UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM CONFIGURAÇÃO E GERENCIAMENTO DE SERVIDORES E EQUIPAMENTOS DE REDES

ROBERTO JOSÉ DE AMORIM

O USO DO PROTOCOLO OPENFLOW EM REDES DEFINIDAS POR SOFTWARE

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA – PR
2012

ROBERTO JOSÉ DE AMORIM

O USO DO PROTOCOLO OPENFLOW EM REDES DEFINIDAS POR SOFTWARE

Monografia de Especialização apresentada ao Departamento Acadêmico de Eletrônica da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Especialista em Configuração e Gerenciamento de Servidores e Equipamentos de Redes"
Orientador: Prof. Fabiano Scriptore de Carvalho.

CURITIBA – PR
2012

RESUMO

AMORIM, Roberto J. **O uso do protocolo OpenFlow em redes definidas por software** - Especialização em Configuração e Gerenciamento de Servidores e Equipamentos de Redes, Universidade Tecnológica Federal do Paraná. Curitiba 2012.

Este projeto apresenta um estudo detalhando a tecnologia emergente OpenFlow como ferramenta para gerenciamento sofisticado do fluxo de dados em uma rede de computadores, de modo a otimizar o uso dos recursos oferecidos por essa rede. O trabalho inclui estudos, simulações, análises de desempenho e descrições dos protocolos e equipamentos de rede, bem como análises da situação atual da tecnologia e as perspectivas para o futuro.

Palavras chave: OpenFlow, redes de computadores, optimização, fluxo de dados, Mininet, desenvolvimento de redes experimentais, análises de desempenho.

ABSTRACT

AMORIM, Roberto J. **O uso do protocolo OpenFlow em redes definidas por software** - Especialização em Configuração e Gerenciamento de Servidores e Equipamentos de Redes, Universidade Tecnológica Federal do Paraná. Curitiba 2012.

This project presents a study detailing the OpenFlow emerging technology as a tool for sophisticated data flow management in a computer network, as a way to optimize the usage of resources offered by this network. The paper includes studies, simulations, benchmarks and descriptions of the protocols and networking equipment, as well as analyses of the technology's current situation and future perspectives.

Key Words: OpenFlow, computer networks, optimization, data flow, Mininet, experimental network development, benchmarks.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ARP	Address Resolution Protocol
DoS	Denial of Service
GB	Gigabyte
HP	Hewlett-Packard
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISO	International Standards Organization
KDE	K Desktop Environment
LAN	Local Area Network
MAC	Machine Address Code
OSI	Open Systems Interconnection
RAM	Random Access Memory
SDN	Software Defined Networks
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UDP	Unified Datagram Protocol
VLAN	Virtual LAN
WWW	Worls Wide Web

1 INTRODUÇÃO

Neste capítulo será tratado o Tema, Delimitação da Pesquisa, Problemas e Premissas, o Objetivo Geral, os Objetivos Específicos, Justificativa, Procedimentos Metodológicos, Embasamento Teórico e a Estrutura deste trabalho.

1.1 TEMA

As redes de computadores são um sucesso, sendo parte do cotidiano de bilhões de pessoas. Assim, as redes são compostas por milhares de equipamentos, podendo chegar a milhões no caso da Internet. Devido a esse grande número, propostas de novos protocolos para essas redes podem se tornar impossíveis de serem testadas em ambientes reais, pois muitas necessitariam de mudanças em todos os equipamentos da rede. Além disso, como as redes de computadores são parte da infraestrutura crítica de residências, escritórios e instituições em geral, essas mudanças poderiam ser descartadas pelos seus administradores. Esses fatores criam barreiras para novas ideias na área de redes de computadores, pois muitas propostas de pesquisadores dessa área acabam não sendo testadas em ambientes reais. Muitos desses pesquisadores consideram que a infraestrutura de rede está “ossificada”, não podendo ser modificada. (McKeon et al., 2008)

Além disso, novas tecnologias têm alterado e complicado a natureza das redes de computador. Por exemplo, as exigências da computação em nuvem e de datacenters de grande porte tornaram muito mais complexa a construção de redes eficientes. (Vaughan-Nichols, 2011)

Para lidar com essas demandas, operadores querem que suas redes sejam mais inteligentes e querem ser capazes de controla-las e gerenciá-las. Em resposta, tem aumentado o interesse por redes definidas por software (SDNs).

SDNs executam o plano de controle de rede em software, em geral em servidores padrão operados separadamente dos dispositivos de rede, tais como switches. Isso dá aos administradores de rede controle mais refinado sobre o tráfego de dados. (OpenFlow Switch Specification, 2011)

Em essência, OpenFlow transfere controle de tráfego da rede da infraestrutura - os switches e roteadores - para os administradores da rede.

Este trabalho aborda as especificações do protocolo OpenFlow, casos de uso, estudos de caso de instituições que adotaram OpenFlow, vantagens e desvantagens do protocolo, perspectivas, simulações de uso e conclusões sobre seus prós e contras.

1.1.1 Delimitação de Pesquisa

Para bom entendimento desta pesquisa será mostrado primeiramente o que é uma Rede de Computadores, como ela funciona e suas principais características. Em seguida serão abordados os conceitos de roteamento e switching, suas aplicações e os seus padrões de funcionamento.

Referente às tecnologias de SDNs, o foco do estudo será o protocolo OpenFlow, que será simulado em redes virtuais de médio e grande porte através do aplicativo MiniNet.

1.2 PROBLEMA E PREMISSAS

Desenvolvimentos recentes têm posto cada vez mais pressão sobre as conexões de dados globais. Vídeo sob demanda, redes sociais, virtualização, computação na nuvem, *offshoring* e outras inovações têm feito com que o tráfego global de dados cresça em média 50% ao ano, com uma expectativa de chegar a 60 exabytes ($60 * 10^{18}$ bytes) em 2016 (ABIresearch, 2011)

Tal pressão exige dos operadores de redes de grande porte controle fino de suas conexões, de modo a garantir que sua capacidade seja aproveitada da maneira mais eficiente e produtiva.

Apoiado nessa preocupação, o foco principal dessa pesquisa visa ajudar a solucionar o seguinte problema:

O protocolo OpenFlow fornece recursos de controle fino de conexões que possam ser usados por empresas e meios acadêmicos? Quais são as vantagens e desvantagens apresentadas pela tecnologia?

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Analisar o protocolo de SDN OpenFlow, suas características, vantagens e desvantagens, além das perspectivas para o seu futuro.

1.3.2 Objetivos Específicos

- Definir o padrão OpenFlow e analisar os protocolos e formato de quadro
- Usar e descrever o software MiniNet e o switch virtual openvSwitch para simular uma rede virtual utilizando OpenFlow.
- Usar o analisador de pacotes Wireshark para dissecar e entender um pacote OpenFlow
- Executar testes de desempenho entre switches virtuais e entre diferentes controladores OpenFlow

1.4 JUSTIFICATIVA

O aumento global no consumo de largura de banda, aliado à necessidade de redes cada vez mais confiáveis e os riscos apresentados por cyber-terrorismo e espionagem industrial exige que equipamentos de rede sejam cada vez mais inteligentes, de modo que possam oferecer controle das conexões, confiabilidade e segurança.

O protocolo OpenFlow tem por objetivo atender a todos esses requerimentos ao permitir que o controle fino de equipamentos de rede passe de rotinas fechadas embutidas pelo fabricante a servidores centralizados sob o controle do administrador de rede. Dessa forma o administrador tem uma visão global da rede e pode agir com rapidez em casos como congestão das conexões, falha de equipamento ou invasão da rede, entre outros problemas.

1.5 PROCEDIMENTOS METODOLÓGICOS

Verificando os critérios de pesquisa proposto por Gil (2010), a pesquisa é de natureza aplicada. Já quanto aos seus objetivos gerais e propostos é uma pesquisa explicativa. E por fim será feita uma pesquisa prática usando simulação e virtualização de switches OpenFlow a fim de executar análises de desempenho.

1.6 EMBASAMENTO TEÓRICO

Com a intenção de mostrar os conceitos sobre Redes de Computadores destacam-se os trabalho bibliográficos de Santos (2008), Peterson e Davie (2011) e Tanembaum (2003). Para a explicação teórica sobre o protocolo OpenFlow será utilizado basicamente o seu documento de especificação (OpenFlow Switch Specification 1.1.0, 2011). Já a descrição de como montar e executar o switch OpenFlow simulado com Mininet será baseada no excelente tutorial em inglês do próprio grupo responsável por divulgação do protocolo OpenFlow (http://www.openflow.org/wk/index.php/OpenFlow_Tutorial)

1.7 ESTRUTURA

O trabalho esta organizado em sete capítulos.

