Uma Arquitetura de Virtualização de Redes Orientada à Migração com Qualidade de Serviço

Diogo Menezes Ferrazani Mattos

PEE / COPPE / UFRJ

Dissertação submetida para a obtenção do título de
Mestre em Engenheira Elétrica
ao Programa de Engenharia Elétrica/COPPE/UFRJ

UMA ARQUITETURA DE VIRTUALIZAÇÃO DE REDES ORIENTADA À MIGRAÇÃO COM QUALIDADE DE SERVIÇO

Diogo Menezes Ferrazani Mattos

Dissertação de Mestrado 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 Mestre em Engenharia Elétrica.

Orientador: Otto Carlos Muniz Bandeira Duarte

Rio de Janeiro
Dezembro de 2012

UMA ARQUITETURA DE VIRTUALIZAÇÃO DE REDES ORIENTADA À MIGRAÇÃO COM QUALIDADE DE SERVIÇO

Diogo Menezes Ferrazani Mattos

DISSERTAÇÃO 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 MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

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

Prof. Nivio Ziviani, Ph.D.

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Prof. Igor Monteiro Moraes, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2012

Mattos, Diogo Menezes Ferrazani

Uma Arquitetura de Virtualização de Redes Orientada à Migração com Qualidade de Serviço/Diogo Menezes Ferrazani Mattos. – Rio de Janeiro: UFRJ/COPPE, 2012.

XIV, 82 p.: il.; 29, 7cm.

Orientador: Otto Carlos Muniz Bandeira Duarte

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2012.

Referências Bibliográficas: p. 76 – 82.

1. Virtualização de Redes. 2. Redes Definidas por Software. 3. Computação em Nuvens. 4. Internet do Futuro. 5. Qualidade de Serviço. 6. Xen. 7. OpenFlow. 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 a Deus pela oportunidade de concretizar mais um ciclo na minha formação profissional. Agradeço à minha família e à minha namorada, que sempre estiveram ao meu lado, por todo carinho e compreensão. Em especial, agradeço aos meus pais, pelo apoio que me dão em todos os momentos e por sempre me motivarem a seguir em frente.

Agradeço aos amigos, em especial, Daniel Vega, Hugo Eiji, Pedro Pisa, Pedro Coutinho, pela amizade e companheirismo e por tornarem a graduação uma experiência repleta de aprendizagem e boas lembranças. Agradeço, também, a todos os amigos que fiz no Grupo de Teleinformática e Automação, pois sempre contribuíram positivamente para a conclusão desse trabalho. Em especial, agradeço a Lyno Ferraz, Lucas Mauricio, Daniel Neto, Pedro Guimarães e Rodrigo Couto.

Agradeço, ainda, a todos os professores que participaram da minha formação. Em especial, agradeço ao meu orientador, professor Otto Duarte, por todos os conselhos, dedicação e principalmente paciência, durante a orientação no mestrado e a gradução. Gostaria de agradecer também aos professores Luís Henrique, Miguel, Aloysio do GTA/UFRJ, aos professores Igor e Pedro Velloso da UFF e aos professores Marcelo Amorim e Michel Abdalla.

Agradeço aos professores Nivio Ziviani, Luís Henrique Costa e Igor Monteiro pela participação na banca examinadora.

Agradeço aos funcionários do Programa de Engenharia Elétrica da COPPE/UFRJ, Maurício, Daniele, Rosa e Arthur pela presteza no atendimento na secretaria do Programa.

Agradeço a todos que participaram de forma direta ou indireta da minha formação profissional. Por fim, agradeço a FINEP, FUNTTEL, CNPq, CAPES, FAPERJ e UOL pelo financiamento deste trabalho.

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

UMA ARQUITETURA DE VIRTUALIZAÇÃO DE REDES ORIENTADA À MIGRAÇÃO COM QUALIDADE DE SERVIÇO

Diogo Menezes Ferrazani Mattos

Dezembro/2012

Orientador: Otto Carlos Muniz Bandeira Duarte

Programa: Engenharia Elétrica

A técnica de virtualização de redes é essencial para prover o compartilhamento dos recursos físicos e permitir a implantação de inovações no núcleo das redes. Uma das principais funcionalidades da virtualização de redes é permitir o remapeamento das topologias das redes virtuais sobre o substrato físico, ou simplesmente migração de redes virtuais, agregando flexibilidade para o ambiente de redes virtuais. Contudo, fornecer a primitiva de migração, garantir isolamento entre redes virtuais e assegurar a oferta de Qualidade de Serviço (QoS) ainda são desafios tecnológicos. Este trabalho propõe o sistema QFlow para isolar o uso de recursos e prover QoS em uma arquitetura híbrida de redes virtuais. O sistema QFlow combina uma ferramenta de virtualização, como o Xen, para implementar o plano de controle, com a interface de programação de aplicação (API) OpenFlow, para o controle do mecanismo de encaminhamento de pacotes. Os dois principais elementos do QFlow são o controlador de uso de recursos e a aplicação de mapeamento de parâmetros de QoS. O sistema QFlow permite oferecer diferenciação de serviços inter-redes virtuais distintas e intrarrede virtual, garantindo recursos a diferentes fluxos de uma rede virtual. Um protótipo foi desenvolvido e a avaliação de desempenho do sistema proposto mostra que a migração de redes virtuais ocorre sem que haja a interrupção ou perdas no encaminhamento de pacotes. Além disso, o controlador de recursos proposto atende aos requisitos de qualidade de serviço das redes virtuais e redistribui de forma ponderada os recursos ociosos da rede.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AN ARCHITECTURE FOR A MIGRATION-ORIENTED NETWORK VIRTUALIZATION WITH QUALITY OF SERVICE

Diogo Menezes Ferrazani Mattos

December/2012

Advisor: Otto Carlos Muniz Bandeira Duarte

Department: Electrical Engineering

Network virtualization is a mandatory technique to provide sharing of physical resources and to innovate in the core of the network. One of the key features of network virtualization is to remap virtual network topologies over the physical substrate, called virtual network migration, which adds flexibility to virtual networking environment. Nevertheless, providing migration primitive, ensuring isolation between virtual networks and guaranteeing Quality of Service (QoS) are still technological challenges. In this work, we propose the QFlow system to isolate resource usage and to provide QoS on hybrid architecture of virtual networks. The QFlow system combines a virtualization tool such as Xen, to implement the control plane, with the OpenFlow Application Programming Interface (API) to control packet-forwarding mechanism. The two main QFlow elements are the resource-usage controller and the QoS parameters mapping application. QFlow system allows offering service differentiation between inter-virtual networks and intra-virtual network, ensuring resources for different flows in a virtual network. We developed a prototype and the performance evaluation of the proposed system shows that migration of virtual networks occurs without interruption or disruption of packet forwarding. Furthermore, the resource controller meets the Quality of Service requirements of virtual networks and weightily redistributes idle network resources.

Sumário

Lista de Figuras

Lista de Algoritmos

Lista de Abreviaturas

API	*Application Programming Interface*, p. 21, 39, 75
ARP	*Address Resolution Protocol*, p. 38, 40, 61
CPU	*Central Processing Unit*, p. 5, 15
DOVE	*Distributed Overlay Virtual Ethernet*, p. 61
E/S	Entrada e Saída, p. 7, 31, 33
I/O	*Input and Output*, p. 7
IP	*Internet Protocol*, p. 10, 36
LAN	*Local Area Network*, p. 35
MAC	*Medium Access Control*, p. 7, 10, 33
MSDP	*Multicast Source Discovery Protocol*, p. 62
NAT	*Network Address Translation*, p. 33
QoS	*Quality of Service*, p. 46
RAM	*Random-Access Memory*, p. 6
SDN	*Software Defined Networking*, p. 59, 75
SSL	*Secure Sockets Layer*, p. 55
SSL	*Secure Sockets Layer*, p. 41
TRILL	*Transparent Interconnection of Lots of Links*, p. 61
TTL	*Time to Live*, p. 34, 58
VLAN	*Virtual Local Area Network*, p. 38, 39
VMM	*Virtual Machine Monitor*, p. 5
vCPU	*Virtual Central Processing Unit*, p. 5

Capítulo 1

Introdução

A Internet é um grande sucesso, porém o seu projeto original não leva em consideração exigências atuais, como segurança, gerenciamento, mobilidade e monitoramento [1–5]. Assim, novos serviços são propostos para atender suas atuais exigências. Contudo, os provedores de serviço de Internet rejeitam a introdução de inovações no núcleo da Internet porque temem que experimentos e alterações no núcleo da rede possam impactar e causar possíveis falhas em serviços já estabelecidos. Portanto, a experimentação de novos serviços e protocolos necessita de um ambiente que se aproxime das condições realísticas da Internet atual, usando a rede de produção ou a própria Internet, mas devem estar suficientemente isolados para protegê-las do mau funcionamento e falhas das novas propostas. Assim, um ambiente de experimentação de redes requer as caraterísticas e escala da Internet atual, mas com o tráfego experimental isolado do tráfego de produção.

1.1 Virtualização de Redes

A virtualização de redes é uma tecnologia essencial para prover um ambiente pluralista de experimentação para propostas para a Internet do Futuro [6–8], em que diversas redes lógicas com requisitos e pilhas de protocolos distintos compartilham a mesma rede física subjacente. A virtualização separa a função desempenhada por um elemento de rede de sua realização física. Assim, permite experimentar diversos protocolos e serviços inovadores no núcleo da rede, pois cada rede virtual, chamada de fatia de rede, é isolada das demais. Cada rede virtual é dedicada a um experimento e tem sua própria pilha de protocolos e arcabouço de controle. As redes virtuais devem ser isoladas e têm diferentes requisitos de qualidade de serviço (QoS) [9, 10]. Iniciativas recentes de experimentação de redes de nova geração, como PlanetLab [11], GENI [12] e Horizon [7], empregam a virtualização de redes como ferramenta de testes entre as tecnologias atuais e novas tecnologias. Contudo, essas iniciativas ainda enfrentam desafios para garantir o desempenho, o isolamento e os

requisitos de qualidade de serviço exigidos pelas aplicações que executam nas redes virtuais.

Esta dissertação foca dois ambientes de virtualização que serão usados para prover virtualização de redes em computadores pessoais: a plataforma Xen e o OpenFlow. A plataforma de virtualização Xen [13] adiciona uma camada de *software* entre o sistema operacional e o *hardware*, o chamado hipervisor ou monitor de máquinas virtuais [14, 15], que cria uma abstração do *hardware* da máquina física que é particionado em diversos ambientes virtuais. Os ambientes virtuais compartilham o acesso aos recursos físicos da máquina. No entanto, cada ambiente virtual acessa o seu próprio sistema operacional, com suas próprias aplicações e pilhas de protocolos. Cada ambiente virtual é chamado de máquina virtual. No caso de usar uma ferramenta de virtualização de computadores para realizar a virtualização de redes, as máquinas virtuais se comportam como roteadores virtuais, que compartilham os recursos de uma máquina física.

O OpenFlow [10] virtualiza fluxos, permitindo que a infraestrutura física de redes Ethernet seja compartilhada pela rede de produção e por redes experimentais. O OpenFlow é um projeto desenvolvido na Universidade de *Stanford* que tem por objetivo implementar uma tecnologia capaz de promover a inovação no núcleo da rede, através da execução de redes de testes em paralelo com a rede de produção. A ideia chave do OpenFlow é promover a inovação em redes universitárias, através da criação de um elemento controlador da rede que realiza a tomada de decisões na rede através de aplicações e particiona a rede física entre os diversos experimentos de acordo um subconjunto de campos dos cabeçalhos dos protocolos de Camada 2, 3 ou superior dos pacotes que trafegam na rede. A tecnologia OpenFlow promove a criação de redes definidas por *software*, usando elementos comuns de rede, tais como comutadores, roteadores, pontos de acesso ou, até mesmo, computadores pessoais[1] [16].

1.2 Objetivos e Contribuições

O principal objetivo desta dissertação é estudar e propor mecanismos eficientes e eficazes para criar e gerenciar redes virtuais.Assim, esta dissertação é dividida duas partes principais. A primeira parte apresenta o conjunto de ferramentas básicas para o desenvolvimento da proposta deste trabalho, mostrados no Capítulo 2. Esse conjunto é composto pelo OMNI (*OpenFlow MaNagement Infrastruture*), uma infraestrutura de controle e monitoramento para uma Rede Definida por Software,

[1]Uma das propostas para a implementação do comutador OpenFlow em computadores pessoais é o Open vSwitch [16], que funciona como um módulo do Linux que implementa o encaminhamento programável de pacotes diretamente no *kernel* do sistema operacional.

baseada em OpenFlow, e pelo XenFlow, um sistema de virtualização de redes que permite uma rede virtual Xen controlar uma matriz de encaminhamento OpenFlow. *OpenFlow MaNagement Infrastruture* é um conjunto de aplicações para uma rede OpenFlow que permite colher estatísticas da rede, gerenciar a criação, apagamento e migração de fluxo e, ainda, permite visualizar e atuar na rede através de uma interface de serviços web ou através de uma interface de usuário baseada na web. O XenFlow permite que protocolos de roteamento comuns da Internet exerçam o encaminhamento em uma rede OpenFlow. O sistema XenFlow foi inicialmente proposto como projeto de fim de curso [17] e, desde então, é estendido para novos cenários de aplicação.

A segunda parte deste trabalho apresenta a proposta QFlow, Qualidade de Serviço no sistema XenFlow, apresentado no Capítulo 3. O QFlow é um sistema de controle de qualidade de serviço e de isolamento de recursos, como CPU, memória e banda, em redes virtuais. O sistema QFlow baseia-se na arquitetura híbrida de virtualização de redes usada no sistema XenFlow [18], que combina a plataforma de virtualização de computadores Xen com a interfaces de programação de aplicação OpenFlow. As principais contribuições do sistema QFlow são: i) a garantia de isolamento e a oferta de qualidade de serviço através de um eficiente controle dos recursos disponíveis; ii) um mecanismo de mapeamento dos parâmetros de Qualidade de Serviço, definidos para cada rede virtual, em recursos do plano de encaminhamento de dados; e iii) um algoritmo de redistribuição de recursos ociosos na rede física entre as redes virtuais, de forma que os recursos sejam redistribuídos proporcionalmente à necessidade e à prioridade de cada rede virtual.

A principal contribuição desta dissertação é o sistema QFlow, Qualidade de Serviço em um sistema híbrido Xen e OpenFlow (QoS + XenFlow). O sistema QFlow proposto nesta dissertação se baseia no modelo híbrido do XenFlow de virtualização de redes [17, 18], combinando as ferramentas de virtualização de computadores Xen e a de programação de redes OpenFlow. No entanto, ao contrário do XenFlow, o QFlow garante o isolamento dos recursos compartilhados pelo roteadores virtuais no domínio privilegiado de controle do Xen, tornando-o robusto a ataque de negação de serviço, e também oferece garantias de qualidade de serviço. O sistema QFlow realiza o compartilhamento eficiente dos recursos ociosos e suporta o atendimento a acordos de nível de serviço (SLAs – *Service Level Agreements*), como por exemplo uma garantia mínima de banda destinada a um roteador virtual. Outra inovação do QFlow em relação às demais propostas de virtualização de redes é a aplicação de algoritmos para o encaminhamento de pacotes em difusão e para o isolamento do espaço de endereçamento de redes virtuais. O QFlow implementa algoritmos de aprendizagem que permitem o comutador OpenFlow de cada nó físico identificar quais redes virtuais estão conectadas a ele e qual interface pertence a qual rede.

A proposta desta dissertação se apoia no paradigma da separação de plano, no qual o encaminhamento e o controle da rede são desacoplados [19, 20]. A ideia central da proposta é que os pacotes de uma rede virtual permaneçam restritos aos nós dessa rede e não interfiram no funcionamento das demais, assim o tráfego de uma rede virtual não interfere no desempenho de outras redes. Um dos objetivos da do sistema QFlow proposto nesta dissertação é a virtualização de redes em que os nós tenham a impressão de que a infraestrutura física é dedicada somente para os pacotes da sua rede virtual. Nesse sentido, são também contribuições desta dissertação: i) a garantia de isolamento da comunicação entre redes virtuais; ii) o mapeamento de funções de isolamento de redes virtuais para primitivas do plano de dados; e iii) a combinação da solução de isolamento da comunicação com a proposta de isolamento de recursos [21], garantindo a virtualização de redes em um ambiente em que haja garantia de recursos para redes virtuais e, ao mesmo tempo, garantia de que os pacotes de uma dada rede sejam somente entregues aos nós destinados a receber.

As principais propostas para prover QoS são baseadas no provisionamento exagerado de recursos da infraestrutura física da rede e na configuração manual de parâmetros de QoS nos dispositivos físicos [22, 23]. O sistema proposto, diferentemente, monitora o uso de recursos por cada rede virtual e, com base nesses dados, calcula a redistribuição dos recursos ociosos na infraestrutura física. As principais propostas para prover o isolamento de redes virtuais visam somente o isolamento de recursos [3, 21, 24] ou se baseiam no encapsulamento dos pacotes de redes virtuais para prover o isolamento da comunicação das redes virtuais [25, 26]. A arquitetura proposta, no entanto, usa o paradigma da separação de planos para garantir o máximo desempenho do encaminhamento de pacotes e realiza tanto o isolamento de recursos quanto o isolamento da comunicação. O isolamento da comunicação é alcançado através da marcação dos pacotes de cada rede virtual com uma etiqueta que indica a qual rede o pacote pertence. O padrão 802.1Q, que define o funcionamento de VLANs (*Virtual Local Area Network*), é usado para marcar os pacotes de acordo com as regras definidas pela aplicação OpenFlow que traduz as regras do plano de controle para o plano de dados. O isolamento de recursos de rede consiste no mapeamento das redes virtuais em filas de encaminhamento definidas no plano de dados.

A ideia chave do QFlow é um controlador de recursos que define valores mínimo e máximo de cada fila e, assim, oferece QoS inter-redes, e também QoS intrarrede. A proposta de isolamento de redes virtuais estende o sistema XenFlow [18, 21], que realiza a separação de planos usando o Xen para executar o plano de controle e OpenFlow para o plano de dados. A proposta, então, conjuga a separação de planos do XenFlow, adicionando a marcação de pacotes por VLAN como esquema

de isolamento de redes virtuais.

Um protótipo do sistema QFlow foi desenvolvido e avaliado. Os resultados mostram que o algoritmo de redistribuição de recursos reduz a ociosidade da rede, quando comparado a outras técnicas, e atribui a banda reservada para cada rede virtual proporcionalmente ao valor contratado por cada rede. O sistema permite, então, o atendimento dos acordos de nível de serviço das redes virtuais, assim como otimiza o uso de recursos da rede física. A avaliação revelou ainda que a abordagem proposta executa tanto o isolamento da comunicação quanto o isolamento do uso de recursos entre redes virtuais.

1.3 Plataforma de Virtualização Xen

O Xen é uma plataforma de virtualização de computadores pessoais, bastante empregada na consolidação de servidores[2]. A arquitetura do Xen é baseada em uma camada de virtualização, localizada sobre o *hardware*, denominada Monitor de Máquina Virtual (VMM – *Virtual Machine Monitor*) ou hipervisor, como pode ser visto na Figura 1.1. Sobre o hipervisor executam os ambientes virtuais, chamados de máquinas virtuais, ou domínios não privilegiados (Domínio U), que acessam recursos de forma independente, como CPU, memória, acesso a disco e a rede. Cada ambiente virtual está isolado dos demais, isto é, a execução de uma máquina virtual não afeta a execução de outra máquina virtual as quais, inclusive, podem ter sistemas operacionais distintos. Há, ainda, um ambiente virtual privilegiado, denominado Domínio 0, que detém a exclusividade do acesso aos dispositivos físicos e, portanto, provê o acesso às operações de Entrada/Saída dos demais domínios e também executa operações de gerência do hipervisor. Já os demais domínios, referenciados como Domínio U ou domínios não privilegiados, não possuem acesso direto ao *hardware*. Sendo assim, os domínios desprivilegiados possuem dispositivos (*drivers*) virtuais, que se comunicam com o Domínio 0 para acessarem os dispositivos físicos.

1.3.1 Virtualização do Processador

A virtualização do processador físico (*Central Processing Unit* - CPU) é realizada através da atribuição de CPUs virtuais (vCPU) às máquinas virtuais. A vCPU é a CPU que os processos que estão sendo executados em uma máquina virtual podem acessar. Os processos da máquina virtual são escalonados pelo sistema operacional da máquina virtual para serem executados nas vCPUs atribuídas àquela máquina virtual [13, 27]. O mapeamento de vCPU em uma CPU real é realizado pelo hiper-

[2]A consolidação de servidores consiste em instalar diferentes servidores em máquinas virtuais isoladas hospedadas por uma mesma máquina física.

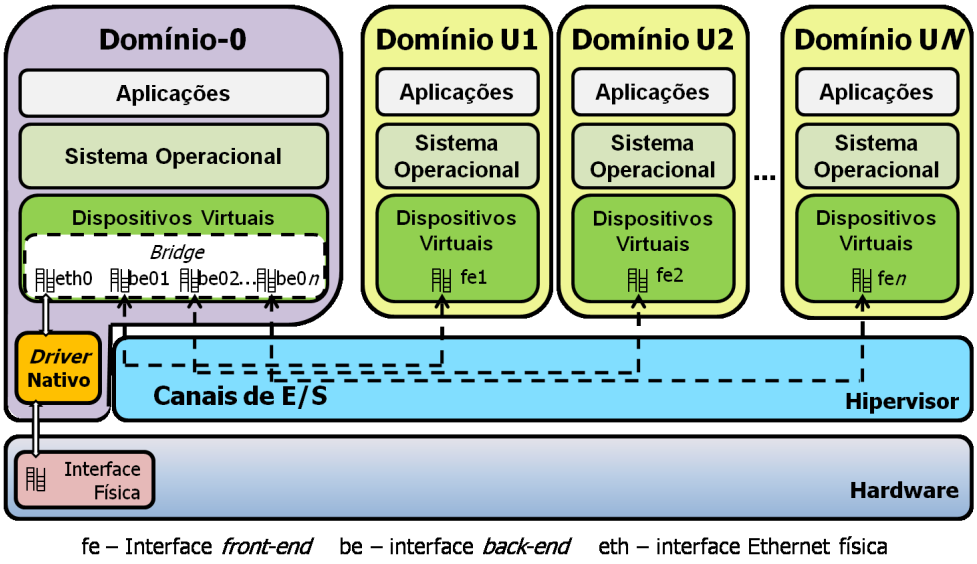


Figura 1.1: Arquitetura da plataforma de virtualização Xen.

visor do Xen. O hipervisor do Xen implementa um mecanismo de escalonamento dinâmico de vCPUs sobre as CPUs reais. O algoritmo padrão de escalonamento de CPU no Xen é o *Credit Scheduler* [28], ou escalonador por crédito, que gera um compartilhamento proporcional de CPU entre as vCPUs. Nesse esquema de escalonamento, os recursos de CPU são alocados para as vCPUs de acordo com pesos definidos para cada máquina virtual.

1.3.2 Virtualização da Memória

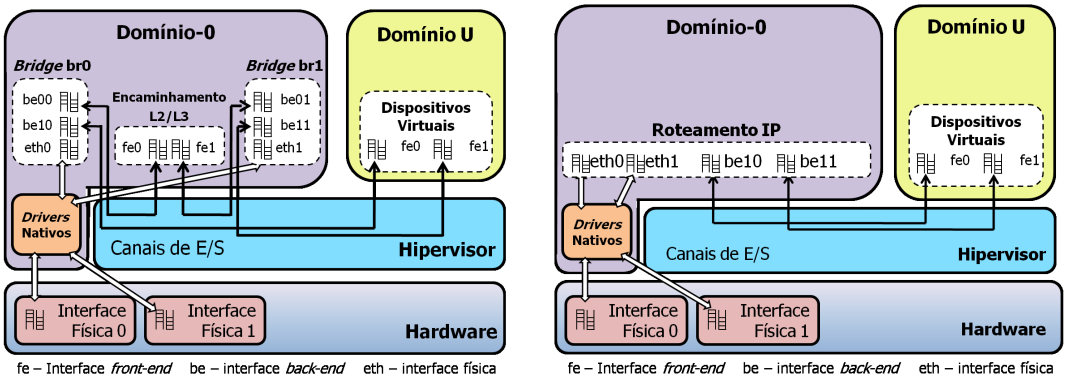
A virtualização do recurso de memória no Xen é estática. A memória RAM da máquina física é dividida entre as máquinas virtuais. Cada máquina virtual recebe uma quantidade fixa de memória no momento de sua instanciação. Atualmente é possível manipular a quantidade de memória que uma máquina virtual tem direito durante a execução da máquina virtual, dentro de um intervalo limitado de memória configurado na inicialização da máquina. A quantidade de memória a ser alocada de forma dedicada à máquina virtual deve respeitar valores de máximo e mínimo definidos em sua configuração inicial e devem, ainda, respeitar a disponibilidade de memória na máquina física. A virtualização da memória exige o mínimo de envolvimento do hipervisor. As máquinas virtuais são responsáveis pelas suas tabelas de páginas de memória. Todas as vezes que uma máquina virtual requisita uma nova tabela de páginas, ela a aloca e a inicializa em seu próprio espaço de memória e a registra no hipervisor Xen, o qual é responsável por garantir o isolamento. O isolamento é realizado pelo hipervisor de modo que cada máquina virtual, ou domínio, acesse apenas a área de memória reservada a ela.

1.3.3 Virtualização da Interface de Rede

A virtualização da interface de rede no Xen é feita demultiplexando os pacotes que entram pela interface física para os domínios não privilegiados e, de forma similar, multiplexando os pacotes que saem desses domínios para as interfaces físicas de rede. A virtualização das operações de entrada e saída nas interfaces de rede se dá da seguinte forma. Os Domínios Us possuem acesso a dispositivos virtuais de entrada e saída, que são controlados por dispositivos (*drivers*) virtuais que fazem requisições ao Domínio 0 para acessarem os dispositivos físicos. Ao contrário dos Domínios Us, o Domínio 0 tem acesso direto aos dispositivos de entrada e saída, através dos controladores de dispositivos (*drivers*) nativos. Dessa forma, ao receber uma requisição de um Domínio U, o Domínio 0 executa a requisição diretamente sobre o controlador de dispositivo (*driver*) nativo. A comunicação entre os dispositivos virtuais dos domínios desprivilegiados e o Domínio 0 é realizada através de uma dupla de interfaces: interface *back-end* e interface *front-end* [29]. Cada domínio desprivilegiado tem interfaces virtuais, chamadas *front-end*, que são utilizadas para todas as comunicações de rede. Essas interfaces virtuais são tratadas pelos sistemas operacionais dos Domínios Us como se fossem interfaces físicas reais. Para cada interface *front-end* criada nos domínios desprivilegiados, é criada uma interface *back-end* no Domínio 0. As interfaces *back-end* atuam como representantes das interfaces dos domínios desprivilegiados no Domínio 0. As interfaces *back-end* e *front-end* se comunicam através de um canal de E/S (*I/O Channel*). A troca de pacotes entre as interfaces dos domínios Us e o Domínio 0 é realizada através da passagem de referência de endereços de memória compartilhados e, portanto, sem cópia de memória. Um mecanismo empregado pelo canal de E/S remapeia a página física contendo o pacote no domínio de destino.

Por padrão, no Xen, a conexão das interfaces *back-end* e as interfaces físicas de rede podem ser realizadas de duas maneiras. A primeira, e também o padrão do Xen, é através do modo comutado (*Bridge*), mostrado na Figura 1.2(a). Nesse modo, são instanciadas pontes (*bridges*) no Domínio 0 e as interfaces *back-end* e as interfaces reais são associadas a elas. Uma ponte (*bridge*) é um comutador por *software*. Assim, o encaminhamento do pacote para interface *back-end* correta é realizado através do encaminhamento do pacote pela interface que responde ao endereço MAC de destino do pacote. Vale ressaltar que são necessárias tantas pontes (*bridges*) quantas são o número de interfaces físicas. O segundo modo de interconexão das interfaces reais e as *back-end* é o modo roteado (*Router*), mostrado na Figura 1.2(b). No modo roteado, o Domínio 0 passa a se comportar como um roteador. Dessa forma, a cada pacote que chega, o Domínio 0 verifica o endereço IP de destino e encaminha o pacote de acordo com as rotas definidas em suas tabelas de roteamento. Assim,

o encaminhamento do pacote para a interface *back-end*, ou física, correta, depende somente da definição correta da rota no Domínio 0 [29].



(a) Modo comutado (*Bridge*).

(b) Modo roteado (*Router*).

Figura 1.2: Virtualização do recurso de rede no Xen.

A virtualização da rede é alcançada no Xen através da instanciação de diversas máquinas virtuais, que correspondem aos elementos virtuais de rede, sobre um mesmo *hardware* físico. Um exemplo de virtualização de redes usando Xen é o caso em que os elementos de rede virtuais instanciados são roteadores virtuais. Nesse caso, como a camada de virtualização do Xen está abaixo dos sistemas operacionais, cada roteador virtual pode ter o seu próprio sistema operacional e cada um detém os seus próprios planos de dados e controle isolados dos demais roteadores. Nessa arquitetura de rede virtual, um roteador virtual pode ser instanciado, configurado, monitorado e desativado sob demanda. O roteador virtual pode, ainda, ser migrado, em funcionamento, usando o mecanismo de migração ao vivo do Xen [30].

