UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES

GUILHERME HIERT DA SILVA

REDES DEFINIDAS POR *SOFTWARE*: aplicações práticas

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2016

GUILHERME HIERT DA SILVA

REDES DEFINIDAS POR *SOFTWARE*: aplicações práticas

Trabalho de Conclusão de Curso de Graduação, apresentado ao Curso Superior de Tecnologia em Sistemas de Telecomunicações, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Lincoln Herbert Teixeira

CURITIBA
2016

TERMO DE APROVAÇÃO

GUILHERME HIERT DA SILVA

REDES DEFINIDAS POR *SOFTWARE*: aplicações práticas

Este trabalho de conclusão de curso foi apresentado no dia 09 de junho de 2016, como requisito parcial para obtenção do título de **Tecnólogo em Sistemas de Telecomunicações**, outorgado pela Universidade Tecnológica Federal do Paraná. O aluno foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Kleber Kendy Horikawa Nabas
Coordenador de Curso
Departamento Acadêmico de Eletrônica

_____
Prof. M.Sc Sérgio Moribe
Responsável pela Atividade de Trabalho de Conclusão de Curso
Departamento Acadêmico de Eletrônica

BANCA EXAMINADORA

_____
Prof. Dr. Kleber Kendy Horikawa Nabas
UTFPR

_____
Prof. Dra. Tania Lucia Monteiro
UTFPR

_____
Prof. Me. Lincoln Herbert Teixeira
Orientador - UTFPR

RESUMO

SILVA, Guilherme H. Da. **Redes Definidas por *Software***: aplicações práticas. 2016. 64 f. trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2016.

Este trabalho tem por objetivo caracterizar uma rede definida por *software* e busca, por meio da pesquisa bibliográfica, apresentar seus conceitos, características, composição e aplicações no mercado. Utilizando-se dos emuladores Mininet e GNS3, integrados ao controlador OpenDaylight implementa-se, em um ambiente de testes, uma topologia híbrida, capaz de demonstrar o gerenciamento remoto do fluxo de dados em um switch OpenFlow.

Palavras chave: Rede Definida por *Software*. OpenFlow. OpenDaylight. Hibrida. GNS3.

ABSTRACT

SILVA, Guilherme H. Da. **Software Defined Network**: practical applications. 2016. 64 f. trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2016.

This work has the objective of characterizing a software defined network and intends, by means of bibliographic research, to present its concepts, characteristics, composition and market application. Using Mininet and GNS3 emulators integrated with OpenDaylight controller implements, in a test ambient, a hybrid topology, able to demonstrate the remote management of the data flow in an OpenFlow switch.

Keywords: Software defined network. OpenFlow. OpenDaylight. Hybrid. GNS3.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API	Application Programming Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
ICMP	Internet Control Message Protocol
IDC	International Data Corporation
IP	Internet Protocol
IPv4	Internet Protocol version 4
MAC	Media Access Control
MPLS	Multiprotocol Label Switching
NIB	Network Information Base
ODL	OpenDaylight
OSPF	Open Shortest Path First
OVS	Open vSwitch
REST	Representational State Transfer
RFC	Request for Comments
ROM	Read only Memory
SDN	Software Defined Network
SD-WAN	Software Defined – Wide Area Network
SNAC	Simple Network Access Control
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
TCAM	Ternary Content Addressable Memory
TCP	Transmition Control Protocol
VLAN	Virtual Local Area Network
WAN	Wide Area Network

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta os elementos necessários para que possa situar o leitor com o tema proposto, são eles: Tema, delimitação do tema, problemas e premissas, o objetivo geral, os objetivos específicos, justificativa, procedimentos metodológicos, embasamento teórico e a estrutura deste trabalho.

1.1 TEMA

A década de 90 foi marcada por um aumento significativo no uso de redes de dados em todo o mundo, desafiando a capacidade dos roteadores existentes, que utilizavam tecnologias de *software* embarcados, incapazes de atender a demanda do encaminhamento de pacotes com taxas de transmissão cada vez maiores. A solução para esse problema se deu com a concepção de uma nova arquitetura, baseada na separação em dois planos, sendo um composto pelo sistema operacional, o *software*, responsável pelo controle; e outro, o hardware, responsável pelo encaminhamento do fluxo de dados (JUNIPER, 2010).

