{N_H}(proximo\_liberado[i])$;
    para $rede = 1$ *até* N_H **faça**
        se $((D[rede] > proximo\_liberado[rede])\,\&(utilizado[rede] < V_l[rede]))$ **então**
        $proximo\_liberado[rede] += sobra \cdot peso[rede]/peso\_total$;
        se $(proximo\_liberado[rede] < D[rede])$ **então**
        $punicao[rede] = 1 - proximo\_liberado[rede]/D[rede]$;
        senão $punicao[rede] = 0$;
    fim
fim

uma demanda inferior ao *proximo_liberado*[ ] tem essa variável reduzida até a sua demanda e a diferença é disponibilizada para as demais redes. O *peso_total* também é recalculado, considerando apenas as redes que devem participar da divisão dos recursos disponibilizados. Em seguida, o *proximo_liberado*[ ] das redes que tem uma demanda excedente e que ainda não usaram toda a reserva a longo prazo é recalculado com base nos recursos disponibilizados proporcionalmente ao peso da rede. Por fim, as punições de todas as redes são atualizadas. Esse processo é repetido até que nenhuma rede possua um *proximo_liberado*[ ] superior a sua demanda. Com isso, os recursos físicos são distribuídos de forma proporcional ao peso e à demanda de cada rede.

- **Atualização do peso**

Outro ponto importante do módulo controlador é o cálculo do valor do peso de cada rede em cada recurso em cada intervalo curto. O peso é uma variável adaptativa, calculada com base no que a rede já utilizou dos recursos (*utilizado*[ ]) e na reserva a longo prazo, como mostrado no Algoritmo 3. A base do algoritmo para o ajuste do peso é dar a cada rede um peso proporcional à razão entre o volume de dados ainda não gasto da reserva a longo prazo da rede e o volume ainda não gasto da reserva a longo prazo de todas as redes. Com isso, as redes que possuem maior reserva a longo prazo disponível ganham prioridade na alocação dos recursos disponíveis, recebendo uma fatia maior dos recursos caso haja demanda. Consequentemente, aumenta-se a probabilidade de que a reserva a longo prazo seja integralmente provida a todas as redes virtuais.

O Algoritmo 3, inicialmente, verifica se é o final de um intervalo longo, e caso seja, zera a variável *contador_longo*, que conta o tempo que já se passou dentro do intervalo longo atual, e zera o vetor *utilizado*[ ]. Se não for o final do intervalo longo, então o algoritmo atualiza o valor do vetor *utilizado*[ ] para cada rede, de acordo com a demanda que foi obtida com a monitoração, como mostrado na linha 7. Em seguida, o contador longo também é atualizado com o tempo gasto no intervalo curto. Depois, na linha 10, é chamada a função *descobre_reserva_longa_total*( ), que calcula o volume de dados que foi reservado para todas as redes até o final do intervalo longo, mas que ainda não foi gasto, descontando o volume de dados reservado a curto prazo até o final do intervalo longo. Na linha 12, verifica-se se a rede ainda tem algum volume de recursos para ser utilizado além da reserva a curto prazo, com a função *descobre_reserva_longa*( ), que é semelhante à função *descobre_reserva_longa_total*( ), mas fazendo o cálculo para apenas uma rede. Se a rede não pode gastar nada além da sua reserva mínima, o seu peso passa a zero. Caso contrário, o peso é calculado de acordo com o que a rede ainda pode gastar como reserva longa e o que as outras redes também podem gastar como reserva

Algoritmo 3: Cálculo do peso para cada rede virtual.

```
1  Entrada: N_H, I_c, I_l, R_c[ ], V_l[ ], utilizado[ ], D[ ], contador_longo
   Saída: peso[ ], contador_longo
2  se contador_longo > I_l então
3  |  contador_longo = 0
4  |  Zerar(utilizado[])
5  senão
6  |  para rede=1 até N_H faça
7  |  |  utilizado[rede]+ = D[rede] · I_c
8  |  fim
9  |  contador_longo+ = I_c
10 |  soma = descobre_reserva_longa_total(utilizado[ ], V_l[ ], R_c[ ], I_l, I_c, contador_longo)
11 |  para rede = 1 até N_H faça
12 |  |  se
   |  |  (descobre_reserva_longa(utilizado[rede], V_l[rede], R_c[rede], I_l, I_c, contador_longo) ≤
   |  |  0 então
13 |  |  |  peso[rede] = 0
14 |  |  senão
15 |  |  |  pontos_totais = V_l[rede] − R_c[rede] · I_l
16 |  |  |  se utilizado[rede] > R_c[rede] · contador_longo então
17 |  |  |  |  pontos_utilizados = utilizado[rede] − R_c[rede] · contador_longo
18 |  |  |  |  peso[rede] = (pontos_totais − pontos_utilizados)/soma
19 |  |  |  senão
20 |  |  |  |  peso[rede] = pontos_totais/soma
21 |  |  |  fim
22 |  |  fim
23 |  fim
24 fim
```

longa. Assim, na linha 15, é calculado o volume de recursos que cada rede recebe além da sua reserva a curto prazo. Na linha 16, é verificado se a rede virtual já gastou um volume de recursos superior a sua reserva a curto prazo até o momento. Caso a rede não tenha utilizado nada além da reserva a curto prazo, então o seu peso é calculado como mostrado na linha 20, na qual o peso é dado como a proporção entre o que a rede pode gastar e o total de recursos que todas as redes juntas ainda podem gastar de acordo com suas reservas. Caso a rede já tenha utilizado algo além da sua reserva a curto prazo, então isso deve ser considerado no cálculo do peso, como descrito nas linhas 17 e 18. A variável *pontos_utilizados* guarda o quanto a rede virtual gastou além da reserva a curto prazo. O peso é então calculado descontando-se esse valor dos *pontos_totais* e dividindo o valor resultante por todos os pontos que ainda podem ser gastos pelas redes.

Com base nesse algoritmo, o peso de cada rede é atualizado de acordo com o uso da rede da sua reserva a longo prazo e do uso das demais redes da reserva a longo prazo. Com isso, espera-se prover uma garantia probabilística da reserva a longo prazo, pois as redes que possuem mais para ser gasto na parte restante do intervalo longo ganham maior acesso aos recursos sobressalentes.

5.2.3 Aplicação da punição

A punição é calculada no VIPER com base nos Algoritmos 1 ou 2, independente do recurso do Dom0 que esteja sendo monitorado. A aplicação da punição, no entanto, depende do tipo de recurso.

Quando existe uma sobrecarga dos recursos de processamento por uma rede virtual, a punição dessa rede é aplicada sobre todas as interfaces de entrada como um percentual de descarte de pacotes. Dessa forma, um percentual de pacotes correspondente à punição deixará de entrar no Dom0 reduzindo os gastos com o encaminhamento de pacotes. Quando existe uma sobrecarga de banda em uma interface de saída, os pacotes destinados àquela interface são descartados. Cabe observar que a punição de rede reduz o consumo de banda na interface de saída, mas os gastos de processamento para o Dom0 se mantêm, sendo contabilizados no gasto de CPU.

O consumo de memória é estimado no VIPER apenas pelo volume de regras de filtragem e de encaminhamento de pacotes no Dom0 de cada rede virtual. De fato, a filtragem e o encaminhamento de pacotes implicam em um gasto de memória no Dom0, mas esse é pequeno e pode ser desconsiderado. Dessa forma, o VIPER contabiliza apenas o tamanho da tabela de encaminhamento e do filtro de pacotes que a rede virtual insere no Dom0. Assim, uma rede virtual que opte por fazer o encaminhamento de pacotes pela máquina virtual tem custo de memória zero para o VIPER, pois essas duas operações serão feitas dentro da máquina virtual. Já as redes que optarem pela separação de planos deverão observar também um volume máximo de regras de filtragem e de tamanho de tabela de encaminhamento. A punição devido ao gasto excessivo de memória implica no descarte percentual de rotas da tabela de encaminhamento. Para impedir a perda de pacotes, uma rota padrão é criada encaminhando o pacote para a máquina virtual, na qual o plano de dados está completo com todas as rotas. Assim, se uma rede que opta pela separação de planos não deseja perder desempenho com o encaminhamento de pacotes pela máquina virtual, deve controlar o número de rotas instaladas no Dom0.

5.2.4 Controle de acesso de roteadores virtuais

O controlador de admissão de novas redes virtuais arbitra o acesso de novos roteadores à máquina física. O número de roteadores virtuais hospedados em uma máquina física influencia a provisão da reserva de volume a longo prazo. De fato, o mecanismo de controle de admissão é não trivial, pois diferentes perfis de rede podem usar a mesma quantidade de recursos a longo prazo. Assim, a diferença entre perfis de rede, o que inclui dados estáticos, como o tamanho do disco e da memória, e dados dinâmicos, como a vazão e o uso de CPU, deve ser levada em

consideração no cálculo da probabilidade de bloqueio de recurso. A probabilidade de bloqueio de recurso é definida como a probabilidade de existir demanda dentro dos limites estabelecidos nos SLAs de uma rede virtual e o recurso não ser provido.

A idéia chave do controlador de admissão é determinar se uma nova rede virtual pode ser hospedada no substrato físico com base na estimava do uso máximo de recursos físicos pelas outras redes virtuais. Essa estimativa é feita de acordo com o que foi contratado pelas redes virtuais e com as observações de uso real de recursos. Os dados monitorados utilizados pelo algoritmo de controle são guardados em histogramas, pois esses são estruturas simples e, portanto, de fácil manipulação. Assim, o controlador de admissão de novas redes virtuais usa as seguintes estruturas para arbitrar a entrada de novas redes no substrato físico:

- **histograma do substrato -** o histograma do substrato guarda a informação de uso do recurso do substrato por todas as redes virtuais;
- **uso médio de recursos por rede virtual -** uma média do uso dos recursos de cada rede virtual é armazenado para determinar a estimativa no aumento da demanda;
- **histograma de uso estimado para a nova rede virtual -** histograma utilizado para verificar se o substrato físico pode ou não aceitar mais uma rede.

Monitoramento e armazenamento

O VIPER armazena um conjunto de histogramas representando o uso do substrato. Novos histogramas são gerados para cada recurso dinâmico monitorado em cada intervalo de monitoração. Um conjunto de histogramas de um mesmo recurso modela a variação da carga ao longo de um período tempo (por exemplo, um dia).

Os histogramas do substrato são necessários para modelar o comportamento do uso agregado de recursos pelas redes virtuais. A idéia é que redes com perfis de tráfego diferentes levam a histogramas de recursos agregados diferentes, mesmo que o padrão de reserva, ou seja, a reserva a curto e a longo prazo, seja o mesmo entre as redes. Assim, duas redes podem ter o mesmo V_l e o mesmo R_c e ter comportamentos completamente diferentes. Consequentemente, o uso de recursos agregado para cada par de perfis de uso será diferente. Com isso, o histograma do substrato modela de forma realística o que está acontecendo com os recursos da rede física em cada intervalo de monitoração.

Um intervalo de monitoração é dado por $K_{adm} \cdot I_l$ intervalos longos, aonde K_{adm} é uma constante arbitrada pelo administrador de infraestrutura. Para evitar uma sobrecarga de armazenamento e de processamento, o sistema proposto seleciona aleatoriamente K_{rand}, o qual é o número de intervalos longos que não serão avaliados

após K_{adm} intervalos longos. Além disso, ao invés de armazenar todos os histogramas, o VIPER verifica a diferença entre o histograma atual e o histograma anterior. Para isso, o VIPER normaliza ambos os histogramas e verifica o maior erro no eixo y (e_m) entre os dois histogramas. Se a condição $|e_m| > E_{adm}$ for verdadeira, onde E_{adm} é um limiar especificado pelo administrador de infraestrutura, então o perfil atual é armazenado e um novo histograma é iniciado. Como consequência, os recursos monitorados que apresentam uma variação de perfil intensa geram mais histogramas.

O algoritmo de controle

O algoritmo do controlador estima a probabilidade de bloqueio de recurso após uma nova rede entrar no substrato físico para cada recurso dinâmico, ou seja, CPU, banda e memória. Essa probabilidade é usada como critério para aceitar ou não uma nova rede virtual.

As entradas para o algoritmo que arbitra a admissão de redes virtuais são os histogramas do substrato, a reserva de volume a longo prazo de cada rede virtual ($V_l[\,] = R_l[\,] \cdot I_l$) e o uso médio de recursos de cada rede virtual, $R_{avg}[\,]$.

O algoritmo de controle de admissão possui quatro passos:

- **Passo 1 -** Estimar o uso de recursos agregado em um histograma.
- **Passo 2 -** Estimar como o aumento de demandas até o limite contratado impacta o histograma de recursos agregados.
- **Passo 3 -** Estimar uma função de probabilidade de uso para a nova rede virtual com base nos valores de reserva contratados.
- **Passo 4 -** Calcular a probabilidade de colisão caso a nova rede estivesse usando o substrato físico.

Passo 1

O Passo 1 consiste basicamente da etapa de monitoramento e armazenamento descrito na Seção 5.2.4. Assim, ao final desse passo, o algoritmo proposto conhece os histogramas de uso agregado de banda, CPU e memória compartilhados para cada intervalo de monitoração.

Passo 2

Nesse passo, o algoritmo proposto estima como ficariam os histogramas do substrato para cada recurso caso todas as redes virtuais estivessem usando o máximo de recursos contratados. A ideia é garantir que sempre existirão recursos físicos para atender a todas as redes, mesmo que existam picos de demanda simultâneos. Soluções baseadas em migração [37] podem ser lentas, pois dependem da busca de

um novo mapeamento entre a topologia virtual e a física, evitando os nós sobrecarregados, e depois da migração dos nós virtuais para os novos nós físicos. Além de lento, esse processo pode causar perdas de pacote, gerando multas para o provedor de infraestrutura. Esse processo sem o uso de um controle de admissão adequado também pode levar a sobrecarga dos novos nós físicos, causando mais perdas e instabilidades nas redes virtuais. Portanto, é importante estimar o impacto de uma nova rede antes de admiti-la em um nó físico.