O capítulo 1 deste trabalho apresenta a Introdução, falando do tema, delimitação da pesquisa, problemas e premissas, objetivos, justificativa, procedimentos metodológicos, embasamento teórico e a estrutura descrita aqui.

O capítulo 2 concentra na fundamentação teórica da pesquisa.

O capítulo 2.1 apresenta uma breve introdução às redes de computadores, modelo OSI, TCP/IP e descrição do funcionamento de switches e roteadores.

O capítulo 2.2 descreve o protocolo OpenFlow, o seu funcionamento e as partes que o compõem (switch OpenFlow, tabela de fluxo, controlador).

O capítulo 3 descreve a pesquisa feita sobre OpenFlow. A configuração e uso de uma máquina virtual, a simulação de uma rede OpenFlow usando o software Mininet, descrição das ferramentas utilizadas, comandos básicos em Mininet, população manual da tabela de fluxo e análise dos pacotes do protocolo.

O capítulo 4 apresenta os testes de desempenho efetuados sobre OpenFlow: teste de desempenho de switches e de vários controladores OpenFlow.

O capitulo 5 contém a conclusão do trabalho, prós, contras e perspectivas acerca do protocolo OpenFlow.

2 REFERENCIAIS TEÓRICOS

Neste capítulo será descrito o Referencial Teórico do trabalho, que contém os seguintes assuntos: Redes de Computadores, o Modelo de Referência OSI, o modelo de Referência TCP/IP, Switching, Roteamento e uma descrição de como funciona o OpenFlow.

2.1 REDES DE COMPUTADORES

Segundo Tanembaum (2003), pode-se conceituar o termo rede de computadores como um conjunto de computadores e outros dispositivos utilizando uma tecnologia que permite a troca de informações compartilhando o mesmo meio físico e lógico. Redes de Computadores podem ser utilizadas para diversos serviços, tanto para empresas quanto para indivíduos. Nas empresas as redes são utilizadas para compartilhar arquivos, impressoras e informações coorporativas, e para as pessoas servem como fonte de informação, pesquisa e diversão (Tanembaum, 2003).

No inicio das redes de computadores cada fabricante possuía sua própria forma de trabalho e sua linha de desenvolvimento tecnológico. Desse modo um hardware do fabricante X só pode ser conectado por meio físico (fio) a outro hardware do mesmo fabricante. Se um dos dispositivos apresentasse problemas e não sendo possível a substituição por outro dispositivo do mesmo fabricante, seria necessário substituir todos os dispositivos daquela rede, o que causava transtornos e gastos elevados (Tanembaum, 2003).

2.1.1 O Modelo de Referência OSI

Visando resolver este problema de interoperabilidade, a interconectividade, a portabilidade e a escalabilidade entre tecnologias e produtos de diferentes fabricantes, foi criado pela ISO (*International Standards Organization*) no ano de 1970 o modelo de referência OSI (*Open Systems Interconnection*), que seria utilizado como padrão para troca de informações entre e dentro das redes de computadores (Tanembaum, 2003). Esse modelo possui sete camadas (figura 1), as

camadas em ordem crescente são: física, enlace de dados, transporte, rede, sessão, apresentação e aplicação. Segue uma breve descrição das sete camadas.

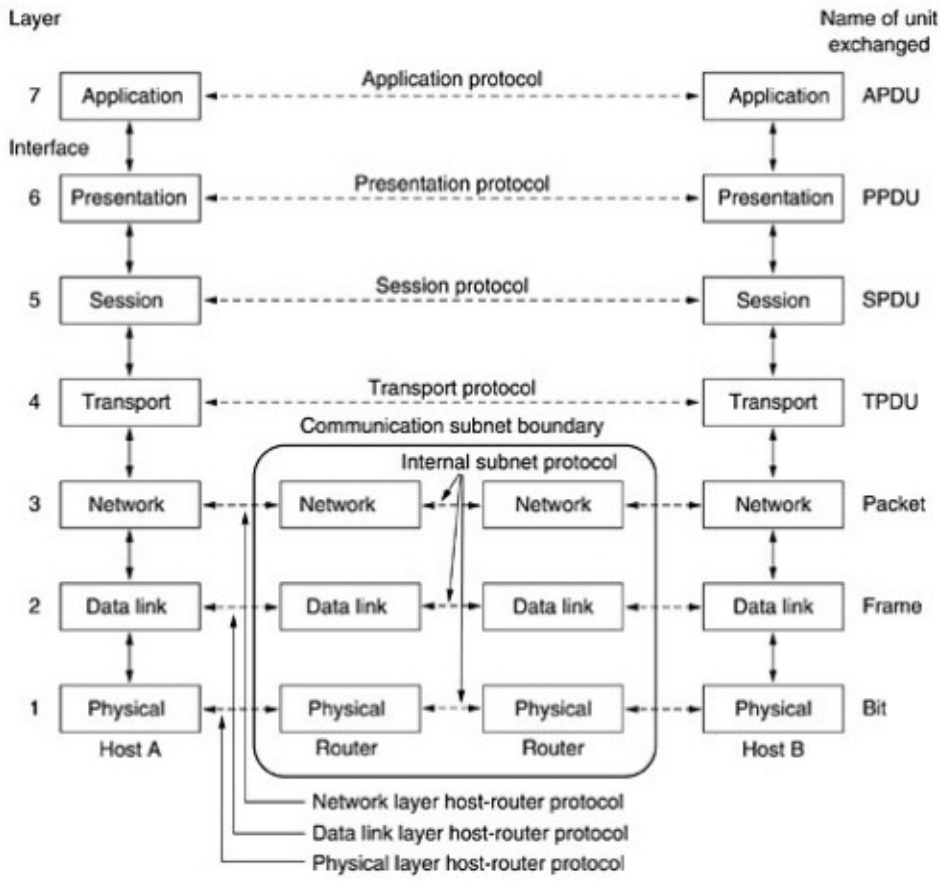

Figura 1 - Modelo de Referência OSI

Fonte: Tanenbaum, 2003

Camada Física: A camada física cuida das características físicas, elétricas, funcionais e procedimentos para ativar, manter e desativar conexões entre duas partes. Ela está ligada diretamente à transmissão de bits primários (bit 0 e bit 1) por um canal de comunicação (Santos, 2008).

Camada de Enlace: Providencia maneiras funcionais e procedimentos para estabelecimento, manutenção e liberação de enlace de dados entre as entidades da rede. Os objetivos são providenciar a transmissão de dados para a camada de rede e detectar, e possivelmente corrigir, erros que possam ocorrer no meio físico (Santos, 2008).

Camada de Rede: A principal função da camada de rede é controlar a operação de rede. Ela estabelece uma conexão lógica entre dois pontos, cuidando do tráfego e roteamentos dos dados da rede (Tanembaum, 2003).

Camada de Transporte: A principal função da camada de transporte é receber dados da camada de sessão, dividi-los em pacotes menores caso haja necessidade, transmitir os mesmos para a camada de rede e garantir que todos os pacotes sejam entregues (Santos, 2008).

Camada de Sessão: A camada de sessão gerencia as atividades das camadas inferiores. Permite que usuários de diferentes máquinas estabeleçam comunicação entre si. A camada sessão cuida de vários serviços, como por exemplo, qual maquina deve transmitir em cada momento, controle de troca de dados e sincronização entre as duas máquinas (Santos, 2008).

Camada de Apresentação: A camada de apresentação é responsável pela conversão dos dados para uma forma que eles sejam entendidos por todos os sistemas envolvidos na comunicação, resolvendo assim problemas de sintaxe entre os sistemas. Também realiza compressão, descompressão, criptografia e descriptografia (Tanembaum, 2003).

Camada de Aplicação: Na camada de aplicação se encontrada os serviços utilizados pelos usuários, como transferência de arquivos, e-mail, gerenciamento de redes e outras facilidades. Um protocolo amplamente utilizado na camada de aplicação é o HTTP (*HyperText Transfer Protocol*) que constitui a base para o WWW (*Word Wide Web)*. Quando acessamos uma pagina na Web, o nome desta pagina é enviada ao servidor utilizando o protocolo HTTP. Depois o servidor transmite a página novamente (Santos, 2008).

2.1.2 O Modelo de Referência TCP/IP

O modelo TCP/IP surgiu para atender às necessidades de conexão da ARPANET, uma rede de pesquisa patrocinada pelo Departamento de Defesa dos Estados Unidos, e aos poucos universidades e repartições publicas foram sendo conectadas a esta rede através de linha telefônica dedicada. Após a criação das redes de rádio e satélite começaram a surgir problemas com a arquitetura existente, por isso foi necessário a criação de um novo modelo de referência, esse modelo

tinha como principal objetivo conectar várias redes de maneira uniforme (Tanembaum, 2003).