Desde então, a *Internet* se tornou conhecida e acessada por uma fração significativa da população, iniciativas de inclusão digital são desenvolvidas em diversas esferas com o objetivo de expandir ainda mais o seu alcance (GUEDES et al, 2012). E apesar de sua evolução, em termos de penetração e de aplicações, a tecnologia, representada pela arquitetura em camadas e pelos protocolos do modelo TCP/IP (*Internet Protocol / Transmition Control Protocol*), não evoluiu suficientemente nos últimos vinte anos (ROTHENBERG et al, 2010).

Trazendo um problema para a comunidade de pesquisa: como grande parte da sociedade depende hoje da *Internet*, testes com novos protocolos e tecnologias não poderiam ser realizados, na rede em geral, devido ao risco de interrupção do serviço (GUEDES et al, 2012).

Guedes et al. (2012) destaca que este cenário levou diversos pesquisadores a afirmarem que a arquitetura de redes de computadores e a rede mundial atingiram

um nível de amadurecimento que as tornaram pouco flexíveis, tendo como resultado o engessamento da *Internet*, complementa Rothenberg et al. (2010).

Ou seja, a rede global tornou-se comercial e os equipamentos de rede verdadeiras “caixas pretas”, baseados no modelo de arquitetura caracterizado por um plano de controle distribuído e verticalizado, que consiste em um *software* fechado sobre *hardware* proprietário (ROTHENBERG et al, 2010).

Em contraste com este modelo “engessado”, avanços na padronização de APIs (*Application Programming Interfaces*) independentes do fabricante, possibilitaram a definição do comportamento da rede em *software*, separando dos dispositivos de rede (*hardware*) a parte lógica (*software*) e centralizando a tomada de decisões, ou seja, como a arquitetura atual é composta por dois planos autocontidos, eles não precisariam estar fechados em um mesmo equipamento, conforme aponta Rothenberg et al. (2010).

Mas para isso acontecer, é necessário que haja uma forma de se programar remotamente; permitindo que o plano de controle possa ser movido para um servidor dedicado (ROTHENBERG et al, 2010). Dessa forma, através de APIs e ferramentas, baseadas no conceito de código aberto (*open source*), seria possível aos administradores customizar o sistema de controle, ajustando precisamente às suas necessidades, especialmente no suporte de aplicações avançadas (JUNIPER, 2010).

A partir desse propósito surge o padrão aberto OpenFlow, proposto pela Universidade de Stanford em 2007, e que tem como objetivo permitir que se utilize equipamentos de rede comercial para pesquisa de novos protocolos de rede, em paralelo com a sua operação normal (GUEDES et al, 2012). Com ele, nasce o conceito de redes definidas por *software* (ROTHENBERG et al, 2010).

Assim, pode-se dizer que a principal característica de uma SDN (*Software Defined Network*) seria a existência de um sistema de controle, que pode ordenar o mecanismo de encaminhamento dos elementos de comutação da rede, por meio de uma API bem definida; ou seja, permite ao *software* a inspeção, definição e alteração dos parâmetros de entrada da tabela de encaminhamento, como é o exemplo de comutadores OpenFlow.

1.1.1 Delimitação do Tema

O conteúdo será abordado a partir da compreensão do conceito de redes definidas por s*oftware*, baseado na separação dos planos de controle e de dados em controladores e comutadores, bem como a apresentação dos recursos e componentes de uma rede SDN, as soluções de controle e protocolos já desenvolvidos, o cenário atual de mercado e as principais aplicações.

Por fim, este trabalho apresentará um estudo de caso, baseado na virtualização de *switches*, habilitados ao protocolo OpenFlow, e roteadores comerciais e na implementação de um controlador centralizado, a fim de demonstrar o funcionamento desta tecnologia. Porém, a simulação não alcançará o âmbito de testes de resiliência, utilizando uma arquitetura de alta disponibilidade.

1.2 PROBLEMAS E PREMISSAS

Tecnologias de engenharia de tráfego mudam de um paradigma do menor esforço, aonde os protocolos de roteamento encontram caminhos mais curtos pelos quais todo o tráfego deve seguir, para um sistema que permite ao gerente controlar o caminho de cada fluxo, cita Guedes et al. (2012), atingindo resultados que minimizam os custos financeiros e impactam positivamente na qualidade dos serviços.

Assim, é possível então uma abordagem que combine as tecnologias de produção e definidas em *software*, permitindo ao operador de rede uma visão global e flexível da rede, e que possibilite a elaboração de soluções redundantes e seguras.