1.4 Comutação OpenFlow

A ideia básica do OpenFlow é dividir a rede em dois planos, o de controle, responsável pela execução dos algoritmos de controle da rede, e o plano de dados, responsável pelo encaminhamento e pela aplicação de políticas a cada pacote. Para tanto, o OpenFlow explora o fato de que a maioria dos fabricantes de comutadores *Ethernet* e de roteadores implementa uma tabela de fluxos, que permite a execução de aplicações de *firewall*, *Network Translation Table* (NAT), qualidade de serviço (QoS – *Quality of Service*), e implementa também a coleta de estatísticas diretamente nos equipamentos [10]. No entanto, cada fabricante tem uma tabela de fluxos diferente. Assim, a proposta do OpenFlow é definir um conjunto de funcionalidades comuns que devem executar em comutadores e roteadores, definindo uma interface padrão de controle do plano de dados. Dessa forma, os comutadores e roteadores

da rede ficam responsáveis somente pelo plano de dados. Já o plano de controle é executado em outro nó, logicamente centralizado, que detém uma visão global da rede. O plano de controle centralizado executa os algoritmos de controle e decide as ações a serem tomadas pelo plano de dados. As informações são trocadas entre o plano de dados e o plano de controle segundo o protocolo OpenFlow.

O OpenFlow associa o encaminhamento eficiente, já que a função de encaminhamento é mantida em *hardware* especializado, com uma interface de controle simples e que permite o desenvolvimento de novas aplicações. Essa arquitetura permite que as funções da rede sejam definidas por aplicações expressas em *software*, enquanto o encaminhamento é realizado por *hardware* especializado. A função de encaminhamento, desempenhada pelo plano de dados, é executada por elementos especializados da rede que apresentam uma tabela de fluxos compartilhada. É através dessa tabela de fluxos compartilhada que o plano de dados é virtualizado. Já a função de controle, exercida pelo plano de controle, é centralizada em outro elemento da rede, o chamado controlador. O controlador executa funções de controle para a rede virtual.

1.4.1 O Protocolo OpenFlow

O OpenFlow é um protocolo simples que provê a comunicação de uma entidade de controle remota com o plano de dados de um comutador. O OpenFlow tem sido empregado em diferentes contextos e com diferentes funcionalidades [18, 23, 31, 32]. Nessa dissertação, o OpenFlow é apresentado como uma ferramenta capaz de se adaptar a diferentes ambientes, tais como: na virtualização de fluxos de uma rede local Ethernet, como uma plataforma básica de experimentação em redes de novas propostas e também como uma ferramenta de gerenciamento e uniformização do controle de redes.

A comunicação entre os elementos de rede e o controlador é definida pelo protocolo OpenFlow. A comunicação é estabelecida através de um canal seguro definido entre o controlador e cada elemento OpenFlow. O protocolo OpenFlow define funções para configurar e monitorar os elementos. O encaminhamento é definido com base em fluxos. Um fluxo é uma sequência de pacotes com um conjunto de características comuns. Quando um pacote chega ao elemento encaminhador, o elemento verifica se o pacote se adéqua a algum fluxo já definido. Em caso positivo, as ações definidas para aquele fluxo são aplicadas ao pacote. Em caso negativo, o pacote é encaminhado para o controlador, que extrai as características do fluxo, a partir do pacote, e cria um novo fluxo, introduzindo-o na tabela de fluxos do elemento OpenFlow. Uma das possíveis ações que o controlador pode definir para um fluxo é que ele siga o processamento normal, seja de comutação na Camada 2, seja roteamento na Camada 3, como se não existisse o protocolo OpenFlow. Essa funcionalidade

garante que a rede de produção execute em paralelo com redes experimentais, sem que estas afetem o funcionamento daquela. A comunicação entre os comutadores OpenFlow e o controlador centralizado ocorre através de um canal seguro baseado no protocolo *Secure Sockets Layer*). A Figura 1.3 mostra a organização de uma rede OpenFlow e a comunicação dos comutadores com o controlador.

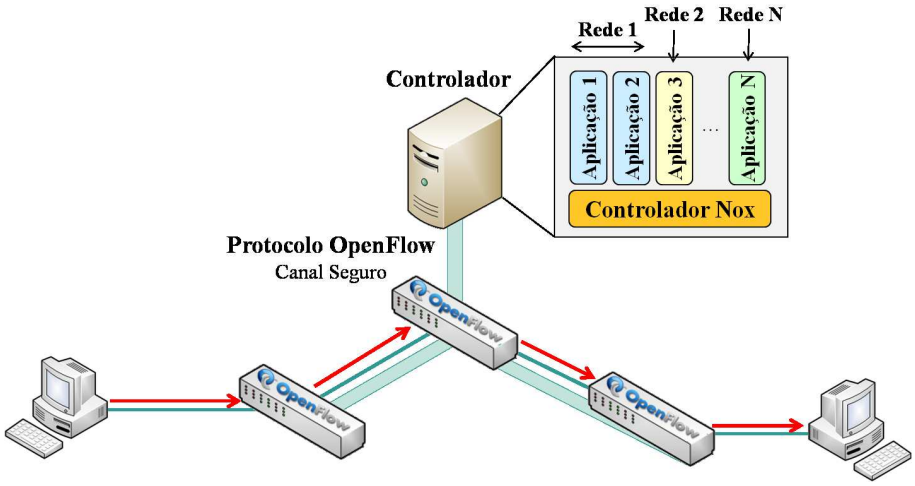


Figura 1.3: Arquitetura de uma rede OpenFlow. Os comutadores OpenFlow comunicam-se com o controlador através do protocolo OpenFlow em um canal seguro, usando o protocolo SSL (*Secure Sockets Layer*). O controlador executa as aplicações de controle de cada rede virtual.

As entradas na tabela de fluxo do OpenFlow são compostas por campos do cabeçalho dos pacotes, contadores e ações. Os campos do cabeçalho são a descrição do fluxo, ou seja, descrevem quais pacotes combinam com aquele fluxo. Esses campos formam uma tupla de doze elementos, que reúne características de pilhas de protocolos de várias camadas do pacote, como pode ser visto na Figura 1.4. No OpenFlow, a regra de encaminhamento de um pacote não se restringe ao endereço de rede, endereço IP, ou endereço físico, endereço MAC, do pacote. Os elementos dessa tupla podem conter valores exatos ou valores coringas, que combinam com qualquer valor que o pacote comparado apresente para o campo. O encaminhamento pode se dar por outras características do pacote, como por exemplo, as portas de origem e destino do protocolo de transporte. Um objetivo futuro do projeto OpenFlow é permitir a criação de campos definidos pelo usuário como um critério de encaminhamento. Dessa forma, será possível definir regras de encaminhamento com base em protocolos experimentais, como protocolos de uma arquitetura pós-IP.

A entrada na tabela de fluxos apresenta a descrição do fluxo, composta pelos campos extraídos do cabeçalho, como mostrado em Figura 1.4, e contadores para monitorar o fluxo. Esses contadores registram os dados referentes ao fluxo descrito, tais como, quantidade de *bytes* transmitidos, duração do fluxo e quantidade de pacotes transmitidos, em cada elemento de encaminhamento. Seguindo os contadores,

Figura 1.4: Definição de um fluxo em um comutador OpenFlow. Os campos que compõem a definição do fluxo são extraídos do cabeçalho do primeiro pacote que é enviado ao controlador.

a entrada na tabela de fluxos ainda tem as ações definidas para cada fluxo. O conjunto de ações relacionadas a cada fluxo é definido pelo controlador para cada elemento de encaminhamento. Esse conjunto define o tratamento dado a todos os pacotes que chegam ao elemento de encaminhamento e combinam com a descrição do fluxo. Essas ações incluem o encaminhamento do pacote em uma determinada porta de saída, mas, também, incluem outras ações como a modificação de campos no cabeçalho dos pacotes.

1.4.2 Componentes de uma Rede OpenFlow

Uma rede OpenFlow é composta basicamente por dois elementos: os comutadores OpenFlow, que realizam as funções do plano de dados; e o controlador OpenFlow, que executa as funções do plano de controle. Dentre os principais controladores da rede OpenFlow, um dos mais usados é o NOX, que é discutido nesta Seção. Existem outros controladores para os comutadores OpenFlow, como é o caso do SNAC [33], além de aplicações que agem como controlador [32, 34]. No entanto, o NOX possui propósito geral e permite que o administrador da rede desenvolva suas próprias aplicações de controle. Um terceiro componente da arquitetura OpenFlow é o FlowVisor. O FlowVisor é uma camada de virtualização do plano de controle, que gera uma abstração do substrato físico e, assim, permite a fragmentação (*slicing*) do substrato entre múltiplos controladores da rede OpenFlow.

Comutador OpenFlow

O comutador OpenFlow é o elemento encaminhador de pacotes que implementa o protocolo OpenFlow. Quando um pacote chega a um comutador, o comutador verifica se o pacote se adéqua a algum fluxo já definido. Em caso positivo, as ações definidas para aquele fluxo são aplicadas ao pacote. Em caso negativo, o pacote é encaminhado para o controlador, que extrai as características do fluxo, a partir desse pacote, e cria uma nova entrada na tabela de fluxos. A tabela de fluxos é uma estrutura de dados do comutador OpenFlow que relaciona a definição dos fluxos a ações a serem tomadas para pertencentes àquele fluxo. Associado a um fluxo, há ainda contadores que determinam o seu estado atual, tais como número de

bytes transmitidos, erros e a duração do fluxo. A ação associada a um fluxo define como o comutador deve tratar cada pacote daquele fluxo. A ação é definida pelo controlador. Uma das possíveis ações que o controlador pode tomar é processar o pacote como se não houvesse o OpenFlow, seja realizando comutação na camada de enlace, seja roteamento na camada de rede. Essa funcionalidade é a que garante que o OpenFlow seja transparente para o tráfego de produção que compartilha um mesmo substrato com redes experimentais. Outras ações que podem ser tomadas pelo controlador são o encaminhamento do pacote em uma dada porta de saída, o direcionamento do pacote a uma fila de prioridades ou a alteração de campos no cabeçalho dos pacotes encaminhados.

As entradas na tabela de fluxo do OpenFlow são compostas por campos do cabeçalho dos pacotes, contadores e ações. Os campos dos cabeçalhos formam a descrição do fluxo. Esses campos formam uma tupla de doze elementos, que reúne características dos protocolos de várias camadas do pacote. No OpenFlow, a regra de encaminhamento de um pacote não se restringe ao endereço de rede, endereço IP, ou endereço físico, endereço MAC. Os elementos dessa tupla podem conter valores exatos ou valores coringas, que se retornam verdadeira para qualquer que seja o valor a ele comparado. Como o fluxo contém informações desde a camada de enlace à camada de transporte, o encaminhamento pode se dar por outras características do pacote, como por exemplo, as portas de origem e destino do protocolo de transporte. Um objetivo futuro do projeto OpenFlow é permitir a criação de campos definidos pelo usuário como um critério de encaminhamento. Dessa forma, será possível definir regras de encaminhamento com base em protocolos experimentais, como os de arquiteturas pós-IP.

Controlador OpenFlow

O controlador é um elemento logicamente centralizado que executa os mecanismos de controle sobre a rede OpenFlow, configurando as tabelas de fluxo dos elementos encaminhadores. O controlador implementa o protocolo OpenFlow para se comunicar com os elementos encaminhadores e, através desse protocolo, envia os comandos de controle da rede. Um dos controladores OpenFlow mais usados é o NOX [35]. O NOX age como um sistema operacional de redes, sobre o qual executam aplicações desenvolvidas pelo administrador. O NOX provê as funções básicas de configuração e monitoramento da rede para as aplicações que executam sobre ele. Dessa forma, o controlador age somente como uma interface entre a rede e as aplicações, provendo uma implementação da interface entre o plano de dados e o plano de controle. Assim, o plano de controle é efetivamente exercido pelas aplicações que executam sobre o NOX. Dessa forma, uma rede virtual no OpenFlow é representada pelo seu conjunto de fluxos, plano de dados, e pelas suas aplicações

de controle, plano de controle.

Uma alternativa ao NOX é o controlador OpenFLow POX. O POX é um controlador OpenFlow desenvolvido seguindo o modelo de controlador do NOX e permite o desenvolvimento de aplicações de controle da rede OpenFlow [36]. A principal diferença entre o POX e NOX é que o POX é totalmente escrito em linguagem Python e, portanto, permite o desenvolvimento de aplicações OpenFlow multiplataformas e apresenta desempenho superior ao NOX clássico que era desenvolvido parte em Python e parte em C++.[3]

O NOX, por ser uma aplicação centralizada, apresenta problemas de escalabilidade e de confiabilidade, já que é um ponto único de falha. Portanto, existem algumas propostas para distribuir o controle de uma rede OpenFlow, como o HyperFlow [37], e para garantir a escalabilidade da rede, como o DIFANE [34].

O controlador distribuído HyperFlow [37] é implementado como uma aplicação do controlador NOX. A distribuição do controle é alcançada instanciando-se diversos controladores dentro de uma mesma rede OpenFlow. Cada instância do controlador NOX é, então, responsável por um conjunto de comutadores OpenFlow e cada comutador é controlador por apenas um dos controladores. Os controladores, por sua vez, executam o mesmo conjunto de aplicações de controle. Entre as aplicações que os controladores executam está a aplicação do HyperFlow. A aplicação HyperFlow é responsável por capturar os eventos gerados em cada partição da rede, controlada por uma dada instância do controlador NOX, e os divulga para os demais controladores da rede através do paradigma de *publish/subscribe*. Para realizar a divulgação, a aplicação HyperFlow usa o WheelFS [38]. O WheelFS é um sistema de arquivos distribuído projetado para fornecer armazenagem de dados para aplicações distribuídas. Com base nesse sistema de arquivos, o HyperFlow define um diretório como canal de divulgação de eventos e as mensagens de divulgação como arquivos. A partir de então, toda a tarefa de divulgação dos eventos é realizada pelo WheelFS, enquanto a o HyperFlow trata do recebimento e entrega dos eventos às aplicações NOX. O controlador HyperFlow proposto é resistente a partições da rede, pois, caso um determinado grupo de controladores perca o contato com outro grupo de controladores, cada grupo continua a exercer o controle restrito à sua rede e, quando as duas partições voltarem a se fundirem, os eventos de cada partição são publicados na outra, permitindo que as partições retornem a um estado de consistência global.

Uma proposta para prover escalabilidade ao controle de uma rede OpenFlow é a DIFANE (*DIstributed Flow Architecture for Networked Enterprises*) [34], em que o controle é centralizado, mas há modificações no plano de dados da rede OpenFlow que forma que o torne mais eficiente e escalável. Nessa proposta, o elemento controlador centralizado é responsável por gerar regras de alto nível, ao invés de manipular

[3]http://www.noxrepo.org/pox/about-pox/.

os pacotes em tempo real, como acontece em outros sistemas que implementam o OpenFlow. A abordagem de instalar regras de comutação sob demanda pode gerar alguns problemas como atrasos, necessidade de grandes *buffers* e maior complexidade nos comutadores para tratar um pacote que não se adéque a uma das regras já definidas. A DIFANE trata os pacotes, que não se adéquem a uma regra já definida, mantendo-os no plano de dados e reduzindo a quantidade de pacotes que não se adéquam a nenhuma regra, através da instalação de regras coringas ao invés de regras específicas. A arquitetura DIFANE consiste de um controlador que gera regras e as aloca em comutadores de autorização. Os comutadores de autorização são um subconjunto dos comutadores da rede. No momento em que um pacote chega a um comutador, esse comutador verifica se o pacote se adéqua a alguma regra já estabelecida. Em caso positivo, o pacote é encaminhado por essa regra. Em caso negativo, o pacote, ou sequência de pacotes, é encaminhado para o comutador de autorização mais próximo, identificado através de um protocolo de roteamento que executa entre os comutadores. A execução desse protocolo permite identificar também a mudança no estado de um enlace. Nos comutadores de autorização, estão definidas regras coringas geradas pelo controlador. As regras coringas são traduções das políticas aplicadas pelo administrador à rede. A tradução e a instalação das regras coringas, nos comutadores de autorização, é realizada pelo controlador. Quando um pacote é encaminhado por uma regra dos comutadores de autorização, esse instala uma nova regra nos comutadores de entrada do pacote.

FlowVisor: Virtualização do Plano de Controle

A virtualização do plano de controle em redes OpenFlow é feita pelo FlowVisor [39]. O FlowVisor é um controlador especial do OpenFlow, que funciona de forma transparente entre os dispositivos em uma rede OpenFlow e os outros controladores, como por exemplo Nox. O FlowVisor permite que mais de um controlador controle a rede OpenFlow, para tanto, o FlowVisor introduz o conceito de fatia. Cada fatia de rede é controlada por um controlador. As fatias são definidas por políticas no FlowVisor. As mensagens de controle trocadas entre os dispositivos e os controladores são intermediadas pelo FlowVisor, que verifica para cada mensagem quais políticas são aplicáveis e verifica, também, se um determinado controlador pode controlar um dado fluxo em um dispositivo.

A Figura 1.5 mostra que o FlowVisor intercepta as mensagens OpenFlow mandadas pelos controladores visitantes (1), compara-as com as políticas de fatiamento da rede (2) e, após, reescreve a mensagem. Esse procedimento propicia que o controle da fatia fique limitado ao controlador que está definido nas políticas do FlowVisor. O procedimento de envio de mensagens dos dispositivos para os controladores visitantes também é semelhante (4). O FlowVisor apenas encaminha as mensagens

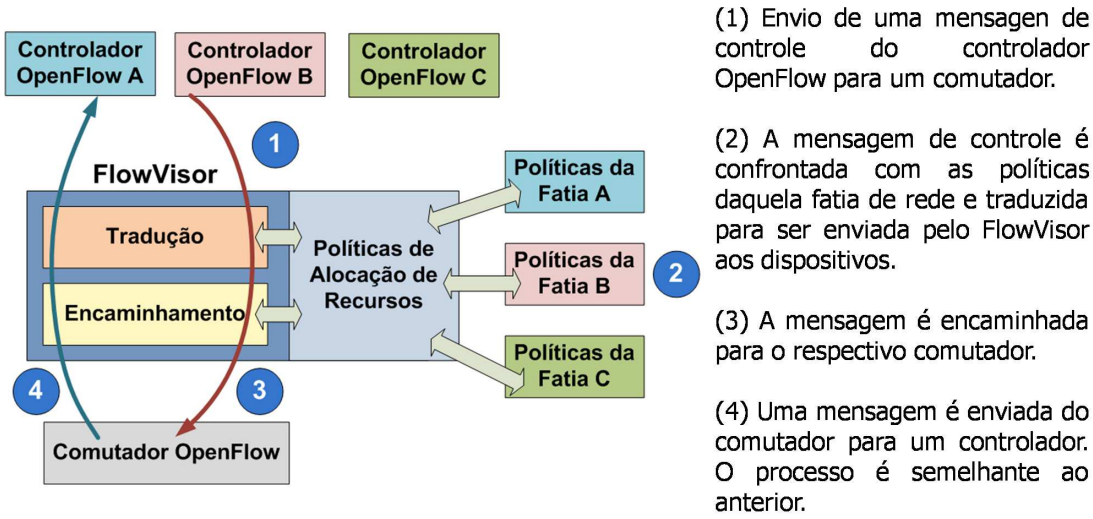


Figura 1.5: Arquitetura do FlowVisor, um controlador especial do OpenFlow, que permite a definição de redes virtuais.

dos dispositivos para os controladores visitantes se as mensagens dos dispositivos forem compatíveis com as políticas de cada fatia de rede que o controlada por um dado controlador. O fatiamento do plano de controle da rede é realizado de forma a manter cada fatia isolada das demais.

O FlowVisor executa a virtualização do plano de controle da rede, isolando o controle, mas compartilhando o plano de dados dos comutadores da rede OpenFlow. O fatiamento da rede realizado pelo FlowVisor é focado no compartilhamento de cinco recursos primitivos da rede [39]: isolamento de banda, descoberta de topologia, engenharia de tráfego, monitoramento de CPU e controle das tabelas de encaminhamento. O FlowVisor executa apenas a virtualização do plano de controle, permitindo que mais de um controlador troque mensagens de controle com um dispositivo OpenFlow. No entanto, cada controlador exerce o controle em uma fatia da rede e, nessa fatia, só um controlador exerce o controle. Dessa forma, o FlowVisor cria ilhas de controle em uma rede OpenFlow, de forma que o controle global da rede fica distribuído, enquanto o controle de cada fatia de rede fica centralizado em um controlador por fatia.

1.5 Trabalhos Relacionados

A tecnologia de virtualização permite a abordagem pluralista de diversas redes virtuais executando sobre um mesmo substrato físico. No entanto, o isolamento e a provisão de QoS em redes virtualizadas ainda são desafios para as principais plataformas de virtualização [3].

Nas redes atuais, os requisitos de qualidade de serviço são mapeados pelo ad-

ministrador da rede, a partir de uma descrição em alto nível, em recursos físicos disponíveis na rede [23]. No entanto, essa é uma tarefa muito complexa e fica sujeita a falhas do administrador. Outro ponto importante na provisão de qualidade de serviço é que os diversos serviços que compartilham uma mesma rede Ethernet convencional, a princípio, não estão isolados entre si, logo, o desempenho de um fluxo pertence a um dado serviço interfere nos demais fluxos dos demais serviços. Dessa forma, um meio de se prover QoS (*Quality of Service*), em um ambiente em que a infraestrutura é compartilhada, consiste na definição de fatias virtuais (*slices*) de rede. Dessa forma, cada equipamento da rede pode servir múltiplos serviços, criando fatias virtuais que podem prover o isolamento estrito do tráfego designado para tal fatia, sem possibilitar interferências com as demais fatias no mesmo equipamento.

Houidi *et al.* propõem um sistema adaptativo para o provimento de redes virtuais [40]. A ideia central é prover recursos sob demanda a redes virtuais, após a verificação da degradação do serviço prestado por cada rede virtual ou após a falha de algum recurso. Para tanto, o sistema utiliza um mecanismo multiagentes distribuído na infraestrutura física para negociar a requisição, realizar a adequação dos recursos e sincronizar os nós fornecedores e as redes virtuais. Outra proposta, o OMNI (*OpenFlow Management Infrastructure*) [41], também se baseia em um sistema multiagentes para prover qualidade de serviço a redes OpenFlow [10]. O OMNI é apresentado com mais detalhes no Capítulo 2. Entretanto, as ações tomadas pelos agentes OMNI restringem-se à migração de fluxos em uma rede OpenFlow. Kim *et al.* propõem um sistema de mapeamento de parâmetros de QoS em recursos disponíveis em comutadores OpenFlow, tais como filas e limitadores de taxa [23]. O objetivo desse sistema é prover QoS em cenários em que a infraestrutura física é compartilhada por redes virtuais com diferentes demandas. Contudo, o controle dos parâmetros de QoS e o mapeamento são centralizados no nó controlador da rede OpenFlow.

Uma outra forma de prover QoS a redes virtuais é apresentada por McIlroy e Sventek [22]. A proposta consiste em uma nova arquitetura de roteador, na qual o roteador é composto por diversas máquinas virtuais, chamadas *Routelets*. Cada *Routelet* é isolado dos demais e os seus recursos são garantidos e limitados. Nessa arquitetura, os fluxos sensíveis a QoS são roteados pelos *Routelets*, cuja prioridade de acesso a recursos do substrato define a qualidade de serviço de cada fluxo que encaminha. Contudo, o encaminhamento de pacotes nessa proposta é realizado pelas máquinas virtuais, o que limita o desempenho de encaminhamento dos *Routelets*.

Fernandes e Duarte [3, 24] propõem também mecanismos de controle e gerenciamento de redes virtuais. Os mecanismos propostos realizam o controle local, ou seja, restrito ao nó provedor do ambiente de virtualização Xen [42]. As propostas baseiam-se no isolamento dos recursos usados por cada ambiente virtual no domí-

nio de controle Xen e na diferenciação dos serviços providos por cada rede virtual. A proposta se restringe à plataforma de virtualização Xen e depende da marcação de pacotes. Paralelamente, outra proposta para prover a diferenciação de serviços em redes virtuais Xen é o *Xen Throughput Control* (XTC) [27]. A ideia chave do XTC é controlar a vazão de transmissão de dados alcançada por cada rede virtual controlando o quanto de processamento é dedicado a cada máquina virtual pelo escalonador de processamento do Xen. Esse controle é possível, pois a taxa de pacotes encaminhada por cada máquina virtual depende do quanto de processamento é dedicado a ela. A proposta XTC, no entanto, não realiza a separação de planos e depende que todos os pacotes sejam encaminhados exclusivamente pela máquina virtual. Nesse sentido, as propostas XNetMon, XNetMan e XTC não garantem o completo isolamento de tráfego entre redes virtuais, pois a separação do tráfego apenas se baseia na classificação de pacotes de acordo com o endereço da rede virtual a que se destinam. Assim, o isolamento se restringe a redes virtuais com espaços de endereçamento disjuntos. Wang *et al.* apresentam o OpenFlow como uma ferramenta de gerência de redes [43]. Nessa proposta, o OpenFlow é usado como uma plataforma de balanceamento de cargas, baseado em comutadores de baixo custo, que multiplexam requisições entre diferentes réplicas de um servidor. A solução proposta usa pesos para fragmentar o espaço de endereçamento IP dos clientes entre os servidores. Assim, de acordo com o IP do cliente, é possível identificar qual réplica deve atendê-lo. A proposta, no entanto, não garante a reserva de recursos, nem a qualidade de serviço dos fluxos.

Hao *et al.*, por sua vez, apresentam a infraestrutura VICTOR (*Virtually Clustered Open Router*) [32] que se baseia em criar um agrupamento de *datacenters*, através de uma infraestrutura de rede virtualizada. A ideia central dessa proposta é usar o OpenFlow como a infraestrutura básica da rede de *datacenters*, de modo a permitir que ao migrar uma máquina virtual de uma localização física para outra, seja possível reconfigurar os caminhos na rede. Essa proposta otimiza o uso da rede realizando migrações de servidores, porém não garante a qualidade de serviço dos fluxos e, também, não isola o uso de recursos decorrente dos diferentes servidores virtualizados.

Al-Fares *et al.* apresentam o desafio do gerenciamento de redes de centros de dados, em especial, no quesito de banda agregada disponibilizada às aplicações [44]. No cenário apresentado por Al-Fares *et al.*, os centros de dados necessitam realizar a interconexão de comutadores para atender a quantidade de máquinas que devem se comunicar. A interconexão de comutadores é necessária, pois nem mesmo os comutadores especializados (*high-end switches*) têm a densidade de portas para atender os cenários dos centros de dados. Dessa forma, a topologia das redes dos centros de dados é tipicamente constituída de uma árvore com múltiplas raízes,

com diversos caminhos com custos iguais entre pares de nós. Vale notar que nesses datacenters, muitas das aplicações, desde computação de pesquisas científicas até o processamento de consultas usando *MapReduce*, requerem uma grande banda de comunicação interna e, para tanto, a rede dos centros de dados devem fornecer uma grande quantidade de banda agregada. Assim, a fim de otimizar a banda agregada fornecida aos serviços dos centros de dados e aproveitar os caminhos de mesmo custos fornecidos pela topologia de árvores com múltiplas raízes, Al-Fares *et al.* propõem o sistema Hedera [44], um sistema dinâmico de escalonamento de fluxos para topologias de comutadores em múltiplas camadas de datacenters. O sistema Hedera coleta as informações de fluxos dos comutadores constituintes da rede, calcula caminhos não conflitantes para os maiores fluxos e roteia novamente o tráfego de acordo com os novos caminhos calculados. O principal objetivo do sistema é maximizar a utilização da rede e minimizar a sobrecarga de escalonamento e o impacto nos fluxos ativos. O sistema é capaz de identificar gargalos na rede, pois possui uma visão global das demandas de tráfego, sendo assim, o sistema é capaz de identificar gargalos que escalonadores locais que comutadores comuns não conseguem identificar. O sistema é projetado para suportar qualquer topologia geral de múltiplas raízes. Embora o sistema Hedera não dependa necessariamente do OpenFlow, um exemplo de implementação é usando a API OpenFlow para controlar os comutadores dos centros de dados. Nesse sentido, a opção pelo OpenFlow como ferramenta de gerência da rede relaciona-se com a facilidade de acesso às informações dos comutadores.