O Modelo TCP/IP foi projetado sem relação às camadas do modelo OSI, e não foi criado para se tornar um modelo de referência padrão, mas sim para atender às necessidades do Departamento de Defesa dos Estados Unidos, cujo requerimento era a manutenção da conexão entre máquinas de origem e destino mesmo que equipamentos e links intermediários ficassem inoperantes (Tanembaum, 2003)

O TCP/IP é composto por quatro camadas (figura 2), a camada de Host, a camada de Inter-Rede, a camada de Transporte e a camada de Aplicação. Segue abaixo a descrição das funcionalidades principais das quatro camadas.

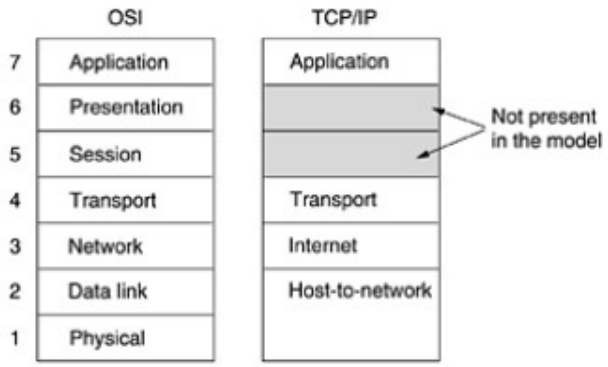

Figura 2 - Modelo de Referência TCP/IP

Fonte: Tanenbaum, 2003

Camada de Acesso à Rede: O Modelo de Referência TCP/IP não especifica detalhadamente o que ocorre nesta camada, apenas determina que o host deve se conectar a rede utilizando algum protocolo que seja possível enviar pacotes IP (*Internet Protocol*), este protocolo não é definido e varia de *host* para *host* e de rede para rede (Tanembaum, 2003)

Camada de *Internet* (Inter-Rede): A função da camada de *Internet* ou inter-rede é garantir que um *host* consiga enviar pacotes em qualquer rede e garantir que esses pacotes trafegarão independentemente até o destino, podendo esse destino ser uma rede diferente. Estes pacotes podem chegar fora de ordem, cabendo às camadas superiores reorganizá-los caso a entrega necessite ser em ordem. A camada de inter-redes define um formato de pacote padrão e um protocolo

denominado IP, entregando esses pacotes aonde forem necessários, atentando para a parte de roteamento, visando evitar congestionamento de dados. A camada inter-redes do TCP/IP é análoga à camada de rede do modelo OSI (Tanembaum, 2003).

Camada de Transporte: A camada de transporte do modelo TCP/IP tem a mesma função que a camada de transporte do modelo OSI, garantir a entrega dos pacotes enviados, garantindo que os *hosts* de origem e destino mantenham uma conversação. Dois protocolos fim a fim foram definidos nesta camada. O primeiro deles é o TCP (*Transmission Control Protocol* – Protocolo de Controle de Transmissão), é um protocolo orientado a conexão que permite a entrega sem erros de um pacote enviado pela origem até o destino. O TCP fragmenta os pacotes em mensagens separadas e encaminha cada mensagem para a camada de inter-redes. Chegando ao destino o TCP monta essas mensagens e envia uma confirmação de entrega para a origem (Peterson and Davie, 2011).

O Segundo protocolo se chama UDP (*User Datagram Protocol*). Ele é utilizado para transporte rápido e de baixa complexidade entre hosts IP. Porém o UDP não garante entrega e nem verificação de dados, ele apenas encaminha o pacote para o destino e o host remetente não recebe confirmação de entrega. Esse serviço do UDP é chamado de sem conexão. (Peterson and Davie, 2011)

Camada de Aplicação: O modelo TCP/IP não possui as camadas de apresentação e de sessão, condensando-as na camada de aplicação encontrada logo acima da camada de transporte. Ela é responsável pela execução dos serviços utilizados pelos usuários como navegação web, transferência de arquivos, e-mail e terminal virtual (telnet) (Tanenbaum, 2003)

2.1.3 Switching

Na definição mais simples, um switch é um mecanismo que permite interconectar links que formam uma rede maior. O switch é um dispositivo com múltiplas entradas e múltiplas saídas que transfere pacotes de uma entrada para uma ou mais saídas. Essas entradas e saídas podem estar conectadas a outros switches de modo a multiplicar a quantidade de conexões disponíveis naquela rede.

Esses equipamentos geralmente agem na camada de enlace de dados do modelo OSI, embora modelos mais sofisticados atuam também na camada de rede.

2.1.4 Roteamento

Algoritmo de roteamento é “a parte do software da camada de rede responsável pela decisão sobre a linha de saída a ser usada na transmissão do pacote de entrada” (Tanenbaum, 2003), ou seja, determina o caminho que um pacote de dados irá seguir da fonte até o destino dentro da rede. Para esta tomada de decisão são levados em consideração alguns fatores como, por exemplo, o número de saltos e o estado do link. Age na camada de rede do modelo OSI.

2.2 OpenFlow

OpenFlow é uma interface aberta para controlar remotamente tabelas de encaminhamento de pacotes em switches, roteadores e pontos de acesso. Atuando sobre primitivas de baixo nível, pesquisadores e administradores de rede podem construir novas redes com propriedades de alto nível. Por exemplo, OpenFlow permite implantar segurança em redes que por padrão não têm segurança, realizar trocas de ponto de acesso sem fio de maneira imperceptível ao usuário (seamless handoff), redes escaláveis em data centers, mobilidade de hosts e redes mais eficientes em termos energéticos e em enlaces de longa distância, para citar alguns exemplos. Um objetivo do OpenFlow é proporcionar redes flexíveis e seguras que apresentam menos problemas de tráfego de dados que as redes atuais e custam menos para implementar e administrar.

2.2.1 Funcionamento do protocolo OpenFlow

Em um roteador ou switch tradicional, o encaminhamento de pacotes (data path – caminho de dados) e as decisões de roteamento (control path – caminho de controle) ocorrem no mesmo dispositivo. Um Switch OpenFlow separa estas duas funcionalidades. A função de data path permanece no switch, enquanto que as

decisões de roteamento são movidas para um Controlador, geralmente localizado em um servidor (a parte definida por software das SDNs). Os Switches OpenFlow e o Controlador se comunicam através do protocolo OpenFlow, geralmente transmitido por um canal criptografado seguro (SSL), que define mensagens como: handshake, definição de VLANs, criptografia, obter status das portas do switch, definir parâmetros etc. (OpenFlow Switch Specification, 2011)

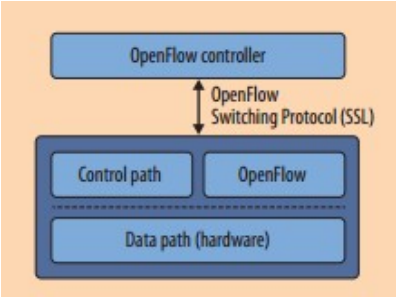

Figura 3 - Esquema do funcionamento de um dispositivo OpenFlow

Fonte: Vaughan-Nichols, 2011

O OpenFlow nos permite controlar o fluxo de dados escolhendo as rotas que os pacotes vão seguir e o processamento a que eles devem ser submetidos. Desta forma, o OpenFlow torna possível experimentar com novos protocolos, novos modelos de segurança, novos esquemas de endereçamento etc. O data path de um Switch OpenFlow consiste no que é chamado Flow Table (Tabela de Fluxo), e as ações associadas com cada entrada de fluxo.

2.2.2 Entradas da Tabela de Fluxo

Cada entrada de fluxo (flow-entry) na tabela de fluxo do switch ou roteador tem uma ação simples relacionada. Podemos citar três ações principais: (OpenFlow Switch Specification, 2011)

- Encaminhar os pacotes deste fluxo para uma determinada porta ou várias portas. Isso permite que os dados sejam direcionados dentro da rede. Esse tipo de ação pode ser executado sem nenhuma penalidade de performance imposta pelo protocolo OpenFlow (isto é, line speed – velocidade do hardware)

- Encapsula e transmite pacotes desse fluxo para um controlador OpenFlow. Os pacotes são enviados ao controlador usando um canal seguro (Secure Channel). Geralmente isso é utilizado para o primeiro pacote de um novo fluxo para permitir ao controlador avaliar se esse fluxo deverá ser adicionado à tabela de fluxo. Ou, dependendo da circunstância, todos os pacotes do fluxo podem ser redirecionados ao controlador para a execução de processamento, monitoramento, etc.
- Descartar pacotes deste fluxo. Isso pode ser usado para mitigar ataques como negação de serviço (DoS) ou bloquear a comunicação entre hosts.