1.3 OBJETIVOS

Nesta seção são apresentados os objetivos geral e específicos do trabalho, relativos ao problema anteriormente apresentado.

1.3.1 Objetivo Geral

Analisar o funcionamento das redes definidas por *Software* e do protocolo OpenFlow.

1.3.2 Objetivos Específicos

Dissertar a respeito dos conceitos de redes definidas por *software* e do protocolo OpenFlow.

Simular um ambiente de testes híbrido, baseado na virtualização de *switches* OpenFlow conectados a um controlador centralizado, por meio de uma rede de produção.

Permitir que o gerenciamento de tráfego (fluxos) seja implementado remotamente através de uma interface *web*.

Analisar a comunicação entre controlador e comutador através da captura de pacotes.

1.4 JUSTIFICATIVA

Ponderando a respeito da arquitetura atual dos equipamentos de rede é possível estabelecer um padrão: um modelo composto por duas camadas ou partes, ou seja, plano de dados e plano de controle; são acessíveis através de APIs limitadas

disponibilizadas pelo fabricante, que possibilita em sua essência apenas o controle de funções predeterminadas, inviabilizando a manipulação de pacotes específicos; como apresenta Jacob (2015).

Entretanto, como esclarece Rothenberg et al. (2010), se a arquitetura é composta por duas camadas, elas não precisam estar inseridas em um mesmo equipamento, o que torna possível a especialização do *software* de controle por meio de um protocolo aberto.

Essa solução foi alcançada no protocolo OpenFlow, que permite a definição de um controlador externo pelo qual o administrador de rede pode manipular o fluxo de dados entre os equipamentos de rede; seja para operar como uma rede de produção, seja para fins acadêmicos, com testes de protocolos em desenvolvimento e fluxos não padronizados sem o risco de interrupção dos serviços.

1.5 PROCEDIMENTOS METODOLÓGICOS

O estudo aqui proposto, segundo Silva e Menezes (2005), quanto à sua natureza, trata-se de uma pesquisa aplicada, apresentando uma abordagem quanti-qualitativa, é “descritiva” e “objetiva gerar conhecimentos para aplicação prática e dirigida à solução de problemas específicos” (SILVA; MENEZES, 2005), envolvendo verdades e interesses locais; considerando a abordagem do problema proposto, trata-se de uma pesquisa quantitativa, pela análise numérica das opiniões e informações coletadas (SILVA; MENEZES, 2005).

1.6 EMBASAMENTO TEÓRICO

O foco deste trabalho está voltado para a bibliografia especializada em SDN, abrangendo os conceitos, arquitetura e componentes envolvidos - incluindo o protocolo OpenFlow. Destacam-se as pesquisas desenvolvidas por Rothenberg et al. em 2010, Guedes et al. em 2012 e o livro publicado recentemente por Comer, em 2015, no Brasil. Como referencial para o desenvolvimento do ambiente de testes

foram utilizados os trabalhos acadêmicos de Amorim (2012), Jacob (2015) e Rechia (2014).

1.7 ESTRUTURA

Este trabalho será dividido em cinco capítulos, conforme a estrutura abaixo apresentada.

Capítulo 1 – Introdução: serão apresentados o tema, as delimitações da pesquisa, os problemas e premissas, os objetivos deste trabalho, a justificativa, os procedimentos metodológicos, o embasamento teórico e a estrutura descrita nesta seção.

Capítulo 2 – Referenciais Teóricos: serão abordados os conceitos sobre a tecnologia de redes definidas por *software*, apresentando a sua arquitetura e os elementos que a compõem, incluindo o protocolo OpenFlow.

Capítulo 3 – Mercado e Aplicações: será apresentada a situação atual do mercado e enumeradas as aplicações para a tecnologia de redes definidas por software, abordando desde soluções para redes domésticas até sua implantação em Data Centers e provedores.

Capítulo 4 – Implementação: será implementado um laboratório de testes para implantação de uma topologia híbrida IP e OpenFlow, demonstrando as configurações específicas e apresentando as características técnicas desta topologia e suas validações.

Capítulo 5 – Conclusão: serão apresentadas a conclusão do trabalho desenvolvido e recomendações para trabalhos futuros.

2 REFERENCIAIS TEÓRICOS

Para enfrentar o problema da “calcificação” da internet, aponta Guedes et al (2012), a comunidade de pesquisa tem investido em recursos que levem à implantação de redes com maiores recursos de programação.