O TRILL (*Transparent Interconnection of Lots of Links*) é uma proposta para realizar o encaminhamento de pacotes na camada de enlace em redes de grande porte como as de Centros de Dados, com suporte a múltiplos caminhos e sem a necessidade de se definir uma árvore de cobertura [26]. O TRILL é um protocolo padronizado pelo IETF (*Internet Engineering Task Force*) e aplica técnicas de roteamento da camada de rede para interconectar comutadores Ethernets na camada de enlace [26]. A interconexão dos comutadores é feita de forma transparente para o protocolo de camada de rede como o IP. Os enlaces interconectados pelo TRILL são vistos pela camada de rede como uma única subrede IP, por exemplo. O TRILL encapsula, com um cabeçalho próprio, os quadros Ethernet a serem encaminhados. Como o quadro Ethernet é mantido inalterado, o TRILL garante a provisão das funcionalidades da camada de enlace, como VLAN (*Virtual Local Area Network*), e permite o uso de *multicast* e *broadcast*. A ideia central do TRILL é evitar a formação de grandes Árvores de Cobertura nas redes dos Centros de Dados e aproveitar a possibilidade de uso de múltiplos caminhos usando técnicas de roteamento em uma rede sobrecamada entre as camadas de enlace e rede. Já a arquitetura de redes para Centros de Dados DOVE (*Distributed Overlay Virtual Ethernet*) é uma proposta de virtualização de redes que provê isolamento entre redes virtuais, capacidade de

reconfiguração dinâmica da rede e de gerenciamento da rede virtual, independentemente da infraestrutura física preexistente no Centro de Dados [25]. A arquitetura DOVE permite a criação de redes virtuais com espaços de endereçamentos isolados e dinâmicas sobre uma infraestrutura física comum. O funcionamento do DOVE baseia-se no encapsulamento de pacotes e na consequente criação de uma rede de sobrecamada para permitir a separação entre rede virtual e a rede física subjacente.

Casado *et al.* apresentam o conceito de hipervisor de rede (*Network Hypervisor*), uma camada de *software* que se estende entre os elementos físicos da rede e gera a abstração de uma rede lógica única [31]. Com a ideia de hipervisor de rede, em vez de a interface controle interagir diretamente com o *hardware* de rede, o controle age sobre a camada de virtualização de rede. Esta abordagem permite que o estado da rede, encaminhamento e configuração, seja dissociada do *hardware* subjacente, permitindo a migração, aumentando a resiliência e possibilitando o gerenciamento mais complexo do estado da rede. A abordagem proposta se baseia na interface de programação de redes OpenFlow e é implementada como um controlador OpenFlow. A ideia do hipervisor de redes baseia-se, então, na alocação de comutadores lógicos distribuídos entre nós físicos e o isolamento entre redes virtuais é dado pelo mapeamento dos recursos virutais no recursos físicos. Contudo, a proposta do hipervisor de rede depende de uma visão global da rede física para realizar o mapeamento de recursos lógicos em recursos físicos, enquanto o sistema QFlow baseia-se em visões locais do estado da rede para garantir a escalabilidade do sistema.

Uma proposta de sistema híbrido de virtualização de redes, com foco no roteamento, é o RouteFlow [45]. O principal objetivo do RouteFlow é permitir que protocolos legados de roteamento exerçam o controle centralizado de uma infraestrutura de rede OpenFlow. O RouteFlow baseia-se em replicar a topologia física de uma rede OpenFlow em um ambiente virtual, pois a topologia da rede física composta por comutadores OpenFlow é repetida em um servidor de virtualização em que cada máquina virtual representa um comutador OpenFlow e executa o protocolo de roteamento. As alterações nas rotas que cada máquina virtual conhece são repassadas ao controlador OpenFlow que executa o encaminhamento de pacotes na rede física. Nesse ambiente virtual composto por máquinas virtuais, cada uma representa um comutador da rede física OpenFlow e executa protocolos de roteamento. O ambiente virtual se comunica com a rede OpenFlow através do controlador da rede. O controlador da rede, por sua vez, é um nó centralizado responsável por programar os planos de dados de todos os comutadores OpenFlow. No entanto, essa proposta não isola os recursos, nem define parâmetros de QoS para cada rede.

O XenFlow [18] é outro sistema híbrido de virtualização de redes que combina as ferramentas de virtualização de computadores Xen e a de programação de redes OpenFlow. O objetivo do XenFlow é a construção de um sistema em que os proto-

colos de roteamento são executados em máquinas virtuais Xen e o encaminhamento dos pacotes é realizado por um comutador OpenFlow. Uma das principais vantagens do XenFlow é a provisão da funcionalidade de migração ao vivo. No entanto, o sistema XenFlow não isola as redes virtuais no plano de encaminhamento dos pacotes, assim como não provê primitivas de qualidade de serviço para as redes virtuais. O XenFlow aplica o conceito de separação de planos [19], mas não realiza o isolamento de recursos ou da comunicação de redes virtuais.

Esta dissertação propõe o sistema QFlow. As principais inovações do QFlow em relação às propostas precedentes são o isolamento de recursos de cada rede virtual no plano de encaminhamento, a introdução de um mecanismo de controle de QoS e um algoritmo de redistribuição dos recursos excedentes de acordo com a prioridade e com o uso de cada rede virtual. O QFlow isola redes virtuais mesmo no cenário de separação de planos, sem a necessidade de criar uma rede sobrecamada, inserindo uma etiqueta de VLAN nos pacotes de cada rede virtual. A proposta estende o sistema XenFlow [18] e aplica o isolamento de recursos do sistema. A ideia central é prover o isolamento de recursos, garantindo a qualidade de serviço de cada rede virtual, e realizar o isolamento da comunicação de redes virtuais através de primitivas do plano de dados, como marcar a VLAN que um pacote pertence, para que seja possível realizar o roteamento em redes virtuais, isolando uma rede virtual das demais, sem que os pacotes dessa rede sejam tratados pelo plano de controle.

1.6 Organização do texto

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 descreve o desenvolvimento da infraestrutura de gerenciamento OMNI, desenvolvida para redes OpenFlow, e discute o desenvolvimento e os desafios da ferramenta de virtualização de redes XenFlow. O Capítulo 3 descreve o sistema QFlow de controle de recursos e garantia de qualidade de serviço e apresenta os resultados do protótipo desenvolvido e apresenta uma discussão sobre o encaminhamento de pacotes em difusão no sistema QFlow. Os resultados obtidos com a aplicação do encaminhamento no QFlow são apresentados no Capítulo 4. O Capítulo 5 conclui este trabalho e apresenta as direções de trabalhos futuros.

Capítulo 2

A Infraestrutura de Gerenciamento OMNI e a Comutação XenFlow

A interface de programação de aplicação (API – *Application Programming Interface*) OpenFlow permite a criação de redes de teste em paralelo com a rede de produção, utilizando equipamentos de rede comerciais [10]. Logo, o OpenFlow oferece suporte à inovação, permitindo o desenvolvimento de novos mecanismos de controle e o seu teste em ambientes reais [29]. Contudo, a utilização do OpenFlow requer a existência de ferramentas que simplifiquem o gerenciamento das redes e facilitem o desenvolvimento de novos mecanismos de controle.

Este capítulo apresenta a ferramenta *OpenFlow MaNagement Infrastructure* (OMNI) [41, 46], uma ferramenta para o controle e o gerenciamento de redes OpenFlow que possibilita o desenvolvimento de aplicações autônomas[1]. O sistema XenFlow de comutação também é apresentado.

[1] A ferramenta OMNI foi desenvolvida no "Projeto Horizon: Um Novo Horizonte para a Internet" que propõe uma arquitetura pluralista para a Internet do Futuro. Neste projeto de pesquisa projetou-se ferramentas para testar novas propostas de Internet do Futuro e desde o início o Grupo de Teleinformática e Automação (GTA) vislumbrou duas possibilidades de oferecer uma plataforma de teste de redes virtuais: com a plataforma de virtualização Xen e com a interface de programação de aplicação OpenFlow. Duas ferramentas foram implementadas no Projeto Horizon: VNEXT [47, 48] e OMNI [41, 49]. O autor desta dissertação de mestrado participa deste projeto desde seu início em 2009. A ferramenta OMNI é, portanto, resultado de um trabalho de equipe e trabalharam também na ferramenta OMNI, Callebe T. Gomes, Natalia C. Fernandes, Vitor T. da Costa, Lucas Henrique Mauricio, Filipe P. B. M. Barretto, Alessandra Y. Portella, Marcelo Duffles Donato Moreira, além dos professores Igor M. Moraes, Miguel Elias M. Campista, Luís Henrique M. K. Costa, e Otto Carlos M. B. Duarte. Deve-se ressaltar o trabalho do Leonardo Pais Cardoso que resultou na funcionalidade de gerenciamento autônomo através de agentes Ginkgo.

2.1 *OpenFlow MaNagement Infrastructure* - OMNI

A OMNI disponibiliza um conjunto de ferramentas de controle e gerenciamento, uma interface web para o administrador atuar sobre essas ferramentas, além de um sistema multiagentes para controlar autonomamente a rede. Entre as ferramentas passivas, destacam-se a ferramenta que coleta as estatísticas dos fluxos e a que coleta dados para obtenção da topologia física. Como ferramentas ativas, responsáveis pelas ações na rede, são disponibilizadas uma interface para a criação, modificação e remoção de regras de encaminhamento para fluxos e também uma interface para migração de fluxos, que permite ao administrador da rede alterar o caminho físico de um fluxo. Ressalta-se que a facilidade de migração proposta garante uma migração de fluxo sem perdas de pacotes. Graças a essa funcionalidade, o administrador pode fazer manutenção preventiva dos seus comutadores ou utilizar técnicas de computação verde para reduzir o número de comutadores ativos. A OMNI também oferece a funcionalidade de gerenciamento autônomo, implementado através de agentes criados com a plataforma Ginkgo [50]. Assim, a ferramenta possui uma interface para que agentes programados pelo administrador possam atuar sobre a rede. Para demonstrar a facilidade de gerenciamento autônomo, foi desenvolvido um agente para detectar problemas no encaminhamento de pacotes e garantir que as redes virtuais não utilizem os enlaces com alta taxa de perda de pacotes. Os testes com a migração mostram que essa funcionalidade apresenta baixo tempo de resposta e é realizada sem causar perdas de conexão ou de pacotes. Experimentos realizados também mostram que os agentes da ferramenta desenvolvida apresentam um baixo tempo de resposta para detecção de falhas relevantes e tomada de atitude para solucionar o problema. Tais resultados obtidos com a ferramenta serão demonstrados com o auxílio de uma aplicação de transferência de vídeo.

2.1.1 As Aplicações do NOX para Gerenciamento da Rede OpenFlow

O principal objetivo da ferramenta OMNI é permitir o controle e o gerenciamento de uma rede OpenFlow. As funções de controle e gerenciamento são desempenhadas por aplicações que executam sobre o NOX, que é um controlador que implementa o protocolo OpenFlow e, assim, age como uma interface entre as aplicações de controle e a rede.

A ferramenta proposta se baseia no conceito de serviços web, permitindo, assim, o controle e o gerenciamento remotos. Logo, uma das aplicações NOX que compõe a ferramenta é um provedor de serviços web que se comunica com as demais aplicações

e exporta as interfaces dessas aplicações como serviços web. Foi desenvolvido também um servidor que disponibiliza uma interface web que apresenta as funções de controle de forma amigável para o administrador da rede. Esse servidor se comunica com o NOX através da aplicação provedora dos serviços web para fazer requisições de serviços na rede. Outra aplicação que acessa os serviços web providos pelo NOX são os agentes responsáveis pelo controle autônomo da rede. Um exemplo de uso da ferramenta proposta é apresentado na Figura 2.1. Essa figura ilustra uma rede OpenFlow genérica hospedando duas redes virtuais, cada uma com o seu respectivo controlador. Cada nó da rede hospeda um agente da OMNI, os quais interagem entre si. É apresentado também o conjunto de aplicações para o NOX propostas e desenvolvidas pelos autores, assim como as aplicações estendidas e as aplicações apenas modificadas e adaptadas.

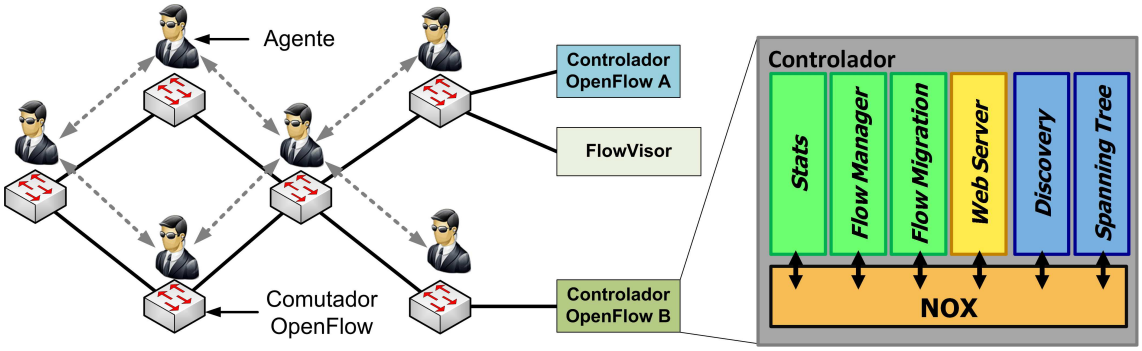


Figura 2.1: Esquema com a utilização da ferramenta OMNI em uma rede OpenFlow com duas redes virtuais.

Para desenvolver a interface de gerenciamento, foram criadas aplicações para o NOX, além de terem sido estendidas algumas das aplicações já existentes, todas escritas em linguagem Python. As aplicações passivas, que executam o sensoriamento da rede, incluem o monitoramento dos fluxos e dos comutadores, através da aplicação *Stats*, e a descoberta de topologia, através da aplicação *Discovery*. As aplicações ativas, que atuam na rede, incluem a *Spanning Tree*, que calcula a árvore de cobertura da rede, a *Flow Manager*, que faz o gerenciamento de fluxos, e a *Flow Migration*, que realiza a migração de fluxos. Todas essas aplicações foram criadas ou modificadas para fornecerem uma saída em *eXtensible Markup Language* (XML) como resultado da ação requisitada. Optou-se pelo uso do XML para que os dados gerados por uma aplicação fossem interpretados de forma simples por outras aplicações, pelos agentes e pelo servidor da interface gráfica. Existe ainda a aplicação *Web Server* que disponibiliza um serviço para fazer chamadas a todas as aplicações de gerenciamento.

Mattos *et al.* [41, 46] propuseram e desenvolveram a aplicação inovadora *Flow Migration* que permite a migração de fluxos sem perdas, uma vez que, por padrão,

essa funcionalidade não é implementada pelo NOX. Mattos *et al.* [41, 46] também propuseram e desenvolveram a aplicação *Flow Manager* que é uma simples interface para chamar as funções para criação e remoção de fluxos disponibilizadas pelo NOX. As aplicações *Stats* e *Web Server* se basearam em um código disponibilizado pelo NOX, mas receberam extensões de funcionalidades significativas. Por fim, as aplicações *Discovery*, disponibilizada pelo NOX, e *Spanning Tree*, obtida no sítio do OpenFlow, foram modificadas para funcionar com diferentes dispositivos físicos, como computadores e roteadores Linksys, e para fornecerem uma saída em XML. As aplicações *Flow Migration*, *Stats* e *Web Server* são descritas com detalhes a seguir.

A Aplicação *Flow Migration*

A aplicação *Flow Migration* realiza a migração de fluxos entre comutadores de uma rede OpenFlow. O objetivo da migração é fazer com que um fluxo que esteja passando por uma determinada sequência de comutadores seja reconfigurado para passar por outra sequência de comutadores. Como o controle da rede é centralizado e o controlador possui acesso às tabelas de fluxos de todos os comutadores, a migração consiste na reconfiguração das tabelas de fluxos dos comutadores. É importante ressaltar que pacotes não são perdidos durante a migração de fluxos, porque o algoritmo de reconfiguração de tabelas proposto estabelece uma determinada ordem para executar as reconfigurações dos comutadores que evita as perdas de pacotes. Esta astúcia permite uma característica muito importante que é a migração sem perdas.

Primeiramente, o algoritmo identifica o caminho atual do fluxo na rede. Em seguida, verifica se há conectividade direta entre todos os comutadores definidos como integrantes do novo caminho do fluxo. O novo caminho é estabelecido caso haja conectividade direta entre os comutadores informados. Caso contrário, o caminho entre os comutadores que não são diretamente ligados é calculado utilizando o algoritmo de Dijkstra, de modo a ser estabelecido um caminho completo com o menor número de saltos. Com base no caminho calculado, o fluxo é adicionado aos comutadores do novo caminho a partir do comutador de saída do fluxo até o de entrada. Nesse último comutador, o controlador, ao invés de adicionar um novo fluxo, modifica o fluxo já definido, de modo que a antiga porta de saída seja modificada para uma nova porta que redireciona o fluxo ao restante do caminho já configurado. Após o encaminhamento do último pacote em trânsito pelo caminho original, o OpenFlow apaga o caminho original do fluxo, pois, por padrão, o OpenFlow apresenta um temporizador associado a cada regra de encaminhamento para apagar a regra quando esta fica sem uso. Assim, o procedimento de migração proposto evita a perda de pacotes.

A Aplicação *Stats*

A aplicação *Stats* é responsável pela obtenção de estatísticas dos comutadores OpenFlow e pela conversão dessas informações para XML, para que elas possam ser disponibilizadas através do serviço web. Essa aplicação envia pedidos para cada um dos comutadores OpenFlow, que respondem com um relatório contendo a sua descrição, suas estatísticas e as estatísticas de cada uma de suas tabelas de fluxos.

A aplicação *Stats* é uma extensão da aplicação *Switchstats* disponibilizada pelo NOX. A aplicação *Switchstats* faz requisições aos comutadores para fornecer a descrição dos comutadores, o número de pacotes recebidos e perdidos por porta do comutador e também por tabela do comutador. Contudo, essa aplicação não disponibiliza informações sobre os fluxos, pois o NOX não implementava o envio das requisições de estatísticas de fluxos para os comutadores. Assim, a aplicação e o NOX foram estendidos para requisitarem as informações sobre cada um dos fluxos instanciados e também sobre os fluxos agregados, tais como: a descrição de cada fluxo, o número de pacotes encaminhados, o número de pacotes perdidos e o tempo de duração do fluxo, o que permite calcular as taxas de encaminhamento e de perda de pacotes. Em relação aos fluxos agregados, são requisitados o número de pacotes e de bytes encaminhados por todos os fluxos de um comutador. Esses dados adicionais são fundamentais para o gerenciamento e para dar suporte à migração de fluxos.

A Aplicação *Web Server*

A versão original do NOX fornece um arcabouço para o desenvolvimento de um servidor web que interage com as demais aplicações. A partir desse arcabouço, podem ser desenvolvidos recursos[2] que implementam sítios ou aplicações sobre o protocolo HTTP. A ferramenta OMNI implementa a aplicação *Web Server* como um recurso nesse arcabouço para prover serviços web de controle da rede. Essa aplicação recebe chamadas de funções sob a forma de URL, as trata e retorna o resultado em mensagens XML. A mensagem XML retornada por cada função pode ser tratada e interpretada por outras aplicações, como as aplicações de interface com o usuário ou de controle autônomo da rede.

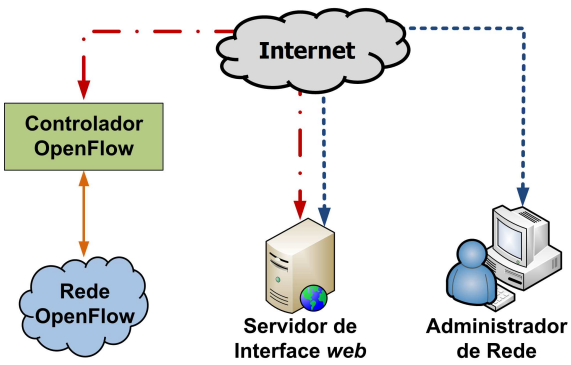
A aplicação *Web Server* interage diretamente com as demais aplicações através de funções internas do próprio controlador NOX. Assim, uma chamada remota à aplicação *Web Server* por uma URL é convertida por tal aplicação em uma chamada de função de outra aplicação que esteja em execução sobre o NOX.

[2]Nomenclatura usada pelo arcabouço do NOX.

2.1.2 A Interface Web

A interface web foi desenvolvida para permitir o acesso amigável do administrador da rede ao monitoramento dos comutadores e para permitir a execução de funções básicas de gerenciamento. A interface web de controle da rede OpenFlow foi criada como um cliente dos serviços web fornecidos pelas aplicações do NOX. A Figura 2.2(a) ilustra a página inicial da interface e a Figura 2.2(b), a comunicação entre o servidor de interface web, o controlador NOX e o administrador da rede.

(a) Página inicial do sítio da ferramenta OMNI.



(b) Comunicação entre o administrador e o controlador através da interface web.

Figura 2.2: Interface gráfica com acesso remoto da ferramenta OMNI.

O aplicativo que implementa a interface web executa tanto um cliente quanto um servidor HTTP. O fluxo de dados nesse aplicativo ocorre da seguinte forma. Ao entrar no sítio provido pela interface web, o administrador tem acesso a um menu com todas as funções gerenciais. Ao acessar uma opção que dependa de dados

da rede OpenFlow, como a opção de estatísticas dos comutadores, o aplicativo da interface web envia uma requisição HTTP ao servidor web do NOX, que trata a requisição e retorna uma mensagem XML. Essa mensagem é recebida pela aplicação de interface web que processa os dados para gerar a visualização. Conforme o administrador interage com a página e executa funções de controle sobre a rede, esses comandos são convertidos em requisições URLs e são enviados para o controlador. Então, o controlador encaminha as requisições para a aplicação pertinente e retorna à interface web a confirmação, ou erro, da execução da função, também sob a forma de uma mensagem XML.

2.1.3 O Sistema Multiagentes

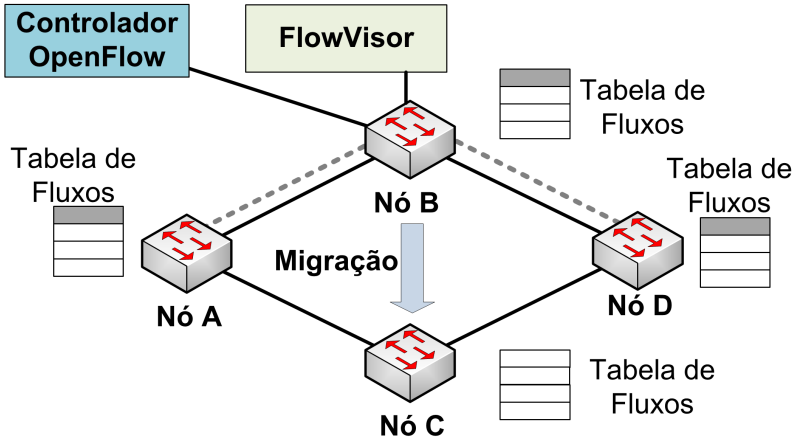
Com base no serviço web disponibilizado através do NOX com a aplicação *Web Server*, é possível desenvolver mecanismos de controle autônomos usando uma plataforma de agentes, tal como a plataforma Ginkgo [50] utilizada na OMNI. Para exemplificar essa funcionalidade, foi especificado um sistema multiagentes[3] que permite a detecção de falhas no encaminhamento em um determinado nó e, após a detecção, o agente orquestra a migração dos fluxos de todas as redes virtuais que estão utilizando tal nó físico. O agente desenvolvido executa em algum nó da rede ou em um *middle box*, caso o equipamento comercial não dê suporte para o agente. O agente consulta o controlador para saber a taxa de pacotes que está sendo perdida pelo nó que ele monitora e armazena a informação em sua base de conhecimento. O agente então se comunica com os demais agentes presentes na rede, troca a informação armazenada e, caso conclua que algum dos enlaces está com a taxa de perdas acima de um limiar predefinido e acima da taxa dos demais enlaces, o que indica algum tipo de falha no compartilhamento do enlace, o agente envia uma ordem para o controlador através do serviço web para que todos os fluxos daquele nó sejam migrados para outros enlaces com maior capacidade.

2.1.4 Avaliação da Ferramenta

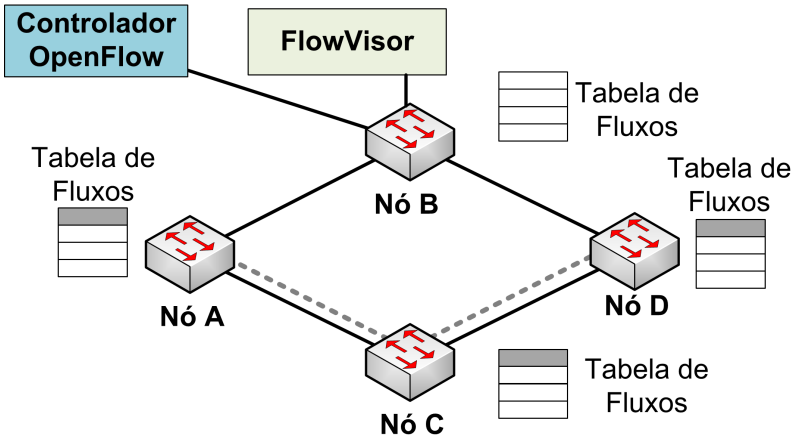
A OMNI foi avaliada para verificar o tempo de resposta e a correção das suas funções. A seguir, são descritos alguns resultados da execução dos testes de migração e atuação do agente na rede do GTA. Resultados semelhantes são apresentados durante a demonstração, com o auxílio de uma aplicação de vídeo para mostrar variações de vazão na rede. Para realizar os testes, foi montada uma rede OpenFlow com computadores pessoais. Essa rede possui quatro comutadores OpenFlow, um FlowVisor e um controlador NOX, como mostrado na Figura 2.3. Os comutadores

[3]Esta aplicação foi desenvolvida principalmente pelo bolsista de iniciação científica Leonardo Pais Cardoso.

OpenFlow e o FlowVisor são executados em máquinas com processador Intel Core 2 Duo com 2 GB de memória. O controlador é executado em uma máquina com processador Intel I7 com 4 GB de memória. Nessa mesma máquina, também são instanciados um agente Ginkgo para cada um dos comutadores OpenFlow.



(a) Rede antes da migração do fluxo de dados que possui o caminho A-B-D.

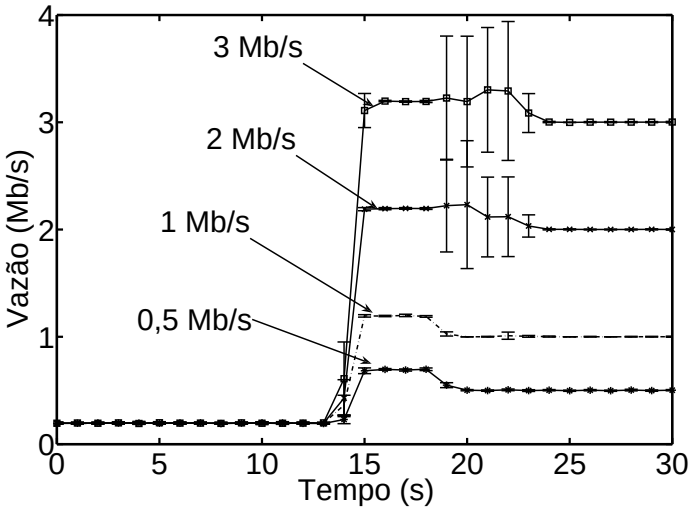


(b) Rede após a migração do fluxo de dados para o caminho A-C-D.

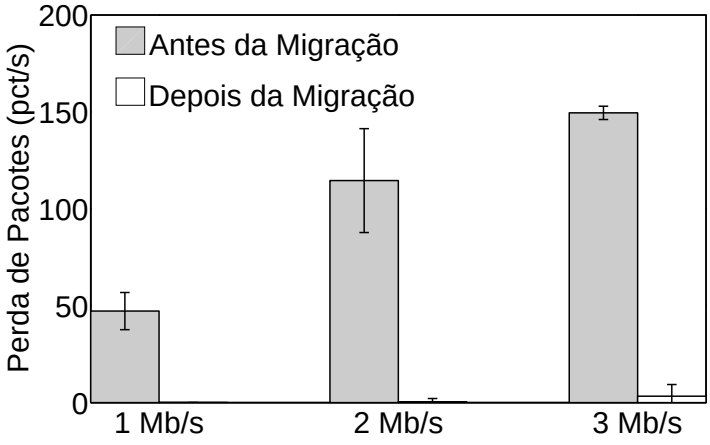
Figura 2.3: Migração do fluxo que passa no caminho A-B-D para o caminho A-C-D.

O primeiro teste mostra a migração de um fluxo que passa pelo caminho formado pelos comutadores A, B e D para o caminho formado pelos comutadores A, C e D. A taxa do fluxo foi variada entre 0,5 e 3 Mb/s. Nesse cenário, o enlace A-B está limitado pelo FlowVisor em uma taxa de 200 kb/s, enquanto que o enlace A-C está limitado a 100 Mb/s. Assim, pelo caminho original, existem perdas de pacotes para taxas superiores a 200 kb/s. Após a migração para o enlace com maior banda, não existem mais perdas de pacotes. Isso é visualizado na demonstração através da transmissão de um vídeo entre a fonte, ligada ao comutador A, e o destino, ligado ao comutador D. A Figura 2.4(a) mostra o resultado desse teste, que foi bem sucedido e não apresentou perdas de pacotes devido à migração. A migração foi executada em

um tempo médio de 419 μs com um desvio padrão de 210 μs, o que mostra o baixo tempo de resposta dessa funcionalidade na OMNI. Como se pode observar, após a migração para o enlace com maior capacidade, a taxa de transmissão se estabilizou no valor da taxa nominal de envio.