O VIPER usa $R_{avg}[\,]$ e $R_l[\,]$ de cada recurso monitorado para estimar os aumentos de demanda, os quais são representados por:

$$\Delta = \sum_{n=1}^{N_H} R_l[n] - R_{avg}[n], \tag{5.10}$$

onde N_H é o número de redes virtuais. A ideia do Δ é estimar o impacto de um aumento de demanda nas redes virtuais sobre o substrato físico. Como base para essa estimativa, supõe-se que a rede pode requisitar recursos até o valor máximo contratado e que o aumento de requisições não altera a forma como os recursos são requisitados. Assim, se a função $f_i(t)$ modelasse o consumo de recursos da rede i ao longo do tempo, então a estimativa de aumento de uso de recursos para essa rede seria dada por $f_i(t) + \Delta_i$, aonde $\Delta_i = R_l[i] - R_{avg}[i]$. Contudo, para o algoritmo proposto, não importa como cada rede aumenta o seu consumo individualmente, mas sim como o uso agregado aumenta quando cada rede virtual está usando o seu máximo de recursos contratados.

Após obter Δ (Eq. 5.10), o controlador atualiza cada histograma do substrato de acordo com o seguinte. Primeiramente, o histograma é deslocado de acordo com Δ, assumindo que $I[i]$ é um intervalo com limite superior $L_s[i]$, e $I[i']$ é um intervalo que contém o valor $L_s[i] + \Delta$, i.e.

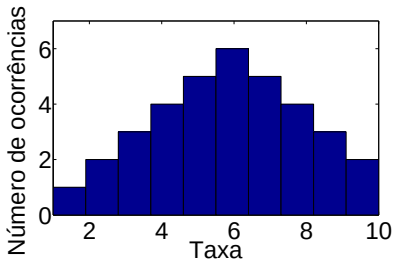
$$L_s[i' - 1] < L_s[i] + \Delta \leq L_s[i']. \tag{5.11}$$

Além disso, assume-se que $H_{sub}(I[i])$ é o número de medidas no intervalo $I[i]$ no histograma do substrato e que N_{int} é o número de intervalos no histograma do substrato. Assim, o substrato do histograma deslocado, H_{sft}, é calculado como $H_{sft}(I[i']) = H_{sub}(I[i])$. O intervalo $I[N_{int}]$ do H_{sft} passa a ser definido como

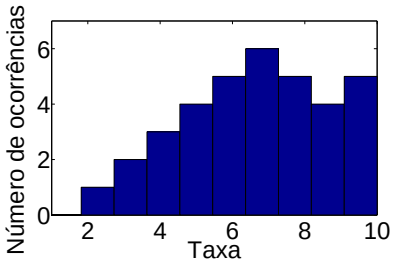
$$L_s[N_{int} - 1] < I[N_{int}] < \infty, \tag{5.12}$$

para manter um número fixo de intervalos. A Figura 5.6 mostra um exemplo de deslocamento de histograma quando $N_{int} = 10$ e $\Delta = 1$.

Depois disso, o controlador calcula uma função de massa de probabilidade para



(a) Histograma do substrato (H_{sub}).



(b) Histograma deslocado (H_{sft}).

Figura 5.6: Exemplo de deslocamento de histograma, quando $N_{int} = 10$ e $\Delta = 1$.

o histograma do substrato deslocado (PMF_{hist}), assumindo que os valores de x são dados por $L_s[\,]$, como mostrado na Figura 5.7.

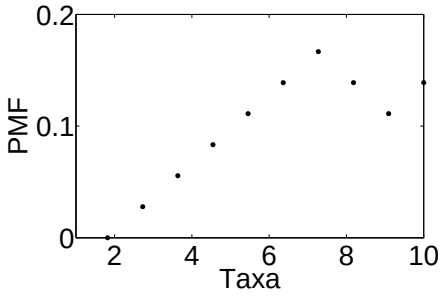


Figura 5.7: Função de massa de probabilidade do histograma do substrato deslocado mostrado na Figura 5.6(b).

Passo 3

O controlador estima a demanda da nova rede virtual baseado em uma distribuição pré-definida. Por exemplo, um controlador de admissão otimista assume uma distribuição de Poisson, enquanto que um controlador de admissão pessimista assume uma distribuição concentrada em taxas de pico.

A distribuição assumida para a nova rede virtual (PMF_{new}) é representada de acordo com os intervalos $I[n]$ do histograma do substrato, assumindo que os valores de x são dados por $L_s[\,]$. Assim, existe uma correspondência direta entre os intervalos de taxa para o histograma do substrato e para a distribuição de probabilidade estimada para a nova rede.

Passo 4

Por fim, a PMF_{hist} é usada pra calcular a probabilidade de bloqueio de recurso, de acordo com a distribuição assumida para a nova rede virtual, PMF_{new}. Considerando que

$$C_t = \{(x_1, x_2) \mid x_1 + x_2 \geq C\}, \quad x_1, x_2 \in [L_s[1], L_s[N_{int}]], \tag{5.13}$$

é o conjunto de tuplas (x_1, x_2) cuja soma $x_1 + x_2$ excede a capacidade do recurso C, então, o controlador calcula a probabilidade de bloqueio de recurso como

$$P_B = \sum_{\forall (x_1,x_2) \in C_t} PDF_{hist}(x_1) \cdot PDF_{new}(x_2). \tag{5.14}$$

Com isso, estima-se os casos em que a soma dos recursos utilizados no histograma do substrato deslocado com os recursos utilizados pela nova rede excede a capacidade máquina física.

O novo roteador virtual é aceito se existem recursos suficientes para os requisitos estáticos da nova rede e se as condições

$$\sum_{n=1}^{N} R_c[n] < C \qquad \text{e} \tag{5.15}$$

$$P_B < P_L \tag{5.16}$$

forem verdadeiras para todos os histogramas de todos os recursos dinâmicos, aonde P_L é a probabilidade de bloqueio de recursos aceita pelo administrador da infraestrutura. Um P_L pequeno garante uma baixa probabilidade de bloqueio de recursos. Contudo, um P_L pequeno também implica em uma baixa eficiência no uso dos recursos físicos, o que reduz os lucros do provedor de infraestrutura.

Capítulo 6

Análise do VIPER*

Esse capítulo mostra os resultados obtidos com a análise dos gerenciadores e do controlador propostos no VIPER. Foi desenvolvido um protótipo para os dois gerenciadores em C++ e Python. Além disso, foi desenvolvido um simulador para avaliar o controle de admissão de acordo com diferentes padrões de tráfego.

Primeiramente, é apresentada a análise realizada para calcular os parâmetros do monitor de uso de processamento por rede virtual. Em seguida, é mostrada uma análise da utilização do gerenciador com alocação de recursos ociosos (GRO) proposto para prover isolamento e segurança em ambientes virtuais. Os parâmetros do gerenciador com alocação exata de recursos (GER) também são analisados. Por fim, ambos os gerenciadores propostos, GRO e GER, são avaliados em termos de eficiência na utilização de recursos e acurácia para atender as especificações de acordo de nível de serviço (SLA). Os gerenciadores propostos são comparados ao controle de banda oferecido pela ferramenta *Traffic Control* do Linux.

6.1 Monitoramento do uso da CPU compartilhada no Domínio 0

Para determinar o custo das operações de encaminhamento de pacotes ($C_j,\ j \in [1,6]$), mediu-se o gasto de CPU no Dom0 para cada uma das possíveis operações:

- $VM - VM$ - encaminhamento de pacotes de máquina virtual para máquina virtual (C_1 e C_4));
- $VM - EXT$ - encaminhamento de pacotes de máquina virtual para máquina externa (C_2);
- $EXT - VM$ - encaminhamento de pacotes de máquina externa para máquina virtual (C_3);

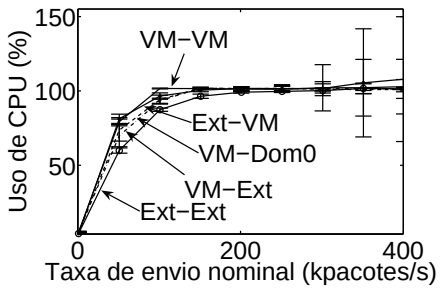
*Esse capítulo é baseado no texto publicado em [24, 25, 40].

- $VM - Dom0$ - encaminhamento de pacotes de máquina virtual para o Dom0 (C_5);
- $EXT - EXT$ - encaminhamento de pacotes entre máquinas externas (C_6).

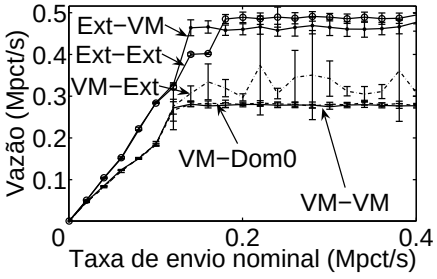
Nessa medida, variou-se a carga oferecida para ser encaminhada e mediu-se o gasto de CPU no Domínio 0. A Figura 6.1(a) mostra o gasto de CPU no Dom0 de acordo com a carga de pacotes nominal gerada na origem, em uma máquina física com processador Intel Core 2 Quad, 4 GBytes de memória RAM, 5 interfaces de rede gigabit, hospedando 2 máquinas virtuais e com kernel Linux 2-6-32. O tráfego é gerado através do módulo do kernel pktgen, sendo caracterizado por segmentos UDP com 64 Bytes de carga útil. Esses parâmetros de tráfego foram escolhidos para aumentar ao máximo o número de pacotes que a máquina física era capaz de processar em certo intervalo de tempo. O Dom0 está configurado com quatro CPUs lógicas, enquanto que cada DomU possui uma CPU lógica. Com isso, o uso de CPU no Domínio 0 é limitado em 400%, das 4 CPUS, nos gráficos a seguir. Observa-se por esse resultado que o Xen utiliza apenas uma CPU para encaminhar pacotes, de forma que o aumento da carga enviada pela origem não implica em um aumento linear de consumo de CPU no Dom0[1]. Com base nesse gráfico, é possível observar que a origem e o destino de cada pacote, assim como a taxa de chegada de pacotes no Domínio 0, interferem no custo de processamento de cada pacote. Assim, processar pacotes relativos a uma comunicação entre máquinas externas é a operação menos custosa e processar pacotes relativos a uma comunicação entre máquinas virtuais é a operação mais custosa. Isso se explica facilmente, porque o encaminhamento de pacotes entre máquinas externas não faz uso da área de memória compartilhada do Xen (*I/O ring*), que é o principal gargalo para o encaminhamento de pacotes pelas máquinas virtuais no Xen [106].

A Figura 6.1(b) mostra a vazão de pacotes no Dom0 em função da taxa nominal gerada. Observa-se que o Dom0 possui um limite de encaminhamento em torno de 500 kpct/s e, em taxas superiores a esse volume, acontecem descartes de pacote. Isso ocorre porque o Domínio 0 utiliza apenas uma CPU para encaminhar pacotes. Assim, embora existam CPUs ociosas no Domínio 0, isso não aumenta a taxa de encaminhamento de pacotes. No caso das transmissões que se originam em máquinas virtuais, observa-se que não foi possível chegar a taxa de 500 kpct/s. Isso ocorre devido à saturação da CPU de máquina virtual, que não consegue gerar uma taxa de pacotes superior a $\approx$ 300 kpct/s. Com base nesses dados é possível estimar o custo por pacote dependendo da origem e do destino do pacote e da taxa de chegada de pacotes nas interfaces físicas, como mostrado na Figura 6.1(c). Esses valores são

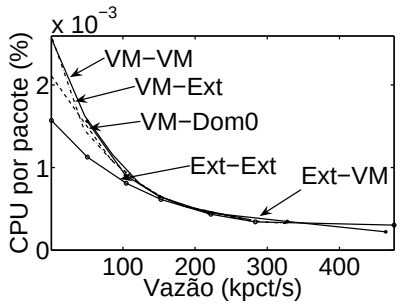
[1]É importante notar, no entanto, que outros processos executados no Dom0 não são impedidos, porque as demais CPUs lógicas estão disponíveis para a execução de outros processos.



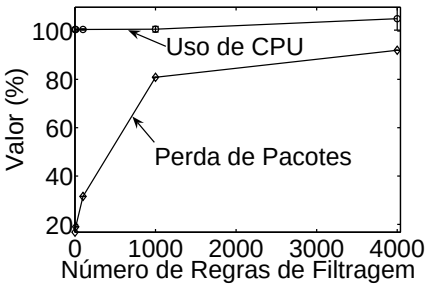
(a) Uso de CPU no Dom0 em função da taxa nominal de envio.



(b) Vazão no Dom0 em função da taxa nominal de envio.



(c) Uso de CPU no Dom0 por pacote em função da vazão.



(d) Impacto do número de regras de filtragem sobre o encaminhamento.

Figura 6.1: Análise dos parâmetros do VIPER para a estimativa do gasto de CPU no Domínio 0 por rede virtual.

utilizados para estimar o uso de CPU nos gerenciadores de recursos compartilhados de acordo com a taxa de entrada de pacotes e com a origem e o destino dos pacotes.