A iniciativa de maior sucesso, foi a definição do protocolo OpenFlow, um padrão aberto, que tem como principal objetivo a separação dos planos de controle e de dados, permitindo a operação de tráfegos de produção em paralelo com fluxos experimentais (GUEDES et al, 2012).

2.1 REDES DEFINIDAS POR *SOFTWARE*

Do ponto de vista histórico, as redes definidas por *software* têm sua origem na definição da arquitetura de redes *Ethane,* que determinava uma forma de se implementar políticas de controle de acesso, de forma distribuída, a partir de um mecanismo de supervisão centralizado, aponta Guedes et al. (2012). Ela foi testada no Campus da Universidade de Stanford, através de um conjunto de equipamentos que serviam mais de 300 hosts (RECHIA, 2014).

Naquela arquitetura, cada elemento de rede deveria consultar o supervisor que decidiria com base em um grupo de políticas globais, como o elemento de encaminhamento deveria tratar o novo fluxo de dados identificado. Essa decisão seria informada a um elemento comutador, programando uma regra na tabela de encaminhamento. Esse modelo foi posteriormente formalizado por alguns autores na forma da arquitetura OpenFlow (GUEDES et al, 2012).

Essa estrutura, conforme descreve Guedes et al. (2010), permite que a rede seja controlada de forma extensível através de aplicações expressas em *software*. Ao novo paradigma se deu o nome de redes definidas por *software*.

2.2 ARQUITETURA

Analisando a arquitetura atual dos roteadores, é possível observar que se trata de um modelo formado em sua essência por dois planos: um composto pelo sistema operacional, o *software*, responsável pelo controle; e outro, o hardware, responsável pelo encaminhamento do fluxo de dados (JUNIPER, 2010).

O primeiro, seria encarregado de tomar as decisões de roteamento que devem ser executadas pelo plano de encaminhamento através de uma API proprietária; sendo que a única interação da gerência com o dispositivo ocorre através de interfaces de configuração, limitando o uso às funcionalidades preestabelecidas pelo fabricante (ROTHENBERG et al, 2010).

É coerente pensar que numa arquitetura caracterizada por dois planos auto compostos, não precisam necessariamente estar inseridos em um mesmo equipamento; para isso basta que exista uma forma padrão de programar o dispositivo de rede remotamente, transferindo o plano de controle para um dispositivo externo com alta capacidade de processamento, mantendo o alto desempenho de encaminhamento exercido pelo *hardware* aliado a flexibilidade de se inserir, remover e especializar o *software* por meio de um protocolo aberto; solução alcançada pelo conceito OpenFlow (ROTHENBERG et al, 2010).

A figura 1 exibe a arquitetura atual dos roteadores contraposta com a arquitetura em equipamentos de uma rede definida por *software*:

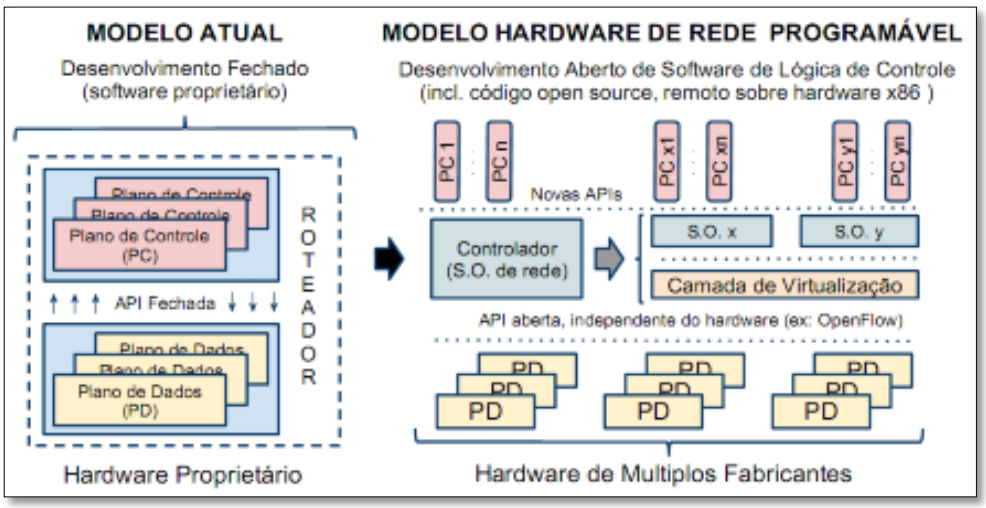

Figura 1: Arquitetura - modelo atual e modelo SDN
Fonte Rothenberg et al. (2010)

Na figura 2, Comer (2015) apresenta de forma mais simples a arquitetura SDN, nela um pequeno módulo é adicionado a um dispositivo que permite que o sistema externo configure o *hardware* subjacente; esse sistema é chamado de controlador.