(a) Vazão do fluxo de dados após a migração manual do fluxo selecionado.



(b) Perdas de pacotes antes e depois do disparo da ação do agente.

Figura 2.4: Resultados dos testes de migração de fluxo utilizando a chamada manual e a chamada pelo agente.

O segundo teste mostra a migração pelo sistema multiagentes. Nesse caso, é variada a taxa de transmissão do fluxo e mede-se a perda de pacotes até que o agente dispare autonomamente a migração do fluxo. O cenário utilizado é o mesmo do teste com a chamada manual da migração. Os resultados desse teste estão na Figura 2.4(b), que mostra que o agente consegue detectar o enlace sobrecarregado corretamente e migra o fluxo de forma a evitar ou reduzir a taxa de perdas de pacotes para valores próximos de zero. Como o agente observa a taxa de perdas

de pacotes no enlace e assume um limiar de perda de 20 pacotes/s maior que nos enlaces observados pelos outros agentes, então não houve variação do tempo para o agente iniciar a migração independente da taxa de envio. O agente desenvolvido faz observações na rede com intervalos de 10 s, e assim, teve um tempo médio para disparar a migração de 29,4 s. Esse tempo pode ser reduzido ao configurar medições mais frequentes pelo agente na rede. Na demonstração ao vivo, são utilizados vídeos com taxas diferentes, de tal forma que o tempo para disparar as migrações irá depender da qualidade do vídeo e da variação ao longo do tempo da taxa do vídeo.

2.2 XenFlow: Um Sistema Híbrido de Virtualização de Redes

Esta seção apresenta o XenFlow[4], um sistema de virtualização híbrido baseado nas plataformas Xen e OpenFlow. O XenFlow garante o suporte à primitiva de migração em um ambiente que facilita o desenvolvimento de inovações. O XenFlow também oferece tanto a opção de controle distribuído ou de controle centralizado da rede. Assim, o XenFlow disponibiliza ferramentas para um remapeamento das topologias virtuais, realizando tanto migração de roteadores virtuais, quanto a migração de enlaces virtuais. No XenFlow, as funções do roteador virtual são divididas em dois planos, o plano de controle e o plano de dados. O plano de controle, que é executado dentro da máquina virtual criada com a plataforma Xen, é responsável por todas as tarefas de controle, como a atualização da tabela de roteamento. Por outro lado, o plano de dados é criado utilizando-se o OpenFlow que é responsável pelo encaminhamento dos pacotes.

A vantagem da utilização da plataforma de virtualização Xen é a criação de um ambiente de virtualização de redes com controle distribuído e com amplo suporte à inovação, pois as máquinas virtuais (roteadores virtuais) são isoladas umas das outras e o administrador da rede virtual pode desenvolver seus mecanismos de controle sem restrições. A vantagem da utilização do OpenFlow é que, por ser um comutador programável, a migração de fluxos é trivial e, portanto, a migração dos enlaces virtuais torna-se uma atividade trivial, permitindo o remapeamento simples de topologias lógicas sobre a topologia física. Além disso, o encaminhamento com base

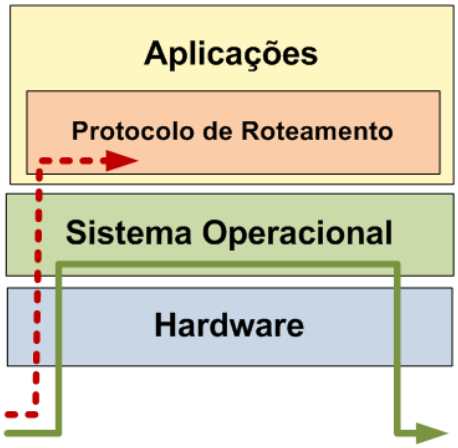
[4]O sistema XenFlow foi inicialmente proposto como Projeto de Final de Curso do autor desta dissertação. Durante o mestrado, no entanto, o sistema evoluiu. As principais evoluções do XenFlow no período foram: i) a troca do encaminhamento no espaço do usuário para o encaminhamento de pacotes pelo módulo do *kernel* Open vSwitch; ii) o desenvolvimento das aplicações de comunicação de atualização da tabela de rotas com SSL; e, iii) a correção do mecanismo de encaminhamento de pacotes em difusão. Além destes melhoramentos, a evolução do XenFlow para garantir o isolamento de recursos, a provisão de Qualidade de Serviço e o controle de filas culminaram na proposta do sistema QFlow que é o foco desta dissertação.

em recursos OpenFlow garante o uso de mecanismos de encaminhamento especialmente desenvolvidos para essa função, o que contribui para um maior desempenho no encaminhamento de pacotes no XenFlow. No XenFlow, cada máquina física está associada a um comutador OpenFlow que liga as máquinas virtuais Xen ao resto da rede e cada máquina virtual Xen funciona como um controlador do comutador.

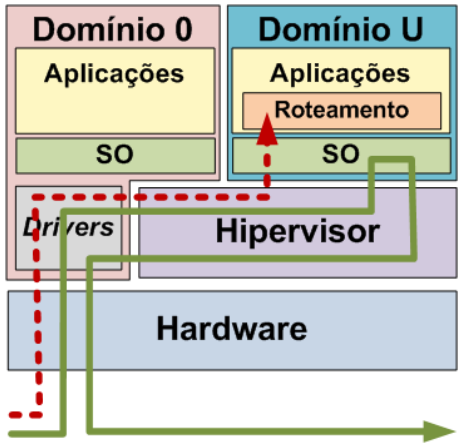
As duas principais vantagens do XenFlow são: a realização da migração de topologias virtuais sem perdas de pacote, herdada da característica de separação de planos do OpenFlow, podendo mapear um enlace lógico sobre um ou mais enlaces físicos, e a possibilidade de fazer o controle distribuído da rede, tal qual nas redes convencionais, herdada do Xen. O XenFlow é inovador em relação ao alcance da migração de um roteador virtual seja aumentado em relação a outras propostas na literatura [19, 20, 30], que se restringem ao nó físico de destino ter os mesmos vizinhos que o nó físico de origem. A facilidade provida pelo XenFlow de mapear um enlace lógico sobre um ou mais enlaces físicos elimina essa restrição. Outra vantagem do sistema proposto é o controle distribuído da rede, que é realizado pelos protocolos de roteamento executando em máquinas virtuais.

2.2.1 Roteamento de Pacotes em Computadores Pessoais

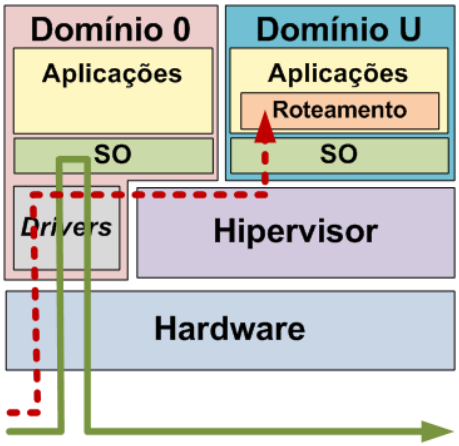
As plataformas de virtualização de computadores pessoais, tal como o Xen, apresentam um enorme sucesso na consolidação de servidores. No entanto, a virtualização das funções de entrada e saída (E/S) ainda são um desafio para essas plataformas. Na virtualização de redes as funções de E/S são fundamentais para se implementar os roteadores virtuais. Esta seção detalha os problemas de comutar pacotes na plataforma Xen. A Figura 2.5 apresenta uma comparação entre quatro formas distintas de se implementar um roteador virtual em uma plataforma de computadores pessoais. A Figura 2.5(a) apresenta o funcionamento básico de um computador pessoal agindo como roteador. Nesse caso, o protocolo de roteamento é implementado como uma aplicação que executa sobre o sistema operacional. A aplicação de roteamento configura políticas no *kernel*, núcleo, do sistema operacional para realizar o encaminhamento dos pacotes diretamente pelo núcleo. Em um cenário virtualizado, Figura 2.5(b), o encaminhamento dos pacotes ocorre dentro de uma máquina virtual. Assim, a virtualização das operações de entrada e saída (E/S) adiciona a sobrecarga, na qual todos os pacotes devem passar pelo Domínio 0, que detém o acesso exclusivo aos dispositivos de E/S, e também pelo hipervisor, a camada responsável pela virtualização. Uma alternativa para evitar a sobrecarga gerada pela virtualização de dispositivos de E/S é a técnica de separação de planos. Essa técnica prevê a execução normal do protocolo de roteamento na máquina virtual, mas o processo de encaminhamento dos pacotes é realizado no Domínio 0, como



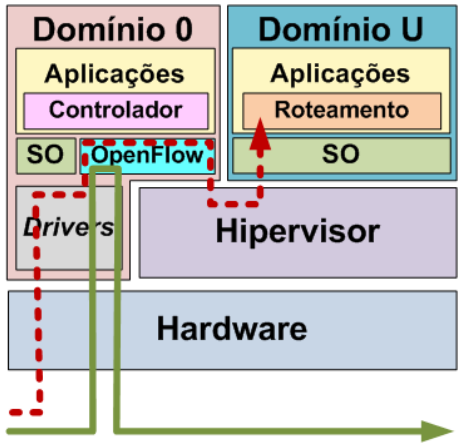
(a) Roteamento usando um computador pessoal sem virtualização. Os pacotes de controle são entregues ao protocolo de roteamento que executa na camada de aplicação. Os pacotes de dados são encaminhados diretamente pelo *kernel* do sistema operacional.



(b) Roteamento através de uma máquina virtual Xen sem separação de planos. Os pacotes de controle são enviados para a máquina virtual e os de dados são encaminhados pela a máquina virtual.



(c) Roteamento através de uma máquina virtual Xen com separação de planos. Os pacotes de controle são enviados para a máquina virtual. Os pacotes de dados são encaminhados diretamente pelo Domínio 0.



(d) Roteamento através do sistema XenFlow. Os pacotes de dados e de controle são encaminhados por um comutador OpenFlow no Domínio 0. Os pacotes de controle são enviados para o roteador virtual a que se destinam e os pacotes de dados são encaminhados diretamente para a interface de saída.

Figura 2.5: Comparação entre quatro propostas de roteamento usando computadores pessoais. As setas tracejadas indicam os fluxos dos pacotes de controle dos protocolos de roteamento. As setas contínuas indicam os fluxos de pacotes encaminhados por cada configuração de roteador.

mostrado na Figura 2.5(c). Nesse caso, o encaminhamento é realizado pelo sistema operacional do Domínio 0, que possui uma cópia das regras de encaminhamento da máquina virtual. Já no sistema XenFlow, o roteamento dos pacotes é realizado de outra maneira, como mostrado na Figura 2.5(d). No XenFlow, tanto o encaminha-

mento dos pacotes de dados para a porta de saída correta, quanto a entrega dos pacotes de controle para as máquinas virtuais de destino, são realizados por um comutador OpenFlow instanciado no Domínio 0. O comutador OpenFlow, por ser programável, oferece maior flexibilidade ao roteamento do que as demais propostas de roteamento apresentadas, já que permite que os pacotes sejam encaminhados de acordo com os seus campos dos cabeçalhos de Camada 2, Camada 3 ou superiores.

2.2.2 Encaminhamento de Pacotes no Xen

O sistema de virtualização Xen permite, por padrão, conforme mostrado no Capítulo 1, três modos principais para realizar o encaminhamento de pacotes entre as interfaces das diferentes máquinas virtuais e as interfaces da máquina física: o modo *bridge*, o modo *router* e o modo NAT (*Network Address Translation*). Esses mecanismos multiplexam (comutam ou roteiam) o tráfego de saída dos pacotes originados pelas diferentes máquinas virtuais e demultiplexam (comutam ou roteiam) o tráfego de chegada de pacotes de uma (ou diversas) interface de entrada E/S com destino às diferentes máquinas virtuais [51]. Os dois principais mecanismos discutidos nesse trabalho são o modo *bridge* e o modo *router*.

A Figura 2.6 compara os modos *router* e *bridge* do Xen. A figura mostra dois roteadores físicos (Roteador Físico 1 e Roteador Físico 2) com duas interfaces físicas Ethernet cada um (`peth0` e `peth1`). Um enlace físico liga a interface física Ethernet `peth1` do Roteador Físico 1 com a interface física Ethernet `peth0` do Roteador Físico 2. O enlace virtual mostrado na figura é estabelecido com a associação do Roteador Virtual 1_1 do Roteador Físico 1 com o Roteador Virtual 2_1 do Roteador Físico 2.

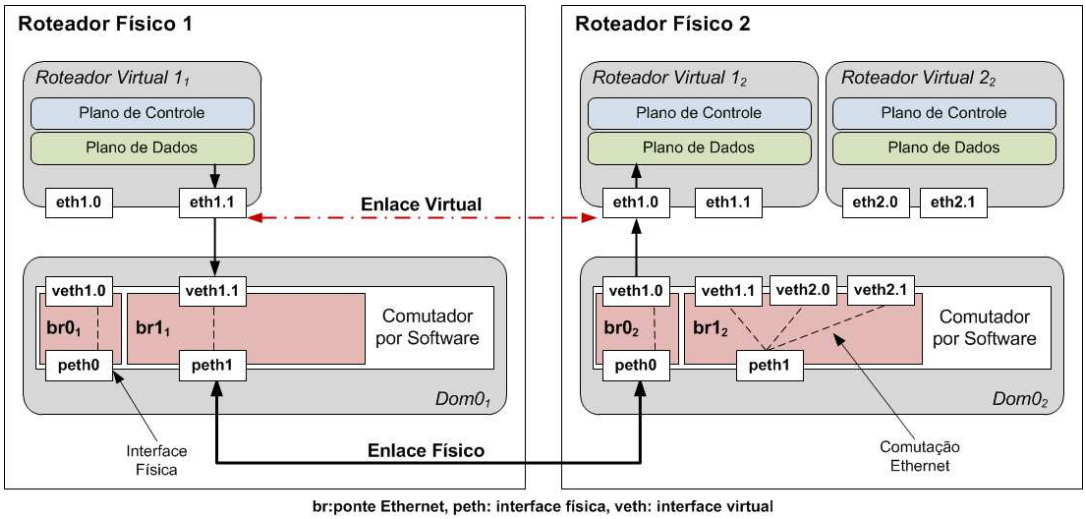
O modo *bridge* define uma ponte Ethernet, *bridge Ethernet*, que interconecta as interfaces da máquina física às interfaces das máquinas virtuais, como mostrado na Figura 2.6(a). Nesse caso, as interfaces físicas são representadas por interfaces virtuais que são ligadas ponto-a-ponto com as interfaces físicas. A ponte é um comutador por software e, portanto, encaminha os pacotes para máquina virtual baseado no encaminhamento da camada de enlace, MAC (*Medium Access Control*). O encaminhamento na camada de enlace ocorre da seguinte forma. O MAC é um endereço plano e, portanto, o endereço de um nó não traduz a localização do nó nem permite a agregação de endereços[5]. Sendo assim, para realizar o encaminhamento de acordo com o endereço MAC, o comutador por software, primeiramente, aprende em que porta cada endereço é acessível. Assim, ao chegar um pacote em um comutador, a

[5]Agregação de endereços é uma propriedade do endereçamento hierárquico na qual um endereço, do espaço de endereçamento, representa o caminho para se alcançar diversos (agregação) endereços individuais. Por exemplo, o prefixo 21 agrega todos os números de telefone do Rio de Janeiro enquanto o prefixo 11 agrega todos os números de telefone de São Paulo.

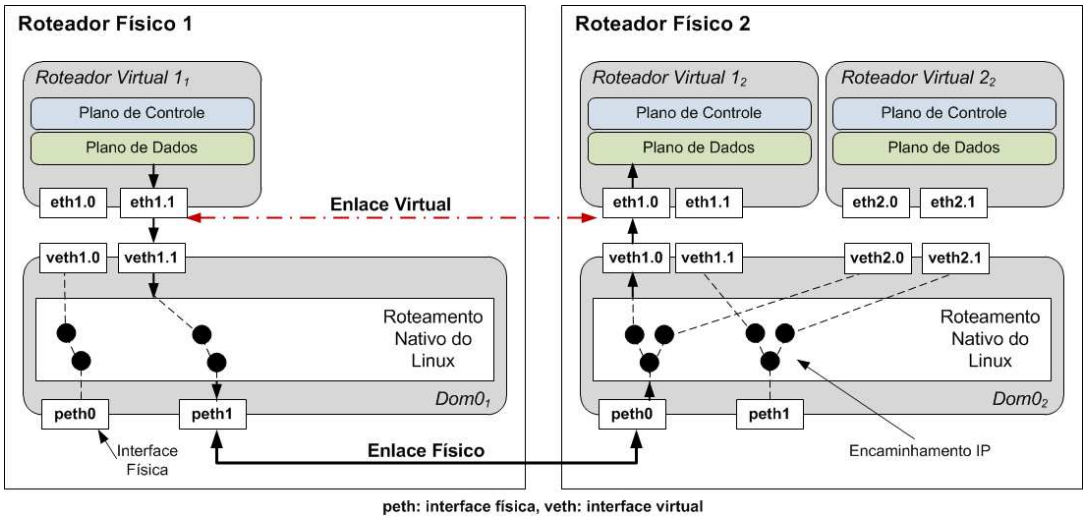
primeira ação tomada é gravar em uma tabela de aprendizagem a porta do comutador e o endereço de origem do pacote, marcando que tal endereço é acessível pela porta de entrada do pacote. Esse é o processo de aprendizagem realizado pelos comutadores tradicionais. Em seguida, quando o pacote vai ser encaminhado para seu destino, o comutador verifica na tabela de aprendizagem em qual porta o endereço de destino do pacote é acessível. Dessa forma, o pacote é enviado somente na porta em que o endereço de seu nó de destino é alcançável. É importante ressaltar que esse processo de encaminhamento requer que o endereço de destino do pacote seja exatamente igual ao de uma das entradas na tabela de aprendizagem, pois, nesse caso, o encaminhamento é feito baseado no endereço MAC, que é um endereço plano e, por isso, não sofre agregação. Assim, o encaminhamento de pacotes entre máquinas virtuais e a rede física é efetuado através de um comutador e, portanto, é como se o conjunto de todas as máquinas virtuais estivessem em um mesmo segmento Ethernet, como se fosse uma "extensão" da rede Ethernet física.

Outro modo de encaminhamento de pacotes no Xen é o modo *router*, mostrado na Figura 2.6(b). O modo *router* encaminha os pacotes para a máquina virtual através do roteamento dos pacotes de acordo com as informações da camada de rede, no caso, baseado no endereço IP. O modo *router* cria um roteador entre a máquina física e as máquinas virtuais. Assim, a máquina física se apresenta como mais um salto na rede IP entre máquina virtual e o restante da rede. O encaminhamento ocorre da seguinte forma. Nesse modo de encaminhamento, as interfaces virtuais, que representam as interfaces das máquinas virtuais no Domínio 0, recebem o mesmo endereço IP que suas respectivas interfaces nos Domínios Us. No modo *router* as interfaces de rede do Domínio 0 não só representam as interfaces reais, mas realmente são as interfaces reais [51]. A consequência é que os pacotes ao chegarem às interfaces de rede reais são encaminhados pelos mecanismos nativos do Linux ao invés de serem primeiramente encaminhadas pelo hipervisor do Xen. Dessa forma, ao chegar um pacote na máquina física, cujo endereço IP de destino é o de uma das máquinas virtuais, esse pacote é roteado para a máquina virtual através dos mecanismos nativos do Linux. Vale ressaltar que o roteamento nativo do Linux envolve algumas etapas. A primeira é verificar na tabela de rotas da máquina física a rota que melhor se adequa ao endereço do pacote encaminhado (algoritmo de *best matching prefix*). Ao ser encontrada a rota que melhor se adequa, o TTL (*Time to Live*) do pacote é decrementado e, então, o pacote IP é enviado no enlace definido pela rota selecionada, o que envolve a reconstrução do pacote da camada de enlace, trocando assim o cabeçalho da camada de enlace para endereçar o pacote ao seu próximo nó na camada de enlace, no caso do Xen, referente à camada MAC.

Essas arquiteturas de rede são suficientes e apresentam um bom funcionamento e também um bom desempenho para consolidação de servidores [42]. O modo *bridge*



(a) Encaminhamento de pacotes de acordo com o modo *bridge* do Xen.



(b) Encaminhamento de pacotes de acordo com o modo *router* do Xen.

Figura 2.6: Comparação entre dois modos de encaminhamento de pacotes no Xen, o modo *bridge* e o modo *router*.

permite que as máquinas virtuais se comuniquem como se estivessem em uma mesma LAN (*Local Area Network*). Já o modo *router* permite a agregação de rotas para um determinado conjunto de máquinas virtuais que estão sobre uma mesma máquina física, diminuindo os requisitos de memória sobre os dispositivos de encaminhamento da infraestrutura física [25], já que as rotas divulgadas na rede física endereçam grupos de máquinas virtuais, ao invés da divulgação dos endereços de todas as máquinas virtuais de forma plana e não agregada, como ocorre no modo *bridge*.

A princípio esses dois modos de multiplexação de pacotes entre interfaces de redes físicas e virtuais são suficientes para que as máquinas virtuais exerçam qualquer função, como por exemplo, inclusive, roteadores virtuais. Contudo, essas arqui-

teturas de rede não são plenamente suficientes para a virtualização de redes. Os modos *bridge* e *router* falham na arquitetura com separação dos planos de controle e de dados. A separação de planos é essencial para realizar a primitiva de migração de redes virtuais sem perda de pacotes [18, 19]. A separação de planos consiste em dividir a tarefa de roteamento em dois planos distintos, um de controle e outro de dados. O processo de controle de construção e alteração de rotas é definido no plano de controle, que é responsável pela execução do protocolo de roteamento e pela atualização da tabela de rotas. Já o plano de dados é responsável apenas pelo encaminhamento dos pacotes, ou seja, por receber os pacotes nas interfaces de entrada e encaminhá-los para as interfaces de saída corretas, de acordo com as rotas definidas no plano de controle. Pisa *et al.* propõem uma migração sem perdas no Xen com a mudança do plano de dados de todos os roteadores virtuais para o Domínio 0 do hipervisor Xen dos roteadores físicos [19]. Dessa forma, a migração do plano de dados ocorre sem afetar o encaminhamento dos pacotes. No entanto, a solução é restritiva, no sentido de que um roteador virtual só pode ser migrado para outro roteador físico que apresente os mesmos vizinhos e que estejam conectados por um mesmo barramento. Essa solução limita a migração a ocorrer somente para roteadores físicos cujas interfaces físicas de rede pertençam às mesmas redes que as do roteador de origem.

A separação de planos é dificultada pelos modos de encaminhamento padrão do Xen. Na separação de planos, o plano de controle é responsável por calcular as rotas, já o plano de dados é responsável por encaminhar os pacotes nas rotas corretas. O controle do protocolo de roteamento é executado em uma máquina virtual, o chamado plano de controle, enquanto o encaminhamento de pacotes IP é realizado no Domínio 0, o plano de dados. A separação dos planos relaciona-se com o roteamento de pacotes, ou seja, o encaminhamento de pacotes de acordo com o cálculo de uma rota mais adequada ao endereço de rede do pacote. Assim, quando o modo *bridge* é considerado, a separação de planos não se aplica, pois essa é uma tarefa que depende das informações da camada de rede. O modo *bridge* encaminha os pacotes entre as interfaces físicas e as máquinas virtuais de acordo com o endereço da camada de enlace do pacote e não considera o endereço da camada de rede, no caso, o IP. Sendo assim, como no modo *bridge* não há o roteamento de pacotes, os pacotes são encaminhados diretamente na Camada 2, como pontes Ethernet.

Já no modo *router*, a arquitetura com separação de planos é viável [19] em relação ao roteamento dos pacotes pelo cabeçalho IP. Cada máquina virtual executa um aplicativo, *daemon*, que verifica as rotas calculadas pelas máquinas virtuais (ou roteadores virtuais) e as comunica ao Domínio 0. No Domínio 0, por sua vez, há um outro aplicativo que recebe as rotas e cria novas tabelas de rota, uma para cada máquina virtual. Assim, cada máquina virtual detém uma cópia de sua tabela de

rotas no Domínio 0. Nas novas tabelas de rota, dedicadas às máquinas virtuais, são inseridas as rotas vindas de cada máquina virtual. O aplicativo do Domínio 0 também cria regras para que os pacotes de cada rede virtual sejam encaminhados de acordo com a tabela da rede referente à rede virtual a que pertencem. A forma mais trivial de classificar a qual rede virtual um pacote pertence é considerando que as redes virtuais possuem espaços de endereçamento IP distintos, o que é restritivo. Dessa forma, então, pode-se identificar a qual rede virtual um pacote pertence verificando-se apenas o seu endereço IP. Assim, ao receber um pacote, o endereço IP de destino do pacote é avaliado e, então, identifica-se a qual rede virtual o pacote pertence. Assim, o Domínio 0 encaminha o pacote de acordo com as novas tabelas de rotas referente à rede virtual do pacote, seguindo o processamento normal do roteamento nativo do Linux [52]. Contudo, a separação de planos usando o modo *router* nativo do Linux apresenta alguns desafios. Dois dos principais desafios são como identificar a qual rede virtual o pacote pertence e como tratar os pacotes com múltiplas destinações, por exemplo, a difusão (*broadcast*) e a destinação múltipla ou difusão seletiva (*multicast*). Estes dois problemas são detalhados a seguir.

Na virtualização de redes um roteador físico hospeda diversos roteadores virtuais. Um pacote que chega por uma das interfaces físicas do roteador físico precisa alcançar o roteador virtual ao qual pertence. A identificação trivial de qual rede um pacote pertence é através de seu endereço IP de destino. No entanto, essa identificação trivial falha na virtualização de redes se houver duas ou mais máquinas virtuais com a mesma faixa de endereçamento IP sobre uma mesma máquina física. Outro caso em que a identificação trivial falha é quando duas redes virtuais distintas usam o mesmo espaço de endereçamento IP, como por exemplo, a rede 10.0.0.0/24 mostrada na Figura 2.7. Os espaços de endereçamento não são disjuntos e, portanto, não há isolamento entre redes virtuais. Assim, se algumas redes virtuais de um mesmo roteador físico compartilham o mesmo espaço de endereçamento, as redes não são isoladas e, portanto, a identificação de qual rede virtual um pacote pertence é dificultada.

O segundo desafio de identificação de pacotes em redes virtuais consiste no encaminhamento de pacotes com endereços multidestinatários ou de difusão (*multicast/broadcast*). Ressalta-se que um pacote multidestinatário ou de difusão possui um endereço específico que designa a multidestinação. A multidestinação (*multicast*) não é uma lista de endereços individuais, mas sim de um grupo e isto ocasiona perda na semântica hierárquica de qual rede um IP pertence. Assim, os pacotes são destinados a endereços IP padrões e não é possível diferenciar entre uma rede ou outra apenas pelo seu endereço IP. Portanto, a solução de usar espaços de endereçamentos isolados para identificar as redes virtuais falha uma vez que o IP *multicast* não pertence à faixa de IP destinado às redes virtuais e, consequentemente, não pode ser

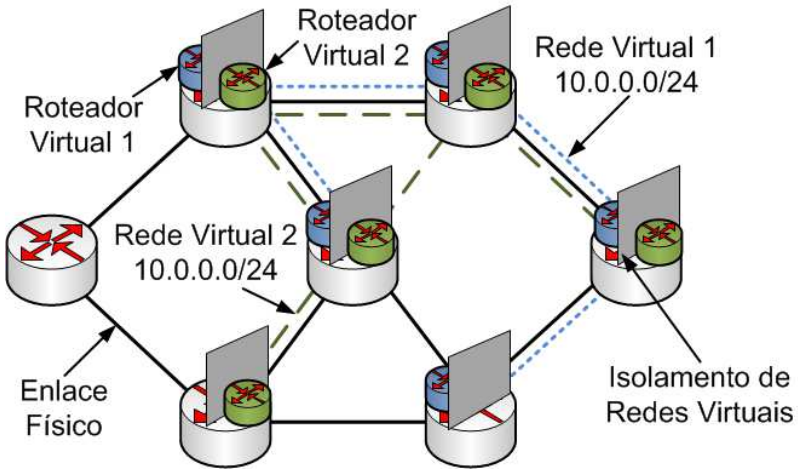


Figura 2.7: Rede física compartilhada por duas redes virtuais. As Redes Virtuais 1 e 2 apresentam o mesmo espaço de endereçamento IP. O isolamento garante que os pacotes de uma das redes virtuais não sejam acessíveis pela outra rede virtual.

encaminhado pelas tabelas de rota do Domínio 0. O problema de encaminhamento do quadro de *broadcast* é ainda mais desafiador, pois há casos em que o quadro de *broadcast* não possui cabeçalho IP, como no caso do ARP (*Address Resolution Protocol*). Dessa forma, a definição de qual rede o quadro *broadcast* pertence deve ser feita na camada de enlace, mas o modo *router* nativo do Linux trata somente a camada de rede. Assim, a virtualização de redes introduz alguns desafios cuja solução depende de ações que considerem o endereçamento da camada de enlace, no caso Ethernet, e permita o encaminhamento pelo endereçamento da camada de rede, IP.