O peso das operações do classificador também é considerado para a estimativa do gasto de CPU no VIPER. Para avaliar o custo de CPU para filtragem e para verificar se a filtragem utiliza a mesma CPU que o encaminhamento de pacotes no Xen, mediu-se a vazão e o gasto de CPU ao se encaminhar um tráfego com carga máxima entre duas máquinas externas roteadas pelo Dom0 de uma terceira máquina física. A Figura 6.1(d) mostra a perda de pacotes e o gasto de CPU em função do número de regras de filtragem no Iptables. Com base nesse gráfico, observa-se que o número de regras de filtragem tem um grande impacto sobre o encaminhamento de pacotes. Observa-se também que a filtragem não utiliza recursos de processamento além da CPU que é utilizada para encaminhar pacotes. Assim, ao invés do uso de CPU no Domínio 0 aumentar com o aumento de regras de filtragem, até que atingisse o patamar de 400%, no qual aconteceriam perdas de pacote, o uso de CPU para o encaminhamento e filtragem fica restrito a 100%. É importante ressaltar que a operação de filtragem é uma operação que deve ser realizada a cada pacote e toda operação a cada pacote onera a capacidade de processamento. Com isso, o aumento do número de regras de filtragem implica uma capacidade menor de encaminhar pacotes no Domínio 0. Assim, se um roteador virtual deseja inserir

regras de filtragem, então ele deve pagar mais caro para compensar a redução da taxa de encaminhamento de pacotes.

Com isso, é possível atribuir a cada rede virtual um custo fixo de CPU para a classificação dos pacotes entre as redes virtuais, além de um custo extra de acordo com o número de regras de filtragem extra que a máquina coloque no Dom0, no caso de utilização de separação de planos.

6.2 Isolamento e segurança com o gerenciador com alocação de recursos ociosos

Essa seção descreve os testes de comprovação de isolamento e de segurança realizados com a proposta de gerenciador com alocação de recursos ociosos, indicado como 'GRO' nos gráficos. Nesses cenários, avalia-se o impacto de ações realizadas por redes virtuais maliciosas ou mal comportadas[2], que, na ausência do gerenciamento proposto, resultam na redução do desempenho no encaminhamento de pacotes das demais redes virtuais. Uma máquina virtual maliciosa tem por objetivo receber mais recursos que as outras máquinas virtuais ou prejudicar o desempenho de uma ou mais máquinas virtuais. É importante ressaltar que a falta de isolamento se constitui em uma enorme vulnerabilidade de segurança da plataforma Xen quando usada em redes virtuais.

Foi desenvolvido um protótipo do VIPER utilizando o GRO para realizar uma prova de conceito sobre como fazer a comunicação segura entre o Dom0 e as máquinas virtuais, medir a efetividade do gerenciador na presença de domínios adversários e verificar a eficiência do gerenciador na divisão dos recursos. O protótipo é implementado em C++ e Python e disponibiliza a separação de planos, a comunicação segura entre domínios e os controles de rede e de CPU no Xen modo *router*. Os planos de dados são criados utilizando-se as diversas tabelas de encaminhamento do *kernel* do Linux. A monitoração dos pacotes e a punição com descarte probabilístico são implementadas com o Iptables [113][3]. Foram escolhidos para a implementação da comunicação segura os protocolos Blowfish [114], por ser considerado um protocolo de criptografia simétrica leve e seguro, e RSA [115], por ser um dos protocolos de criptografia assimétrica mais utilizado. O descarte residual nas interfaces virtuais, descrito na Seção 5.2.2, foi estimado com base na taxa de pacotes que causava danos ao funcionamento do Dom0, chegando-se a uma probabilidade de descarte de $0,0009$.

[2]Por simplicidade, os roteadores maliciosos ou mal comportados são chamados de adversários daqui em diante.

[3]O Iptables é uma ferramenta do Linux que serva para classificar e descartar pacotes. Seu uso mais comum é para a construção de *firewalls*

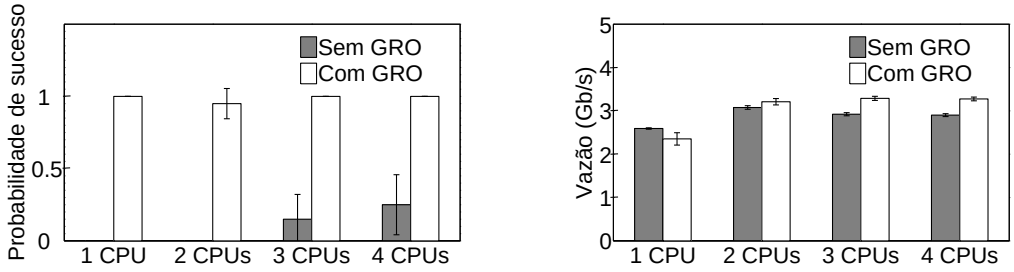
Os testes foram realizados em uma máquina, doravante denominada roteador, com processador Intel Core2 Quad com 4GB de memória RAM, utilizando o Xen 3.4-amd64 no modo *router*. Esse roteador possui cinco interfaces físicas de rede Ethernet com banda de 1Gb/s cada. São instanciadas quatro máquinas virtuais, cada uma delas com uma CPU virtual e 128 MB de memória e executando um sistema operacional Debian com *kernel* Linux 2.6-26-2. As máquinas virtuais são configuradas com cinco interfaces de rede, para um mapeamento simples das interfaces virtuais nas interfaces reais. O número de CPUs virtuais no Dom0 varia de acordo com o teste e não existem restrições de memória para esse domínio. As CPUs físicas estão compartilhadas por todas as CPUs virtuais, cabendo ao hipervisor o escalonamento das CPUs virtuais nas CPUs reais. Os testes usam duas máquinas externas, que geram ou recebem pacotes, cada uma com uma interface de rede de 1Gb/s. Essas máquinas estão ligadas ao roteador e se comunicam apenas por intermédio do roteador.

O primeiro teste realizado objetiva medir a disponibilidade do procedimento de atualização do plano de dados realizado com segurança. A segurança é importante para impedir que DomUs maliciosos se passem por DomUs comuns para inserir informações falsas nos planos de dados dos domínios atacados. Para tornar a separação de planos convencional segura, adicionou-se a ela o protocolo de comunicação segura do VIPER, descrito na Seção 5.1.3, que usa a criptografia assimétrica, baseada no RSA, para estabelecer uma chave de sessão simétrica. Depois de estabelecida a chave de sessão, o domínio virtual autenticado se comunica com o Domínio 0 de forma totalmente segura usando o algoritmo Blowfish de criptografia simétrica. Nesse teste, considera-se que a chave de sessão já foi estabelecida.

O teste de disponibilidade do procedimento de atualização é considerado bem sucedido se o DomU consegue atualizar o seu plano de dados usando o protocolo de comunicação segura independente das demais operações que o Dom0 esteja executando. Em outras palavras, isto significa que nenhuma operação consegue impedir a atualização do plano de dados e, portanto, o mecanismo proposto provê segurança porque é inviável qualquer possibilidade de ataque deste tipo. Assim, comparou-se o mecanismo convencional de separação de planos com a separação de planos no VIPER que se serve do protocolo de comunicação segura e do gerenciador de rede propostos. O teste consiste de um máximo de três tentativas do $DomU_1$ para atualizar o plano de dados, enquanto o $DomU_2$ envia dados em alta taxa para o $DomU_1$, usando a ferramenta Iperf com comunicação via TCP. O cenário procura simular a ação de um roteador virtual, o $DomU_2$, como um domínio adversário tentando impedir que um roteador comum, o $DomU_1$, opere normalmente. No gerenciador com alocação de recursos ociosos (GRO), o $DomU_1$, que tenta fazer a atualização, tem $Perc_1 = 0,5$ de reserva de recursos, enquanto que o domínio adversário, chamado

de $DomU_2$, possui $Perc_2 = 0,3$ de reserva de recursos. As demais máquinas virtuais não possuem recursos reservados. Todos os DomUs possuem peso $W = 500$.

A Figura 6.2(a) mostra a probabilidade de sucesso de atualização do plano de dados, enquanto que a Figura 6.2(b) mostra o volume de dados transmitido entre as máquinas virtuais, ambas assumindo um intervalo de confiança de 95%.



(a) Probabilidade de sucesso na atualização do plano de dados.

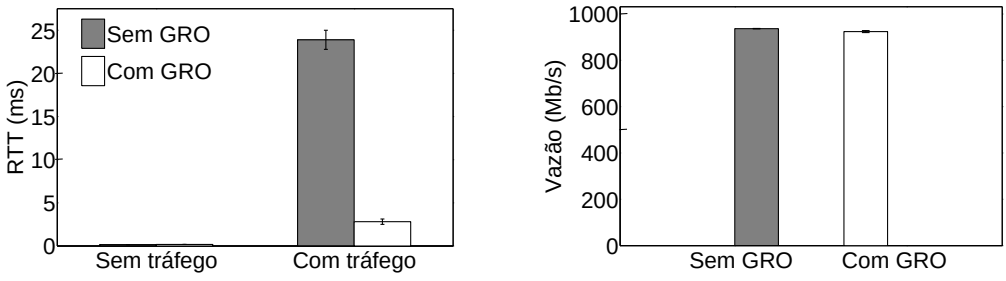
(b) Volume de dados transmitidos na conexão entre DomUs.

Figura 6.2: Disponibilidade para o $DomU_1$ da atualização do plano de dados com segurança quando existe um tráfego entre o $DomU_2$ e o $DomU_1$.

A Figura 6.2(a) indica que o ataque para impedir a atualização do plano de dados é efetivo quando não se usa o GRO. Sem o uso do GRO, o ataque continua efetivo mesmo quando se disponibiliza um maior número de CPUs, procurando oferecer ao Dom0 maior capacidade de processamento para tratar os pacotes trocados entre as máquinas virtuais e os cálculos criptográficos. O teste sinaliza que um domínio adversário consegue atacar outro domínio qualquer simplesmente fazendo transferências de dados para este domínio atacado, pois ele exaure os recursos para encaminhamento de pacotes do Dom0. Por outro lado, o gerenciador com alocação de recursos ociosos (GRO) se mostra bem eficaz. Os resultados com o GRO mostrados na Figura 6.2(a) indicam um aumento de até 100% na probabilidade de sucesso de atualização do plano de dados. O GRO é eficaz porque limita o tráfego de ataque proveniente do $DomU_2$ quando este usa excessivamente a CPU para enviar dados em alta taxa e, ao mesmo tempo, reserva a CPU necessária para o envio das mensagens de atualização e os cálculos criptográficos no $DomU_1$. A Figura 6.2(b) mostra que o GRO gera uma punição no tráfego do $DomU_2$ para o $DomU_1$, para garantir que os recursos de CPU não sejam exauridos, e por isto a vazão com o GRO com uma CPU é menor que a vazão da separação de planos convencional. Ao se aumentar o número de CPUs, essa punição é naturalmente relaxada. Com isso, o volume de tráfego com o GRO passa a ser maior do que o obtido com a separação de planos convencional, sem o GRO. Nesta condição, devido à CPU do $DomU_1$ ser reservada, garantindo que tanto as atualizações de plano de dados quanto os ACKs da conexão TCP iniciada pelo $DomU_2$ são entregues, a perda de ACKs quando não se utiliza o GRO causa um impacto maior no tráfego do que as limitações de envio

impostas pelo controle proposto ao $DomU_2$ devido ao consumo de CPU. Assim, a utilização do gerenciador de recursos do VIPER garante uma separação de planos segura e sempre disponível, ao mesmo tempo em que assegura uma conexão com alto desempenho entre as máquinas virtuais, devido à arquitetura proposta com o GRO. De fato, o teste mostra que a separação de planos com segurança só pode ser considerada disponível quando existe algum tipo de mecanismo limitador do uso de CPU do Dom0 pelos roteadores virtuais.

O segundo teste realizado avalia o atraso inserido na transmissão de dados pelo GRO, quando comparado à separação de planos convencional, assumindo um intervalo de confiança de 95% nos resultados. O objetivo é verificar o quanto a sobrecarga do GRO afeta o atraso de acordo com a carga de trabalho do Dom0. Uma vez que não está se medindo a justiça na divisão dos recursos, foi criada apenas uma rede virtual com 100% dos recursos do Dom0. O teste consiste de dois experimentos que medem o *Round Trip Time* (RTT) na comunicação entre duas máquinas externas. No primeiro experimento, não existe tráfego de fundo, enquanto que, no segundo experimento, um tráfego de fundo TCP foi gerado com a ferramenta Iperf entre as duas máquinas externas. Os resultados de ambos os experimentos são mostrados na Figura 6.3(a). Observa-se que, sem tráfego de fundo, a transmissão de dados apresenta um valor de RTT muito baixo para as duas configurações. No entanto, quando existe um tráfego de fundo, a CPU do Dom0 é sobrecarregada, aumentando o tempo de resposta do sistema, o que implica em criação de filas no Dom0 e, consequentemente, um aumento do RTT. Os resultados mostram que o controle de CPU e banda provido pelo GRO impede que a CPU seja sobrecarregada e, com isso, o desempenho com o GRO é bem superior, resultando em uma redução do RTT em mais de oito vezes. É importante observar que, embora o controle do GRO implique em descarte de pacotes, esses descartes não causam um grande impacto sobre o tráfego, como mostra a Figura 6.3(b).



(a) Impacto do VIPER sobre o RTT.

(b) Vazão do tráfego de fundo.

Figura 6.3: Atraso na comunicação entre máquinas externas em função do tráfego encaminhado pelo Dom0, utilizando-se uma CPU virtual no Dom0.