Cada entrada da tabela de fluxo possui três campos: (OpenFlow Switch Specification, 2011)

1. Uma regra que define o fluxo, isto é, como determinar quais pacotes são parte daquele fluxo. A regra consiste basicamente em fazer a associação a partir dos cabeçalhos do pacote;
2. a ação que define o que deve ser feito com os pacotes e
3. estatísticas que, entre outros dados, mantêm registrado o número de bytes e pacotes para cada fluxo e o tempo decorrido desde o último pacote que foi associado ao fluxo (de modo a suprimir fluxos inativos da tabela, melhorando o processamento e o consumo de memória)

A regra de fluxo é composta de 10 campos conforme mostrado abaixo. Cada campo pode ser definido como um curinga para agregar fluxos.

Switch Port	MAC source	MAC dest	Eth type	VLAN ID	IP source	IP dest	IP prot	TCP srcPort	TCP destPort

Tabela 1 - Campos da regra de fluxo

Alguns exemplos de entradas da tabela de fluxo e respectivas ações a serem tomadas são mostradas abaixo:

Switch Port	MAC source	MAC dest	Eth type	VLAN ID	IP source	IP dest	IP prot	TCP srcPort	TCP destPort	Ação
*	*	00:1d:...	*	*	*	*	*	*	*	porta2

Tabela 2 - Exemplo da tabela de fluxo: Switching

Switch Port	MAC source	MAC dest	Eth type	VLAN ID	IP source	IP dest	IP prot	TCP srcPort	TCP destPort	Ação
porta4	00:3b:...	00:1d:...	800	vlan1	1.2.3.4	5.6.7.8	4	4465	80	porta1

Tabela 3 - Exemplo da tabela de fluxo: Flow Switching

Switch Port	MAC source	MAC dest	Eth type	VLAN ID	IP source	IP dest	IP prot	TCP srcPort	TCP destPort	Ação
*	*	*	*	*	*	*	*	*	22	descartar

Tabela 4 - Exemplo da tabela de fluxo: Firewall

Na Tabela 2 está representado o funcionamento clássico de um switch da camada 2: se um quadro tem um determinado MAC de destino, ele deve sair por uma determinada porta. Nenhuma outra característica do pacote é relevante.

A Tabela 3 demonstra um controle mais sofisticado de fluxo: apenas pacotes que atendam vários critérios sairão pela porta 1.

A Tabela 4 representa um caso de uso típico de Firewall. Todos os pacotes com porta de destino 22 (SSH) serão descartados.

2.2.3 O processamento dos pacotes pelo OpenFlow

A maneira mais simples de processar os pacotes de um fluxo numa rede OpenFlow é forçar todos esses pacotes a passar pelo controlador. Para isso o controlador não precisa manipular a tabela de fluxo no switch – basta determinar que o dispositivo redirecione todos os pacotes para o controlador por padrão. Tal cenário poderia ser útil para testar funcionalidades de um novo protocolo em desenvolvimento mas não seria interessante na implantação de uma grande rede para produção dada a sobrecarga que causaria no controlador. A alternativa é redirecionar apenas o primeiro pacote do fluxo ao controlador e deixar que este decida sobre a ação que deve ser tomada. Após definir a regra do fluxo e a ação na tabela de fluxo do dispositivo os pacotes subsequentes daquele fluxo serão rapidamente processados já que a tabela fica no próprio dispositivo e não no controlador. (OpenFlow Switch Specification, 2011)

2.2.4 O Controlador OpenFlow

Um controlador OpenFlow adiciona e/ou remove entradas da tabela de fluxo de um ou mais switches sob seu controle. Por exemplo, um controlador estático seria uma aplicação simples rodando em um PC que estaticamente determina fluxos que interligam vários hosts durante um experimento. Mas podemos imaginar também um cenário com controladores mais sofisticados rodando em servidores dinamicamente adicionando e removendo fluxos baseados em análises de pacotes que trafegam pela rede. (OpenFlow Switch Specification, 2011)

Existe uma gama de controladores OpenFlow sendo desenvolvidos para atender todo tipo de caso de uso, desde aplicativos simples usados em redes pequenas de cunho acadêmico, passando por aplicativos complexos usados em grandes projetos de pesquisa e desenvolvimento até aplicativos com funcionalidades corporativas que implementam controladores OpenFlow para ambiente de produção em grandes empresas. Alguns controladores para fins de pesquisa que podem ser citados são: POX (programado em Python), Beacon (Java), Floodlight (Java), Maestro (Java), Trema (Ruby) e NOX (C++/Python). Já a NEC Corporation fornece o software ProgrammableFlow Management Console para redes corporativas.

3 OPENFLOW NA PRÁTICA

Neste capítulo será descrita uma experiência simples para emular dispositivos OpenFlow em um computador rodando o sistema operacional Linux além de testes usando essa rede emulada.

3.1 PRÉ-REQUISITOS

É necessário um computador com pelo menos 1GB (recomendados 2GB ou mais) de memória RAM e pelo menos 5GB de espaço livre em disco.

A fim de garantir um ambiente de testes homogêneo ele será montado dentro de uma máquina virtual rodando no aplicativo gratuito e livre VirtualBox, fornecido pela Oracle. Dessa forma o sistema operacional host da máquina pode tanto ser Windows como Linux e Mac OS.

As instruções a seguir pressupõem o uso de um sistema operacional host rodando Windows.

A imagem da máquina virtual é fornecida pelo próprio grupo de desenvolvimento da especificação OpenFlow e pode ser obtida no seguinte link: http://www.openflow.org/downloads/cs244-VM-0107.zip

Essa máquina virtual é baseada na distribuição Ubuntu versão 10.10 e já contém todas as ferramentas necessárias para emular e controlar um switch OpenFlow.

Além da máquina virtual é necessário obter e instalar o servidor X Xming para visualização da interface do Wireshark e do Floodlight e o cliente SSH putty.

3.2 CONFIGURANDO A MÁQUINA VIRTUAL

Comece descompactando o arquivo zip que contém a imagem da máquina virtual (cs244-VM-0107.zip) podendo para isso usar o Windows Explorer. A imagem é um arquivo .vdi de cerca de 3.5GB

Uma vez descompactada a imagem precisamos configurar máquina virtual no VirtualBox:

- Inicie o VirtualBox.

- Selecione o botão New.
- Digite um nome para sua máquina virtual, escolha o sistema operacional Linux e versão Ubuntu. Clique em Próximo.
- Selecione 512MB de memória e clique em Próximo
- Selecione “Utilizar um disco rígido virtual existente” e localize o arquivo cs244.vdi que foi extraído do arquivo zip. Depois clique em criar.
- Selecione a máquina virtual recém criada no VirtualBox e clique no botão Configurações. Vá até Rede -> Adaptador 2. Marque “Habilitar Placa de Rede” e selecione a opção “Placa de rede exclusiva do hospedeiro (host-only)”, conforme a imagem:

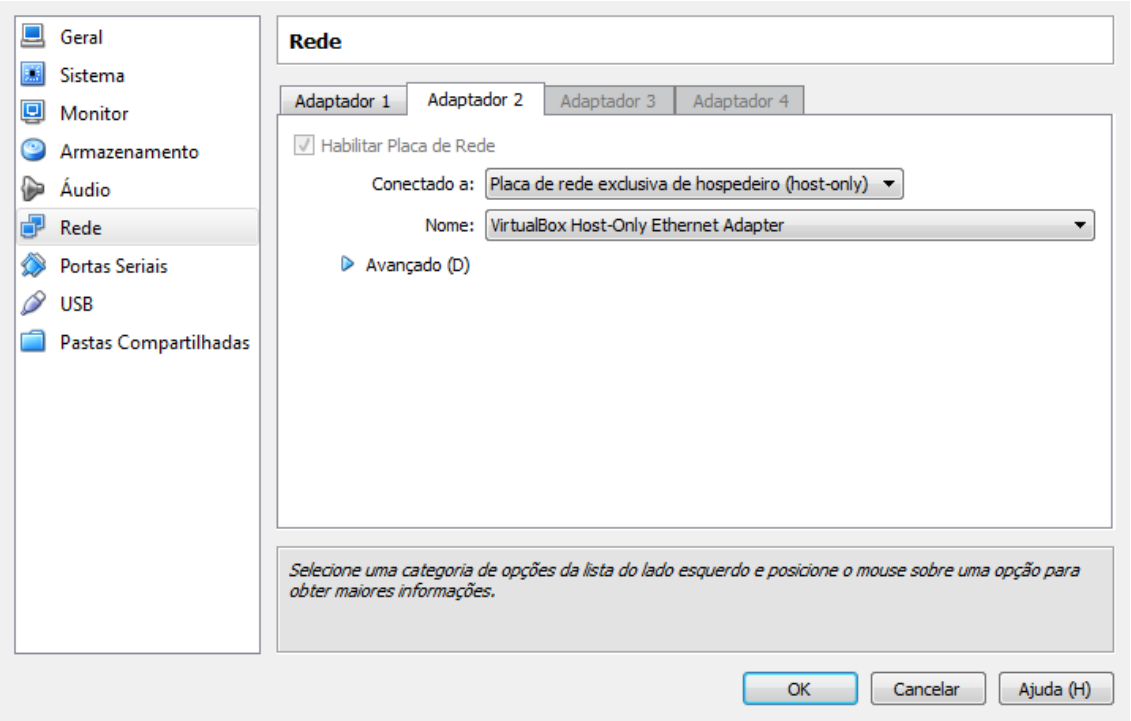

Figura 4- Configuração de rede no VirtualBox

Fonte: Autoria própria

Depois de clicar OK você poderá executar sua máquina virtual. Pressione “Iniciar” ou dê um duplo clique na máquina criada. Feito o boot você poderá fazer login com o usuário e senha openflow.