Uma possível solução para separar o tráfego de uma rede virtual do tráfego das demais é realizar a multiplexação e demultiplexação de pacotes entre interfaces de rede virtuais e físicas através da marcação de pacotes com etiquetas, ou *tags*, de VLAN, como provido pelo Open vSwitch [16].

O Open vSwitch é um comutador por software desenvolvido para interligar ambientes virtuais, como mostra a Figura 2.8. O Open vSwitch age como uma ponte Ethernet, porém com algumas opções especiais, como por exemplo o uso de VLAN, que é o acréscimo de uma informação para a multiplexação/demultiplexação no cabeçalho do pacote. Assim, a ideia básica ao se usar o Open vSwitch é marcar as interfaces de rede virtuais com uma etiqueta (*tag*) comum às interfaces que pertençam a um mesmo domínio de *broadcast*, ou seja, um mesmo enlace Ethernet. Essa etiqueta, com o identificador da VLAN, é aplicada a todos os pacotes que saem da interface de rede virtual marcada com ela. O funcionamento, então, é simples. Um pacote, ao sair de uma máquina virtual para o Domínio 0, é marcado com a etiqueta da VLAN a que a sua interface de rede virtual pertence, ou seja, por onde o pacote chega ao Domínio 0. Assim, a etiqueta é um identificador do enlace virtual

ou do domínio de *broadcast* virtual. O pacote, depois de marcado, é enviado na rede física. Ao chegar no Domínio 0 do roteador virtual de destino, o Domínio 0 verifica qual é a etiqueta daquele pacote e só o entrega nas interfaces de rede virtuais que tiverem a mesma etiqueta do pacote. Esse procedimento garante que os espaços de endereçamento de cada rede virtual sejam independentes. Dessa forma, duas redes virtuais podem possuir a mesma faixa de endereçamento sem que uma interfira na outra, garantido o isolamento do espaço de endereçamento e dos pacotes de *broadcast*. O Open vSwitch, para se adaptar ao ambiente virtualizado, age como uma ponte Ethernet que executa algumas funções mais sofisticadas, como segmentar a rede virtual em diversas VLANs.

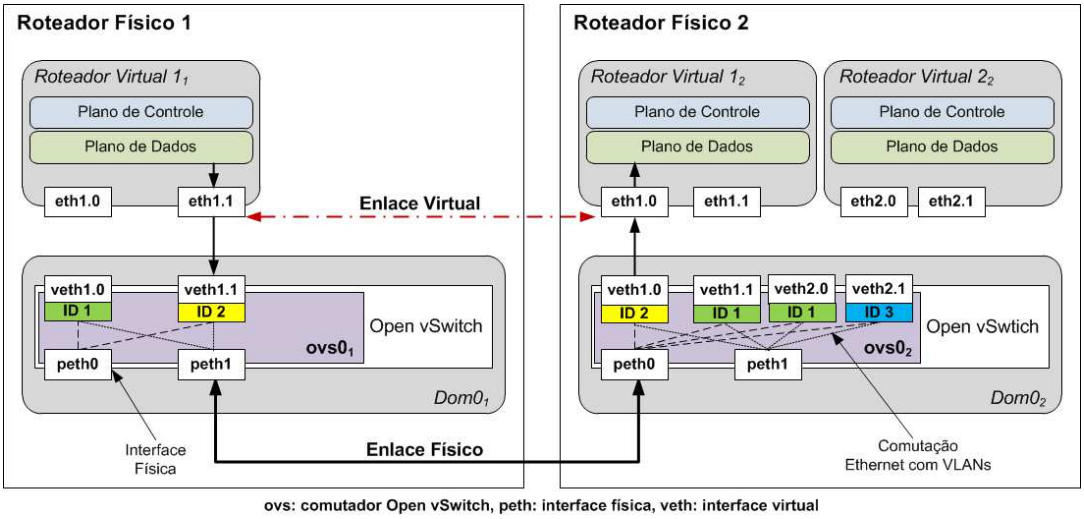


Figura 2.8: Encaminhamento de pacotes usando Open vSwitch no Xen.

O Open vSwitch oferece mecanismos, como a marcação de VLAN, que permitem virtualizar a rede e, ao mesmo tempo, isolar cada sub-rede IP, sem que uma sub-rede interfira nas demais [25]. Portanto, o Open vSwitch permite identificar cada rede virtual e fazer a difusão de pacotes *broadcast* e *multicast* dentro de uma rede virtual. Entretanto, como o Open vSwitch funciona de forma semelhante a uma ponte Ethernet, o desafio de realizar a separação de planos é o mesmo que no modo *bridge* nativo do Xen. O Domínio 0 deve ser capaz de realizar o encaminhamento de pacotes considerando tanto os campos da camada de enlace, Ethernet, para realizar a separação dos espaços de endereçamento de cada rede e encaminhar os pacotes de *broadcast* e *multicast*, assim como os campos da camada de rede, IP, para encaminhar de acordo com a melhor rota.

O Open vSwitch é capaz de realizar o processamento de pacotes em diversas camadas seguindo a interface de programação de aplicação (API – *Application Programming Interface*) OpenFlow [10]. O OpenFlow, por sua vez, é uma API para o controle de plataformas de encaminhamento que se baseia em campos das camadas de enlace, de rede e superiores para definir as regras de encaminhamento dos paco-

tes. Logo, uma solução para fazer a separação de planos com o isolamento de redes virtuais é usar o Open vSwitch se comportando como um comutador compatível com a API OpenFlow e programar esse comutador para fazer a separação de planos com isolamento. Essa é a ideia básica do XenFlow [18].

2.2.3 O Sistema XenFlow

O XenFlow é um sistema híbrido de roteamento de pacotes para redes virtuais que permite realizar a separação de planos no Xen usando como plano de dados um comutador OpenFlow.

A separação de planos no XenFlow é alcançada da seguinte maneira. O sistema XenFlow é composto de três programas principais: o XenFlow `Client`, o `Server` e a aplicação que roda sobre o controlador OpenFlow, no caso o NOX. Vale ressaltar que ao contrário da arquitetura centralizada do OpenFlow tradicional, em que todos os comutadores da rede são controlados por um único controlador centralizado, na arquitetura de virtualização de redes do XenFlow, o controle é distribuído, sendo que cada máquina física que hospeda roteadores virtuais apresenta uma instância do controlador NOX que é responsável por controlar somente o seu comutador OpenFlow local de acordo com as informações de rotas geradas pelos roteadores virtuais que hospeda. Assim, no XenFlow a relação entre controlador e comutador controlado é de um controlador para cada comutador, gerando um controle distribuído na rede OpenFlow.

Na máquina virtual, executa o XenFlow `Client` que é um aplicativo que executa em segundo plano e verifica se há atualização na tabela de rotas ou na tabela ARP da máquina virtual e as envia para XenFlow `Server`, no Domínio 0 da máquina física que a hospeda. O XenFlow `Server` é basicamente um procurador, *proxy* que recebe as conexões de todos os clientes, trata as mensagens, as responde e repassa as informações de todos os clientes concentradas e resumidas para a aplicação que executa sobre o NOX. A aplicação do NOX, `XenFlow App`, mantém a estrutura de dados para armazenar as tabelas de rota de cada máquina virtual e, também, faz a tradução das rotas em fluxos OpenFlow. Os componentes do sistema XenFlow estão explicitados na Figura 2.9.

Há dois motivos principais para fazer o `XenFlow Client` se conectar ao `XenFlow Server` ao invés de se conectar diretamente na aplicação do NOX: i) o NOX não apresenta bom desempenho ao tratar conexões *socket* em aplicações, pois o NOX é um único processo em uma única *thread*. Embora isso garanta o estado de coerência da rede, pois só há uma ação sendo executada por vez, ao tratar diversas conexões, o NOX precisa interromper o seu processamento principal. Por isso, o número de *sockets* atendidos pelo NOX é apenas um e com mensagens resumidas de todas as

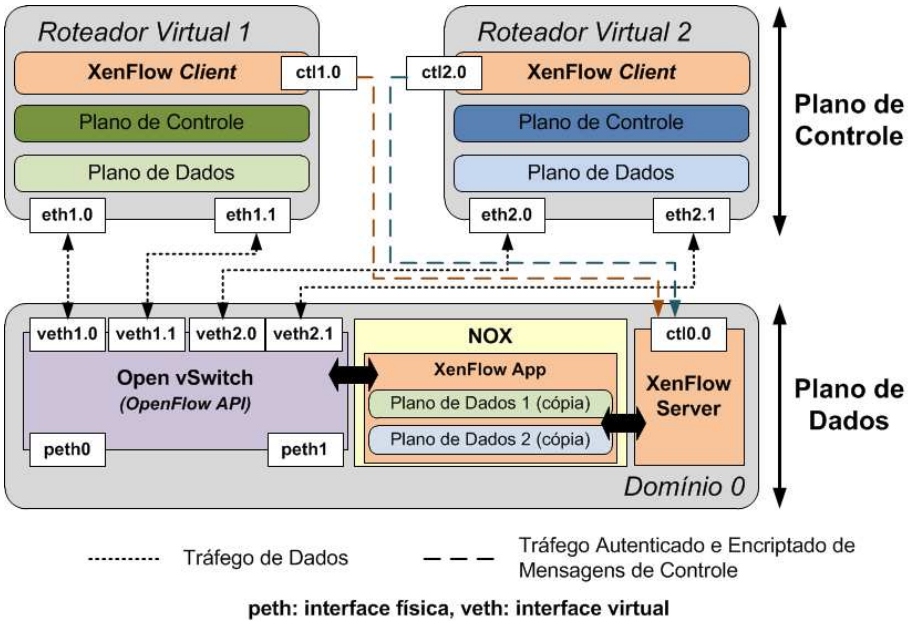


Figura 2.9: Componentes do sistema XenFlow. Toda comunicação de controle do XenFlow é encriptada e autenticada seguindo através do protocolo SSL.

máquinas virtuais; ii) o XenFlow pode ser usado para criar uma arquitetura para roteadores por software, isto é, o plano de dados OpenFlow pode ser composto por diversos comutadores físicos (*high-end*) e o plano de controle por diversas máquinas virtuais executando sobre diferentes servidores físicos. Nesse caso, O XenFlow *Server* age como um concentrador entre as máquinas virtuais e aplicação do NOX que controla os comutadores físicos do plano de dados.

As aplicações XenFlow *Cliente* e *Server* foram implementadas em linguagem Python e executam no espaço de usuário. A comunicação entre o XenFlow *Client* e o XenFlow *Server* ocorre através de interfaces de redes dedicadas à comunicação entre máquinas virtuais e Domínio 0. Todas as comunicações de controle do XenFlow são encriptadas e autenticadas através da troca de certificados, usando o protocolo SSL v3.0 (*Secure Sockets Layer*).

Tradução de Rotas em Fluxos

A tradução de uma rota em fluxo ocorre da seguinte forma. Ao chegar um pacote no comutador por *software* (Open vSwitch) de um Domínio 0, se não existir um fluxo ao qual o pacote se adeque, o pacote é enviado ao NOX, conforme o funcionamento normal do OpenFlow. No NOX, o pacote é processado pela aplicação `XenFlow App` que verifica a qual máquina virtual o pacote se destina de acordo com o seu endereço MAC de destino. Identificada a máquina virtual, o endereço IP de destino do pacote é verificado. Se o endereço IP de destino do pacote se adéqua a alguma rota daquela máquina virtual, a `XenFlow APP` extrai o IP do próximo salto através da rota na

tabela de rotas da máquina virtual identificada que melhor se adeque ao endereço do pacote, algoritmo de *best match*. Após extrair o IP do próximo salto do pacote na rede, `XenFlow App` verifica na cópia da tabela ARP da máquina virtual, para a qual o pacote se destina, se a máquina virtual já conhece o mapeamento do endereço IP no endereço MAC do próximo salto. Conhecendo o MAC, a `XenFlow App` introduz um novo fluxo no plano de dados OpenFlow. O novo fluxo é introduzido de acordo com os campos do pacote que dispararam o cálculo do novo fluxo e as ações associadas a esse novo fluxo são trocar os endereços MAC de origem e destino do pacote e encaminhar o pacote na porta de saída adequada. O encaminhamento no XenFlow é exemplificado pela Figura 2.10.

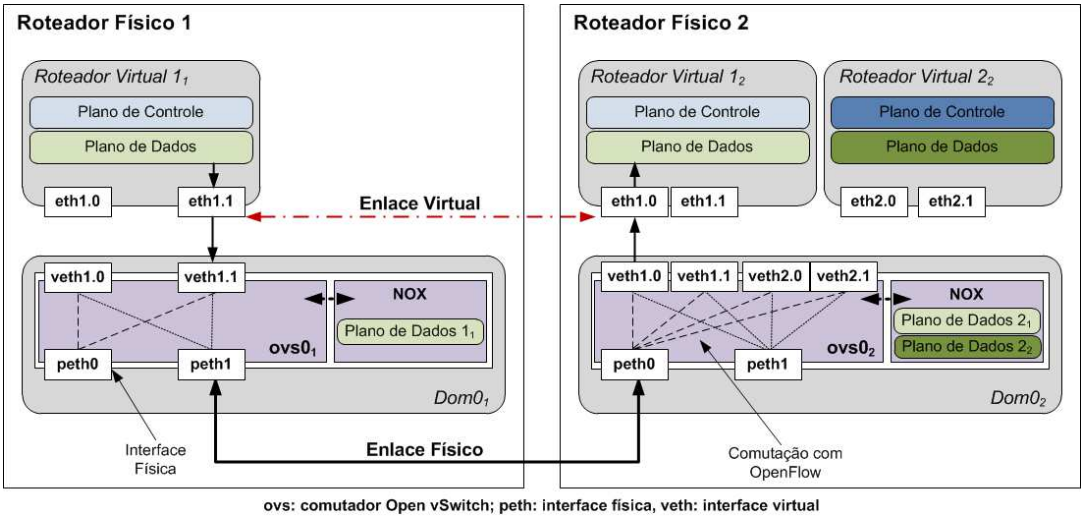


Figura 2.10: Encaminhamento de pacotes no XenFlow.

As ações configuradas para cada fluxo no comutador OpenFlow correspondem ao roteamento do pacote. A troca dos endereços MAC marca a troca do enlace pelo qual o pacote está sendo encaminhado. Nesse sentido, as ações são colocar o endereço MAC de origem do pacote como sendo o endereço da interface pelo qual o pacote seria encaminhado pela máquina virtual e colocar como MAC de destino do pacote como o endereço MAC consultado na tabela ARP da máquina virtual, este endereço é a tradução do IP do próximo salto para o MAC do próximo salto do pacote na rede. A descoberta de em qual porta física do comutador OpenFlow o pacote modificado deve ser encaminhado é feita pelo mecanismo de aprendizagem do comutador. Assim, quando um pacote chega ao comutador, este armazena o endereço MAC de origem do pacote e por qual porta o aquele chegou. Assim, o comutador sempre sabe em qual porta um MAC é acessível.

Se qualquer uma dessas fases falhar, a `XenFlow App` encaminha o pacote como se fosse um comutador comum, ou seja, verifica qual o MAC de destino do pacote e o envia, tal como ele chegou, na porta física do comutador em que o MAC de destino é alcançável. Esse comportamento faz com que o XenFlow se comporte tanto como

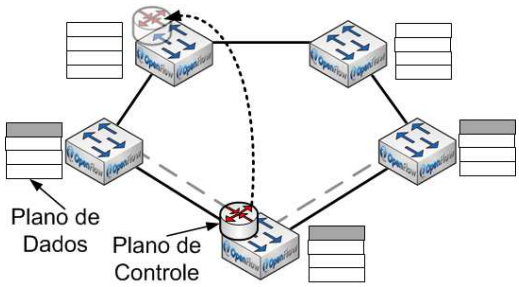
um roteador, quando conhece uma rota para o pacote, ou como um comutador, quando alguma das etapas de processamento do pacote falha. No caso de o nó físico da rede não conhecer nenhuma rota para o pacote em processamento e, também, não conhecer em que porta o MAC de destino é acessível, o pacote é inundado na rede. O comportamento de inundação é o comportamento padrão de um comutador que não conhece ainda como alcançar o endereço MAC de destino.

Migração de Redes Virtuais

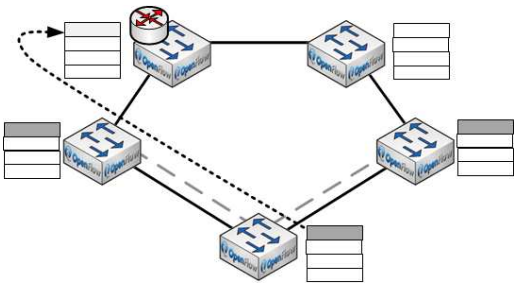
A migração de roteadores virtuais sem perdas é um desafio, pois o processo de migração exige que o roteador seja suspenso por algum tempo. No XenFlow, o processo de migração é dependente da migração do Xen e, portanto, também exige a suspensão do roteador virtual por um determinado tempo. Contudo, como o XenFlow implementa a separação de planos, a suspensão temporária é somente do plano de controle. Portanto, o plano de dados, que está no Domínio 0, continua a executar mesmo enquanto ocorre a migração. Somente depois de o plano de controle ser restaurado na máquina de destino da migração que o plano de dados da máquina de origem é desligado.

Em uma rede XenFlow, um enlace virtual pode ser mapeado em um ou mais enlaces físicos. O encaminhamento é feito por uma tabela de fluxos programada dinamicamente pelo Controlador NOX. Assim, a topologia da rede virtual fica desassociada da topologia da rede física. Sendo assim, a migração de nós virtuais em uma rede XenFlow, mostrada na Figura 2.11, possui três etapas: migração do plano de controle, reconstrução do plano de dados e migração de enlaces [18]. O plano de controle é migrado entre dois nós físicos da rede, através do mecanismo de migração ao vivo de máquinas virtuais convencional do Xen [30]. Em seguida, a reconstrução do plano de dados é realizada da seguinte forma. A aplicação XenFlow *Client*, que se conecta ao XenFlow *Server* referente à máquina física em que o roteador virtual está hospedado, detecta que houve a mudança de qual servidor físico está hospedando o roteador virtual migrante. Essa mudança é detectada pela quebra temporária da conexão entre o XenFlow *Client* e *Server*. Ao detectar a quebra, o XenFlow *Client* se reconecta, agora no novo servidor já na máquina física de destino da migração, e envia todas as informações de suas tabelas de rotas e ARP. Ao receber essas informações, o XenFlow *Server* as repassa ao XenFlow *APP* que, por sua vez, reconfigura o NOX local da máquina física de destino da migração para executar o plano de dados à máquina virtual migrante. Assim, todos os pacotes que chegarem à máquina física de destino da migração e que são destinados a máquina virtual migrante são tratados de acordo com as informações de controle calculadas pelo plano de controle previamente já migrado. Depois da migração do plano de dados e da reconstrução do plano de controle, ocorre a migração dos enlaces. A migração de enlaces ocorre

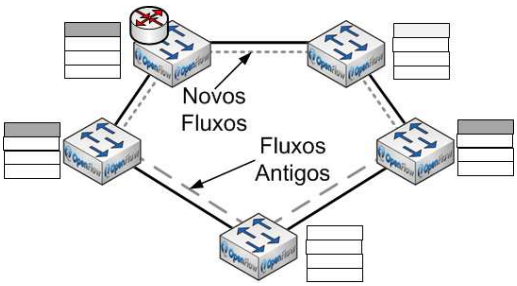
nos comutadores OpenFlow dos Domínios 0 e nos outros comutadores da rede. A migração de enlaces ocorre de forma a criar um caminho comutado entre os vizinhos a um salto lógico do roteador virtual até o roteador físico de destino da migração. Para tanto, o roteador virtual migrado envia um pacote *ARP reply* com um endereço MAC de destino especial (`AA:AA:AA:AA:AA:AA`). No sistema XenFlow, esse endereço é reservado e esse pacote tem prioridade em relação aos demais pacotes *ARP reply* enviados na rede na aprendizagem de em qual enlace o roteador virtual migrado é acessível. Esse procedimento de atualização da localização de um roteador virtual na rede associado ao comportamento dual, roteador e comutador, de um nó XenFlow resultam na primitiva de migração de roteadores virtuais sem perda de pacotes ou interrupção do serviço de encaminhamento de pacotes.



(a) Migração do plano de controle do roteador virtual.



(b) Reconstrução do plano de dados de acordo com o plano de controle migrado.



(c) Novos fluxos são definidos na máquina física de destino da migração.

Figura 2.11: Mecanismo de migração do XenFlow.

Ao final do procedimento de migração de um roteador virtual, alguns fluxos ainda podem estar sendo encaminhados pelo nó físico de origem da migração, pois os fluxos já definidos na rede não são migrados. No caso específico desses fluxos já definidos, esses fluxos não são alterados pelo procedimento de migração. Contudo, ao fim do procedimento de migração, os novos fluxos são encaminhados pelo nó de destino da migração. Assim, conforme os fluxos antigos são encerrados, novos são criados e encaminhados pelo novo caminho. No intuito de evitar que fluxos fiquem sendo encaminhados por tempo indeterminado pelo caminho antigo, ao definir um novo fluxo na rede, o XenFlow define um *hard timeout*, isto é um argumento na

definição de um fluxo OpenFlow que define o tempo máximo de duração daquele fluxo. Caso o fluxo dure mais do que o definido em seu *hard timeout*, o próprio comutador OpenFlow exclui o fluxo e reenvia o próximo pacote do fluxo ao NOX para que este o redefina. Assim, o tempo máximo de uso de um caminho antigo após uma migração de um roteador virtual é igual ao *hard timeout* definido na rede.

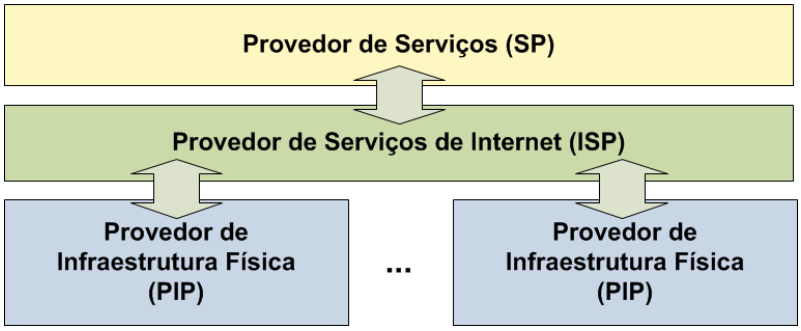
Como por algum tempo os dois planos de dados podem estar ativos, é possível que haja o desordenamento de pacotes, mas, como o encaminhamento é feito por fluxos, não há duplicação, exceto de pacotes de *broadcast* e *multicast*.

Capítulo 3

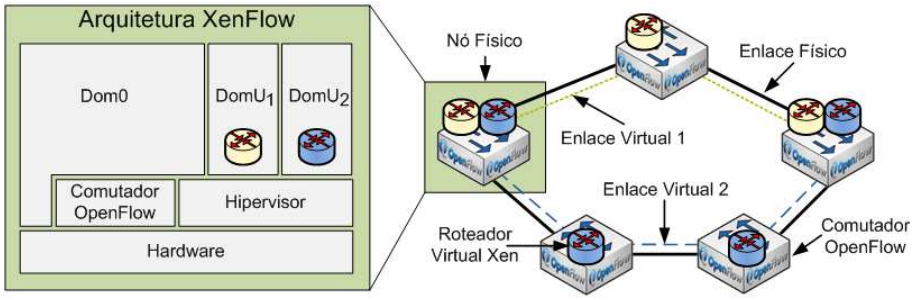
QFlow: Controle de Recursos e Garantia de Qualidade de Serviço

No modelo atual da Internet, os provedores de serviços (*Service Provider* - SP) fornecem serviços através da Internet e os provedores de serviço de Internet (*Internet Service Provider* – ISP) fornecem conectividade à Internet aos seus clientes. Contudo, projeções para os novos modelos de negócio da Internet apontam para necessidade de desacoplar as funções de fornecer e manter os equipamentos físicos de rede das funções de prover conectividade fim-a-fim e serviços na Internet, criando a necessidade de um novo ator, o provedor de infraestrutura física (*Physical Infrastructure Provider* – PIP) [53]. A principal proposta para realizar o desacoplamento de funções é a virtualização de redes, pois permite que um ISP execute múltiplos serviços fim-a-fim sobre equipamentos de rede de diferentes provedores de infraestrutura [6], como mostrado na Figura 3.1(a). Nesse novo cenário, os acordos de nível de serviço (*Service Level Agreements* – SLAs) estabelecidos entre os ISPs e seus clientes devem ser mapeados na rede física. Assim, para garantir uma determinada qualidade de serviço a uma rede virtual, é necessário obter garantias tanto do provedor de rede virtual (ISP) quanto do provedor de infraestrutura física (PIP), como também é necessário um mapeamento entre os parâmetros estabelecidos pelo ISP e os recursos fornecidos pelo PIP. O sistema QFlow proposto permite a garantia e o gerenciamento de Qualidade de Serviço (QoS - *Quality of Service*) através do plano de encaminhamento de dados e também realiza o mapeamento dos parâmetros de QoS do plano de controle da rede virtual para o plano de dados. Portanto, o sistema QFlow oferece garantias de qualidade de serviço atendendo a esse novo modelo de negócios.

Este capítulo apresenta uma extensão do modelo híbrido de virtualização de redes XenFlow [18]. O modelo XenFlow de virtualização de redes permite que um provedor de infraestrutura forneça roteadores virtuais aos provedores de serviço de Internet e esses, por sua vez, usem os roteadores virtuais para prover conectivi-



(a) Modelo de serviços com três atores na arquitetura da Internet de nova geração: provedor de serviço, provedores de serviço de Internet e provedor de infraestrutura física.



(b) Arquitetura XenFlow com duas redes virtuais.

Figura 3.1: Modelo de virtualização de redes com XenFlow no qual um comutador OpenFlow interconecta as interfaces virtuais às interfaces físicas.

dade fim-a-fim a seus clientes. Nesse contexto, um roteador virtual é um ambiente virtualizado com interfaces semelhantes às de uma máquina real e executa sobre o substrato físico compartilhado com outros roteadores virtuais, como mostra a Figura 3.1(b). O modelo de virtualização XenFlow é dividido em dois componentes principais: o plano de controle Xen e o plano de dados OpenFlow. O plano de controle Xen é o ambiente virtualizado ao qual o ISP contratante tem acesso e executa os seus protocolos de roteamento. Já o plano de dados OpenFlow é a estrutura de encaminhamento compartilhada por todos os roteadores virtuais hospedados sobre um mesmo substrato físico. O compartilhamento dos planos de dados é sujeito a ataques de negação de serviço, pois uma ou mais redes podem exaurir os recursos do roteador físico de forma intencional ou não. Além disso, a interação entre o plano de controle e o plano de dados restringe-se à informação das rotas calculadas no plano de controle que são repassadas ao plano de dados. As políticas de QoS definidas no plano de controle precisam ser refletidas no real encaminhamento dos dados no plano de dados que agora é compartilhado por todos os roteadores virtuais.

O sistema QFlow [21] controla o uso dos recursos do plano de dados do modelo de virtualização XenFlow. Assim, o sistema proposto aplica o isolamento das redes virtuais e, ao mesmo tempo, possui controle dos recursos físicos garantindo oferta

de qualidade de serviço por rede virtualizada. Os recursos controlados são o processamento, a memória e a banda passante de cada roteador virtual. O controle de processamento é feito diretamente pelo escalonador do Xen, limitando-se o quanto da CPU é atribuído para cada roteador virtual. O controle de memória é realizado limitando-se o número de rotas exportadas para o plano de dados. O controle de banda é realizado por dois algoritmos que garantem uma taxa mínima para oferta de qualidade de serviço e uma taxa máxima para otimizar o uso de recursos de rede, redistribuindo a capacidade ociosa dos enlaces entre os roteadores virtuais. O controle das taxas mínima e máxima de cada rede virtual é realizado através da instanciação de filas associadas às interfaces físicas e do mapeamento das interfaces de cada roteador virtual nas filas de cada interface física.

3.1 O Sistema QFlow

O QFlow é um sistema que provê isolamento e qualidade de serviço em redes virtualizadas. O QFlow garante isolamento das redes virtuais ao oferecer e garantir recursos mínimos para cada uma das redes virtuais. Assim, uma determinada rede virtual não pode consumir os recursos de outra rede virtual evitando assim ataques de negação de serviço (*Denial of Service* – DoS). O sistema QFlow oferece também qualidade de serviço através do mapeamento dos parâmetros de acordos de nível de serviço (*Service Level Agreements* – SLAs) do plano de controle para os parâmetros do plano de dados das redes virtuais, controlando o consumo de recursos de cada rede virtual. Os recursos básicos de processamento, de memória e de banda passante usados por cada rede virtual são aqueles que podem ser controlados localmente [39] pelo QFlow.