6.3 Análise do gerenciador com alocação exata de recursos

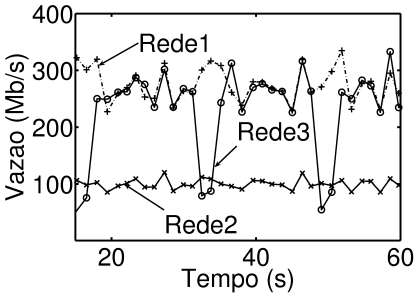
Foi desenvolvido um protótipo do gerenciador com alocação exata de recursos, doravante chamado de GER, em C++ para análise da proposta no Xen-4.0 configurado no modo *router*. O monitoramento dos recursos de CPU, que são estimados com base no volume de pacotes encaminhados, e de rede foi feito utilizando-se a ferramenta Iptables. A mesma ferramenta também foi utilizada para a aplicação das punições. Para fazer o controle dos parâmetros de QoS, utilizou-se a ferramenta *Traffic Control* (TC)[4]. O protótipo foi implementado em uma máquina física com processador Intel Core 2 Quad, 4GBytes de memória RAM, 5 interfaces de rede gigabit, hospedando 3 máquinas virtuais com sistema operacional Debian 2.6.32-5. O Dom0 está configurado com quatro CPUs lógicas, enquanto que cada DomU possui uma CPU lógica. Com base nesse protótipo foram desenvolvidos testes utilizando-se três máquinas externas conectadas à máquina física que executa o Xen com o VIPER configurado com o GER. O tráfego é gerado através do módulo do kernel do Linux 'pktgen', sendo caracterizado por pacotes UDP com 1472 B de carga útil, para garantir o tamanho máximo do quadro Ethernet. Nesse teste, não se utilizou pacotes pequenos, porque optou-se por avaliar o controle da banda. Por essa mesma razão, optou-se por utilizar o UDP, para que a fonte possa controlar a taxa de envio independente de algoritmos de controle de fluxo. 5

Primeiramente, analisou-se o funcionamento do GER controlando três redes virtuais com configurações diferentes, mas demandas iguais a $D[i] = 300\ Mb/s,\ 0 < i < 4$, e sem requisitos de QoS no provedor de infraestrutura. Lembrando que a demanda correspondente à reserva a curto prazo R_c deve ser sempre atendida e também que a demanda média não pode ultrapassar a taxa média da reserva a longo prazo, R_l, para estar integralmente conforme ao contrato de nível de serviço (SLA).

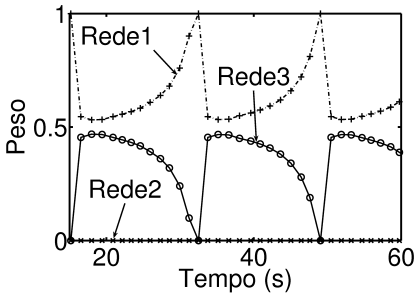
A Rede 1 possui uma reserva a curto prazo $R_c[1] = 50\ Mb/s$ e uma reserva a longo prazo $R_l[1] = 350\ Mb/s$, de forma que sua demanda deve ser provida integralmente, pois está de acordo com o SLA uma vez que $(D < R_l)$. A Rede 2 possui uma reserva a curto prazo igual a reserva a longo prazo no valor de $R_c[2] = R_l[2] = 100\ Mb/s$, apresentando uma demanda superior ao contratado e especificado no SLA, pois $(D > R_l)$. Já a Rede 3 simula uma rede com grande volume de tráfego, mas sem nenhuma prioridade ou requisito de atraso, de forma que a $R_c[3] = 0$ e $R_l[3] = 250\ Mb/s$. Portanto, a Rede 3 também apresenta demanda superior ao seu SLA. Nenhuma das redes virtuais possui reserva exclusiva e os recursos de

[4]O TC é a ferramenta mais utilizada no Linux para fazer o controle de tráfego. O TC disponibiliza um conjunto de políticas e conformadores para limitar o tráfego de entrada e controlar o tráfego de saída de uma máquina física.

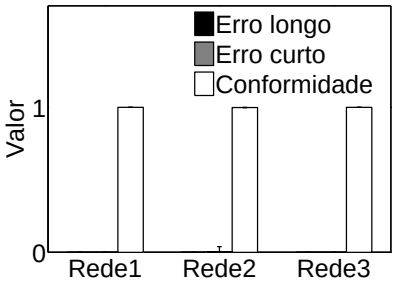
CPU e memória foram divididos igualmente entre as três redes. Os tráfegos da Rede 1 e da Rede 2 são transmitidos de uma máquina externa para outra através de roteamento pelo Dom0, pois se assume que essas redes utilizam a separação de planos. Já o tráfego da Rede 3 vem do roteador virtual para a máquina externa, simulando um tráfego gerado ou encaminhado pelo roteador virtual. O enlace de saída é compartilhado pelas três redes e é o objeto dessa análise. Neste e nos demais experimentos, o GER está configurado com o intervalo curto $I_c = 1\ s$ e o intervalo longo $I_l = 15s$, os quais foram obtidos através de testes com o gerenciador para se obter o melhor desempenho. Assim, nos experimentos realizados o controle de conformidade do SLA é realizado em janelas saltitantes de duração igual ao intervalo longo de $I_l = 15s$. A duração total do teste é de 200 s.



(a) Vazão no Dom0 por rede virtual.



(b) Variação do peso por rede virtual.



(c) Erros e conformidade por rede virtual.

Figura 6.4: Operação do GER com diferentes padrões de rede virtual, assumindo demanda de 300Mb/s para cada rede.

A vazão (V_i) obtida com as três redes ao longo do tempo está apresentada na Figura 6.4(a). Uma vez que a Rede 2 possui apenas a reserva a curto prazo, o GER limitou o seu uso de forma constante em 100 Mb/s. Já a Rede 1 teve a sua demanda atendida em todos os momentos, enquanto que a Rede 3 teve sua demanda bloqueada ao atingir o valor total de dados permitido pela reserva longa em cada intervalo longo. O uso da reserva longa pelas redes é regido de acordo com o peso adaptativo da rede, como descrito na Seção 5.2.2. A Figura 6.4(b) mostra a evolução do valor do peso. O peso da Rede 2 é sempre zero, porque essa rede não tem direito

a usar nada além da reserva a curto prazo. O peso da Rede 1 aumenta enquanto que o peso da Rede 3 diminui, porque a Rede 3 possui uma demanda média superior a sua reserva a longo prazo e a Rede 1 possui uma demanda média inferior a sua reserva a longo prazo. De fato, o peso se adapta proporcionalmente entre as redes de acordo com a fatia da reserva longa que cada rede ainda pode gastar. Com isso, o peso das redes que possuem uma demanda inferior à sua reserva a longo prazo tende a aumentar enquanto que o peso das redes que possuem uma demanda superior à sua reserva a longo prazo tende a diminuir.

São parâmetros definidos para verificar o atendimento das SLAs para cada uma das redes virtuais:

- Erro a curto prazo da rede i, $E_c[i]$;
- Erro a longo prazo da rede i, $E_l[i]$;
- Conformidade com relação ao SLA da rede i, $C_f[i]$.

O erro a curto prazo da rede i, $E_c[i]$, é calculado a cada intervalo curto, I_c, e demonstra o quanto a reserva a curto prazo não foi atendida. Analogamente, o erro a longo prazo da rede i, $E_l[i]$, é calculado no final de cada intervalo longo, I_l, e demonstra o quanto a reserva longa não foi atendida. Assim,

$$E_c[i] = max(0, 1-V[i]/min(R_c[i], D[i])) \quad e \quad E_l[i] = max(0, 1-V[i]/min(R_l[i], D[i])). \tag{6.1}$$

A conformidade com relação ao SLA da rede i, $C_f[i]$, é calculada também ao final de I_l e demonstra um valor médio de atendimento ao SLA. Assim,

$$C_f[i] = 1 - \frac{media(E_c[i])}{2} - \frac{E_l[i]}{2}. \tag{6.2}$$

A Figura 6.4(c) mostra que todas as redes praticamente não apresentaram erros a curto ou a longo prazo e, portanto, tiveram sua conformidade atendida ao longo de todos os intervalos longos do teste, pois o GER oferece um controle eficiente dos recursos compartilhados na máquina física. A Rede 2 apresenta um pequeno erro com relação ao erro curto apenas devido ao primeiro intervalo longo, no qual o GER ainda estava se ajustando à demanda.

O segundo teste realizado compara o GER com outras ferramentas da literatura. O objetivo desse teste é avaliar qual mecanismo que atende aos SLAs especificados da forma mais eficiente, ou seja, consumindo menos recursos do roteador. Para tanto, montou-se um cenário no qual existem duas redes virtuais, a Rede 1 utilizando separação de planos e a Rede 2 utilizando roteamento pela máquina virtual. As redes possuem $R_c[1] = R_c[2] = 100\ Mb/s$, $R_l[1] = 600\ Mb/s$ e $R_l[2] = 200\ Mb/s$.

Ambas as redes possuem uma demanda superior aos parâmetros de suas SLAs, $D_1 = D_2 = 1\ Gb/s$, em tráfegos entre duas máquinas externas. Uma vez que a máquina externa receptora é comum às duas redes, então existe o compartilhamento do enlace de saída. O GER foi comparado ao GRO, pois esse também realiza o controle de recursos entre redes virtuais, embora o GRO não possua a opção pela especificação de uma reserva longa ou o controle adaptativo dos recursos utilizados. Além disso, para comparar as propostas de controladores desta tese, o GRO e o GER, com outras propostas, utilizou-se o controlador de tráfego (TC) com a disciplina de filas *Hierarchical Token Bucket* (HTB). Procurando simular comportamentos do HTB próximos ao do GER e do GRO, foram realizados três perfis diferentes de controle de tráfego utilizando-se a ferramenta TC com o HTB. O perfil TC1 possui reserva mínima $R_m[i] = R_l[i]$ e permite que ambos os tráfegos cheguem a 1 Gb/s caso haja demanda e disponibilidade de recursos. O perfil TC2 utiliza $R_m[i] = R_c[i]$, mas limita o uso do enlace à taxa média da reserva a longo prazo da rede, $R_l[i]$, sendo o mais próximo do comportamento do GER. O perfil TC3 utiliza $R_m[i] = R_c[i]$ e permite que os recursos sejam utilizados até o limite do enlace caso haja demanda e disponibilidade, sendo o mais próximo do comportamento do GRO. Dessa forma, é possível avaliar o desempenho do HTB com diversos tipos de configurações e compará-lo com o controle do GER e do GRO de forma justa. O TC, embora seja uma ferramenta precisa para o controle de tráfego, também não possui controle adaptativo, além de não fazer o controle da divisão dos recursos compartilhados de memória e CPU como o fazem o GER e o GRO. Dessa forma, para tornar a comparação do controle de recursos da rede mais justa, tanto no GER quanto no GRO não foram feitas restrições quanto ao uso de CPU e memória para nenhuma das redes. Por fim, também se analisou o resultado quando nenhum controle era aplicado às redes virtuais ($S/$).

A Figura 6.5 apresenta os resultados do segundo teste, mostrando a conformidade e a ocupação do enlace de saída. A ocupação é calculada como a razão entre o volume de dados utilizado em cada perfil e o volume de dados máximo transmitido no enlace de saída. Apenas o GER e os perfis TC1 e TC2 conseguiram prover conformidade máxima a ambas as redes. Com os demais perfis, a Rede 2 acabou sendo privilegiada devido à baixa capacidade de diferenciar o uso dos recursos além da reserva mínima de cada rede. Os perfis TC1 e TC2 apresentaram bons resultados de conformidade, porque são adaptados a diferenciar o tráfego com base na taxa de uso a longo prazo. O TC1 garante a reserva a longo prazo como reserva mínima para cada rede e o TC2 limita o uso de recursos de cada rede no valor da reserva a longo prazo. Contudo, apenas o GER e o TC2 apresentaram baixa ocupação do enlace, o que é uma característica positiva importante para as redes virtualizadas. O TC1, o TC3 e o GRO permitem que todos os recursos disponíveis sejam usados

sempre que houver demanda, aumentando a ocupação do meio.

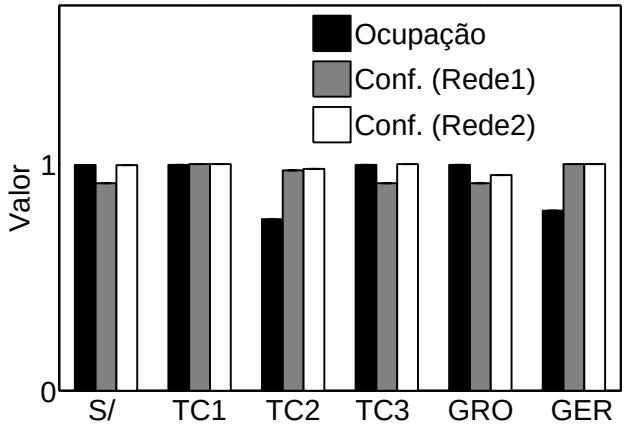


Figura 6.5: Conformidade assumindo duas redes com demanda máxima.

O terceiro teste realizado utiliza os mesmos perfis com as mesmas configurações para as redes, mudando apenas a demanda. Nesse caso, $D_1 = 1\ Gb/s$ durante todo o teste, enquanto que $D_2 = 0$ durante os primeiros dois terços de I_l e $D_2 = 1\ Gb/s$ durante 3 s no último terço de I_l. Este perfil de tráfego da Rede 2 simula o comportamento de uma rede com tráfego em rajadas, no qual a rajada equivale ao total de recursos contratados a longo prazo para aquela rede. É importante ressaltar que, nesse cenário, o erro a longo prazo da Rede 2 não pode ser nulo, porque o atendimento total da demanda necessitaria do uso global da banda, mas a Rede 1 possui uma reserva a curto prazo para ser atendida. Assim, uma perda mínima de 100 Mb/s era esperada para o erro longo da Rede 2. Os resultados na Figura 6.6 mostram que o erro a longo prazo do GER foi o menor de todos, apresentando um ganho de mais de cinco vezes quando comparado ao perfil TC2, que apresentou um ganho semelhante ao do GER no teste anterior. Além disso, o GER apresentou a segunda menor ocupação, mostrando que o GER é positivo tanto para o operador da rede, que deseja o cumprimento do contrato, quanto para o provedor de infraestrutura, que deseja um baixo uso dos recursos físicos para que mais redes possam ser alocadas. Portanto, o GER se mostrou a ferramenta mais adequada para o controle de tráfego das redes virtuais por apresentar um controlador acurado devido à sua capacidade de adaptação, além de possuir um conjunto de configurações de SLAs mais amplo que todas as outras ferramentas analisadas.