3.2 ACESSANDO A MÁQUINA VIRTUAL

A máquina virtual do tutorial não tem um ambiente gráfico de trabalho (KDE, Gnome) para reduzir seu tamanho. Entretanto, todos os exercícios poderão ser feitos por meio de acesso SSH e redirecionamento do display gráfico dos programas através do servidor X (Xming no caso do Windows). Para iniciar o encaminhamento do X (X forwarding), é necessário descobrir o endereço IP da máquina virtual.

Acessando o console da máquina virtual pelo VirtualBox, após fazer login com o usuário e senha openflow, execute o comando:

```
$ ifconfig
```

Você deverá ver três interfaces (eth0, eth1, lo), eth0 e eth1 devem ter endereços IP atribuídos. Tome nota do endereço IP atribuído a eth1.

Execute o Xming (nesse primeiro momento ele não cria nenhuma janela na tela, apenas um ícone na bandeja do sistema) e então o Putty. Na primeira janela do Putty digite o IP da interface eth1, então vá à janela Connection->SSH->X11 e clique em 'Enable X11 Forwarding' como mostrado na figura abaixo, então clique em Open.

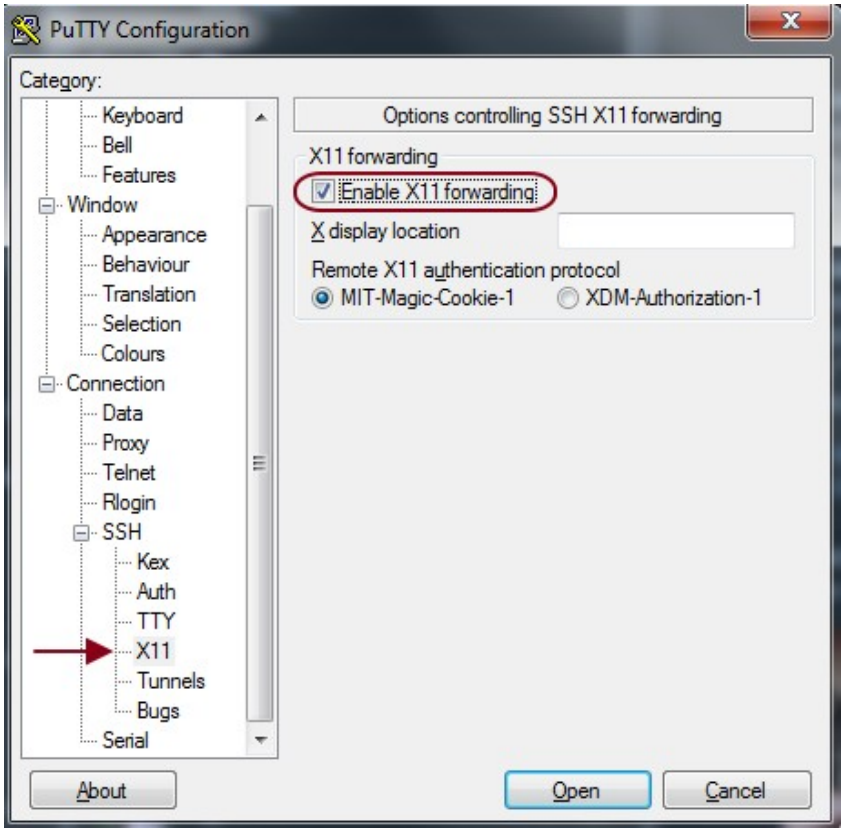

Figura 5 - Configuração do Putty para encaminhamento X11

Fonte: Autoria própria

Faça login mais uma vez com a conta e senha openflow. Então execute o comando:

```
$ xterm
```

Uma janela do terminal básico deve aparecer. Isso é sinal que foi um sucesso a preparação para simular o OpenFlow. Feche a janela do xterm.

3.3 AS FERRAMENTAS UTILIZADAS

A máquina virtual fornecida pelo grupo OpenFlow inclui uma série de utilitários de rede pré-instalados:

- Controlador OpenFlow: situa-se acima da interface OpenFlow. A distribuição de referência do OpenFlow inclui um controlador que age como um Switch Ethernet em combinação com um switch OpenFlow.
- Switch OpenFlow: situa-se abaixo da interface OpenFlow. A distribuição de referência OpenFlow inclui uma opção de switch em software rodando como processo do usuário. Já openvSwitch é uma outra opção de switch em software, mas baseado no kernel do Linux.
- dpctl: utilitário de linha de comando que envia mensagens do protocolo OpenFlow ao switch, útil tanto para visualizar o status do switch quanto para manualmente inserir entradas da tabela de fluxo.
- Wireshark: utilitário gráfico genérico para captura e visualização de pacotes de rede, funciona muito bem com OpenFlow.
- iperf : utilitário usado para testar a velocidade de uma conexão TCP.
- Mininet: plataforma de emulação de rede. O Mininet cria uma rede virtual OpenFlow - com controlador, switches, hosts e links - em uma única máquina real ou máquina virtual.

3.4 SIMULANDO UMA REDE SIMPLES COM O MININET

Agora será simulada uma rede simples composta de três hosts e um switch OpenFlow (e, mais tarde, um controlador OpenFlow)

O comando para criar essa rede usando o mininet é:

```
$ mn --topo single,3 --mac --switch ovsk --controller remote
```

o que define uma topologia contendo 3 hosts e um único switch (do tipo openvSwitch), configura os endereços MAC de cada host de modo a serem parecidos com seus IPs, e aponta para um controlador remoto cujo endereço padrão é o localhost.

O Mininet mostra na tela os passos que está seguindo para criar a rede virtual de acordo com os parâmetros definidos e ao final apresenta seu próprio prompt de comando na tela, que é:

`mininet>`

conforme a imagem:

Figura 6 - início de uma sessão Mininet

Fonte: Autoria própria

Uma vez no prompt vários comandos específicos do mininet podem ser executados, por exemplo:

`mininet> nodes`

para ver a lista de nós disponíveis no console

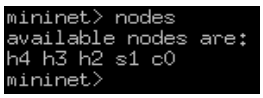

Figura 7 - Lista de nós disponíveis nesta simulação

Fonte: Autoria própria

`mininet> help`

para ver a lista de comandos disponíveis e

`mininet> h2 ifconfig`

Para executar um único comando (ifconfig) em um nó virtual (h2).

Uma grande quantidade de outros comandos e opções de inicialização podem ser pesquisados na documentação oficial do Mininet.

3.5 USANDO O COMANDO DPCTL

dpctl é um utilitário incluído na distribuição de referência OpenFlow que permite visibilidade e controle sobre a tabela de fluxo de um switch. Ele é especialmente útil para identificar erros por permitir exibir o estado e os contadores dos fluxos. A maioria dos switches OpenFlow inicia com uma porta de escuta passiva (em sua configuração atual a porta 6634), na qual é possível interagir com o switch sem ter que passar pelo controlador.