O sistema QFlow se serve do paradigma da separação de planos para obter alto desempenho no encaminhamento de pacotes [29, 42]. Nesse paradigma, as máquinas virtuais agem como plano de controle de roteadores, executando protocolos de roteamento. Já o plano de dados de todos os roteadores virtuais é executado de forma centralizada no Domínio 0, que é uma máquina virtual com privilégios especiais e que acessa diretamente as interfaces físicas de rede da máquina física. No entanto, esse paradigma de separação de planos faz com que todas as redes virtuais compartilhem um mesmo plano de dados, violando o requisito de isolamento de cada ambiente virtual e com isso possibilitando ataques de negação de serviço.

Como foi mencionado, o QFlow se baseia no modelo de virtualização do XenFlow o que significa ter um comutador OpenFlow atuando no plano de dados. O FlowVisor [39] é uma das técnicas de isolamento usada no OpenFlow. Contudo, o FlowVisor não controla e não isola os fluxos de cada roteador virtual que estão agora compartilhados no plano de dados do sistema QFlow, pois o FlowVisor foi proposto

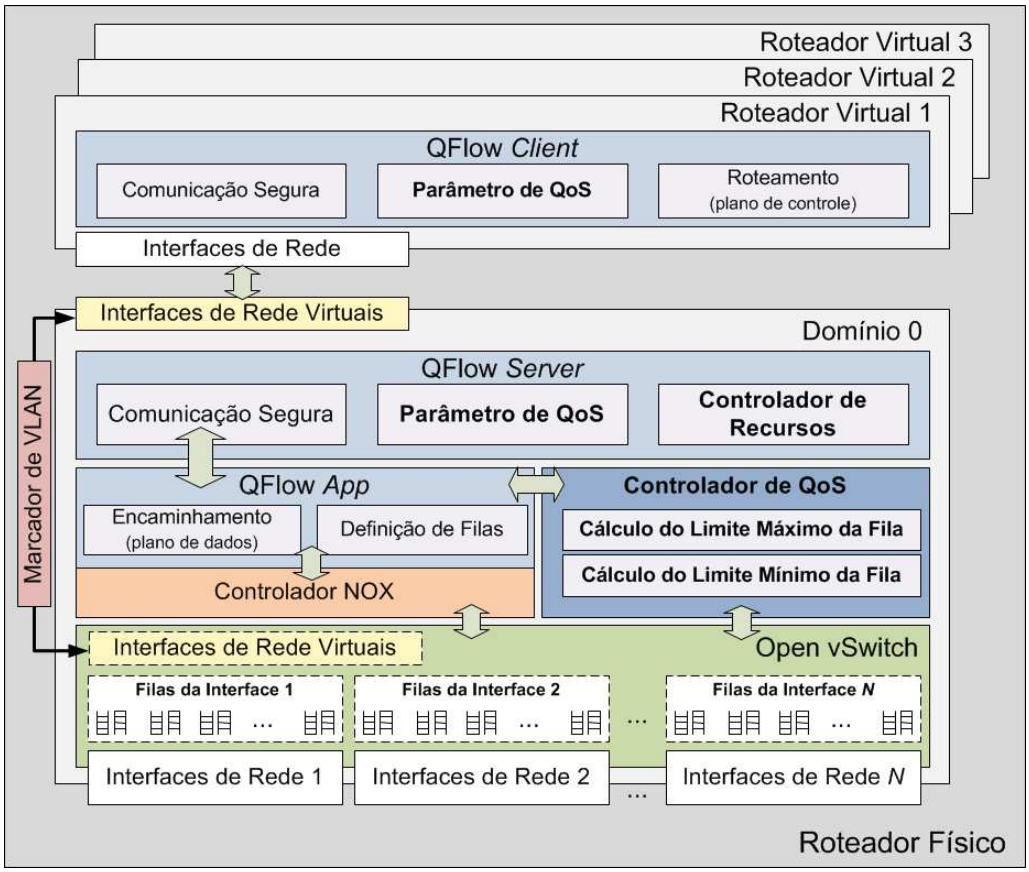


Figura 3.2: Arquitetura do sistema QFlow. O *QFlow Server* executa no roteador físico, que hospeda um conjunto de roteadores virtuais com um *QFlow Client* em cada um deles.

para controlar os recursos de diferentes controladores OpenFlow. Portanto, o isolamento de recursos no QFlow requer a criação de mecanismos de encaminhamento dedicados a cada roteador virtual no plano de dados, no Domínio 0, e também mecanismos de controle. A solução adotada no QFlow é o uso de filas de encaminhamento associadas aos roteadores virtuais com políticas de controle do tipo HTB (*Hierarquical Token Bucket*). A política de filas HTB é implementada pelo Open vSwtich.

A arquitetura do sistema QFlow, mostrada na Figura 3.2, é um computador pessoal executando a plataforma de virtualização Xen e a matriz de comutação de pacotes baseada na interface de programação de aplicação (*Application Programming Interface* – API) OpenFlow. As máquinas virtuais Xen executam protocolos de roteamento e comportam-se como roteadores virtuais. Neste trabalho, a plataforma de roteamento extensível XORP (*eXtensible Open Router Platform*) [54] permite que o roteador virtual execute os principais protocolos de roteamento e, ainda, permite a inclusão de novos protocolos. O roteador virtual executa o módulo *QFlow Client* que verifica as atualizações na sua tabela de rotas e na sua tabela ARP (*Address Resolution Protocol*). A tabela de rotas é a estrutura de dados que armazena as

informações das rotas, calculadas pelos protocolos de roteamento executando no roteador virtual, e a tabela ARP é aquela que armazena o mapeamento dos IPs em endereços MAC conhecidos pelo roteador virtual. As informações coletadas na máquina virtual são, então, encaminhadas para o módulo *QFlow Server* que executa no Domínio 0. O módulo *QFlow Server* gera resumos das informações de todos os roteadores virtuais e os repassa para a aplicação *QFlow App*. A aplicação *QFlow App* executa sobre o controlador NOX do OpenFlow, instanciado, também, no Domínio 0 de cada nó QFlow.

A principal inovação do QFlow em relação às outras propostas é a introdução do módulo de controle de uso de recursos por cada rede virtual e do módulo de controle de filas, mapeando as primitivas de Qualidade de Serviço em recursos disponíveis no plano de dados. O plano de dados do roteador QFlow é um comutador por *software* Open vSwitch [16] com suporte ao controle pela API OpenFlow. É importante ressaltar que no QFlow todo processo de comunicação entre módulos é criptografado e autenticado usando o esquema de distribuição de chaves públicas (*Public Key Infrastructure* – PKI) e o padrão SSL 3.0 (*Secure Sockets Layer*), o que garante a confidencialidade das mensagens enviadas pelos roteadores virtuais, não havendo a possibilidade de um roteador virtual bisbilhotar (*eavesdroping*) as mensagens dos outros.

3.2 O Controlador de Uso de Recursos

Os três recursos controlados pelo QFlow são o processamento, a memória e a banda passante por interface de rede usados por cada roteador virtual. A seguir é explicado como o sistema controla cada um dos recursos citados.

3.2.1 O Controle de Processamento

O controle do processamento de cada roteador virtual é executado de forma estática. Para tanto, o QFlow usa os parâmetros do escalonador do Xen para definir a prioridade de execução de cada roteador virtual em relação aos demais e o quanto de tempo contínuo de processamento cada roteador pode receber. O uso dos mecanismos de escalonamento nativos do Xen é suficiente para garantir o isolamento na carga de processamento de cada roteador virtual, como mostrado em [29]. Contudo, o encaminhamento de pacotes pelo Domínio 0 pode acarretar uma sobrecarga de processamento no Domínio 0 que não é controlada pelos parâmetros do escalonador do Xen [3]. Nesse sentido, a limitação da carga de processamento por roteador virtual no Domínio 0 é realizada através da restrição da capacidade de encaminhamento de pacotes, através do controle de banda associado aos roteadores virtuais.

3.2.2 O Controle de Memória

O sistema QFlow apresenta duas áreas de memória distintas. A primeira é dedicada ao roteador virtual e suas aplicações, representada pela área de memória da máquina virtual Xen. A segunda é uma área de memória compartilhada no Domínio 0. O isolamento na área de memória dedicada é realizado pelos próprios mecanismos do Xen. Entretanto, no Domínio 0 há a necessidade de outros mecanismos para limitar o uso de memória de cada roteador virtual. É importante ressaltar que o uso intencional de memória no Domínio 0 por uma das redes virtuais para exaurir recursos de memória das outras redes é um ataque de negação de serviço.

No Domínio 0, um roteador virtual consome memória para armazenar uma cópia de sua tabela de rotas e de sua tabela ARP, além de consumir memória para armazenar os seus fluxos instanciados na tabela de fluxos do OpenFlow. Assim, o controle de memória para cada roteador é realizado limitando-se o número máximo de entradas que um roteador virtual pode ter na sua tabela de rotas exportada para o Domínio 0, assim como limita-se também o número máximo de fluxos associados a um único roteador virtual. Nesses casos, para evitar perda de pacotes por causa da falta de rotas ou de fluxos no Domínio 0, são adicionados rotas e fluxos padrões que forçam todo novo fluxo a ser encaminhado pelo roteador virtual ao invés de ser encaminhado pelo plano de dados. Portanto, evita-se a negação de serviços por roteadores virtuais mal intencionados transferindo o encaminhamento para a máquina virtual que exceder a sua quota. Logo, o procedimento apenas torna mais lento o encaminhamento de pacotes, que agora vai ser feito no domínio do usuário, o que é interessante para os roteadores que excedem a sua quota, mas não estão agindo de forma mal intencionada.

3.2.3 O Controle de Banda

O controle da banda de cada roteador virtual é realizado através da associação das máquinas virtuais a filas na matriz de encaminhamento OpenFlow. O OpenFlow permite a criação de filas isoladas, em cada interface de rede. Cada fila tem dois limites definidos: o mínimo reservado e o máximo permitido. O mínimo reservado é o limite mínimo garantido para aquela fila no compartilhamento do enlace representado pela interface de rede em que a fila está inserida. O limite máximo é o valor máximo que a taxa de transmissão daquela fila pode assumir.

O mecanismo de mapeamento dos fluxos de cada roteador virtual para a fila correspondente é realizado pela aplicação *QFlow App*, que mapeia as rotas em fluxos e, também, ao definir a interface de saída para o fluxo, associa o fluxo à fila dedicada ao roteador virtual pelo qual o fluxo foi encaminhado. Contudo, o controle de filas exercido pelo OpenFlow é estático e não considera a prioridade nas filas para

redistribuir a capacidade excedente do enlace. Dessa forma, uma das contribuições deste trabalho é um controlador de filas que executa a redistribuição da capacidade ociosa excedente do enlace de acordo com a prioridade de cada fila e inversamente proporcional a quanto a fila está usando além da capacidade contratada.

3.2.4 O Controlador de Filas

A política do controle de filas usada nessa dissertação maximiza a probabilidade de uma rede virtual ter acesso aos recursos contratados, quando forem necessários, e minimiza a probabilidade de os recursos permanecerem ociosos, redistribuindo-os entre as redes virtuais ativas no momento. O controlador de filas é responsável pela redistribuição da capacidade ociosa do enlace entre as filas instanciadas, a qual é feita em dois níveis. O primeiro caso é quando a capacidade total do enlace é maior do que o somatório de todos os limites mínimos das filas associadas àquele enlace. Nesse caso, o controlador age sobre o limite mínimo das filas. O segundo caso é quando cada fila possui a sua capacidade mínima definida, porém o uso total do enlace é menor do que o somatório dos limites mínimos reservados para as filas. Nesse caso, o controlador age sobre o limite máximo de cada fila. Em ambos os casos, a redistribuição considera a prioridade de cada rede e, no QFlow, a prioridade de uma rede é o valor relativo da banda contratada pela dada rede dividido pelo somatório dos valores de banda contratados por todas as redes que compartilham o enlace. O controlador mede o uso de cada fila regularmente a cada t_{med} segundos, armazena os últimos k_{med} valores de uso para cada fila e, sobre esses dados, calcula a redistribuição da capacidade ociosa da fila.

O Controle do Limite Mínimo Garantido

O controle do limite mínimo redistribui os recursos não atribuídos a filas segundo dois critérios:

- o roteador virtual que contrata uma maior capacidade de encaminhamento tem **maior peso** na redistribuição de recursos ociosos;
- o roteador virtual que está usando relativamente mais recursos do que contratou tem **menor peso** na redistribuição de recursos ociosos.

A redistribuição dos recursos ociosos entre as filas é proporcional aos pesos calculados para cada fila, como é mostrado no Algoritmo 1. A entrada do algoritmo de controle é dada pelas seguintes variáveis:

- *Porta*: representa o conjunto de filas associado a uma dada interface de rede;

- $LimMin$: vetor que representa os valores mínimos de transmissão garantidos no OpenFlow para cada uma das filas;
- $Carga$: vetor que representa os valores médios de carga para cada fila da interface controlada no momento em que o algoritmo está sendo aplicado.

A ideia central do Algortimo 1 é calcular um valor relativo α que representa o quão longe o valor da carga atual da fila está do valor mínimo contratado e garantido por SLA. Se α é positivo, a carga atual da fila é menor do que o mínimo contratado. Caso seja negativo, a carga atual é maior que o mínimo contratado. Sendo assim, α mede um percentual de recursos usados por cada fila além do mínimo contratado. O valor de α, então, assume tanto valores positivos quanto negativos. O valor de α é então mapeado para o intervalo de $[0, 1]$ e passa a ser denominado γ. Os valores negativos de α são mapeados para o intervalo $[0, 0, 5)$, os positivos, no intervalo $(0, 5, 1]$ e $\alpha = 0$ é mapeado em 0,5. Com posse dos valores de γ para todas as filas associadas a uma dada interface, os pesos de redistribuição são calculados através da normalização dos valores γ, de forma que a soma de todos os pesos das filas de uma interface seja sempre igual a 1.

Algoritmo 1: Cálculo dos pesos de redistribuição de recursos nas filas.

Entrada: $Porta, LimMin[\,], Carga[\,]$
Saída: $Pesos[\,]$
$Zerar(\alpha[\,])$; $Zerar(\gamma[\,])$;
para $fila \in Porta$ **faça**
  $\alpha[fila] = \frac{LimMin[fila] - Carga[fila]}{\sum_i^{i \in Porta} LimMin[i]}$
fim
para $fila \in \alpha$ **faça**
  se $(\alpha[fila] > 0)$ **então** $\gamma[fila] = 0,5 \cdot \frac{\alpha[fila]}{Max(\alpha)} + 0,5$
  se $(\alpha[fila] = 0)$ **então** $\gamma[fila] = 0,5$
  se $(\alpha[fila] < 0)$ **então** $\gamma[fila] = 0.5 \cdot \frac{Min(\alpha) - \alpha[fila]}{Min(\alpha)}$
fim
$total_\gamma = \sum_i^{i \in \gamma} i$
para $fila \in \gamma$ **faça**
  $Pesos[fila] = \frac{\gamma[fila]}{total_\gamma}$
fim

O Controle do Limite Máximo Permitido

O controle do limite máximo permitido para cada rede virtual tem como objetivo otimizar o uso do enlace de acordo com a prioridade de cada rede virtual. O cálculo do valor máximo permitido é realizado somando-se a carga média de todas as filas

em um enlace e, após, subtraindo-se esse valor da capacidade do enlace. O valor resultante é a capacidade que deve ser redistribuída. A redistribuição dada por

$$max_{fila} = min_{fila} + Cap_{ociosa} * \frac{min_{fila}}{\sum_{i}^{i \in Porta} min_i}, \tag{3.1}$$

em que Cap_{ociosa} é a capacidade ociosa do enlace, garante que os recursos não utilizados por uma fila serão atribuídos às demais filas proporcionalmente ao mínimo contratado por cada uma.

3.3 A Provisão de QoS

A provisão da qualidade de serviço no sistema QFlow é realizada pelo controle de banda. A banda não utilizada do enlace é redistribuída de acordo com a prioridade de cada rede virtual. Além disso, dependendo de como é feito o mapeamento entre redes virtuais e filas, é possível definir-se duas modalidades diferentes para aplicar a QoS em redes virtuais:

- **Qualidade de Serviço Inter-rede** é quando associa-se cada rede virtual a uma única fila. Nesse caso, os parâmetros de QoS são apenas definidos para a rede virtual, permitindo diferenciar os serviços de uma rede virtual em relação aos serviços de outra;

- **Qualidade de Serviço Intrarrede** é quando associa-se um conjunto de filas a um único roteador virtual. Cada fila tem os seus próprios parâmetros de QoS e, dependendo da configuração de cada roteador virtual, um determinado grupo de fluxos é mapeado em cada uma das filas desse roteador virtual. Esse esquema de mapeamento de conjuntos de fluxos em filas permite que seja reservada uma determinada banda para fluxos mais sensíveis dentro de uma mesma rede virtual. O caso da QoS intrarrede é uma generalização do QoS interrede para um conjunto de fluxos e, portanto, não é excludente com o primeiro caso.

O mapeamento de fluxos em filas é realizado pela aplicação *QFlow App*, que executa sobre o NOX. No entanto, a definição de qual fila é dedicada a cada roteador virtual e, no caso intrarrede, quais características dos fluxos são consideradas para mapear cada fluxo em uma fila são configuradas no módulo *QFlow Server* e comunicadas para a *QFlow App*.

3.4 O Isolamento do Espaço de Endereçamento de Redes Virtuais com Separação de Planos

A separação de planos no sistema QFlow é alcançada da seguinte maneira. Na máquina virtual, executa o `QFlow Client` que é um aplicativo que executa em segundo plano e verifica se há atualização na tabela de rotas ou na tabela ARP da máquina virtual e as envia para o `QFlow Server`[1], no Domínio 0 da máquina física que a hospeda. O `QFlow Server` é basicamente um procurador, *proxy*, que recebe as conexões de todos os clientes, trata as mensagens, as responde e repassa as informações de todos os clientes concentradas e resumidas para a aplicação que executa sobre o controlador do comutador OpenFlow que permite o desenvolvimento de aplicações em *Python*. A aplicação desenvolvida sobre o controlador OpenFlow é a que mantém a estrutura de dados para armazenar as tabelas de rota de cada máquina virtual e, também, faz a tradução das rotas em fluxos OpenFlow. Contudo, como o Open vSwtich não é capaz de marcar os pacotes com a etiqueta de VLAN, padrão 802.1Q, ao mesmo tempo em que permite o controle pelo OpenFlow, um elemento importante da arquitetura proposta é o marcador de VLAN que se insere entre as interfaces de rede das máquinas virtuais e o comutador Open vSwitch que implementa o plano de dados OpenFlow.

A ideia chave da proposta se baseia no uso do modo OpenFlow, para ter ao mesmo tempo informações da Camada Ethernet, de VLAN e da Camada IP através dos 12 campos da especificação OpenFlow, e também na inserção do marcador de VLAN entre as interfaces virtuais e comutador OpenFlow. Assim, o pacote que é enviado ou recebido por uma máquina virtual obrigatoriamente deve marcado, ou desmarcado, com a etiqueta da VLAN a que pertence. O elemento marcado de VLAN é configurado no momento de criação das máquinas virtuais com o identificado (ID) da rede virtual que a interface de rede virtual pertence. A função do marcador de VLAN é inserir a etiqueta de VLAN em todo o pacote que saia da máquina virtual em direção ao comutador por *software* e de retirar a etiqueta de VLAN de todos os pacotes que saem do comutador em direção à máquina virtual. Esse funcionamento torna a marcação e o encaminhamento por VLAN transparente para as máquinas virtuais. O marcador de VLAN também é o responsável por garantir que os pacotes que não pertençam a uma dada rede virtual cheguem a máquinas virtuais de outras redes, pois o marcador de VLAN descarta os pacotes que chegam até ele, mas não têm a etiqueta de VLAN com o identificador correto.

A tradução de uma rota em fluxo ocorre da seguinte forma. Ao chegar um pa-

[1] A comunicação entre o *QFlow Client* e o *QFlow Server* ocorre através de interfaces de redes dedicadas à comunicação entre máquinas virtuais e Domínio 0. Todas as comunicações de controle são encriptadas e autenticadas através da troca de certificados, usando o protocolo SSL v3.0 (*Secure Sockets Layer*).

cote no comutador por *software* (Open vSwitch) de um Domínio 0, se não existir um fluxo ao qual o pacote se adeque, o pacote é enviado ao controlador OpenFlow, conforme o funcionamento normal do OpenFlow. No controlador OpenFlow, o pacote é processado pela aplicação que realiza a separação de planos e verifica a qual máquina virtual o pacote se destina de acordo com o seu endereço MAC de destino. Uma vez identificada a máquina virtual, o endereço IP de destino do pacote é verificado. Se o endereço IP de destino do pacote se adequar a alguma rota daquela máquina virtual, a aplicação extrai o IP do próximo salto através da rota na tabela de rotas da máquina virtual identificada que melhor se adeque ao endereço do pacote, algoritmo de *best match*. Após extrair o IP do próximo salto do pacote na rede, a `QFlow Application` verifica a cópia da tabela ARP da máquina virtual, para a qual o pacote se destina, se a máquina virtual já conhece o mapeamento do endereço IP no endereço MAC do próximo salto. Conhecendo o endereço MAC, a `QFlow Application` define um novo fluxo no plano de dados OpenFlow. Contudo, o roteamento ocorre entre redes distintas e, assim, entre VLANs diferentes. Assim, ao identificar a interface de saída do roteador virtual em que o pacote deve ser encaminhado, a `QFlow Application` identifica também a VLAN de saída do pacote. Para tanto, a `QFlow Application` consulta qual o marcador de VLAN está associado à interface virtual de rede, pela qual o pacote seria encaminhado caso realmente fosse encaminhado pela máquina virtual, recupera o novo identificador de VLAN do pacote e adiciona o novo fluxo no plano de dados OpenFlow. O novo fluxo é introduzido de acordo com os campos do pacote que dispararam o seu cálculo e as ações associadas a esse novo fluxo são trocar os endereços MAC de origem e destino do pacote, a troca do identificador de VLAN do pacote e, por fim, encaminhar o pacote na porta de saída adequada.

As ações configuradas para cada fluxo no comutador OpenFlow correspondem ao roteamento do pacote. A troca dos endereços MAC marca a troca do enlace pelo qual o pacote está sendo encaminhado. Nesse sentido, as ações são colocar o endereço MAC de origem do pacote como sendo o endereço da interface pelo qual o pacote seria encaminhado pela máquina virtual e colocar como MAC de destino do pacote como o endereço MAC consultado na tabela ARP da máquina virtual, este endereço é a tradução do IP do próximo salto para o MAC do próximo salto do pacote na rede. A troca do identificador de VLAN corresponde à troca do segmento Ethernet em que o pacote é difundido. A descoberta de em qual porta física do comutador OpenFlow o pacote modificado deve ser encaminhado é feita pelo mecanismo de aprendizagem do comutador. Assim, quando um pacote chega ao comutador, este armazena o endereço MAC de origem do pacote, por qual porta o aquele chegou e um temporizador associada a essa entrada. Desde que o temporizador esteja válido, o comutador sempre sabe em qual porta um endereço MAC está disponível. Se

qualquer uma dessas fases falhar, a `QFlow Application` encaminha o pacote como se fosse um comutador comum, mantendo o mesmo identificador de VLAN. Esse comportamento faz com que a arquitetura proposta se comporte tanto como um roteador, quando conhece uma rota para o pacote, ou como um comutador, quando alguma das etapas de processamento do pacote falha. No caso de o nó físico da rede não conhecer nenhuma rota para o pacote em processamento e, também, não conhecer em que porta o MAC de destino é acessível, o pacote é inundado na rede. O comportamento de inundação é o comportamento padrão de um comutador que não conhece ainda como alcançar o endereço MAC de destino.

O isolamento do espaço de endereçamento de redes virtuais depende da garantia de que os pacotes de uma rede virtual não sejam encaminhados a outras redes virtuais e que os pacotes enviados em difusão, *broadcast*, alcancem somente as estações pertencentes a essa rede virtual. Assim, os responsáveis por garantirem o isolamento mesmo no cenário de separação de planos são o marcador de VLAN e a aplicação `QFlow Application`, pois todo pacote encaminhado recebe uma etiqueta de VLAN e a troca da etiqueta ocorre sempre que houver o roteamento do pacote no plano de dados.

Ao se instanciar uma nova máquina virtual, cada interface virtual da máquina recebe o identificador da rede a qual ela pertence, chamado ID da rede. O ID da rede identifica o domínio de *broadcast* Ethernet a que aquela interface pertence. Esse procedimento assemelha-se ao descrito no isolamento de redes virtuais com o Open vSwitch. O ID da rede que a interface pertence é então atribuída ao marcador de VLAN ao qual a interface está associada. O marcador de VLAN atribui a etiqueta aos pacotes que saem da máquina virtual para o comutador por *software* e retira a etiqueta ao encaminhar os pacotes para a máquina virtual. O marcador de VLAN também o responsável por descartar os pacotes em difusão que cheguem até ele, mas não apresentam o identificador correto da rede.

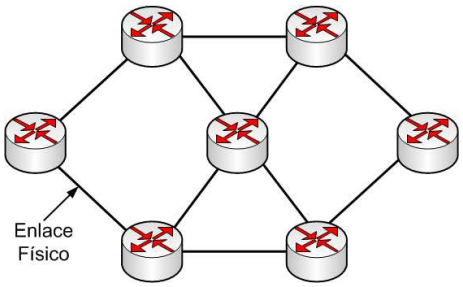
3.5 O Encaminhamento de Pacotes no QFlow

O protocolo Ethernet, quando foi originalmente criado, destinava-se a conectar estações em único enlace e, por isso, é um protocolo de camada de enlace. Contudo, com a crescente adoção do protocolo Ethernet, diversos serviços foram criados para serem executadas diretamente sobre este protocolo. Esses serviços, no entanto, estavam limitados aos cenários de uso do Ethernet, que por sua vez limitava a conectividade entre as estações à distância máxima de alguns quilômetros.

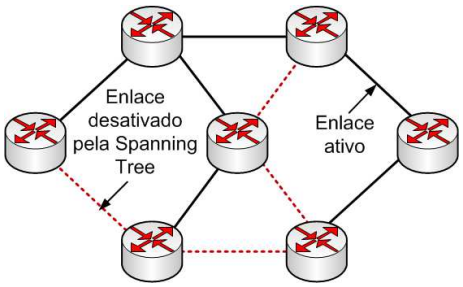
A criação de redes maiores tanto em distância entre os nós como em número de nós requer o emprego de comutadores, que são elementos encaminhadores de camada de enlace. No entanto, o encaminhamento na camada de enlace traz mais desafios do

que o encaminhamento na camada de rede, como por exemplo, o roteamento IP. Os principais desafios do encaminhamento na camada de enlace são a possibilidade de laços (*loops*), a impossibilidade de agregação de endereços e de descoberta de novos nós. A ausência de um campo de contagem de saltos no cabeçalho Ethernet, como o TTL (*Time to Live*) do cabeçalho IP, força a existência de mecanismos extras para evitar a formação de laços. A forma de endereçamento do protocolo Ethernet é plana e, por isso, não reflete a localização geográfica dos nós e não permite a agregação de endereços. A ausência de protocolos de descoberta de novos nós e de protocolos de roteamento em camada de enlace também dificultam o encaminhamento de pacotes na camada de enlace.

Uma maneira simples adotada para encaminhar pacotes na camada de enlace foi a comutação transparente. Os nós encaminhadores aprendem por qual porta um determinado nó final é acessível e passam a utilizar essa porta para encaminharem tráfego para esse nó. No entanto, esse método funciona somente quando há apenas um caminho entre quaisquer pares de nós finais. Assim, para permitir o funcionamento do encaminhamento em camada de enlace para topologias arbitrárias, foi criado o algoritmo de árvore de cobertura (*Spanning Tree Algorithm*) [55], que garante a formação de uma árvore geradora da rede. O algoritmo de árvore de cobertura garante a formação de caminhos na rede sem laços ao custo de desativar enlaces da rede física, como mostrado na Figura 3.3. O algoritmo de árvore de cobertura faz com que todo o tráfego da rede fique restrito a um único caminho, pertencente à árvore de cobertura. Uma consequência desse algoritmo é a sua instabilidade. A perda de conectividade em um enlace pertencente à árvore de cobertura pode significar o recálculo de toda a árvore e consequentes mudanças em toda a sua organização [55]. Dessa forma, o algoritmo de árvore de cobertura apresenta comportamento instável em redes extensas interligadas por comutadores.



(a) Rede física com todos os enlaces ativos, resultando em múltiplos caminhos e possibilidade de laços.



(b) Rede física com enlaces desativados pelo protocolo da *Spanning Tree*, resultando em um único caminho para se chegar a todos os nós e, portanto, impossibilidade de laços.

Figura 3.3: O protocolo de *Spanning Tree* em uma rede com enlaces redundantes.