Nesse sentido, é possível determinar que uma rede definida por *software* é caracterizada por um controlador externo que pode controlar a estrutura de encaminhamento dos elementos de comutação da rede através de uma interface de programação, ou seja, os elementos de comutação permitiriam a um dispositivo remotamente localizado a inspeção, definição e alteração dos parâmetros da tabela de encaminhamento. Como ocorre, por exemplo, com comutadores OpenFlow. (GUEDES et al, 2012).

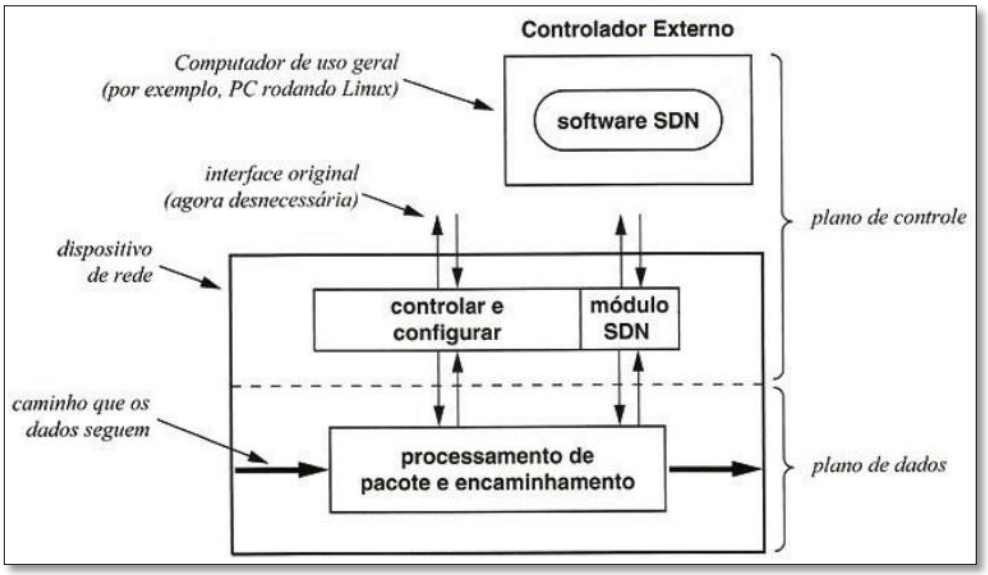

Figura 2: Arquitetura básica SDN com um controlador externo
Fonte: Comer (2015)

Portanto, uma rede programável com OpenFlow consiste, basicamente, em equipamentos de rede habilitados para que o estado das tabelas de fluxos possa ser instalado através de um canal seguro, seguindo as decisões de um controlador em *software* (ROTHENBERG, 2010).

Na seção seguinte será abordado, conforme descreve Guedes (2012), o princípio básico das redes definidas por *software*: possibilitar a programação dos elementos de rede, o que se baliza na manipulação de pacotes baseados no conceito de fluxos, ou seja, sequências de pacotes que compartilham atributos com valores bem definidos.

2.3 COMUTADORES E TABELAS DE FLUXOS

Um fluxo é constituído pela combinação de campos do cabeçalho do pacote a ser processado pelo dispositivo. As “Tuplas”, estruturas de dados ordenadas e que não permitem alteração após sua criação, podem ser formadas por campos das camadas de enlace, de rede ou de transporte, segundo o modelo TCP/IP, complementa Rothenberg (2010).

Em resumo, cada entrada na tabela de fluxos do *hardware* de rede consiste em regra, ações e contadores; onde a regra seria formada com base na definição do valor de um ou mais campos do cabeçalho do pacote, associando um conjunto de ações que definirão o processamento e encaminhamento de cada pacote; alimentando estatísticas de utilização, registradas pelos contadores, que seriam utilizadas para remover fluxos inativos (ROTHENBERG et al, 2010).

Nesse contexto, a mínima unidade de informação, no plano de dados da rede, seriam as entradas da tabela de fluxos, interpretadas como decisões, pelo *hardware,* do plano de controle (*software*), apresenta Rothenberg (2010).