Um exemplo de comando usando dpctl é:

```
$ dpctl show tcp:127.0.0.1:6634
```

O comando 'show' conecta ao switch e descarrega o estado e capacidade de suas portas, como mostrado na imagem:

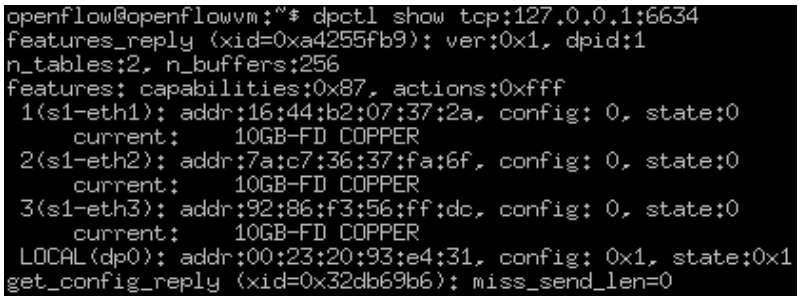

Figura 8 - Estado do switch virtual OpenFlow

Fonte: Autoria própria

Outro comando útil é:

```
$ dpctl dump-flows tcp:127.0.0.1:6634
```

Após conectar ao switch ele descarrega a tabela de fluxos em execução no momento:

Figura 9 - Tabela de fluxo vazia

Fonte: Autoria própria

A tabela de fluxo está vazia. Assim os hosts não conseguirão se comunicar entre si. Para começarmos a popular a tabela podemos rodar os seguintes comandos:

```
$ dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2
$ dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1
```

Isto irá transmitir pacotes vindos da porta 1 para a porta 2 e vice-versa. Agora podemos verificar que a tabela de fluxos foi populada:

```
openflow@openflowvm:~$ dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0xb425be9a): flags=none type=1(flow)
  cookie=0, duration_sec=18s, duration_nsec=655000000s, table_id=0, p
eout=0,in_port=1,actions=output:2
  cookie=0, duration_sec=3s, duration_nsec=968000000s, table_id=0, pr
out=0,in_port=2,actions=output:1
```

Figura 10 - Tabela de fluxos populada

Fonte: Autoria própria

e agora é possível fazer ping entre os hosts:

```
mininet> h2 ping -c2 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_req=1 ttl=64 time=1.38 ms
64 bytes from 10.0.0.3: icmp_req=2 ttl=64 time=0.054 ms

--- 10.0.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 0.054/0.720/1.386/0.666 ms
```

Figura 11 - Usando ping para verificar conectividade

Fonte: Autoria própria

3.5 USANDO O WIRESHARK

O Wireshark é um programa que captura e analisa o tráfego de rede e o organiza por protocolos.

Para executá-lo, a partir da linha de comando execute (como root):

```
$ wireshark &
```

Clique em Capture->Interfaces na barra de menu. Clique no botão Start ao lado de 'lo', que é a interface de loopback. Alguns pacotes começarão a ser capturados. Agora, crie um filtro para o tráfego de controle do OpenFlow, digitando 'of' na Filter box perto do topo. Pressione o botão Apply para aplicar o filtro para todo o tráfego que estiver sendo capturado.

Agora que o Wireshark já está configurado e capturando pacotes iniciaremos o controlador de referência OpenFlow, executando:

```
$ controller ptcp:
```

Isso iniciará um controlador simples que atua como um switch de auto-aprendizado sem instalar entradas manuais na tabela de fluxos. Começarão a aparecer mensagens na janela do Wireshark referentes a mensagens Hello, que são

responsáveis por informar switch e controlador a versão do protocolo OpenFlow que cada um está rodando.

Com o controlador a postos, podemos testar a conectividade da rede usando ping. Nesse caso, convém habilitar o seguinte critério na Filter box para filtrar apenas os pacotes que interessam a essa simulação:

of && (of.type != 3) && (of.type != 2)

Ao executar o ping no console mininet primeiramente será vista uma requisição ARP que não terá associação na tabela de fluxo, essa irá gerar uma mensagem broadcast Packet-Out. Em seguida a resposta ARP volta; com ambos os endereços MAC dos hosts agora conhecidos pelo Controlador, ele então pode empurrar uma entrada de fluxo para o switch através da mensagem Flow-Mod. O switch então empurra entradas de fluxo para os pacotes ICMP. Requisições ping subsequentes vão direto pelo 'datapath', e não geram mais mensagens extras; com as entradas de fluxo conectado H2 e H3 já descarregados no switch, não há mais envolvimento do controlador.

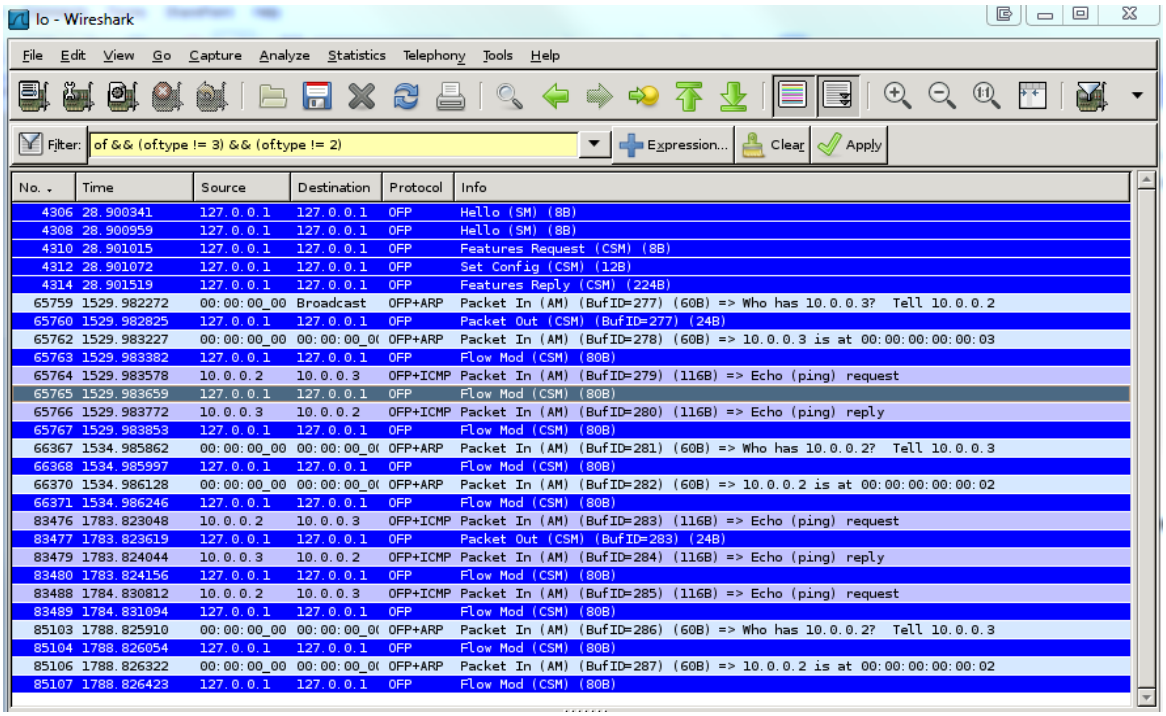

Figura 12 - Interface do Wireshark com pacotes OpenFlow

Fonte: Autoria própria

Na próxima figura vemos um pacote OpenFlow sendo dissecado pelo Wireshark. O cabeçalho contém versão, tipo, tamanho e ID de transação do pacote. Já a seção “Flow Modification” apresenta a associação que deve ser estabelecida, o

comando (é uma inserção de fluxo), o tempo até descartar o fluxo na tabela (60s), prioridade e mais abaixo a ação a ser tomada (saída do pacote pela porta 2 do switch)