3.5.1 O Encaminhamento em Redes Virtuais

Árvore de Cobertura no OpenFlow

O protocolo de Árvore de Cobertura é um algoritmo distribuído. Quando controlando uma rede que segue o paradigma de Rede Definida por Software (*Software Defined Networking* – SDN), uma variação do protocolo executa de forma centralizada em um nó controlador, como o implementado pela aplicação de *Spanning Tree* do OpenFlow [56]. A ideia básica da aplicação de árvore de cobertura para o OpenFlow é calcular uma árvore geradora para os nós da rede OpenFlow com base na topologia da rede. Os enlaces pertencentes à árvore geradora calculada são então habilitados como *flood*, ou seja, permite o encaminhamento de pacotes em difusão. Os demais enlaces são desabilitados. Essa abordagem é possível em redes OpenFlow, pois o controlador tem a visão geral da rede, inclusive da topologia completa da rede. Contudo, essa abordagem sofre das mesmas deficiências do protocolo *Spanning Tree* convencional, pois desabilita enlaces da rede e depende da convergência de um algoritmo de descoberta de topologia para a geração correta da árvore de cobertura. Essa abordagem ainda restringe o seu uso em ambientes de rede em que o controle da rede seja centralizado.

3.5.2 O Encaminhamento em Difusão no QFlow

O problema dos pacotes em difusão assume uma ótica diferente no QFlow. Os nós de uma rede QFlow agem de maneira distribuída e devem ser capazes de definir os caminhos para o encaminhamento de pacotes Ethernet de forma autônoma, assim como fazem as pontes Ethernet. Um nó QFlow deve ser capaz de aprender em que porta cada outro nó da rede é acessível e realizar o encaminhamento de quadros Ethernet em difusão sem que haja a formação de laços (*loop*) na rede. Dessa forma, o comportamento de um nó QFlow para o encaminhamento de pacotes Ethernet se divide em duas etapas. A primeira é o aprendizado de caminhos, na qual o nó descobre por qual porta alcança cada um dos outros nós da rede. A segunda é o encaminhamento de pacotes em difusão.

O Aprendizado de Caminhos

O funcionamento do aprendizado de caminhos nos nós QFlow se baseia no processamento dos pacotes recebidos. Ao receber um pacote, por uma dada porta, o nó QFlow verifica se já conhece o endereço MAC de origem desse pacote. Caso não o conheça, o nó cria e armazena uma entrada em sua tabela de encaminhamento, relacionando o endereço MAC aprendido à porta pela qual o pacote foi recebido. Essa entrada na tabela de aprendizagem define que o MAC aprendido está acessível

na porta relacionada. Cada entrada na tabela de aprendizagem é também relacionada a um temporizador que determina a validade da entrada. Esse primeiro caso é igual ao aprendizado de um comutador Ethernet. O segundo caso de aprendizado do MAC é quando o endereço MAC de origem do pacote já está referenciado na tabela de aprendizagem. Nesse caso, há duas possibilidades: a porta pela qual chegou o pacote é igual à porta aprendida anteriormente, ou o pacote chegou por outra porta que não é a aprendida. Na primeira hipótese, a única ação a ser tomada é a atualização do temporizador referente à entrada do MAC na tabela de aprendizagem. Considerando a segunda hipótese, na qual a porta pela qual o pacote chegou é diferente da aprendida, o algoritmo de aprendizagem verifica a necessidade de atualizar a entrada na tabela de aprendizagem. A entrada só é atualizada caso o tempo de validade da entrada previamente definida tenha expirado. Caso contrário, o processamento do pacote continua sem que nenhuma ação de aprendizagem seja tomada. O funcionamento do algoritmo de aprendizagem é ilustrado pelo Algoritmo 2. A ideia principal desse algoritmo é o nó QFlow manter uma referência atualizada para porta em que pode alcançar cada nó da rede.

Algoritmo 2: Atualização da tabela de aprendizagem com a chegada de um novo pacote.

Entrada: $Pacote, Porta\_Entrada$
Saída: $Aprendizado\_MAC[\,]$

```
1  MAC_CACHE = 5s
2  Aprendizado_MAC = Aprendizado_MAC_anterior
3  MAC_src = Pacote[MAC_src]
4  se (MAC_src ∉ Aprendizado_MAC) então
5  |  Aprendizado_MAC[MAC_src] = (Porta_Entrada, time())
6  fim
7  senão
8  |  Porta_Aprendida = Aprendizado_MAC[MAC_src][0]
9  |  se (Porta_Aprendida = Porta_Entrada) então
10 |  |  Aprendizado_MAC[MAC_src][1] = time()
11 |  fim
12 |  senão
13 |  |  Temporizador = Aprendizado_MAC[MAC_src][1]
14 |  |  se (Temporizador − time() > MAC_CACHE) então
15 |  |  |  Aprendizado_MAC[MAC_src] = (Porta_Entrada, time())
16 |  |  fim
17 |  |  senão
18 |  |  |  Retorna Aprendizado_MAC
19 |  |  fim
20 |  fim
21 fim
22 Retorna Aprendizado_MAC
```

Outro ponto principal no algoritmo de aprendizagem é o tratamento especial dado ao Pacote de Apresentação. O Pacote de Apresentação é um pacote *ARP Reply* (*Address Resolution Protocol*) com um endereço de destino específico e definido. O Pacote de Apresentação é um *ARP Reply* enviado periodicamente pelas máquinas virtuais que estão executando sobre a plataforma de virtualização QFlow para o comutador por software do servidor que a hospeda. O Pacote de Apresentação, no entanto, não é repassado para a rede, apenas é descartado após o processo da aprendizagem. A semântica desse pacote é que a máquina virtual está se apresentando ao domínio de controle, informando em quais interfaces virtuais, suas interfaces de rede estão mapeadas. De uma forma simples, o Pacote de Apresentação informa ao controlador QFlow qual porta do comutador local do domínio de controle, Domínio 0 no Xen, está ligada a cada máquina virtual, sob qual IP e qual MAC. Essa informação é importante no momento da migração, pois permite uma reação rápida da rede à mudança do remapeamento da topologia lógica sobre a física.

O Pacote de Apresentação tem prioridade na definição da aprendizagem de MACs. Ao receber um Pacote de Apresentação, o nó QFlow adiciona a nova entrada à tabela de aprendizagem, independentemente de haver outra entrada ainda com o tempo de validade sem expirar. Esse comportamento é importante para a migração de máquinas virtuais.

O Encaminhamento de Pacotes em Difusão

Cada nó QFlow apresenta o seu próprio controlador. Dessa forma, como o controle da rede QFlow é distribuído, a aplicação do OpenFlow para *Spanning Tree*, centralizada, não atende à rede QFlow, já que essa aplicação pressupõe que o controle da rede é centralizado e que o nó controlador detenha controle global da rede. Paralelamente, o funcionamento do protocolo de *Spanning Tree* comum não é adequado ao QFlow, pois implica na desativação de enlaces da rede para evitar laços, o que pode causar quebra de conectividade nas redes virtuais e limitar a banda agregada da rede. Já as alternativas TRILL [26] e DOVE [25] exigem o encapsulamento de pacotes. O encapsulamento de pacotes limita a compatibilidade do sistema àqueles que reconheçam o encapsulamento. No caso do QFlow, um nó é capaz de se comunicar com um nó de rede atual, como um roteador, sem que haja a necessidade de um nó especial para desencapsular ou alterar os pacotes. O QFlow usa somente os dados dos cabeçalhos dos pacotes para realizar as decisões de aprendizado e de encaminhamento.

Assim, para realizar o encaminhamento de pacotes em difusão sem que haja a formação de laços na rede, o QFlow depende da construção distribuída de uma árvore para o encaminhamento dos pacotes de difusão. Para tanto, é aplicado um algoritmo simples. O algoritmo de difusão de pacotes é o seguinte. Ao receber um pacote, o

Algoritmo 3: Encaminhamento de pacotes em difusão no QFlow.

Entrada: $Pacote, Porta\_Entrada$

```
1  MAC_src = Pacote[MAC_src]
2  Porta_Aprendida = Aprendizado_MAC[MAC_src][0]
3  se (Tipo(Pacote) = broadcast) então
4  |  se Porta_Entrada = Porta_Aprendida então
5  |  |  Encaminhamento_Difusão(Pacote, Porta_Entrada)
6  |  fim
7  |  senão
8  |  |  Descarte_Pacote(Pacote)
9  |  fim
10 fim
11 senão
12 |  Encaminhamento_XenFlow(Pacote, Porta_Entrada)
13 fim
```

comutador marca em qual porta o pacote chegou e o endereço MAC do nó que o enviou. Caso essa porta seja a mesma que o comutador usaria para enviar um pacote para o nó que originou o envio de pacotes em difusão, o pacote recebido é então enviado nas demais portas desse comutador. No caso em que o pacote é recebido por uma porta que não é a que o comutador usaria para acessar o nó que iniciou a difusão, o pacote é então ignorado. Esse procedimento é para evitar que demais cópias de um mesmo pacote sejam reenviadas na rede, já que os pacotes em difusão só são enviados uma única vez por cada nó QFlow. O encaminhamento de pacotes em difusão no QFlow é realizado de acordo com o Algoritmo 3. O Algoritmo 3 baseia-se na técnica de construção de uma árvore para o encaminhamento de pacotes em difusão de acordo com o caminho reverso do nó de encaminhamento em relação ao nó gerador da difusão, a técnica chamada *Reverse Path Forwarding*, usada nos protocolos de roteamento *multicast* como MSDP (*Multicast Source Discovery Protocol*) e PIM-SM (*Protocol Independent Multicast - Sparse-Mode*/abbrevPIM-SM*Protocol Independent Multicast - Sparse-Mode*).

Capítulo 4

Avaliação de Desempenho do Sistema QFlow

O protótipo do sistema QFlow desenvolvido se serve do Xen 4.0 para prover os planos de controle e do Open vSwitch 1.2.2 para prover a função de encaminhamento no plano de dados do sistema. O Open vSwitch [16] é configurado para ser usado pelo controlador NOX [35] do OpenFlow. A aplicação, `QFlow Application`, que realiza a separação de planos e o direcionamento dos pacotes para as filas adequadas no plano de encaminhamento de dados foi escrita em Python e executa sobre o NOX. O controlador de qualidade de serviço, o qual realiza os ajustes dos valores máximo e mínimo de cada fila, foi escrito em Python e interage diretamente com as interfaces de configuração do Open vSwitch. As ferramentas `Iperf`[1] e `Tcpdump`[2] foram usadas para realizar as medidas de desempenho do sistema.

Quatro computadores pessoais compõem o cenário dos experimentos: um executa o protótipo QFlow ligando-se a outros três. Esse computador com QFlow tem a função de receber/encaminhar pacotes para outros dois computadores que geram ou recebem tráfego. O quarto computador controla o ambiente de testes. Todos os computadores possuem processadores Intel Core 2 Quad 2,4 GHz e 4 GB de memória RAM. Cada computador possui, no mínimo, 3 interfaces de rede sendo que todas são configuradas para funcionarem a 100 Mb/s, uma vez que havia também interfaces de 1 Gb/s. O computador que desempenha a função de encaminhador e executa o sistema QFlow hospeda três máquinas virtuais que realizam a função de roteador. Cada máquina virtual é configurada com uma CPU virtual, 128 MB de memória RAM e executa o Debian Linux 2.6-32-5. As máquinas virtuais executam os protocolos de roteamento através da plataforma XORP [54], contudo, durante os testes, foram configuradas rotas estáticas. Os resultados apresentados nessa seção são médias de 10 rodadas de cada experimento, com intervalo de confiança de 95%.

[1] http://iperf.sourceforge.com.
[2] http://www.tcpdump.org.

O roteamento no sistema QFlow é realizado no plano de dados e consiste no direcionamento de cada pacote para a fila de qualidade de serviço adequada, assim como na troca dos endereços MAC de origem e de destino de cada pacote entrante no roteador. O MAC de origem é trocado para o endereço MAC da interface de saída do pacote do roteador e o MAC de destino é trocado pelo o endereço MAC da interface de entrada no próximo salto do caminho do pacote na rede.

4.1 Migração de Roteador Virtual

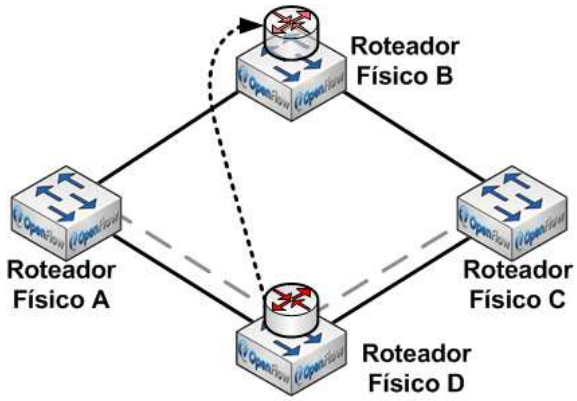


Figura 4.1: Topologia usada no experimento de migração de um roteador virtual.

O experimento da migração no sistema QFlow tem como objetivo avaliar o comportamento do encaminhamento de pacotes na rede durante a migração do roteador virtual que os encaminha. A topologia de rede utilizada nesse experimento é mostrada na Figura 4.1. O experimento consiste em realizar uma transferência de dados UDP a taxa de 6 Mb/s do Roteador Físico A para o Roteador Físico C, sendo que o tráfego é encaminhado pelo roteador virtual inicialmente hospedado no Roteador Físico D. Após 30 s, ocorre a migração do Roteador Virtual que estava hospedado no Roteador Físico D para ser hospedado pelo Roteador Físico C. A duração total da transmissão de dados é de 120 s. A Figura 4.2 apresenta o resultado do experimento. O resultado revela que a migração realizada pelo aplicativo de migração nativo do Xen, indicada na figura por `Xen`, implica a interrupção do encaminhamento de dados por aproximadamente 50 s. Essa interrupção no encaminhamento dos pacotes, na migração nativa do Xen, ocorre devido à desativação da máquina virtual que está encaminhando os dados por um determinado período de tempo até que sua memória seja transferida para máquina física de destino, mantendo-se a consistência. Assim, na migração nativa do Xen os pacotes são encaminhados necessariamente pela máquina virtual e, portanto, a interrupção da máquina virtual resulta na interrupção do encaminhamento dos pacotes. A migração sem perdas só é possível no modelo de separação de planos, no qual os pacotes são encaminhados pelo Domínio 0, plano

de dados, de acordo com as informações do plano de controle. Assim, enquanto a máquina virtual é migrada para o roteador físico de destino, o plano de dados permanece ativo no roteador físico de origem encaminhando os pacotes das conexões previamente estabelecidas. O sistema QFlow emprega o paradigma da separação de planos e a Figura 4.2 demonstra que a migração de um roteador virtual no sistema QFlow não introduz perda de pacotes.

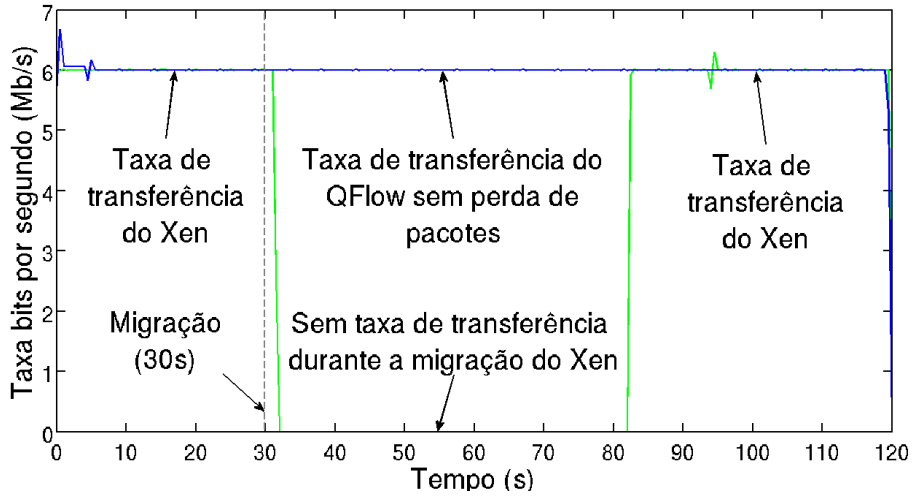
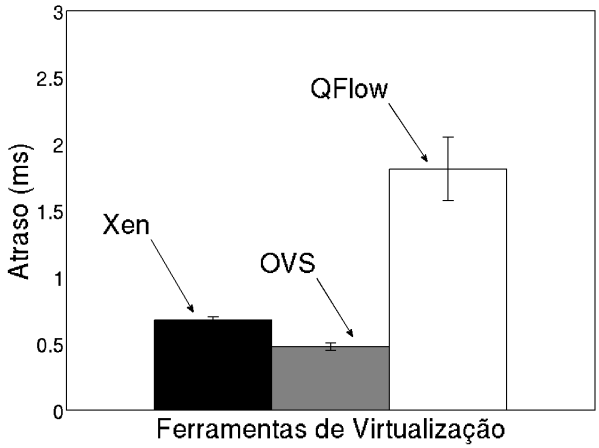


Figura 4.2: Comparação entre a migração no Xen e no QFlow. Após 30 s o roteador virtual que encaminha um tráfego de 6 Mb/s é migrado. No QFlow, a migração ocorre sem que haja a interrupção do encaminhamento dos dados.

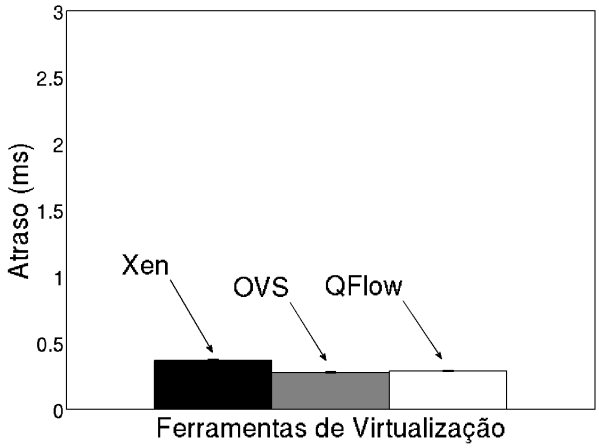
4.2 Qualidade de Serviço

O primeiro experimento de Qualidade de Serviço tem como objetivo avaliar o atraso introduzido pelo sistema QFlow no encaminhamento dos pacotes. A Figura 4.3 compara o atraso introduzido pelo roteamento QFlow com o atraso do Open vSwitch atuando como comutador, referenciado como `OVS`, e também com o atraso quando a função de roteamento é executada na máquina virtual Xen, referenciado como `Xen`. Neste experimento, o Open vSwitch foi configurado para somente comutar os pacotes entre as interfaces do nó encaminhador, agindo de forma semelhante a uma *bridge*; o roteamento pelo Xen foi configurado para os pacotes serem roteados pela máquina virtual e, por questão de justiça na avaliação, o encaminhamento dos pacotes da interface de rede física para a interface de rede da máquina virtual é realizado por uma instância de comutador por *software* Open vSwitch; e o QFlow realiza a separação de planos, o isolamento de recursos e a garantia de QoS baseado na aplicação do *QFlow App*. Os resultados de atraso mostrados na Figura 4.3(a) revelam que o QFlow introduz um aumento do atraso no encaminhamento do primeiro pacote da ordem de 1 ms, quando comparado com as outras

ferramentas. Esse atraso é referente ao encaminhamento do primeiro pacote de cada fluxo para o NOX, para que esse processe as informações do pacote e instale um novo fluxo no comutador OpenFlow do plano de dados. No encaminhamento dos demais pacotes, o atraso introduzido pelo QFlow é próximo ao atraso intrínseco do encaminhamento pelo Open vSwitch, já que o fluxo já está instalado no plano de dados. Isso mostra que as ações de troca de endereços MAC e direcionamento de pacotes a filas não interferem de forma significativa no atraso de processamento dos pacotes. No Xen, o atraso de encaminhamento dos pacotes é superior aos demais, pois cada pacote passa duas vezes pelo plano de dados, uma na chegada ao nó físico, quando é redirecionado ao plano de controle, e outra quando deixa o plano de controle, volta ao plano de dados e só então é transmitido no enlace de saída.



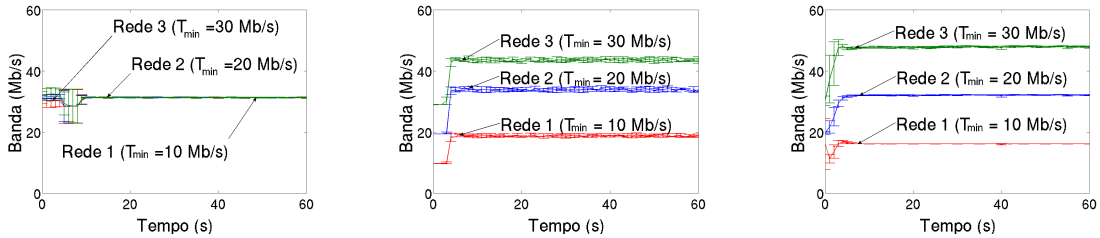
(a) Primeiro pacote do fluxo.



(b) Demais pacotes do fluxo.

Figura 4.3: Comparação dos atrasos introduzidos pelo encaminhamento de pacotes de um fluxo.

O segundo experimento avalia o comportamento dos três sistemas, QFlow, Open



(a) Encaminhamento pela máquina virtual Xen. Os fluxos não são diferenciados na situação de saturação do meio físico

(b) Encaminhamento pelo Open vSwtich. Os fluxos são diferenciados e isolados, atendendo a taxa mínima.

(c) Encaminhamento pelo QFlow. Os fluxos são diferenciados e isolados, atendem a taxa mínima e o valor restante é dividido proporcionalmente.

Figura 4.4: Distribuição da capacidade disponível no enlace entre redes virtuais.

vSwitch e Xen, para a diferenciação de pacotes, que é uma forma de se avaliar o isolamento entre os diferentes roteadores virtuais e também a oferta de qualidade de serviço. A plataforma Xen não é capaz de prover diferenciação de serviço em redes virtuais nativamente porque não provê qualquer tipo de controle de banda. No cenário com Open vSwitch, para realizar diferenciação de pacotes foi usada a mesma aplicação de direcionamento de pacotes a filas usada no QFlow, porém não é usado o controlador de banda do QFlow. Esse experimento consiste em criar três redes virtuais, `Rede 1`, `Rede 2` e `Rede 3`, cada uma com, respectivamente, limite mínimo garantido de 10 Mb/s, 20 Mb/s e 30 Mb/s de banda. Para a realização do experimento, cada rede virtual é definida como um fluxo entre o gerador de tráfego e o receptor. O gerador de tráfego de cada uma das três redes transmite, a uma taxa constante, 60 Mb/s de tráfego UDP de pacotes de 1472 B[3] para cada rede virtual. A banda total requerida pelas três redes virtuais é então de 180 Mb/s e, portanto, superior a capacidade dos enlaces de 100 Mb/s. A Figura 4.4 mostra como os limites são fornecidos em cada sistema. Nos três casos percebe-se a saturação dos enlaces de 100 Mb/s. No Xen, os limites mínimos não são respeitados. No Open vSwitch, cada rede virtual é direcionada a uma fila e, portanto, tem o limite mínimo da fila garantido. Contudo, o restante da banda não atribuída do enlace é igualmente dividido entre os fluxos, não sendo possível definir uma relação de prioridade entre eles. Já no QFlow, a redistribuição dos recursos ociosos é realizada de forma a privilegiar as redes virtuais com maior limite mínimo, ou seja, privilegiar a de maior prioridade. Como mostrado na Figura 4.4, a banda não usada do enlace é proporcionalmente distribuída entre as redes virtuais.

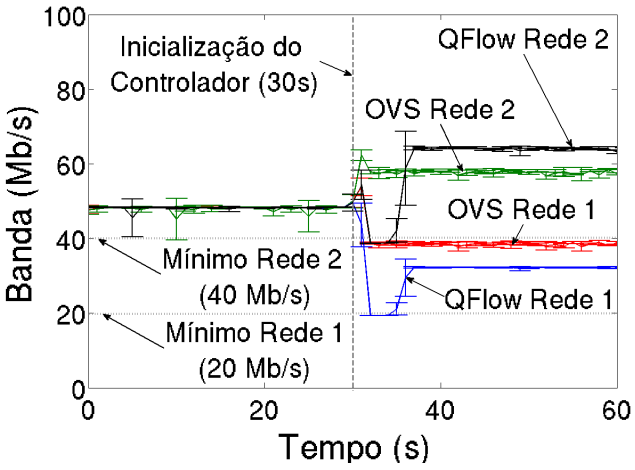
A divisão dos recursos ociosos do enlace deve respeitar a prioridade de cada rede virtual. Para avaliar o quanto o sistema QFlow respeita a prioridade de cada rede

[3]A carga útil dos pacotes gerados é de 1472 B, que somados aos cabeçalhos do UDP e do IP resulta um tamanho total de 1500 B, que corresponde ao tamanho máximo de conteúdo de um quadro Ethernet.

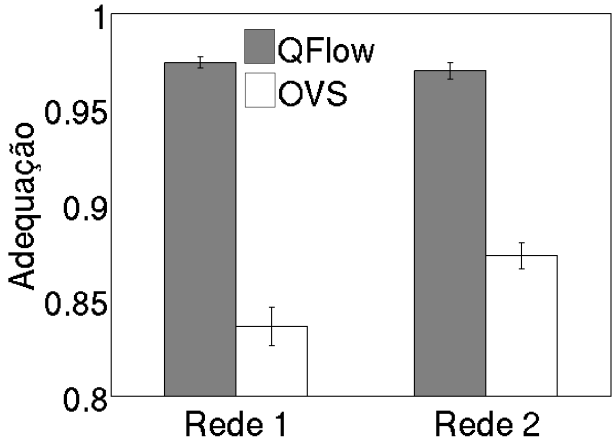
virtual, foi definida a métrica

$$adequacao = 1 - \left| 1 - \frac{Banda\ Obtida}{Banda\ Esperada} \right|, \tag{4.1}$$

que calcula o percentual de acerto da redistribuição de recursos de acordo com a prioridade de cada rede. A métrica determina o quão próximo a banda destinada a uma rede virtual está ao seu valor ideal estimado.



(a) Controlador iniciado após 30 s.



(b) Adequação das redes ao SLA.

Figura 4.5: Adequação dos controlador de QoS aos SLAs, comparado ao comportamento do Open vSwtich com parâmetros de QoS.

O terceiro experimento verifica a eficácia do controlador de QoS do QFlow, no que se refere a uma redistribuição diferenciada e proporcional de recursos, e compara os resultados com a distribuição de recursos ociosos do Open vSwitch. Para tanto, configurou-se o Open vSwitch para direcionar os fluxos de cada rede virtual para uma determinada fila e configurou-se a banda mínima garantida de cada fila de acordo com a definição da rede e o valor máximo, por questão de justiça na comparação com

o QFlow, foi configurado para a banda máxima do enlace, 100 Mb/s. No QFlow, definiu-se a banda mínima reservada para cada fila e a banda máxima é estabelecida pelo controlador. A Figura 4.5 apresenta o experimento para dois fluxos UDP de pacotes de 1472 B para duas redes virtuais diferentes, uma com limite mínimo de 20 Mb/s e outra com limite mínimo de 40 Mb/s. A dinâmica do experimento é a seguinte. O experimento é iniciado em 0 s. No início do experimento, nenhum limite de banda é definido e o enlace é compartilhado totalmente pelas duas redes. Em 30 s, são definidos os limites mínimo e máximo das filas no cenário do Open vSwitch e é iniciado o controlador de QoS no cenário do QFlow. O experimento é encerrado aos 60 s.

A Figura 4.5 evidencia a eficácia da redistribuição diferenciada e proporcional a qualidade de serviço especificada na abordagem QFlow. A rede com maior banda mínima possui uma fila com maior peso na redistribuição de recursos e, consequentemente, tem acesso a uma maior parcela de largura da banda da repartição do excedente dos recursos ociosos do enlace. Na abordagem nativa do Open vSwitch os recursos livres no enlace são igualmente divididos entre todas as filas. Já a Figura 4.5(b) mostra a `adequação`, após a definição dos limites de cada fila. A `adequação` do QFlow é, em torno de, 14% superior a do Open vSwitch, chegando a 97%. Assim, o controlador de QoS do QFlow mostra-se mais apto a fornecer qualidade de serviço para fluxos com prioridade do que o uso simplesmente das primitivas de QoS do OpenFlow implementadas pelo Open vSwitch.

4.3 Isolamento de Redes Virtuais

No experimento de isolamento de redes virtuais, dois computadores pessoais compõem o cenário dos experimentos, conforme mostrado na Figura 4.6. Ambos executam o protótipo do sistema proposto. Nos computadores pessoais foram instanciadas três máquinas que virtuais que apresentam a função de emissor, encaminhador e receptor de pacotes. Todos os computadores possuem processadores Intel Core 2 Quad 2.4 GHz e 4 GB de memória RAM. Cada computador possui, no mínimo, 2 interfaces de rede sendo que todas são configuradas para funcionarem a 100 Mb/s, uma vez que havia também interfaces de 1 Gb/s. As três máquinas virtuais que realizam a função de roteador são configuradas com uma CPU virtual, 128 MB de memória RAM e executa o Debian Linux 2.6-32-5. Os resultados apresentados são médias, com intervalo de confiança de 95%.