Importante destacar que a abstração da tabela de fluxos ainda está sujeita a refinamentos, com o objetivo de aproveitamento máximo do *hardware*, permitindo a concatenação de várias tabelas já disponíveis, como, por exemplo, tabelas IP, *Ethernet* e MPLS (*Multiprotocol Label Switching*) (ROTHENBERG et al, 2010).

2.4 CANAL SEGURO

Para que a rede não sofra interferências externas, tais como ataques de elementos mal-intencionados, o meio de comunicação entre os elementos de rede e o controlador deve ser seguro, garantindo a confiabilidade das informações. Normalmente, a interface de acesso recomendada é o protocolo SSL (do termo em inglês *Secure Socket Layer*). Porém em ambientes virtuais e experimentais, alternativas de interfaces (passivas ou ativas) como podem ser utilizadas, devido a sua simplicidade de utilização, pois não necessitam de chaves criptográficas (ROTHENBERG et al, 2010).

2.5 CONTROLADOR

É o *software* responsável por tomar as decisões, adicionar e remover as entradas na tabela de fluxo de acordo com o objetivo desejado. Exerce a função de uma camada de abstração da infraestrutura física facilitando a criação de aplicações e serviços para o gerenciamento da rede (ROTHENBERG et al, 2010). Guedes et al. (2012) utiliza o termo “Sistema Operacional” e define como sendo esse elemento o mais importante na definição de uma rede definida por *software*, talvez mais importante até que o próprio protocolo OpenFlow.

Em teoria, seria possível a instalação de um novo sistema de controle em um dispositivo de rede, sem a necessidade de adicionar mais *hardware*; através do carregamento de um novo *software* no equipamento; uma vez que na maioria dos sistemas de controle atuais, um processador embutido executa o programa de controle a partir do *software* armazenado em uma ROM (do termo em inglês para memória apenas de leitura). Mas isto seria impraticável, uma vez que muitas execuções de *software* de controle são específicas para o *hardware* subjacente, sendo que o plano de controle deve ser especializado para cada dispositivo de *hardware* (COMER, 2015).

O *software* envolvido, afirma Guedes et al. (2012) apesar de poder ser uma aplicação especialmente desenvolvida, na realidade tende a ser organizado com base em um controlador de aplicação geral, em torno do qual se constroem aplicações específicas para o fim de cada rede; permitindo inclusive, que haja um divisor de visões para que as aplicações sejam separadas entre diferentes controladores.

Um dos pontos fortes de SDNs é a formação de uma visão centralizada das condições de rede, sobre a qual é possível desenvolver análise detalhadas. Contudo, é importante esclarecer que essa noção de visão centralizada não exige que o controlador opere fisicamente instalado em um único ponto da malha de rede. Deve-se abstrair para uma visão global do sistema, onde o controle pode muito bem ser implementado de forma distribuída, o que permite desenvolvimentos voltados para grandes sistemas, como *data center* de grandes corporações; garantindo escalabilidade e alta disponibilidade das SDNs (GUEDES, 2012).

A centralização do controle em conjunto com uma visão geral e consistente da rede permitem a inovação e uma fácil implementação de serviços diretamente no

núcleo da rede, todavia a centralização física do controle em SDNs pode implicar em desafios à segurança bem como prejudicar o desempenho. A escalabilidade também é prejudicada tanto no distanciamento geográfico entre comutadores e controlador, que não estariam próximos, quanto ao número de elementos de rede já que o número de controladores pode não ser suficiente para responder a todas as requisições dos comutadores (MATTOS et al, 2015).

Na abordagem distribuída, Comer (2015) levanta questões como a quantidade de controladores necessários e como deveriam ser distribuídos, dependendo dos tipos de dispositivos de rede a serem controlados e do *software* SDN utilizado. Mattos et al, (2015) acrescenta que estas questões norteiam a busca para soluções de controle otimizadas, devendo considerar uma função objetivo que pode maximizar a distribuição de controladores para tolerar falhas, ao passo que pode também aumentar o desempenho na medida em que a latência e o tempo de configurações de fluxos são reduzidos.

Nesse sentido, há também o desafio da distribuição do estado na rede de controle que consiste na comunicação entre os roteadores (MATTOS et al, 2015). Este cenário pode ser melhor observado na figura 3.