Por fim, foi avaliado o módulo de qualidade de serviço (QoS) do VIPER. O QoS é importante porque se assume que a Internet do Futuro deve dar suporte a soluções de rede que a Internet atual não é capaz de oferecer e o provimento de qualidade de serviço é um dos principais desafios da Internet atual. Assim, as infraestruturas virtualizadas precisam dar suporte à QoS para que soluções para esse problema possam ser aplicadas. A Figura 6.7 mostra o impacto da utilização

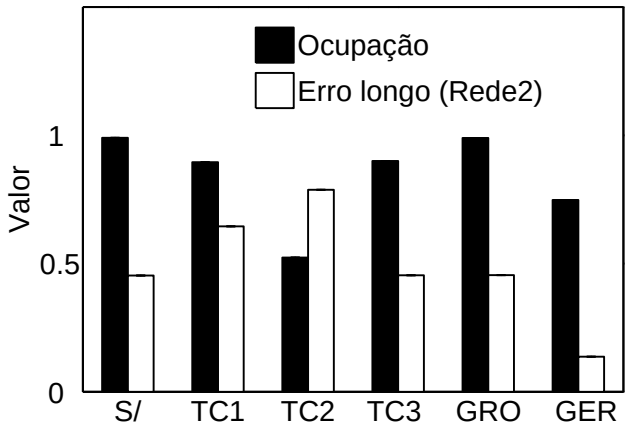


Figura 6.6: Erro a longo prazo da Rede 2 assumindo demanda em rajada.

das premissas de QoS pelo provedor de redes virtuais. Nesse cenário, composto por duas redes virtuais, assume-se que a Rede 1 deseja garantir um atraso máximo para todo o seu tráfego. Assume-se também que a Rede 1 tenha como parâmetros para o enlace de saída $R_c[1] = 50\ Mb/s$ e $R_l[1] = 400\ Mb/s$, enquanto que a Rede 2 possui $R_c[2] = 100\ Mb/s$ e $R_l[2] = 600\ Mb/s$. A CPU e a memória foram divididas igualmente entre as redes. A Rede 1 possui uma demanda $D_1 = 50\ Mb/s$ e a Rede 2 possui $D_2 = 1\ Gb/s$. Foi escolhido um valor alto para D_2, porque um volume alto de tráfego dificulta o provimento de QoS para a Rede 1. O tráfego da Rede 2 veicula entre duas máquinas externas com roteamento pelo Dom0, enquanto que o tráfego da Rede 1, que deve ser priorizado, é roteado pela máquina virtual também entre duas máquinas externas. O enlace de saída é compartilhado pelos tráfegos das Redes 1 e 2. A Figura 6.7 mostra os resultados para o *Round Trip Time* (RTT) com e sem o uso do GER, comparando um cenário aonde os dois tráfegos possuem a mesma prioridade (S/ prio) a outro cenário onde os pacotes da Rede 1 tem prioridade máxima (Prio). O uso do módulo de QoS do provedor mesmo sem a utilização do controle do GER garantiu uma redução de mais de 10 vezes no atraso do tráfego privilegiado. Ao se utilizar o VIPER com módulo de QoS e os demais módulos, o GER garante que a Rede 2 não exceda o uso de rede ou CPU, reduzindo o volume de dados processados e, com isso, reduzindo ainda mais o atraso de transmissão. O ganho com relação à redução do atraso ao se utilizar o VIPER com todos os módulos é de mais de 18 vezes quando comparado com o cenário sem GER e sem prioridade, de mais de 1.8 vezes quando comparado ao cenário sem GER com prioridade e de mais de 2.3 vezes quando comparado ao uso do GER sem prioridade. Com isso, fica clara a necessidade de se prover QoS pelo provedor de infraestrutura e os resultados comprovam a enorme eficácia do VIPER como ferramenta para o provimento de QoS.

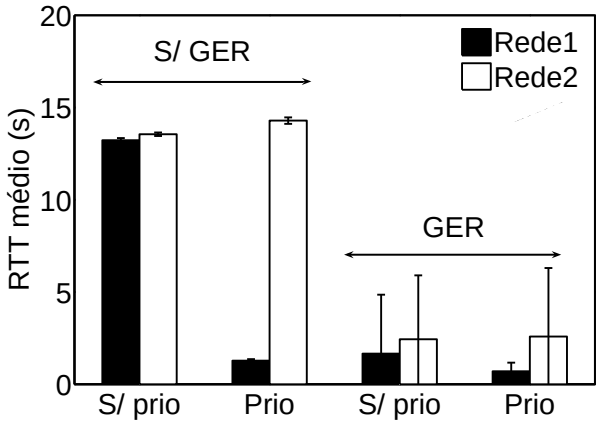
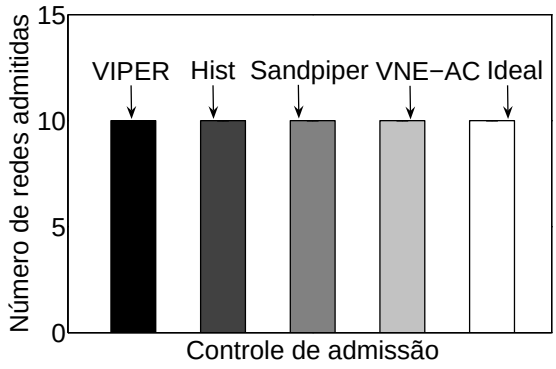


Figura 6.7: RTT de acordo com os parâmetros de QoS usados.

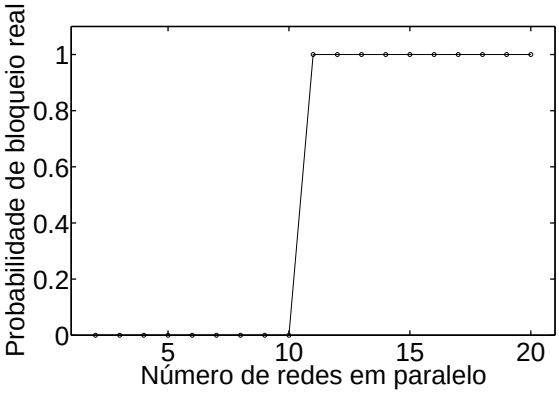
6.4 Resultados da simulação do controle de admissão

O controle de admissão de redes virtuais do VIPER foi implementado em C++ e testado com diferentes padrões de tráfego, assumindo o gerenciador com alocação exata de recursos (GER) descrito na Seção 5.2.2. Além disso, o mecanismo de controle de admissão do Sandpiper [37] e do VNE-AC [43], explicados na Seção 4.11, também foram implementados para comparação com a proposta. A taxa de pico do Sandpiper foi escolhida como $p_k = 95\%$, seguindo a recomendação dos autores da proposta. O limiar superior do VNE-AC foi definido como o valor médio da reserva de volume a longo prazo. O VIPER também foi comparado com uma variação da proposta que não usa o deslocamento Δ (Eq. 5.10), indicado como 'Hist' nos gráficos, para se verificar a eficácia deste procedimento.

A avaliação da eficiência da cada mecanismo é feita medindo o número de redes virtuais que cada mecanismo admite para serem hospedadas no roteador físico, assumindo-se que todas as redes apresentam os mesmos parâmetros de reserva de recursos e que o número de intervalos do histograma do VIPER é $N_{int} = 30$. Também foi verificado o número ideal de redes em cada cenário, o qual é definido como o número de redes para o qual a probabilidade de bloqueio de recursos é menor do que $P_L = 0.05$. A medida do número ideal de redes é feita através da geração de tráfegos para cada rede virtual e a verificação de probabilidade real de bloqueio de recurso ao longo do tempo. Foram realizadas 30 rodadas para cada experimento, simulando-se a vazão em um enlace de saída, medida em pacotes/s. Os resultados são apresentados com o desvio padrão correspondente.



(a) Admissão de redes virtuais com uma demanda média constante.



(b) Probabilidade de bloqueio de recurso de acordo com o número de redes em paralelo.

Figura 6.8: Controle de admissão, assumindo redes virtuais com tráfego modelado por uma distribuição de Poisson e probabilidade de bloqueio de recursos máxima de 5%.

6.4.1 Impacto do tipo de demanda

No primeiro experimento, avaliaram-se redes virtuais cujas demandas são modeladas por processos de Poisson. A demanda de todas as redes virtuais é ≈ 100 Mb/s, a qual também é taxa média da reserva de volume a longo prazo, R_l. A Figura 6.8(a) mostra que o VIPER, assim como as outras propostas, permitiu tantas redes quanto a estimativa ideal. Assim, em redes com pouca variação da demanda, todos os mecanismos são capazes de fazer corretamente o controle de admissão. A Figura 6.8(b) mostra que o controle de admissão nesse cenário ocorre de forma abrupta, pois com 10 redes não existe probabilidade de bloqueio de recursos e com 11 redes a probabilidade passa a 100%.

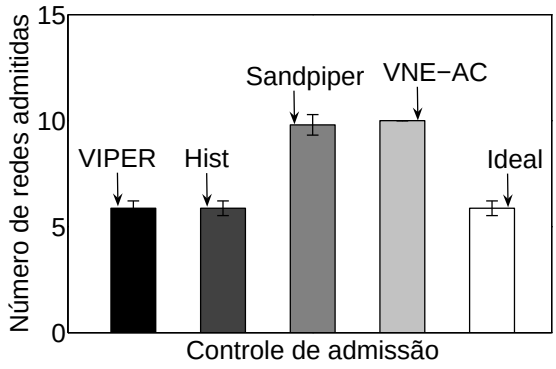
Para avaliar o funcionamento das propostas em ambientes com uma maior variabilidade, também se simulou redes com perfis de tráfego em *on-off*, cujos resultados estão na Figura 6.9. Novamente, todas as redes possuem o mesmo perfil de demanda. A nova rede é estimada pelo VIPER como uma distribuição de Poisson, embora possua o mesmo comportamento que as outras redes. O tráfego *on-off* é

gerado com base em uma distribuição exponencial com $\mu = 1/3$. Cada valor gerado com a distribuição exponencial indica um período de tempo durante o qual o tráfego pode estar ligado ou desligado. O tráfego ligado é modelado com uma distribuição de Poisson com $\lambda \approx 200$ Mb/s e o tráfego desligado apresenta uma demanda sempre igual a zero. A Figura 6.9(a) mostra que o Sandpiper e o VNE-AC admitem mais redes virtuais, até 10 redes, no mesmo roteador físico. No entanto, isto não significa que o controle de admissão destes dois mecanismos é melhor, porque para este número de redes virtuais a Figura 6.9(b) mostra que a probabilidade de bloqueio é de $\approx 50\%$. O Sandpiper superestimou os recursos ociosos devido à natureza 'on-off' do tráfego e permitiu a admissão de até ≈ 10 redes. O VNE-AC apresentou um comportamento semelhante ao do Sandpiper, mas por razões diferentes. O VNE-AC trata exatamente da mesma forma o cenário das Figuras 6.8 e 6.9, pois ele não se baseia em dados reais para fazer a sua previsão de demanda. Assim, redes com perfis diferentes são tratadas como iguais, o que gera erros. O VIPER admitiu o número ideal de redes, garantindo uma baixa probabilidade de bloqueio de recursos e um uso eficiente dos recursos físicos. Esses resultados se devem, principalmente, ao algoritmo de controle de admissão proposto, que estima a probabilidade de bloqueio com base no padrão de uso real do substrato físico.

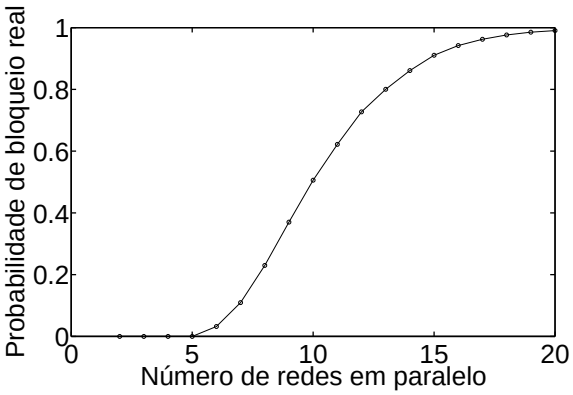
6.4.2 Impacto do Δ

O segundo experimento avalia redes que mudam o seu padrão de demanda ao longo tempo. Com, isso, é possível avaliar o impacto do uso do Δ no VIPER. Nesse cenário, considera-se que a vazão das redes virtuais, modeladas como processos de Poisson, aumenta com o tempo, até que elas atingem o limiar da reserva de volume a longo prazo. Assume-se que o gerenciador de recursos garante que as redes virtuais não excedem a sua reserva. A requisição de admissão da nova rede virtual é enviada quando a taxa da demanda corresponde à metade do valor médio da reserva de volume a longo prazo, R_l. A Figura 6.10(a) mostra que o 'Hist' e o 'Sandpiper' admitem todas as 20 redes virtuais que foram analisadas, pois esses mecanismos não consideram que a demanda da rede pode aumentar com o tempo. Assim, a probabilidade de bloqueio de tráfego é de 100% nesses dois controles de admissão quando todas as demandas crescem até o limiar da reserva de volume a longo prazo, como mostrado na Figura 6.10(b). Ambos o VIPER e o VNE-AC consideraram a variação na demanda, atingindo resultados próximos ao ideal.

Com base nesses resultados, o controlador de admissão do VIPER foi o único que se mostrou eficiente em todos os cenários, garantindo a admissão do maior número possível de redes virtuais, sem violar o limiar estabelecido pelo administrador da infraestrutura para a probabilidade de bloqueio de recursos. O Sandpiper e o VNE-



(a) Admissão de redes virtuais com perfil *on-off* de tráfego.



(b) Probabilidade de bloqueio de recurso de acordo com o número de redes em paralelo.