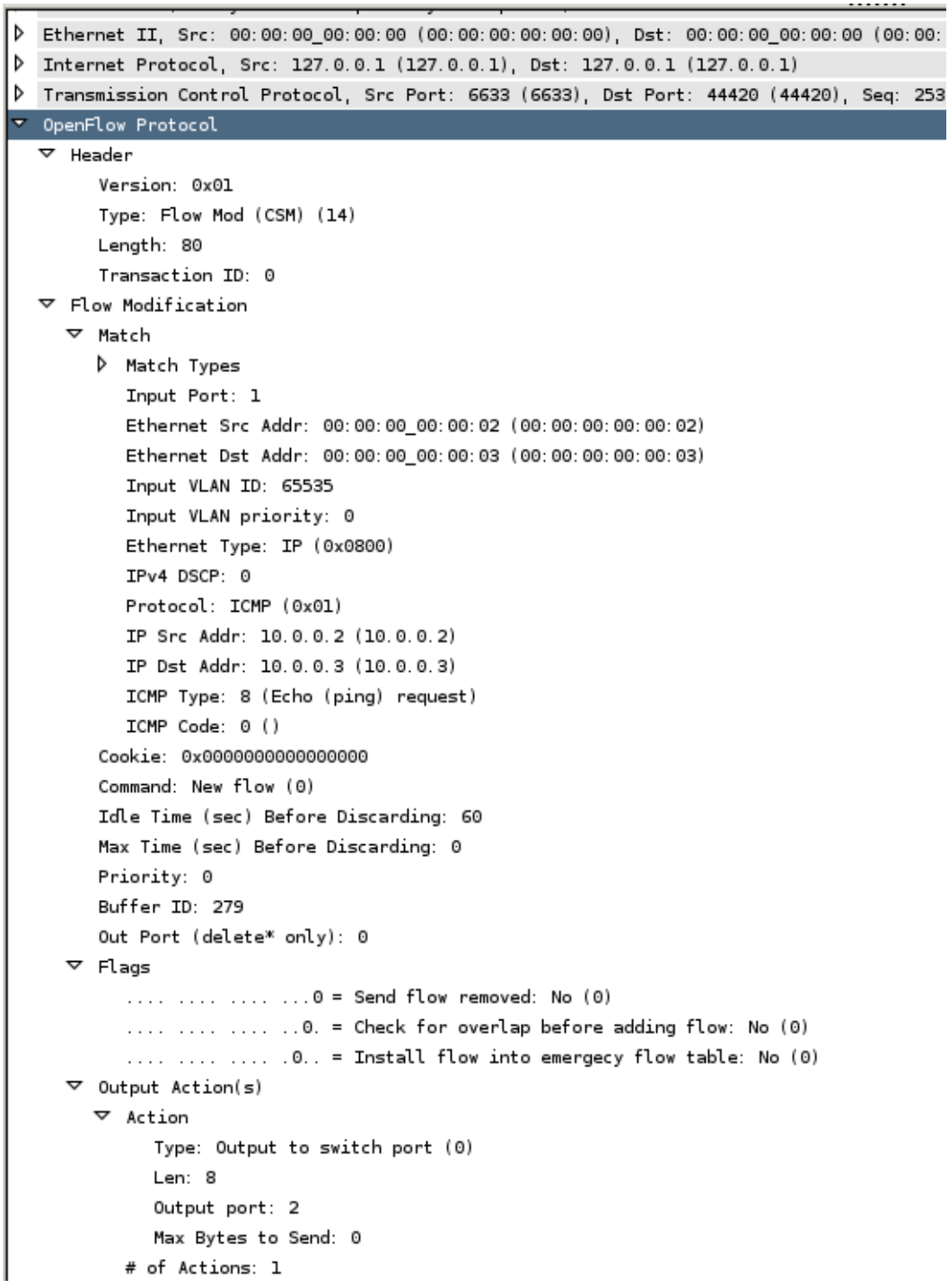

Figura 13 - Um pacote OpenFlow dissecado pelo Wireshark

Fonte: Autoria própria

Uma peculiaridade interessante a se notar nessa experiência é que uma vez estabelecido o datapath, execuções subsequentes do ping são processadas numa velocidade sensivelmente maior já que o controlador não precisa mais aprender o caminho entre os hosts – a informação já estará armazenada na tabela de fluxo do switch. Por outro lado, essa informação é bastante volátil – na configuração padrão, ela dura 60 segundos de inatividade naquele fluxo. Portanto, se esperarmos mais de 60 segundos para enviar o ping, o controlador precisará reaprender o caminho entre os hosts.

Este é um exemplo do uso em modo reativo do OpenFlow, quando os fluxos são descarregados do controlador para o switch em resposta a pacotes individuais. Alternativamente, uma tabela de fluxo completa pode ser descarregada para o switch antes dos pacotes aparecerem. Este é o chamado modo proativo, que evita gastar o tempo de ida e volta ao controlador e os atrasos com inserções na tabela de fluxo. Uma desvantagem relativa é que fluxos não contemplados pelos já descritos na tabela precisarão ser encaminhados ao controlador de qualquer jeito para análise e tomada de decisões.

4 TESTES DE DESEMPENHO OPENFLOW

Ao se analisar uma tecnologia nova é importante avaliar variáveis que interfiram no seu desempenho, de modo a selecionar as melhores alternativas a cada caso de uso.

4.1 DESEMPENHO DE SWITCHES VIRTUAIS

A máquina virtual distribuída pelo grupo OpenFlow vem com duas opções de switches virtuais para uso com o Mininet: um switch rodando em espaço de execução do usuário e o openvSwitch, que roda no espaço do kernel.

Podemos reaproveitar o comando para criar a nossa rede simulada com um switch e 3 hosts. A seguinte linha de comando executa o Mininet com openvSwitch:

```
$ mn --topo single,3 --mac --switch ovsk --controller remote
```

uma vez montada a simulação e já no prompt do mininet rodamos o comando:

```
mininet> iperf
```

Este comando do Mininet executa um servidor TCP iperf em um host virtual, e em seguida, executa um cliente iperf em um segundo host virtual. Uma vez conectados, eles geram uma rajada de pacotes entre si e comunicam os resultados:

```
root@openflowvm:/home/openflow# sudo mn --topo single,3 --mac --switch ovsk
*** Adding controller
*** Creating network
*** Adding hosts:
h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(s1, h2) (s1, h3) (s1, h4)
*** Configuring hosts
h2 h3 h4
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h2 and h4
*** Results: ['2.20 Gbits/sec', '2.21 Gbits/sec']
mininet> exit
*** Stopping 3 hosts
h2 h3 h4
*** Stopping 1 switches
s1...
*** Stopping 1 controllers
*** Done
completed in 15.036 seconds
```

Figura 14 - rodando iperf no switch em modo kernel

Fonte: Autoria própria

Findo o teste com o openvSwitch, podemos testar com o switch em modo usuário:

```
$ mn --topo single,3 --mac --switch user --controller remote
```

E os resultados são:

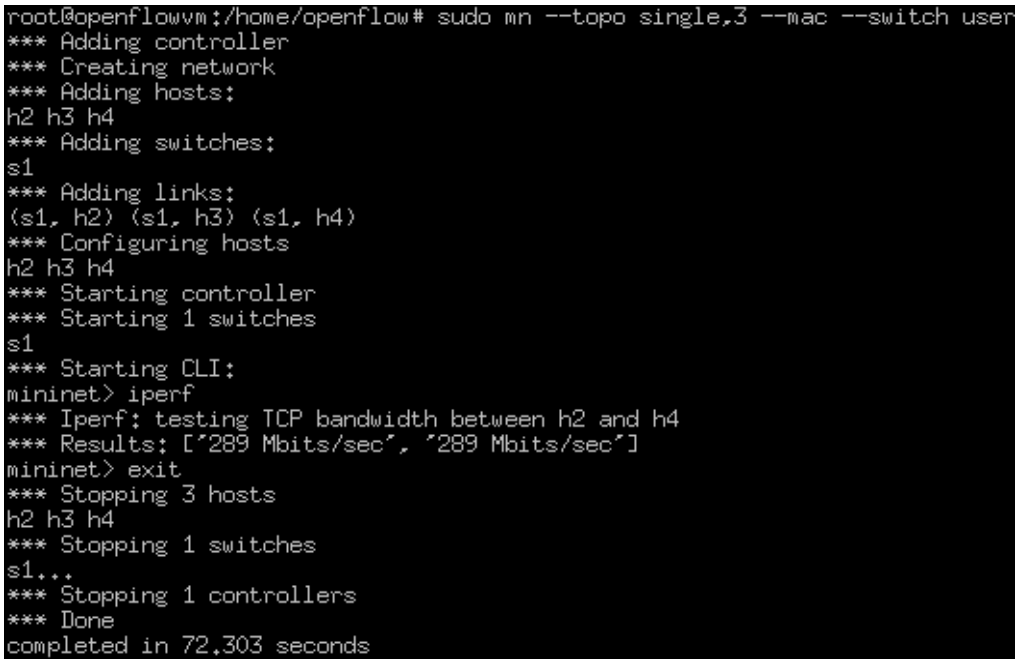

Figura 15 - rodando iperf no switch em modo usuário

Fonte: Autoria própria

O switch em modo kernel é mais de 7.5 vezes mais rápido que o switch em modo usuário. O motivo é que no modo usuário os pacotes precisam cruzar o espaço de memória do usuário para o espaço do kernel e voltar a cada pulo (“hop”) enquanto no modo kernel os pacotes permanecem o tempo inteiro no espaço de memória do kernel enquanto atravessam o switch. Por outro lado o switch de modo usuário é mais fácil de programar e modificar (não existem as restrições impostas a programas rodando no espaço do kernel).

Esse teste foi executado num computador Intel Core2Duo 8400 3ghz (apenas um núcleo de CPU alocado para a máquina virtual) com 8GB de RAM (512MB alocados para a máquina virtual) usando o controlador padrão da distribuição OpenFlow.