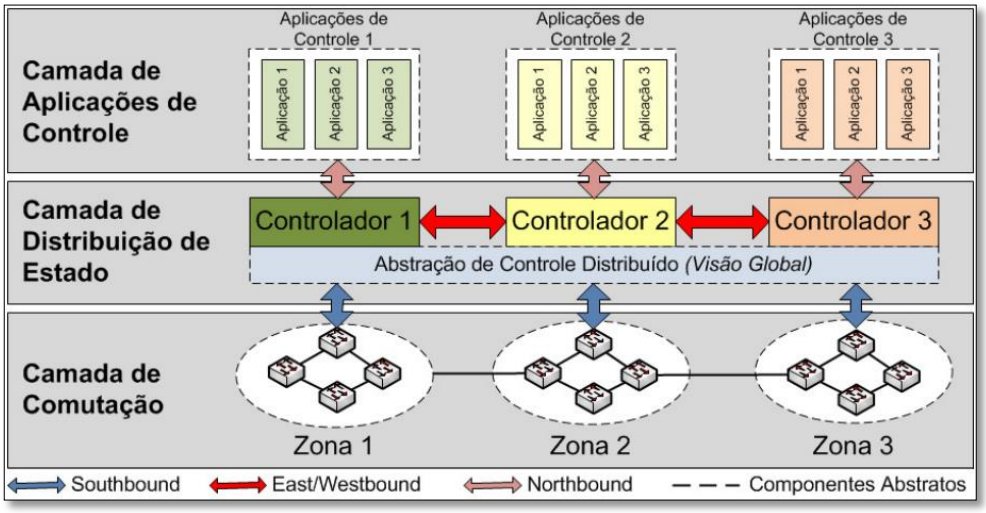

Figura 3: Rede Definida por *Software* com controle distribuído
Fonte: Mattos (2015).

A ideia é que a rede seja dividida em três camadas lógicas. A de comutação seria representada pelos elementos de encaminhamento de pacotes (*switches*), esses elementos armazenam as tabelas de encaminhamento e, conforme ocorre no OpenFlow, manteriam também as estatísticas relativas aos fluxos utilizando

contadores. A camada de distribuição, ou “Sistema Operacional da Rede”, constitui a comunicação entre os controladores, divulgando os eventos e o estado local de cada controlador; esta camada pode ser designada a um controlador, ao qual é atribuída a responsabilidade de manter uma cópia atual da rede.

A última camada lógica é a de aplicações de controle, que se comunica utilizando a camada intermediária, e pode ser executada em um nó específico ou em múltiplos nós (MATTOS et al, 2015).

2.5.1 Principais Controladores

Nesta seção serão enumerados os principais controladores utilizados para aplicações SDN e, de maneira abreviada, as suas principais caraterísticas.

2.5.1.1 NOX

Atualmente a maioria dos projetos de pesquisa na área de SDNs são baseados em NOX, um sistema operacional simples, que tem como principal função o desenvolvimento de controladores eficientes em C++.

Ele opera sobre o conceito de fluxo de dados, onde o primeiro pacote é analisado e a tabela de encaminhamentos consultada para determinar a política a ser aplicada. O controlador realiza o gerenciamento das regras instaladas nos comutadores fornecendo a base para o gerenciamento de eventos de rede (GUEDES et al, 2012).

Guedes descreve ainda, que o NOX permanece como um ambiente adequado para implementações que tenham demandas mais elevadas em termos de desempenho; enquanto uma opção baseada neste o modelo, o POX, desenvolvido para utilizar a linguagem Python consiste em uma interface mais ‘elegante’ e simples.

2.5.1.2 Trema

Trata-se de uma implementação OpenFlow voltada ao desenvolvimento de controladores utilizando as linguagens C e Ruby; sendo seu principal objetivo facilitar a criação de controladores, permitindo inclusive, adicionar manipuladores de mensagem para a classe de controle.

Seu foco não é fornecer uma implementação específica de controlador OpenFlow. Também dispõe de um emulador de redes OpenFlow integrando, que automaticamente prepara *switches* OpenFlow e *hosts* para testar aplicações de controlador (GUEDES et al, 2012).

2.5.1.3 Onix

É um controlador de uso restrito e proprietário desenvolvido em parceria pela Nicira, Google e NEC, com o objetivo de permitir o controle de redes de grandes dimensões de forma confiável.