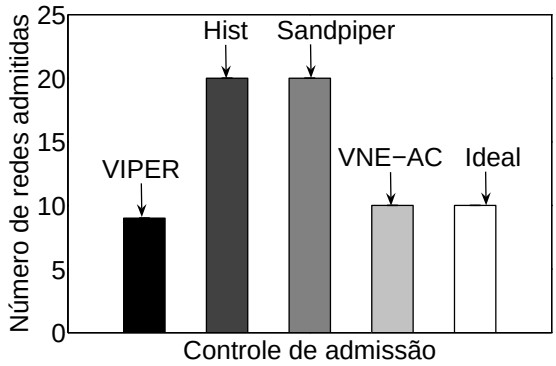
Figura 6.9: Controle de admissão, assumindo redes virtuais com tráfego *on-off* e uma probabilidade de bloqueio de recursos máxima de 5%.

AC se mostraram ineficientes quando a demanda da rede tem uma alta variabilidade, permitindo a admissão de um número de redes virtuais cuja demanda por recurso excede a capacidade física.

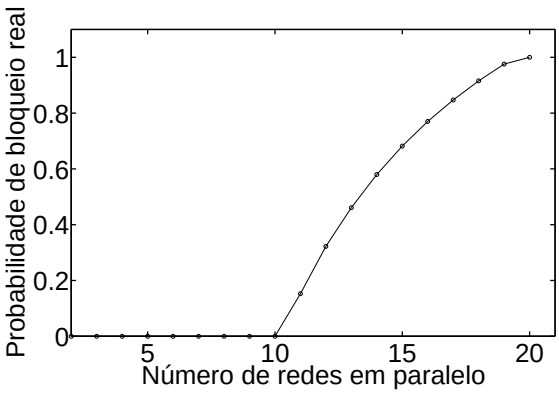
6.5 Discussão

A análise realizada mostra que os algoritmos de controle do VIPER são eficientes e garantem o provimento das SLAs para as redes virtuais. O controle dos recursos compartilhados nas operações de rede não se restringe apenas a banda, como na maioria das ferramentas de controle de rede, mas também inclui o processamento e o uso de memória, garantindo a construção de ambientes seguros e isolados.

O controle de admissão proposto foi o único entre os analisados que não sobrecarregou o *hardware* com redes virtuais, possibilitando o provimento correto dos SLAs para cada uma das redes. Além disso, o controle proposto também evita a subutilização do *hardware*, garantindo uma alta eficiência no uso dos recursos, o que é importante para o provedor de infraestrutura.



(a) Admissão de redes virtuais com perfil de tráfego crescente ao longo do tempo.



(b) Probabilidade de bloqueio de recurso de acordo com o número de redes em paralelo.

Figura 6.10: Controle de admissão, assumindo redes virtuais com tráfego crescente ao longo do tempo e probabilidade de bloqueio de recursos máxima de 5%.

Dessa forma, os algoritmos de controle propostos se mostram como uma alternativa eficiente e de uso simples para o controle de recursos em plataformas de virtualização. De fato, o sistema proposto não está correlacionado a versões do kernel e também não depende de *hardware* específico, o que simplifica a sua difusão e uso.

Os algoritmos propostos para controle de recursos compartilhados e controle de admissão não são específicos da plataforma Xen. De fato, esses algoritmos são aplicáveis a qualquer plataforma de virtualização de rede, já que em todas as plataformas de virtualização é importante a existência do controle dos recursos compartilhados. O controle de recursos compartilhados realizado pelo gerenciador do VIPER só não é necessário em plataformas que fazem o controle das operações de entrada e saída classificando os pacotes por rede em *hardware* [116]. Esse tipo de abordagem, contudo, depende de *hardware* específico, aumentando os custos e inserindo mais dificuldades na implementação do ambiente de rede virtualizado.

Portar o VIPER para outras plataformas de virtualização de computadores pessoais para a criação de redes virtuais é simples, uma vez que a monitoração e a

aplicação das punições são feitas com aplicações padrões do Linux. No caso de uso de outros sistemas operacionais, essas ferramentas precisariam ser substituídas por outras similares. Da mesma forma, portar o VIPER para plataformas como o OpenFlow também não apresenta um nível de dificuldade muito elevado. Nesse caso, ao invés de monitorar as interfaces e aplicar punições diretamente, todo o controle da rede deve ser feito por meio das primitivas disponibilizadas pelo FlowVisor [85]. Assim, a observação do número de pacotes encaminhados, da banda utilizada e do tamanho da memória em uso deve ser feita por meio das primitivas de monitoramento das redes OpenFlow, como disponibilizado em aplicações como a "Stats" do Nox [88, 95]. Os parâmetros para restringir a banda ou o uso da memória devem ser calculados segundo o algoritmo proposto e aplicados utilizando a interface do FlowVisor. O controle de admissão também é realizado com base nas primitivas de monitoramento da rede OpenFlow, que permitem obter os mesmos dados que foram descritos para o Xen, garantindo a construção dos histogramas de uso de recursos.

Portanto, o sistema proposto é genérico e eficiente, sendo uma importante ferramenta para a criação de redes virtuais com isolamento, segurança e qualidade de serviço. De fato, o VIPER permite uma definição ampla de SLAs, ao mesmo tempo em que garante o seu provimento, mesmo sob diferentes padrões de demanda. Além disso, os módulos para inserção de qualidade de serviço em nível de operador de redes virtuais e provedor de infraestrutura garantem a construção de uma arquitetura flexível o suficiente para atender demandas de redes virtuais com requisitos de banda e atraso, desde que se use em conjunto com o VIPER mecanismos para criar caminhos com qualidade de serviço fim-a-fim.

Capítulo 7

Conclusões

Os desafios para a Internet do Futuro são enormes, pois os problemas da Internet atual, como segurança, qualidade de serviço, mobilidade, entre outros, devem ser resolvidos. Além disso, as redes desafiadoras, como redes ad hoc, redes tolerantes a atrasos e desconexões etc., devem ser incorporadas aos sistemas de comunicações convencionais. Uma forte corrente de pesquisadores defende que a solução para todos estes problemas passa por um núcleo de rede flexível e robusto. Outros vão ainda além e defendem também a inteligência no núcleo. Esta tese abordou a autoconfiguração de endereços em redes ad hoc e o provimento de isolamento e qualidade de serviço para as redes virtuais.

Na primeira parte dessa tese, foi desenvolvida uma proposta para autoconfiguração de endereços em redes ad hoc chamada de *Filter-based Addressing Protocol* (FAP). O FAP traz como inovação o uso de filtros de endereço para representar o conjunto de endereços alocados na rede. Além do emprego de filtros como idéia-chave, o FAP usa o conceito de "assinaturas" para identificar partições na rede. As assinaturas resumem a lista de endereços alocados de uma partição. O uso de filtros e suas assinaturas se revelou muito eficaz na alocação de endereços, na detecção de partições e na garantia de unicidade de endereço na união de partições. As propostas existentes se baseavam em identificadores de partição arbitrários.

O FAP foi comparado com três protocolos da literatura, sendo eles o DAD [47], o DAD-PD [48] e o MANETconf [54]. Todos os três protocolos apresentam problemas para solucionar todas as colisões de endereço na rede, em especial durante processos de união de partições. Outro problema é a alta sobrecarga de controle gerada tanto pelo MANETconf quanto pelo DAD-PD. As propostas para endereçamento em redes ad hoc introduzidas nessa tese poderiam ser usadas para reduzir o impacto desses problemas nos três protocolos. O principal problema observado no DAD foi a ausência de um mecanismo para identificar processos de união de partição. O DAD-PD tem como objetivo estender o DAD solucionando esse problema. Contudo, o DAD-PD também apresentou colisões, além de uma alta sobrecarga de controle e,

em alguns casos, o protocolo chegou a ficar instável. Se os filtros de endereço e suas assinaturas fossem utilizados no DAD-PD ao invés dos identificadores aleatórios, esse protocolo teria uma detecção de partição mais precisa além de poder corrigir falhas causadas por perdas de mensagens. As técnicas propostas pelo FAP também poderiam beneficiar o MANETconf, permitindo a detecção de inconsistências nas listas de endereço alocado. Se, ao invés de utilizar identificadores de partição aleatórios, o MANETconf utilizasse as assinaturas das listas de endereços alocados, esse protocolo poderia detectar uniões de partições mal feitas, assim como a exclusão indevida de nós. Embora essa medida não reduza a alta sobrecarga de controle do MANETconf, ela pelo menos resolveria as colisões de endereço.

Por essas razões, a proposta de endereçamento desenvolvida traz contribuições significativas para a autoconfiguração de redes ad hoc. De fato, por facilitarem a detecção de partições, os filtros e suas assinaturas podem ser usados para simplificar e reduzir sobrecargas em vários mecanismos autônomos das redes ad hoc.

A segunda parte dessa tese abordou as redes virtuais. Foi proposto, desenvolvido e avaliado o desempenho de um mecanismo chamado *Virtual network Isolation, Policy Enforcement, and Resource sharing system* (VIPER), que disponibiliza um conjunto de ferramentas para criar ambientes de redes virtualizados. O VIPER se destaca pelo conjunto de funcionalidades de segurança e qualidade de serviço que oferece e, também, pelo alto desempenho que apresenta. O isolamento, a segurança e o provimento dos acordos de nível de serviço são garantidos pelos algoritmos de gerenciamento de recursos compartilhados e pelo protocolo de comunicação segura. Além disso, um nível adicional de qualidade de serviço é provido com os módulos de QoS do operador e QoS do provedor, os quais garantem uma diferenciação de tráfego por parte de ambos o operador de redes virtuais e o provedor de infraestrutura, em ambientes com ou sem separação de planos. O VIPER também provê um controlador de admissão de redes virtuais, que garante que o *hardware* não seja sobrecarregado e que os SLAs sejam providos.

Uma característica importante dos algoritmos do VIPER é que eles são independentes de plataforma de virtualização de redes. De fato, sempre que existem recursos compartilhados, é necessário algum tipo de controle para especificar quantidades de recursos por fatia de rede. O VIPER provê uma divisão de recursos que leva em consideração o que foi contratado e a demanda de cada rede.

Os dois gerenciadores de recursos compartilhados permitem que o provedor de infraestrutura opte entre disponibilizar todos os seus recursos ociosos para aumentar a satisfação do cliente, que tem acesso a um volume de recursos superior ao contratado, ou garantir o provimento apenas dos recursos contratados com precisão. A segunda abordagem é favorável em cenários em que, embora existam recursos ociosos nas extremidades da rede, a vazão agregada nos enlaces de saída do provedor

de infraestrutura pode ser superior ao volume de recursos disponível, devido ao uso excessivo pelos clientes. Assim, a abordagem do gerenciador com alocação exata de recursos simplifica o gerenciamento dos recursos físicos e o redimensionamento das redes de acordo com o crescimento da demanda dos clientes por contrato.

As duas abordagens para gerenciadores de recursos propostas foram implementadas e testadas em ambientes de redes virtuais com diferentes perfis de uso. Entre as análises realizadas, destaca-se a avaliação do isolamento das redes virtuais utilizando-se o gerenciador com alocação de recursos ociosos em ambientes que comportam redes adversárias que tomam atitudes que podem prejudicar o desempenho das outras redes virtuais. Além disso, também foi analisado o erro no provimento dos SLAs contratados pelos clientes. Essa análise mostrou que o gerenciador com alocação exata de recursos é capaz de controlar os recursos compartilhados com precisão, garantindo uma alta conformidade com os SLAs. A proposta foi comparada com diversos padrões de configuração do *Hierarchical Token Bucket* do TC e se mostrou mais eficiente no controle de tráfego em todos os cenários analisados.

Outro ponto importante observado na análise do VIPER é o custo do sistema proposto no que diz respeito ao consumo de recursos. O VIPER foi implementado no Xen assumindo um monitoramento feito com base em regras de filtragem. A separação de planos, a classificação de pacotes por rede e a punição também utilizam o módulo de classificação de pacotes do Linux, o Iptables. A análise realizada mostra que existe balanceamento entre o número de regras de filtragem e o desempenho no encaminhamento de pacotes. Assim, o monitoramento e a classificação causam um gasto de recursos que pode limitar o uso do VIPER com muitas redes virtuais. Abordagens de implementação alternativas incluem o uso do Click, através da construção de um módulo próprio para monitoração e encaminhamento de pacotes, ou ainda a classificação de pacotes por *hardware*.

O VIPER também introduz um conjunto de parâmetros para modelar os SLAs de uma rede. Dentre esses, destacam-se a reserva de taxa a curto prazo, que especifica uma taxa mínima que deve ser provida para a rede virtual quando existir demanda, e a reserva de volume a longo prazo, que determina o volume de recursos ao qual uma rede virtual tem acesso ao longo de um intervalo longo. O conjunto de características que foi selecionado se mostrou flexível o suficiente para modelar diversos padrões de consumo de recursos, sem implicar com isso em uma grande complexidade. Esse conjunto de características foi utilizado para a construção do controlador de admissão de redes virtuais. De fato, o controlador de admissão tem um papel essencial para garantir o provimento da reserva de volume a longo prazo. Se um número excessivo de redes for admitido, então a probabilidade de bloqueio de recursos aumenta, dificultando o provimento integral da reserva de volume a longo prazo.

As análises desenvolvidas por meio de simulação mostraram que a proposta do controlador de admissão do VIPER é mais eficiente do que outras propostas da literatura. De fato, permitir o acesso de um número de redes menor que o ideal é ruim, pois a infraestrutura física fica subutilizada, mas permitir a entrada de mais redes do que o máximo permitido é muito pior, pois os SLAs das redes virtuais não serão providos. Na análise do controle de admissão, o Sandpiper [37] e o VNE-AC [43] introduziram um erro de 66,7% no cenário de tráfego 'on-off' e o Sandpiper introduziu um erro ainda maior no cenário de demanda crescente. O controle de admissão do VIPER foi a única proposta capaz de obter com uma pequena margem de erro o valor ideal do número de redes admitidas em todos os cenários analisados e ao mesmo tempo garantir que o limiar da probabilidade de bloqueio de recursos fosse respeitado.