O primeiro experimento avalia a capacidade do sistema QFlow para reagir à definição de fluxos em rajadas. Para tanto, a rede de testes foi configurada para uma máquina virtual hospedada no nó físico 1 se comunicar com outra máquina virtual hospedada no nó físco 2, através de um roteador virtual também hospedado

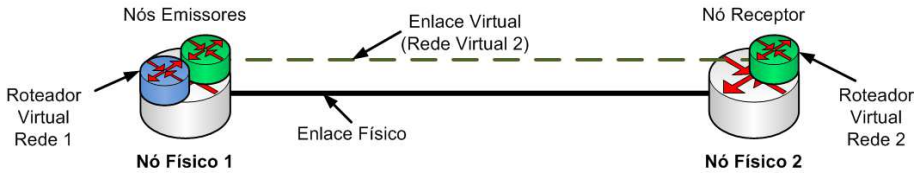


Figura 4.6: Instalação experimental. As máquinas virtuais no nó físico 1 agem como geradores de pacote das redes 1 e 2. As máquinas virtuais no físico 2 agem como receptores.

no nó físico 2. A geração de novos fluxos foi realizada pelo aplicativo httperf[4] para gera uma taxa constante de conexões HTTP por segundo. O resultado do teste avalia a taxa de conexões atendidas. A Figura 4.7 evidencia que a proposta de isolamento de redes virtuais alcança, nesse cenário, uma taxa de aproximadamente 700 fluxos/s, que é superior ao cenário em que o encaminhamento em Camada 2 do OpenFlow padrão, executado pela aplicação de comutação `forwarding.l2_-learning` do controlador OpenFlow POX[5]. O cenário OpenFlow foi testado sob duas hipóteses: usando a marcação de VLANs, referenciado na figura como `OpenFlow com Isolamento`; e sem a marcação de VLANs, `OpenFlow sem Isolamento`. Em ambos os casos o desempenho do OpenFlow foi inferior ao da arquitetura proposta. A melhora introduzida pela proposta deve-se à implementação da separação entre planos de controle e dados, enquanto no OpenFlow age como um comutador somente, enviando os pacotes ao roteador virtual.

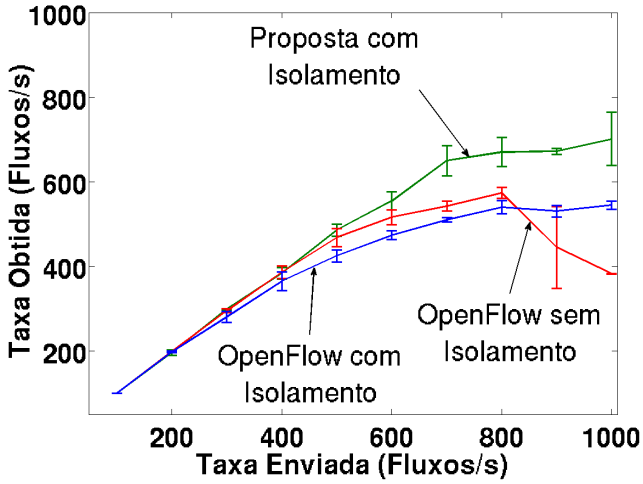
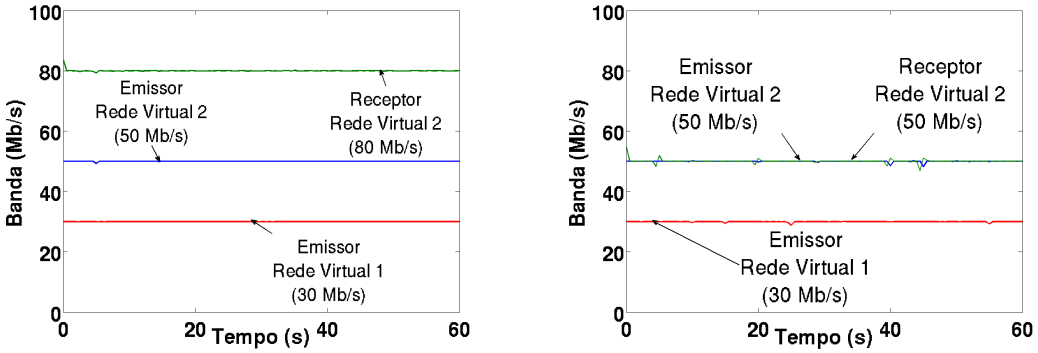


Figura 4.7: Taxa de fluxos configurados com sucesso em relação a taxa de novos fluxos submetidos ao sistema. Os experimentos foram realizados entre uma máquina virtual hospedada no nó físico 1 se comunicando com duas outra máquinas virtuais hospedada no nó físico 2.

O segundo experimento verifica a eficácia do mecanismo isolamento da comu-

[4]http://www.hpl.hp.com/research/linux/httperf/
[5]http://www.noxrepo.org/pox/about-pox/.

nicação de redes virtuais. A ideia central desse experimento é definir duas redes virtuais. A Rede Virtual 1 é composta por uma máquina virtual hospedada no Roteador Físico 1. A Rede Virtual 2 é composta por uma máquina virtual hospedada no Roteador Físico 1 e outra hospedada no Roteador Físico 2. Ambos os nós do Roteador Físico 1 são emissores de dados UDP de 1472 B. Todas as máquinas virtuais foram configuradas para pertencerem ao mesmo espaço de endereçamento IP e enviam dados para um mesmo IP de destino. No entanto, como as redes virtuais são isoladas, espera-se que o fluxo da Rede Virtual 1 não interfira no fluxo da Rede Virtual 2. A Figura 4.8(a) mostra que no cenário sem isolamento, o nó receptor da Rede Virtual 2 recebe tanto os pacotes da Rede Virtual 2, como os pacotes da Rede Virtual 1, rompendo com o isolamento da comunicação de redes virtuais. Já a Figura 4.8(b) demonstra que a proposta isola o espaço de endereçamento de endereçamento de cada rede virtual, pois o tráfego da Rede Virtual 1 não interfere no tráfego da Rede Virtual 2, já que o receptor da Rede Virtual 2 só recebe os pacotes pertencentes à sua rede.



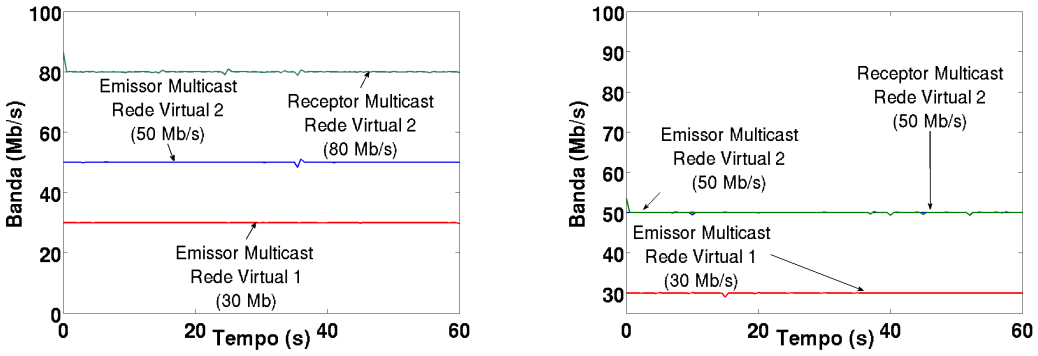
(a) Sem isolamento de comunicação.

(b) Com isolamento da comunicação.

Figura 4.8: Encaminhamento de pacotes para a Rede Virtual 1 e para Rede Virtual 2 que possuem o mesmo endereço IP. a) redes não isoladas - as duas redes recebem os pacotes encaminhados para o IP comum, pois a Rede 2 recebe a soma (80 Mb/s) da taxa relativa aos pacotes da Rede 1 (30 Mb/s) mais a taxa de pacotes da Rede 2 (50 Mb/s). b) redes isoladas - cada rede virtual só recebe os pacotes que lhe são destinados, pois a Rede 2 só recebe a taxa de 50 Mb/s.

A Figura 4.9 mostra o isolamento da comunicação entre redes virtuais no cenário de uma comunicação *multicast*. Nesse caso, o cenário é semelhante ao do experimento anterior. No entanto, ao invés de os nós emissores enviarem pacotes para um dado endereço IP *unicast*, os nós agem como fontes de dados *multicast* para um dado grupo. O nó receptor age como sorvedouro desse grupo. Esse teste é importante, pois protocolos de roteamento, como o OSPF (*Open Shortest Path First*), usam comunicação *multicast* para descoberta de topologia, logo, é importante que os pacotes *multicast* de uma dada rede virtual não sejam encaminhados para outras redes. A

Figura 4.9(a) mostra que no caso sem isolamento de comunicação, o nó receptor da Rede Virtual 2 recebe tanto os pacotes de sua rede, quanto os da Rede Virtual 1. Contudo, ao usar o isolamento de comunicação de redes proposto, o nó receptor da Rede Virtual 2 só recebe os pacotes enviados pelo nó emissor de sua mesma rede, como mostrado na Figura 4.9(b).



(a) Sem isolamento de comunicação.

(b) Com isolamento da comunicação.

Figura 4.9: Encaminhamento de pacotes para um endereço de um grupo IP *multicast*. a) redes não isoladas - as duas redes recebem os pacotes encaminhados para o grupo *multicast* comum, pois a Rede 2 recebe a soma (80 Mb/s) da taxa relativa aos pacotes da Rede 1 (30 Mb/s) mais a taxa de pacotes da Rede 2 (50 Mb/s). b) redes isoladas - cada rede virtual só recebe os pacotes que lhe são destinados, pois a Rede 2 só recebe a taxa de 50 Mb/s.

Capítulo 5

Conclusões

Este trabalho propôs o sistema QFlow, que é capaz de isolar o consumo de recursos de cada rede virtual e ainda provê Qualidade de Serviço (QoS) tanto para uma rede virtual, como para fluxos, ou conjunto de fluxos, de uma mesma rede virtual. O sistema baseia-se em uma arquitetura híbrida de redes virtuais em que o plano de controle executa em uma máquina virtual Xen e o plano de dados é realizado em uma matriz de comutação de pacotes que implementa a interface de programação de aplicação OpenFlow. A ideia chave do sistema QFlow é atribuir filas de encaminhamento no plano de dados a uma rede virtual e realizar o controle de banda sobre as filas no plano de dados. Um protótipo do sistema foi implementando e avaliado. Os resultados demonstram que o mapeamento dos pacotes nas filas de encaminhamento adequadas a cada rede virtual e aos seus requisitos de QoS introduz um atraso de aproximadamente 1 ms somente no primeiro pacote de cada fluxo. Contudo, esse atraso é apenas para o primeiro pacote de um fluxo e não afeta os demais pacotes do fluxo. O controlador de recursos do QFlow alcançou a utilização máxima dos enlaces aliada a uma eficiência de distribuição proporcional de recursos de aproximadamente 97%, o que é 14% superior à eficiência do Open vSwitch.

O sistema QFlow ainda implementa a separação do espaço de endereçamento das redes virtuais através do emprego de um algoritmo de aprendizagem das redes virtuais e da divulgação das máquinas virtuais que participam de uma dada rede. Tais mecanismos associados a um algoritmo de encaminhamento de pacotes de difusão permitem ao QFlow encaminhar os pacotes de uma dada rede virtual somente para os roteadores virtuais que pertencem a ela. O sistema QFlow provê o isolamento completo entre redes virtuais, isolando tanto a comunicação de redes virtuais quanto o consumo de recursos de cada rede virtual. Assim, um das contribuições dessa dissertação é realizar o roteamento dos pacotes entre as redes virtuais distintas sem que o pacote deixe o plano de dados. O isolamento das redes virtuais ocorre adicionando a identificação da VLAN de destino, associando um identificador de VLAN (*Virtual*

Local Area Network) a cada segmento de rede virtual às ações tomadas no plano de dados ao rotear pacotes. Logo, a ideia chave do isolamento de redes virtuais no sistema QFlow se baseia no uso do Open vSwitch no modo OpenFlow, para ter ao mesmo tempo informações da Camada Ethernet, de VLAN e da Camada de Rede, IP, através dos 12 campos da especificação OpenFlow, e também na inserção do marcador de VLAN entre as interfaces virtuais e comutador OpenFlow.

A avaliação do protótipo do sistema QFlow demonstra que o sistema proposto alcançou uma taxa de definição de fluxos por segundo superior à alcançada pela aplicação padrão de comutação do OpenFlow. Por fim, os resultados mostram que o isolamento da comunicação entre redes virtuais é alcançado mesmo no cenário em que as redes virtuais executam aplicações de *multicast* ou compartilham o mesmo espaço de endereçamento IP.

Vale ainda ressaltar que um dos preceitos do OpenFlow é permitir o desenvolvimento de inovações nas redes dos *campi* universitários. Logo, o OpenFlow é uma ferramenta de experimentação em redes. Assim, as ferramentas apresentadas são reunidas em uma arquitetura de virtualização de redes e formam um ambiente de experimentação de redes que contempla os conceitos de virtualização de redes e de Redes Definidas por *Software*. Dessa forma, um dos desdobramentos desse trabalho é a ferramenta FITS[1], *Future Internet Testbed with Security* [57]. O FITS é uma arquitetura de *software* reutilizável e extensível capaz de implantar um ambiente de experimentação para propostas de Internet do Futuro. O principal objetivo do FITS é proporcionar uma infraestrutura aberta, compartilhada e de propósito geral para testes, que permita avaliar o desempenho de propostas inovadoras. O ambiente de experimentação proposto oferece uma abordagem pluralista, na qual os pesquisadores podem optar por terem acesso total às redes virtuais espalhadas nos diversos nós físicos, através da abordagem de virtualização de redes do QFlow. O ambiente de experimentação proposto também permite escolher entre o encaminhamento de pacotes através das máquinas virtuais ou adotar uma abordagem de separação de planos, em que o encaminhamento de pacotes é realizado pelo domínio privilegiado, de acordo com as informações de controle oriundas da máquina virtual. Assim, na abordagem de separação de planos, os pacotes de dados são encaminhados para seu destino final sem que passarem pela máquina virtual. A abordagem separação de planos aumenta a capacidade de encaminhamento da rede virtual e também permite à máquina virtual liberar recursos para executar outros processos ao invés de processar o encaminhamento de pacotes. Outra característica inovadora do ambiente de experimentação proposto neste trabalho é a migração de redes virtuais. A

[1]A rede de testes FITS desenvolvida pelo Grupo de Teleinformática e Automação (GTA). Em parceria com outras instituições no Brasil e na Europa, o GTA opera esta ferramenta de teste. Atualmente, o FITS conta com o desenvolvimento de inovações e funcionalidades realizados por diversos participantes do GTA.

migração de rede virtual permite que os pesquisadores, ao usarem o ambiente de experimentação proposto, possam realocar suas redes virtuais, sem que haja perda de pacotes ou interrupção de suas experiências.

No FITS, os usuários podem optar por realizar a experimentação com base em uma Rede Definida por *Software* baseada em OpenFlow. O paradigma de Redes Definidas por Software (SDN – *Software Defined Networking*) se baseia na ideia de separar a função do elemento de rede da sua realização física. O paradigma do SDN prevê que o encaminhamento de pacotes deve ser realizado por *hardware* especializado em encaminhar pacotes associado a uma interface de programação que permite a redefinição da lógica de encaminhamento de pacotes de acordo com um *software* de controle. Esse ideia implica em transformar o *hardware* de rede em um equipamento encaminhador de pacotes de propósito geral, enquanto a lógica da função a ser realizada por tal *hardware* é definida em um *software*. O principal representante desse novo paradigma é o OpenFlow. O OpenFlow é uma interface de programação de aplicação (API) entre o plano de dados de um comutador e o plano de controle em nó controlador. A ideia inicial do OpenFlow era permitir o gerenciamento remoto e eficiente dos comutadores em uma rede, já que os demais protocolos eram muito complexos e complexos de serem usados [58]. Assim, basicamente, a compatibilidade de um comutador com o protocolo OpenFlow reside na implementação em software do protocolo, mapeando as primitivas do protocolo em recursos dos dispositivos, como tabelas de fluxos e filas, e permitindo que o dispositivo se comunique com o controlador, implantado em outro nó físico, como um computador pessoal.

Como resultado deste trabalho foram publicados dois artigos internacionais [14, 41], quatro artigos nacionais [18, 21, 46, 57] e foram aceitos para publicação uma revista internacional [15] e uma revista nacional [59].

Como trabalhos futuros, pretende-se desenvolver algoritmos de controle de admissão de novas redes virtuais em nós QFlow, assim como desenvolver algoritmos de otimização de alocação de redes virtuais sobre a infraestrutura da rede física. Outro trabalho futuro é explorar o controle de outras métricas como o atraso e a variação do atraso entre os roteadores em um determinado caminho, como o realizado por Kim *et al.* [23].

Referências Bibliográficas

[1] MOREIRA, M., FERNANDES, N., COSTA, L., et al. "Internet do futuro: Um novo horizonte", *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2009*, pp. 1–59, 2009.

[2] CAMPISTA, M. E. M., FERRAZ, L., MORAES, I., et al. "Interconexão de Redes na Internet do Futuro: Desafios e Soluções". In: *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC 2010 - Minicursos*, maio 2010.

[3] 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*, maio 2011.

[4] SZEGEDI, P., FIGUEROLA, S., CAMPANELLA, M., et al. "With Evolution for Revolution: Managing FEDERICA for Future Internet Research", *IEEE Communications Magazine*, v. 47, n. 7, pp. 34–39, jul. 2009.

[5] MODESTO, M., PEREIRA, A., ZIVIANI, N., et al. "Um novo retrato da web brasileira". In: *XXXII Seminário Integrado de Software e Hardware - SBC/SEMISH'2005*, jul. 2005.

[6] 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, 2007. ISSN: 0146-4833.

[7] "Horizon Project: A New Horizon to The Internet". , nov. 2012. Disponível em: <`http://www.gta.ufrj.br/horizon`>.

[8] "Projeto ReVir Redes Virtuais na Internet do Futuro". , nov. 2012. Disponível em: <`http://www.gta.ufrj.br/revir`>.

[9] KELLER, E., REXFORD, J. "The Platform as a service model for networking". In: *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pp. 4–9. USENIX Association, 2010.

[10] 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, 2008. ISSN: 0146-4833.

[11] CHUN, B., CULLER, D., ROSCOE, T., et al. "Planetlab: An Overlay Testbed for Broad-coverage Services", *ACM Computer Communication Review*, v. 33, n. 3, pp. 3–12, 2003.

[12] TURNER, J. "A proposed architecture for the GENI backbone platform". In: *Architecture for Networking and Communications systems, 2006. ANCS 2006. ACM/IEEE Symposium on*, pp. 1–10. IEEE, 2006.

[13] BARHAM, P., DRAGOVIC, B., FRASER, K., et al. "Xen and the art of virtualization". In: *ACM SIGOPS Operating Systems Review*, v. 37, pp. 164–177. ACM, 2003.

[14] MATTOS, D. M. F., FERRAZ, L. H. G., COSTA, L. H. M. K., et al. "Evaluating virtual router performance for a pluralist future internet". In: *Proceedings of the 3rd International Conference on Information and Communication Systems*, ICICS '12, pp. 4:1–4:7, Irbid, Jordan, 2012. ACM. ISBN: 978-1-4503-1327-8. doi: 10.1145/2222444.2222448. Disponível em: <http://doi.acm.org/10.1145/2222444.2222448>.

[15] MATTOS, D. M. F., FERRAZ, L. H. G., COSTA, L. H. M. K., et al. "Virtual Network Performance Evaluation for Future Internet Architectures", *Journal of Emerging Technologies in Web Intelligence - JETWI*, v. 4, n. 4, pp. 304–314, 2012.

[16] PFAFF, B., PETTIT, J., KOPONEN, T., et al. "Extending networking into the virtualization layer", *Proceedings of the 8th ACM Workshop on Hot Topics in Networks*, out. 2009.

[17] MATTOS, D. *XenFlow: Um Sistema de Processamento de Fluxos Robusto e Eficiente para Redes Virtuais*. Relatório técnico, Grupo de Teleinformática e Automação / Escola Politécnica / Universidade Federal do Rio de Janeiro - Projeto Final de Curso de Graduação em Engenharia da Computação e Informação - GTA-11-28, mar. 2011. Disponível em: <http://www.gta.ufrj.br/ftp/gta/TechReports/Diogo11/Diogo11.pdf>.

[18] 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*, maio 2011.

[19] PISA, P., FERNANDES, N., CARVALHO, H., et al. "OpenFlow and Xen-Based Virtual Network Migration". In: Pont, A., Pujolle, G., Raghavan, S. (Eds.), *Communications: Wireless in Developing Countries and Networks of the Future*, v. 327, *IFIP Advances in Information and Communication Technology*, Springer Boston, pp. 170–181, 2010.

[20] WANG, Y., KELLER, E., BISKEBORN, B., et al. "Virtual routers on the move: live router migration as a network-management primitive", *ACM SIGCOMM Computer Communication Review*, v. 38, n. 4, pp. 231–242, 2008. ISSN: 0146-4833.

[21] MATTOS, D. M. F., DUARTE, O. C. M. B. "QFlow: Um Sistema com Garantia de Isolamento e Oferta de Qualidade de Serviço para Redes Virtualizadas". In: *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2012*, abr. 2012.

[22] MCILORY, R., SVENTEK, J. "Resource virtualisation of network routers". In: *Workshop on High Performance Switching and Routing*, HPSR, pp. 15–20. IEEE, 2006.

[23] KIM, W., SHARMA, P., LEE, J., et al. "Automated and scalable QoS control for network convergence", *Proceedings of the Internet Network Management/Workshop on Research on Enterprise Networking*, abr. 2010.

[24] FERNANDES, N., DUARTE, O. "XNetMon: Uma arquitetura com segurança para redes virtuais", *Anais do X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pp. 339–352, 2010.

[25] BARABASH, K., COHEN, R., HADAS, D., et al. "A case for overlays in DCN virtualization". In: *Proceedings of the 3rd Workshop on Data Center-Converged and Virtual Ethernet Switching*, pp. 30–37. ITCP, 2011.

[26] PERLMAN, R., EASTLAKE 3RD, D., DUTT, D., et al. "Routing Bridges (RBridges): Base Protocol Specification". . RFC 6325 (Proposed Standard), jul. 2011. Disponível em: <`http://www.ietf.org/rfc/rfc6325.txt`>. Updated by RFCs 6327, 6439.

[27] COUTO, R. S., CAMPISTA, M. E. M., COSTA, L. H. M. K. "XTC: A Throughput Control Mechanism for Xen-based Virtualized Software Routers". In: *IEEE Global Communications Conference (GLOBECOM'2011)*, Houston, Texas, USA, dez. 2011.

[28] ONGARO, D., COX, A. L., RIXNER, S. "Scheduling I/O in virtual machine monitors". In: *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE'08, pp. 1–10, Seattle, WA, USA, 2008. ACM. ISBN: 978-1-59593-796-4.

[29] FERNANDES, N., MOREIRA, M., MORAES, I., et al. "Virtual networks: Isolation, performance, and trends", *Annals of Telecommunications*, pp. 1–17, 2010. ISSN: 0003-4347.

[30] CLARK, C., FRASER, K., HAND, S., et al. "Live migration of virtual machines". In: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 273–286. USENIX Association, 2005.

[31] CASADO, M., KOPONEN, T., RAMANATHAN, R., et al. "Virtualizing the network forwarding plane". In: *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, PRESTO'10, pp. 8:1–8:6. ACM, 2010. ISBN: 978-1-4503-0467-2. doi: 10.1145/1921151.1921162.

[32] HAO, F., LAKSHMAN, T., MUKHERJEE, S., et al. "Enhancing dynamic cloud-based services using network virtualization". In: *Proceedings of the 1st ACM workshop on Virtualized Infrastructure Systems and Architectures*, pp. 37–44. ACM, 2009.

[33] "The SNAC OpenFlow controller." , nov. 2012. Disponível em: <`http://www.openflow.org/wp/snac/`>.

[34] YU, M., REXFORD, J., FREEDMAN, M., et al. "Scalable flow-based networking with DIFANE". In: *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, pp. 351–362. ACM, 2010.

[35] 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, 2008. ISSN: 0146-4833.

[36] GUEDES, D., VIEIRA, L., VIEIRA, M., et al. "Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores", *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, pp. 160–210, maio 2012.

[37] TOOTOONCHIAN, A., GANJALI, Y. "HyperFlow: A distributed control plane for OpenFlow". In: *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*. USENIX Association, 2010.

[38] STRIBLING, J., SOVRAN, Y., ZHANG, I., et al. "Flexible, wide-area storage for distributed systems with WheelFS". In: *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pp. 43–58. USENIX Association, 2009.

[39] SHERWOOD, R., GIBB, G., YAP, K., et al. *FlowVisor: A network virtualization layer*. Relatório técnico, Tech. Rep. OPENFLOW-TR-2009-01, OpenFlow Consortium, 2009.

[40] HOUIDI, I., LOUATI, W., ZEGHLACHE, D., et al. "Adaptive virtual network provisioning". In: *Proceedings of the second ACM SIGCOMM workshop on Virtualized Infrastructure Systems and Architectures*, pp. 41–48. ACM, 2010.

[41] MATTOS, D., FERNANDES, N. C., COSTA, V., et al. "OMNI: OpenFlow MaNagement Infrastructure". In: *2011 International Conference on the Network of the Future (NoF'11)*, pp. 52–56, Paris, França, nov. 2011.

[42] EGI, N., GREENHALGH, A., HANDLEY, M., et al. "Towards high performance virtual routers on commodity hardware". In: *Proceedings of the 2008 ACM CoNEXT Conference*, pp. 1–12. ACM, 2008.

[43] WANG, R., BUTNARIU, D., REXFORD, J. "OpenFlow-based server load balancing gone wild". In: *Proceedings of the 11th USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Hot-ICE. USENIX Association, mar. 2011.

[44] AL-FARES, M., RADHAKRISHNAN, S., RAGHAVAN, B., et al. "Hedera: Dynamic flow scheduling for data center networks". In: *Proceedings of the 7th USENIX conference on Networked systems design and implementation*. USENIX Association, 2010.

[45] NASCIMENTO, M. R., ROTHENBERG, C. E., DENICOL, R. R., et al. "RouteFlow: Roteamento Commodity Sobre Redes Programáveis", *Revista Brasileira de Redes de Computadores e Sistemas Distribuídos*, v. 4, n. 2, pp. 21–31, 2011.

[46] 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'12*, maio 2011.

[47] PISA, P. S., COUTO, R. S., CARVALHO, H. E. T., et al. "VNEXT: Virtual NEtwork management for Xen-based Testbeds". In: *2011 International Conference on the Network of the Future (NoF'11)*, pp. 41–45, Paris, França, nov. 2011.

[48] "VNEXT: Virtual NEtwork management for Xen-based Testbeds". , nov. 2012. Disponível em: <http://www.gta.ufrj.br/vnext>.

[49] "OMNI: OpenFlow MaNagement Infrastructure". , nov. 2012. Disponível em: <http://www.gta.ufrj.br/omni>.

[50] GINKGO NETWORKS. *Ginkgo Distributed Network Piloting System*. Relatório técnico, Ginkgo Networks, set. 2008.

[51] EGI, N., GREENHALGH, A., HANDLEY, M., et al. "Evaluating Xen for router virtualization". In: *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pp. 1256–1261. IEEE, 2007.

[52] WANG, Y., VAN DER MERWE, J., REXFORD, J. "VROOM: Virtual routers on the move". In: *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking*. ACM, 2007.

[53] CARAPINHA, J., JIMÉNEZ, 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.

[54] HANDLEY, M., HODSON, O., KOHLER, E. "XORP: An open platform for network research", *ACM SIGCOMM Computer Communication Review*, v. 33, n. 1, pp. 53–57, 2003.

[55] TOUCH, J., PERLMAN, R. "Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement". . RFC 5556 (Informational), maio 2009. Disponível em: <http://www.ietf.org/rfc/rfc5556.txt>.

[56] GIBBS, G. *Basic spanning tree*. http://www.openflow.org/wk/index.php/Basic_-Spanning_Tree, Acessado em outubro de 2012.

[57] MATTOS, D. M. F., MAURICIO, L. H., CARDOSO, L. P., et al. "Uma Rede de Testes Interuniversitária a com Técnicas de Virtualizacao Híbridas". In: *Salão de Ferramentas do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'12*, abr. 2012.

[58] CASADO, M., FREEDMAN, M., PETTIT, J., et al. “Ethane: Taking control of the enterprise”, *ACM SIGCOMM Computer Communication Review*, v. 37, n. 4, pp. 1–12, 2007.

[59] MATTOS, D. M. F., DUARTE, O. C. M. B. “Isolamento e Oferta de Qualidade de Serviço em Redes Virtualizadas”, *Revista Brasileira de Redes de Computadores e Sistemas Distribuídos - RB-RESD*. A ser publicado.