4.2 DESEMPENHO DE CONTROLADORES OPENFLOW

Como já mencionado na seção 2.2.4 O Controlador OpenFlowhá uma boa variedade de controladores disponíveis, tanto comerciais quanto gratuitos, tanto os de código livre quanto os proprietários. Aqui será avaliado o desempenho de três controladores. Eles são:

- NOX: um controlador programado em C++ e Python desenvolvido e disponibilizado pela empresa Nicira Networks
- Beacon: um controlador programado em Java desenvolvido por David Erickson da Universidade Stanford
- Maestro: um controlador programado em Java desenvolvido por por Zheng Cai da Universidade Rice

Para sobrecarregar os controladores é usado o aplicativo cbench, parte da distribuição padrão OpenFlow, emulando 32 switches conectados a um controlador que enviam mensagens packet-in (1 milhão de mensagens por switch) e esperam que mensagens flow-mod sejam enviadas de volta, na seguinte configuração:

```
$ cbench -c localhost -p 6633 -m 10000 -l 10 -s 32 -M 1000000 -t
```

E estes são os resultados:

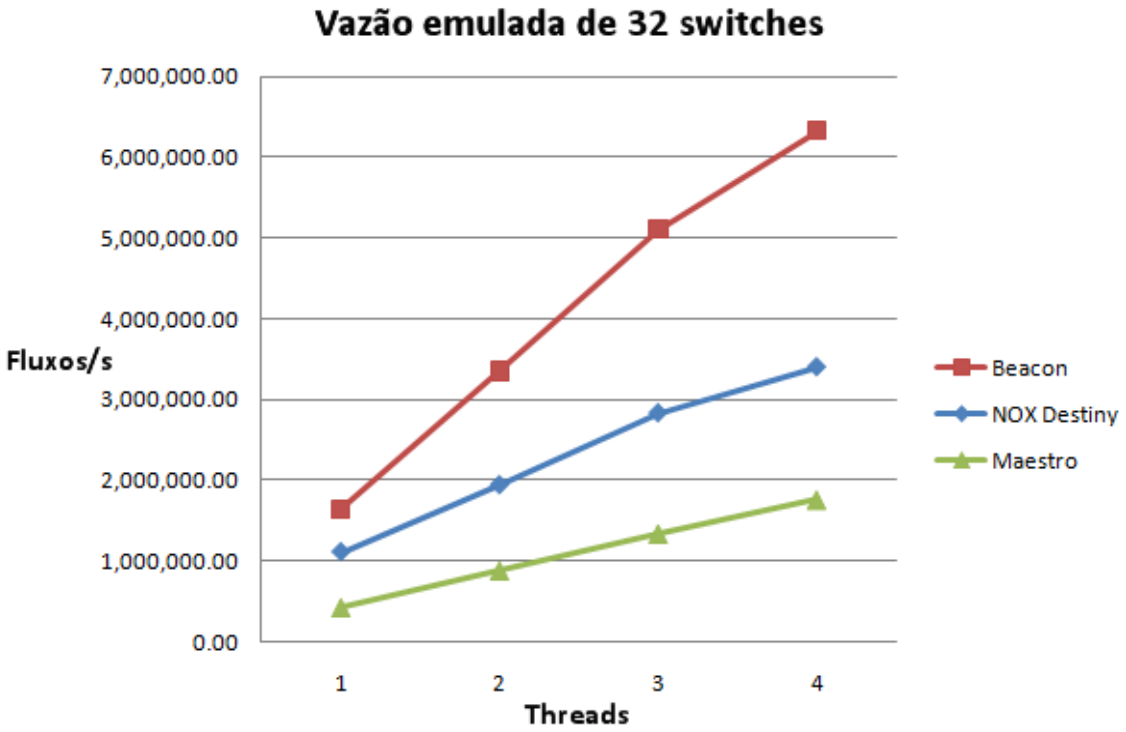

Figura 16 - Desempenho de controladores OpenFlow

Fonte: Autoria própria

No teste executado o controlador Beacon apresentou o melhor desempenho entre os três controladores por uma larga margem, particularmente em situações com vários controladores rodando em paralelo (multithreaded).

Esse teste foi executado num computador Intel Core2Duo 8400 3ghz (apenas um núcleo de CPU alocado para a máquina virtual) com 8GB de RAM (512MB alocados para a máquina virtual).

5 CONCLUSÃO

O protocolo OpenFlow oferece vários benefícios importantes:

- Desempenho e custo: ao remover dos equipamentos o processamento de controle, OpenFlow permite aos switches focar em transmitir tráfego o mais rapidamente possível
- Implementação e teste de novas funcionalidades: OpenFlow permite que administradores de rede lancem mão de novas funcionalidades usando a arquitetura de redes existente implementando-as em software.

Por usar uma arquitetura aberta essas funcionalidades poderiam ser implementadas em todos os switches que suportam OpenFlow a partir do controlador central, independente de fabricante e modelo.

- Segurança e gerenciamento: o controlador centralizado do OpenFlow dá a administradores uma visão unificada da rede, o que possibilita melhor gerenciamento e segurança. Por permitir uma visão clara do fluxo de tráfego o protocolo facilita a identificação de intrusões e gargalos.

O protocolo OpenFlow também permite a administradores priorizar diferentes tipos de tráfego e desenvolver políticas de como a rede deve lidar com congestão, falha de equipamentos e de links.

OpenFlow parece ter seu futuro garantido. Fabricantes de equipamentos de rede como NEC, IBM, HP (antiga 3Com), Brocade e Juniper já têm switches que suportam o protocolo disponíveis no mercado ou em fase de testes. Também há vários fornecedores de software controlador que atendem diferentes demandas empresariais ou acadêmicas. Além disso, como demonstrado neste trabalho, existem aplicativos para várias distribuições de Linux que permitem transformar a máquina num controlador ou num switch virtual OpenFlow.

Por outro lado, por se tratar de uma tecnologia ainda muito nova e em desenvolvimento há riscos. Uma preocupação frequentemente mencionada é se a tecnologia seria capaz de atender redes muito grandes e complexas, já que o controle é centralizado em um único servidor. Nas redes atuais, todos os switches embutem seus controladores, e por isso a rede escala linearmente conforme novos switches são adicionados.

Outra preocupação é com segurança. Como o controle de dezenas, centenas ou até milhares de switches é centralizado em um servidor, bastaria um atacante tomar controle desse servidor para afetar a rede inteira.

Finalmente, há a preocupação que fabricantes de equipamentos implementem apenas funcionalidade básica para vender seus switches como capazes de lidar com OpenFlow, enquanto funcionalidades mais avançadas seriam fornecidas como extensões proprietárias, cancelando a vantagem da interoperabilidade entre diferentes fabricantes.

REFERÊNCIAS BIBLIOGRÁFICAS

ABIresearch **Surge in Video Will Drive Global Data Traffic to More Than 60,000 Petabytes in 2016**. White paper, Londres, 2011.

COMER Douglas E. **Redes de Computadores e internet** 4ª ed. Porto Alegre, Bookman, 2007.

FOROUZAN A. Behrouz. **Comunicação de Dados e Redes de Computadores**. 3ª ed. Porto Alegre, Bookman, 2004.

GENI: Global Environment for Network Innovations, http://www.geni.net/ - Accessado em Novembro/2012.

GIL Antonio C. **Como elaborar projetos de pesquisa**. 5ª ed. São Paulo, Atlas, 2010.

HELLER, Brandon. **OpenFlow Switch Specification v. 1.1.0**. 2011. Disponível em http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf

MCKEOWN, Nick et. al. **NOX: Towards an operating system for networks.** ACM SIGCOMM Computer Communication Review, 2008.

MCKEOWN, Nick et. al. **OpenFlow: Enabling Innovation in Campus Networks.** ACM SIGCOMM Computer Communication Review, 38(2):69–74, 2008. Disponível em http://www.openflow.org/documents/openflow-wp-latest.pdf

PETERSON, Larry L. and DAVIE, Bruce S. **Computer Networks – a systems approach**. 5th ed. Burlington, Elsevier, 2012.

RODRIGUES Paulo Henrique. **Quadro Comparativo Camadas ISO-OSI e TCP/IP**, UNIP/SP, 2009. Disponível em: http://paulohrodrigues.blogspot.com/2009/09/tabelas-iso-osi-e-tcpip.html

SHIMONISHI, Hideyuki et. al. **Programmable Network Using OpenFlow for Network Researches and Experiments**. The Sixth International Conference on Mobile Computing and Ubiquitous Networking, 2012.

TANENBAUM, Andrew S. **Redes de Computadores**. 4ª ed. São Paulo, Elsevier, 2003.

Vários autores. **OpenFlow Tutorial.** Disponível em (última visualização: 03/Dez/2012) http://www.openflow.org/wk/index.php/OpenFlow_Tutorial

VAUGHAN-NICHOLS, Steven J. **OpenFlow: The Next Generation of the Network?** Computer Magazine, vol. 44, ed. 8, pp. 13-15, Agosto de 2011.