Os ganhos observados com o controle de admissão do VIPER se devem, em especial, por esse mecanismo utilizar dados reais do uso do substrato e também ao cálculo do valor Δ, que compensa as variações nas demandas do substrato de acordo com o perfil de uso de recursos de cada rede virtual.

Uma observação importante para ser feita à respeito do VIPER é que o sistema proposto faz apenas um controle local dos recursos, ou seja, por nó. Contudo, uma rede virtual é um conjunto de nós e a reserva de recursos deve ser projetada globalmente. De fato, nenhuma reserva global é bem sucedida sem um módulo de controle de recursos por nó, como o provido pelo VIPER. Isso contudo, ainda não soluciona todo o problema da alocação de recursos físicos para uma rede virtual. Por exemplo, se uma rede possui um caminho formado pela sequência dos nós A, B e C, é preciso estudar a alocação de recursos para evitar que o nó A seja punido enquanto que os nós B e C possuem recursos sobressalentes.

O controle de admissão também deve considerar a visão global da rede. Um algoritmo de mapeamento de redes virtuais busca uma sequência de nós físicos que seja suficiente para criar a topologia virtual. Contudo, esse mapeamento só pode ser feito se o controle de admissão de cada um dos nós físicos permitir a alocação de recursos para os nós virtuais da nova rede. Basta que um dos nós físicos não possa receber a nova rede para que todo o mapeamento seja invalidado. Nesse caso, uma nova busca deve ser feita e um novo mapeamento deve ser proposto.

Dessa forma, as propostas discutidas nessa tese trazem contribuição significativa dentro da área de Internet do Futuro. Como trabalhos futuros, pretende-se expandir a análise das propostas com valores de consumo de roteadores reais. Outro ponto interessante para ser abordados é a utilização de diferentes módulos de monitoramento que tenham uma sobrecarga de consumo de recursos físicos menor.

Outra análise importante é o uso dos algoritmos propostos em outras plataformas, como a plataforma do OpenFlow. Nesse caso, os algoritmos propostos seriam

utilizados para atuar sobre os recursos controlados pelo FlowVisor.

Uma outra questão não abordada nessa tese é o mapeamento de redes virtuais sobre redes físicas. Esse algoritmo deve considerar o controle de admissão do VIPER para determinar se um mapeamento é adequado ou não. Técnicas como o balanceamento de cargas podem ser estudadas para um mapeamento de redes mais flexível sobre redes físicas sobrecarregadas.

A pesquisa iniciada nessa tese também pode evoluir em outras direções como o provimento de segurança e de qualidade de serviço de uma forma mais ampla.

Referências Bibliográficas

[1] RUBINSTEIN, M. G., ABDESSLEM, F. B., CAVALCANTI, S. R., et al. "Measuring the capacity of in-car to in-car vehicular networks", *IEEE Communications Magazine*, v. 47, n. 11, pp. 128–136, nov. 2009.

[2] ITU. *ITU Internet Reports 2005: The Internet of Things - Executive Summary*. In: Report, International Telecommunication Union, Geneva, 2005.

[3] HE, H., ZHU, Z., MAKINEN, E. "A Neural Network Model to Minimize the Connected Dominating Set for Self-Configuration of Wireless Sensor Networks", *IEEE Transactions on Neural Networks*, v. 20, n. 6, pp. 973 –982, jun. 2009.

[4] REXFORD, J., DOVROLIS, C. "Future Internet Architecture: Clean-Slate Versus Evolutionary Research", *Communications of the ACM*, v. 53, n. 9, pp. 36–40, 2010.

[5] MOREIRA, M. D. D., FERNANDES, N. C., COSTA, L. H. M. K., et al. "Internet do Futuro: Um Novo Horizonte". In: *Minicursos do Simpósio Brasileiro de Redes de Computadores*, SBRC '09, pp. 1–59, Rio de Janeiro, Brazil, maio 2009.

[6] CLARK, D. D., WROCLAWSKI, J., SOLLINS, K. R., et al. "Tussle in cyberspace: defining tomorrow's Internet", *IEEE/ACM Transactions on Networking*, v. 13, n. 3, pp. 462–475, jun. 2005.

[7] DEERING, S. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460, dez. 1998.

[8] GUNDERSON, S. H. "Measuring the current state of IPv6 for ordinary users". In: *Global IPv6 statistics*, RIPE 57, oct 2008.

[9] FERNANDES, N. C., MOREIRA, M. D. D., VELLOSO, P. B., et al. "Ataques e Mecanismos de Segurança em Redes Ad Hoc". In: *Minicursos do Simópsio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg'2006)*, pp. 49–102, ago. 2006.

[10] DA SILVA, R. I., LEITE, J. C. B., FERNANDEZ, M. P. "Extending the Lifetime of Ad Hoc Wireless Networks", *Lecture Notes in Computer Science*, v. 4114, pp. 1324–1331, 2006.

[11] OLIVEIRA, C. T., MOREIRA, M. D. D., RUBINSTEIN, M. G., et al. "Redes Tolerantes a Atrasos e Desconexões". In: *Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'2007)*, 2007.

[12] OLIVEIRA, T. C., BRAGA, R. B., TAVEIRA, D. M., et al. "A Predicted-contact Routing Scheme for Brazilian Rural Networks". In: *IFIP Wireless Days Conference*, Dubai, United Arab Emirates, 2008. IEEE Computer Society.

[13] GARETTO, M., FIGUEIREDO, D. R., GAETA, R., et al. "A modeling framework to understand the tussle between ISPs and peer-to-peer file-sharing users", *Performance Evaluation*, v. 64, pp. 819 – 837, 2007.

[14] VERDI, F. L., ESTEVE, C., PASQUINI, R., et al. "Novas Arquiteturas de Data Center para Cloud Computing". In: *Minicursos - Livro Texto - SBRC 2010*, SBRC '10, pp. 103–152, 2010.

[15] BERTINI, L., B., J. C., MOSSÉ, D. "Power optimization for dynamic configuration in heterogeneous web server clusters", *The Journal of Systems and Software*, v. 83, pp. 585 – 598, 2010.

[16] FERNANDES, N. C., DUARTE, O. C. M. B. "CHARADAS: Uma proposta para uso de CHAve de grupo no Roteamento Através de Distribuição Assimétrica Segura". In: *XXV Simpósio Brasileiro de Telecomunicações (SBrT'07)*, set. 2007.

[17] FERNANDES, N., DUARTE, O. C. M. B. "An Efficient Group Key Management for Secure Routing in Ad Hoc Networks". In: *IEEE Globecom 2008 Computer and Communications Network Security Symposium (GC'08 CCNS)*, pp. 1–5, dez. 2008.

[18] FERNANDES, N. C., DUARTE, O. C. M. B. "A Lightweight Group-Key Management Protocol for Secure Ad-Hoc-Network Routing", *Computer Networks*, v. 55, n. 3, pp. 759–778, fev. 2011.

[19] FERNANDES, N. C., DUARTE, O. C. M. B. "Controle de Acesso Auto-Organizável e Robusto Baseado em Nós Delegados para Redes Ad Hoc". In: *Anais do VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg'08*, pp. 3 –16, set. 2008.

[20] FERNANDES, N. C., MOREIRA, M. D., DUARTE, O. C. M. B. "A Self-Organized Mechanism for Thwarting Malicious Access in Ad Hoc Networks". In: *The 29th Conference on Computer Communications (IEEE INFOCOM miniconference 2010)*, pp. 1–5, San Diego, California, USA, apr 2010.

[21] FELDMANN, A. "Internet clean-slate design: what and why?" *ACM SIGCOMM Computer Communication Review*, v. 37, n. 3, pp. 59–64, jul. 2007.

[22] GENI PLANNING GROUP. "GENI Design Principles", *IEEE Computer*, v. 39, n. 9, pp. 102–105, set. 2006.

[23] FIUCZYNSKI, M. E. "PlanetLab: overview, history, and future directions", *SIGOPS Oper. Syst. Rev.*, v. 40, n. 1, pp. 6–10, 2006.

[24] FERNANDES, N. C., MOREIRA, M. D. D., DUARTE, O. C. M. B. "XNetMon: A Network Monitor for Securing Virtual Networks". In: *IEEE International Conference on Communications (ICC 2011 - Next Generation Networking and Internet Symposium)*, ICC'11 NGNI, 2011.

[25] FERNANDES, N. C., DUARTE, O. C. M. B. "Provendo Isolamento e Qualidade de Serviço em Redes Virtuais". In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2011*, pp. 1–14, may 2011.

[26] CARVALHO, H. E. T., FERNANDES, N. C., DUARTE, O. C. M. B. "Um Controlador Robusto de Acordos de Nível de Serviço para Redes Virtuais Baseado em Lógica Nebulosa". In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2011*, pp. 1–14, may 2011.

[27] COUTO, R. S., CAMPISTA, M. E. M., COSTA, L. H. M. K. "XTC: Um Controlador de Vazão para Roteadores Virtuais Baseados em Xen". In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2011*, pp. 1–14, may 2011.

[28] FREITAS, R. B., DE PAULA, L. B., MADEIRA, E., et al. "Using virtual topologies to manage inter-domain QoS in next-generation networks", *International Journal of Network Management*, v. 20, pp. 111–128, May 2010.

[29] BHATIA, S., MOTIWALA, M., MUHLBAUER, W., et al. "Trellis: A platform for building flexible, fast virtual networks on commodity hardware". In:

Proceedings of the Workshop on Real Overlays and Distributed Systems (ROADS), pp. 1–6, dez. 2008.

[30] BOURGUIBA, M., HADDADOU, K., KORBI, I. E., et al. "A Container-based I/O for Virtual Routers: Experimental and Analytical Evaluations". In: *IEEE International Conference on Communications (ICC 2011)*, out. 2011.

[31] EGI, N., GREENHALGH, A., HANDLEY, M., et al. "Evaluating Xen for Router Virtualization". In: *International Conference on Computer Communications and Networks*, ICCCN'07, pp. 1256–1261, ago. 2007.

[32] CUNHA, D. O., DUARTE, O. C. M. B., PUJOLLE, G. "A Cooperation-Aware Routing Scheme for Fast Varying Fading Wireless Channels", *IEEE Communications Letters*, v. 12, pp. 794–796, oct 2008. ISSN: 1089–7798.

[33] FERNANDES, N. C., DUARTE, O. C. M. B. "Autoconfiguração de Endereços Baseada em Filtros de Bloom para Redes Ad Hoc". In: *XXVI Simpósio Brasileiro de Redes de Computadores (SBRC'08)*, pp. 273–286, maio 2008.

[34] FERNANDES, N. C., MOREIRA, M. D., DUARTE, O. C. M. B. "An Efficient Filter-based Addressing Protocol for Autoconfiguration of Mobile Ad Hoc Networks". In: *The 28th Conference on Computer Communications (IEEE INFOCOM 2009)*, pp. 2464 – 2472, Rio de Janeiro, RJ, Brazil, abr. 2009.

[35] FERNANDES, N. C. *Controle de Acesso Distribuído para Redes Ad Hoc*. Tese de Mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, fev. 2008.

[36] FEAMSTER, N., GAO, L., REXFORD, J. "How to lease the Internet in your spare time", *ACM SIGCOMM Computer Communication Review*, v. 37, n. 1, pp. 61–64, jan. 2007.

[37] WOOD, T., SHENOY, P., VENKATARAMANI, A., et al. "Black-box and gray-box strategies for virtual machine migration". In: *Proceedings of the 4th USENIX conference on Networked systems design & implementation (NSDI'07)*, pp. 229–242. USENIX Association, 2007.

[38] FERNANDES, N. C., MOREIRA, M. D. D., MORAES, I. M., et al. "Virtual Networks: Isolation, Performance, and Trends", *Annals of Telecommunications*, v. 66, n. 5–6, pp. 339–355, 2011.

[39] CARAPINHA, J., JIMENEZ, J. "Network Virtualization - a View from the Bottom". In: *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '09, pp. 73–80, New York, NY, USA, 2009. ACM.

[40] FERNANDES, N. C., DUARTE, O. C. M. B. "XNetMon: Uma Arquitetura com Segurança para Redes Virtuais". In: *Anais do X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, SBSEG '10, pp. 339–352, Fortaleza, CE, Brazil, out. 2010.

[41] EGI, N., GREENHALGH, A., HANDLEY, M., et al. "Fairness issues in software virtual routers". In: *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, PRESTO '10, pp. 33–38, ago. 2008.

[42] ALMESBERGER, W. "Linux network traffic control - Implementation overview". In: *5th Annual Linux Expo*, pp. 153–164, 1999.

[43] FAJJARI, I., AITSAADI, N., PUJOLLE, G., et al. "VNE-AC: Virtual Network Embedding Algorithm based on Ant Colony Metaheuristic". In: *ICC 2011 Next Generation Networking and Internet Symposium (ICC'11 NGNI)*, pp. 1–6. IEEE, jun. 2011.

[44] PARNO, B., PERRIG, A., GLIGOR, V. "Distributed detection of node replication attacks in sensor networks". In: *IEEE Symposium on Security and Privacy*, pp. 49–63, maio 2005.

[45] THOMSON, S., NARTEN, T. *IPv6 stateless address autoconfiguration.* RFC 2462, dez. 1998.

[46] NARTEN, T., DRAVES, R. *Privacy Extensions for Stateless Address Autoconfiguration in IPv6.* RFC 3041, jan. 2001.

[47] PERKINS, C. E., ROYERS, E. M., DAS, S. R. *IP address autoconfiguration for ad hoc networks.* Internet Draft, jul. 2000.

[48] FAN, Z., SUBRAMANI, S. "An Address Autoconfiguration Protocol for IPv6 Hosts in a Mobile Ad Hoc Network", *Computer Communications*, v. 28, n. 4, pp. 339–350, mar. 2005.

[49] FAZIO, M., VILLARI, M., PULIAFITO, A. "IP address autoconfiguration in ad hoc networks: design, implementation and measurements", *Computer Networks*, v. 50, n. 7, pp. 898–920, 2006.

[50] ERIKSSON, J., FALOUTSOS, M., KRISHNAMURTHY, S. V. "DART: dynamic address routing for scalable ad hoc and mesh networks", *IEEE/ACM Transactions on Networking*, v. 15, n. 1, pp. 119–132, 2007.

[51] KIM, D., JEONG, H.-J., TOH, C. K., et al. "Passive Duplicate Address-Detection Schemes for On-Demand Routing Protocols in Mobile Ad Hoc Networks", *IEEE Transactions on Vehicular Technology*, v. 58, pp. 3558–3568, sep 2009.

[52] VAIDYA, N. H. "Weak duplicate address detection in mobile ad hoc networks". In: *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pp. 206–216, New York, NY, USA, 2002. ACM.

[53] KIM, H., KIM, S. C., YU, M., et al. "DAP: Dynamic Address Assignment Protocol in Mobile Ad-hoc Networks". In: *IEEE International Symposium on Consumer Electronics (ISCE 2007)*, pp. 1–6. IEEE, jun 2007.

[54] NESARGI, S., PRAKASH, R. "MANETconf: configuration of hosts in a mobile ad hoc network". In: *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2002)*, v. 2, pp. 1059–1068. IEEE, jun. 2002.

[55] ZHOU, H., NI, L., MUTKA, M. "Prophet address allocation for large scale MANETs". In: *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2003)*, v. 2, pp. 1304–1311. IEEE, mar 2003.

[56] LAUFER, R. P., VELLOSO, P. B., CUNHA, D. O., et al. "Towards Stateless Single-Packet IP Traceback". In: *32nd IEEE Conference on Local Computer Networks (LCN'2007)*, pp. 548–555, out. 2007.

[57] FAN, L., CAO, P., ALMEIDA, J., et al. "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol", *IEEE/ACM Transactions on Networking*, v. 8, n. 3, pp. 281–293, jun. 2000.

[58] CAMPISTA, M. E. M., MORAES, I. M., ESPOSITO, P., et al. "The Ad Hoc Return Channel: a Low-Cost Solution for Brazilian Interactive Digital TV", *IEEE Communications Magazine*, v. 45, n. 1, pp. 136–143, jan. 2007.

[59] BARAN, P. "On Distributed Communications Networks", *IEEE Transactions on Communications Systems*, v. 12, n. 1, pp. 1–9, mar. 1964.

[60] CLARK, D., BRADEN, R., SOLLINS, K., et al. *New Arch: Future Generation Internet Architecture*. Relatório técnico, USC Information Sciences Institute Computer Networks Division, MIT Laboratory for Computer Science and International Computer Science Institute (ICSI), ago. 2004.

[61] BLUMENTHAL, M. S., CLARK, D. D. "Rethinking the Design of the Internet: the End-to-End Arguments vs. the Brave New World", *ACM Transactions on Internet Technology*, v. 1, n. 1, pp. 70–109, ago. 2001.

[62] HALLIDAY, J., ARTHUR, C. *Clean-Slate - An Interdisciplinay Research Program*. http://cleanslate.stanford.edu/index.php, Accessed in January, 2011.

[63] DOVROLIS, C. "What would Darwin think about clean-slate architectures?" *SIGCOMM Comput. Commun. Rev.*, v. 38, n. 1, pp. 29–34, 2008.

[64] ANDERSON, T., PETERSON, L., SHENKER, S., et al. "Overcoming the Internet Impasse through Virtualization", *IEEE Computer*, v. 38, n. 4, pp. 34–41, abr. 2005.

[65] WANG, Y., DER MERWE, J. V., REXFORD, J. "VROOM: Virtual Routers On the Move". In: *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking*, pp. 1–7, nov. 2007.

[66] WANG, Y., KELLER, E., BISKEBORN, B., et al. "Virtual Routers on the Move: Live Router Migration as a Network-Management Primitive". In: *ACM SIGCOMM*, pp. 231–242, ago. 2008.

[67] ACHEMLAL, M., ET AL. *Virtualisation Approach: Concept*. Relatório Técnico D-3.1.1, 4WARD - Architecture and Design for the Future Internet, set. 2010.

[68] SCHAFFRATH, G., WERLE, C., PAPADIMITRIOU, P., et al. "Network virtualization architecture: proposal and initial prototype". In: *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '09, pp. 63–72, New York, NY, USA, 2009. ACM.

[69] VAUGHAN-NICHOLS, S. "New Approach to Virtualization Is a Lightweight", *Computer*, v. 39, n. 11, pp. 12 –14, nov. 2006.

[70] LAUREANO, M. A. P., MAZIERO, C. A. "Virtualização : Conceitos e Aplicações em Segurança". In: *Minicursos do VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pp. 1–49, 2008.

[71] QEMU. *QEMU - Open Source processor emulator*. http://wiki.qemu.org/Main_Page, acessado em fevereiro de 2011.

[72] Xen. *Xen.org*. http://www.xen.org/, acessado em fevereiro de 2011.

[73] VMware. *VMware*. http://www.vmware.com/, acessado em fevereiro de 2011.

[74] Virtual PC. *Windows Virtual PC*. http://www.microsoft.com/windows/virtual-pc/, acessado em fevereiro de 2011.

[75] VirtualBox. *Welcome to VirtualBox*. http://www.virtualbox.org/, acessado em fevereiro de 2011.

[76] RIONDATO, M. *FreeBSD Handbook*. http://www.freebsd.org/doc/handbook/jails.html, acessado em fevereiro de 2011.

[77] OpenVZ. *Welcome to OpenVZ Wiki!* http://wiki.openvz.org/Main_Page, acessado em fevereiro de 2011.

[78] VServer. *Linux VServer*. http://linux-vserver.org/Overview, acessado em fevereiro de 2011.

[79] PlanetLab. *PLANETLAB - An Open platform for developing, deploying, and accessing planetary-scale services - User's Guide*. http://www.planet-lab.org/doc/guides/user, acessado em fevereiro de 2011.

[80] VServer. *VNET: PlanetLab Virtualized Network Access*. http://www.planet-lab.org/doc/vnet, acessado em fevereiro de 2011.

[81] OpenFlow. *The OpenFlow Switch Consortium*. http://www.openflowswitch.org/, acessado em fevereiro de 2011.

[82] MATTOS, D. M. F., FERNANDES, N. C., DUARTE, O. C. M. B. "XenFlow: Um Sistema de Processamento de Fluxos Robusto e Eficiente para Migração em Redes Virtuais". In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2011*, pp. 1–14, may 2011.

[83] MENON, A., COX, A. L., ZWAENEPOEL, W. "Optimizing Network Virtualization in Xen". In: *USENIX Annual Technical Conference*, pp. 15–28, maio 2006.

[84] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., et al. "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, v. 38, n. 2, pp. 69–74, abr. 2008.

[85] SHERWOOD, R., CHAN, M., COVINGTON, A., et al. "Carving Research Slices out of Your Production Networks with OpenFlow", *ACM SIGCOMM Computer Communication Review*, v. 40, n. 1, pp. 129–130, 2010.

[86] PFAFF, B., HELLER, B., TALAYCO, D., et al. *OpenFlow Switch Specification Version 1.0.0 (Wire Protocol 0x01)*. Relatório técnico, Stanford University, dez. 2009.

[87] NASCIMENTO, M. R., ROTHENBERG, C. E., SALVADOR, M. R., et al. "QuagFlow: partnering Quagga with OpenFlow". In: *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, SIGCOMM '10, pp. 441–442, New York, NY, USA, 2010. ACM.

[88] GUDE, N., KOPONEN, T., PETTIT, J., et al. "NOX: Towards an Operating System for Networks", *ACM SIGCOMM Computer Communication Review*, v. 38, n. 3, pp. 105–110, jul. 2008.

[89] KOHLER, E., MORRIS, R., CHEN, B., et al. "The Click modular router", *Operating Systems Review*, v. 5, n. 34, pp. 217–231, dez. 1999.

[90] MATEO, M. P. *OpenFlow Switching Performance*. Tese de Mestrado, Politecnico Di Torino, Torino, Italy, jul. 2009.

[91] CHISNALL, D. *The Definitive Guide to the Xen Hypervisor*. EUA, Prentice Hall, 2007.

[92] MORAES, I. M., PISA, P. S., CARVALHO, H. E. T., et al. "VNEXT: Uma Ferramenta de Controle e Gerenciamento para Redes Virtuais Baseadas em Xen". In: *Salão de Ferramentas do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2011*, pp. 1–8, may 2011.

[93] PISA, P. S., COUTO, R. S., CARVALHO, H. E. T., et al. "VNEXT: Virtual NEtwork management for Xen-based Testbeds". In: *A ser publicado no 2nd IFIP International Conference Network of the Future - NoF'2011*, pp. 1–5, nov 2011.

[94] MATTOS, D. M. F., FERNANDES, N. C., CARDOSO, L. P., et al. "OMNI: Uma Ferramenta para Gerenciamento Autônomo de Redes OpenFlow". In: *Salão de Ferramentas do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2011*, pp. 1–8, may 2011.

[95] MATTOS, D. M. F., FERNANDES, N. C., DA COSTA, V. T., et al. "OMNI: OpenFlow MaNagement Infrastructure". In: *A ser publicado no 2nd IFIP*

International Conference Network of the Future - NoF'2011, pp. 1–5, nov 2011.

[96] JONES, M. T. *Anatomy of the libvirt virtualization library - An API for easy Linux virtualization.* Relatório técnico, IBM, jan. 2010.

[97] GONG, Z., GU, X., WILKES, J. "PRESS: PRedictive Elastic ReSource Scaling for cloud systems". In: *6th International Conference on Network and Services Management (CNSM 2010)*, pp. 9–16. IEEE, out. 2010.

[98] MENASCE, D. A., BENNANI, M. N. "Autonomic Virtualized Environments". In: *Proceedings of the International Conference on Autonomic and Autonomous Systems (ICAS '06)*, pp. 28–, Washington, DC, USA, 2006. IEEE Computer Society.

[99] PAYER, H., RÖCK, H., KIRSCH, C. "Get What You Pay For: Providing Performance Isolation in Virtualized Execution Environments". In: *European Systems Conference (EuroSys)*, pp. 1–2, 2010.

[100] HAN, S.-M., HASSAN, M. M., YOON, C.-W., et al. "Efficient Service Recommendation System for Cloud Computing Market". In: *ICIS'09: Proceedings of the 2nd International Conference on Interaction Sciences*, pp. 839–845, 2009.

[101] JIN, X., CHEN, H., WANG, X., et al. "A Simple Cache Partitioning Approach in a Virtualized Environment". In: *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, pp. 519–524, 2009.

[102] XU, J., ZHAO, M., FORTES, J., et al. "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches." *Cluster Computing*, v. 11, pp. 213–227, 2008.

[103] MENG, X., PAPPAS, V., ZHANG, L. "Improving the scalability of data center networks with traffic-aware virtual machine placement". In: *2010 Proceedings of IEEE INFOCOM*, pp. 1–9, 2010.

[104] KELLER, E., GREEN, E. "Virtualizing the Data Plane Through Source Code Merging". In: *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pp. 9–14, ago. 2008.

[105] COUTO, R. S., CAMPISTA, M. E. M., COSTA, L. H. M. K. "XTC: A Throughput Control Mechanism for Xen-based Virtualized Software Routers". In: *Accepted for publication in the Proceedings of the IEEE GLOBECOM'11*, Houston, Texas, USA, dez. 2011.

[106] BOURGUIBA, M., HADDADOU, K., PUJOLLE, G. "Evaluating Xen-based virtual routers performance", *International Journal of Communication Networks and Distributed Systems*, v. 6, pp. 268–282, 2011.

[107] GUPTA, D., CHERKASOVA, L., GARDNER, R., et al. "Enforcing performance isolation across virtual machines in Xen". In: *Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware (Middleware '06)*, pp. 342–362, New York, NY, USA, 2006. Springer-Verlag New York, Inc.

[108] MCILORY, R., SVENTEK, J. "Resource virtualisation of network routers". In: *Workshop on High Performance Switching and Routing*, pp. 1–6, New Yok, NY, USA, 2006. IEEE.

[109] GENI. *GENI Spiral 1 Annual Report 2009*. Relatório técnico, National Science Foundation, dez. 2009.

[110] HOUIDI, I., LOUATI, W., ZEGHLACHE, D., et al. "Adaptive Virtual Network Provisioning". In: *Proceedings of the 2nd ACM workshop on Virtualized Infrastructure Systems and Architectures*, VISA '10, pp. 41–48, New York, NY, USA, set. 2010. ACM.

[111] KOUNAVIS, M. E., CAMPBELL, A. T., CHOU, S., et al. "The Genesis Kernel: A Programming System for Spawning Network Architectures", *IEEE Journal on Selected Areas in Communications*, v. 19, n. 3, pp. 511–26, mar. 2001.

[112] LISCHKA, J., KARL, H. "A virtual network mapping algorithm based on subgraph isomorphism detection". In: *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '09, pp. 81–88, New York, NY, USA, 2009. ACM.

[113] Iptables. *Netfilter: firewalling, NAT, and packet mangling for Linux*. http://www.netfilter.org/, acessado em fevereiro de 2011.

[114] SCHNEIER, B. *The Blowfish Encryption Algorithm*. http://www.schneier.com/blowfish.html, acessado em fevereiro de 2011.

[115] KALISKI, B., STADDON, J. *PKCS #1: RSA Cryptography Specifications Version 2.0*. RFC 2437, out. 1998.

[116] ANWER, M. B., NAYAK, A., FEAMSTER, N., et al. "Network I/O fairness in virtual machines". In: *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, VISA '10, pp. 73–80, New York, NY, USA, 2010. ACM.