Graças ao modelo de controle distribuído hierárquico, provê abstrações para particionar e distribuir o estado de rede, endereçando as questões de escalabilidade e tolerância a falha, que surge quando se utiliza o modelo de controle centralizado (GUEDES et al, 2012)

A visão global da rede é representada por um grafo de todas as entidades presentes na rede física e provida por uma estrutura de dados denominada NIB (*Network Information Base*), o centro do sistema e a base para o modelo de distribuição ONIX.

As aplicações são implementadas com base na leitura e atualização desta base, e com particionamento e replicação entre os servidores do sistema é possível prover escalabilidade a rede e a consistência das atualizações.

Sobre a base: é controlada pela aplicação, utilizando-se de dois repositórios com compromissos estabelecidos entre resiliência e desempenho (GUEDES et al, 2012)

2.5.1.4 SNAC (*Simple Network Access Control*)

É um controlador OpenFlow que utiliza interface *web* para monitorar a rede. Ele incorpora uma linguagem de definição de políticas flexível que permite realizar buscas com base em padrões de alto nível para especificação de regras de controle de acesso (GUEDES et al, 2012).

2.5.1.5 OpenDaylight

É uma plataforma de código aberto desenvolvida pelo consórcio Linux Foundation com a finalidade de atender desde ambientes desenvolvidos para testes até grandes redes de produção, suportando uma grande variedade de protocolos de controle, incluindo OpenFlow; provendo uma ampla diversidade de serviços de rede e permitindo a fácil adição de novas funcionalidades, na forma de plug-ins, serviços de rede e aplicações (OPENDAYLIGHT.ORG, 2016).

O projeto conta com parceiros como: Brocade, Cisco, Citrix, Dell, HP, Intel, Juniper, Microsoft, NEC, Red Hat, VMWare, que dentre outros contribuem para o seu desenvolvimento (OPENDAYLIGHT.ORG, 2015).

Tem por objetivo acelerar a adoção do SDN em redes de produção. Destaca-se na categoria de controladores utilizando-se de protocolos abertos como o OpenFlow para estabelecer comunicação com os equipamentos da rede.

Fornece por meio de uma interface baseada em REST (do termo em inglês para transferência de estado representacional) o controle por parte das aplicações e permite a administração dos fluxos dos nós controlados (JEFFERSON, 2015).

Uma vez enumerados os componentes de uma SDN é preciso definir o padrão pelo qual o controlador atribuirá as regras nas tabelas de fluxos dos comutadores através de um canal seguro, essa comunicação é possível através do protocolo OpenFlow.

2.6 O PROTOCOLO OPENFLOW

Trata-se de um protocolo aberto, concebido para comunicação e troca de mensagens entre controlador e os elementos de rede, conforme explica Rothenberg et al. (2010), permitindo que se utilize equipamento de redes comerciais para pesquisa e experimentação de novos protocolos de rede em paralelo com a operação de redes “de produção” complementa Guedes et al. (2012).

Isso é alcançado graças a definição de uma interface de programação que permite ao desenvolvedor controlar diretamente os elementos de encaminhamento de pacotes de um determinado comutador, que pode ser um equipamento de rede comercial, que normalmente possui um poder de processamento maior do que equipamentos concebidos virtualmente em ambientes de testes (GUEDES et al, 2012).

2.6.1 Características

O OpenFlow apresenta uma tecnologia de virtualização de rede, na qual comutadores podem ser configurados para lidar com o tráfego de rede especializado, como protocolos experimentais fora do padrão IPv4 (*Internet Protocol version* 4), e tráfego de rede de produção; este último de vital importância para o OpenFlow por ser responsável pela comunicação entre os comutadores; ou seja, assim como uma rede de gerência SNMP (*Simple Network Management Protocol*) necessita que todos os elementos de rede estejam inseridos em uma rede gerenciável, o OpenFlow necessita que todos os comutadores estejam conectados ao controlador (COMER, 2015).

Atualmente existem duas versões do protocolo OpenFlow que, em princípio serão denominadas “básica” e “avançada”; apesar das especificações e funcionalidades que diferenciam a primeira da segunda, três características são comuns às duas versões, que sejam: a comunicação utilizada entre comutadores e controlador, tipos de comutadores e parâmetros que podem ser configurados além do formato das mensagens trocadas durante a comunicação entre elementos da rede OpenFlow e o controlador (COMER, 2015).

Define que o controlador e os comutadores utilizem TCP para se comunicar, no entanto não há necessidade de que haja uma conexão física direta entre eles, podendo ser utilizada uma rede de produção; entretanto especifica que seja utilizado um canal seguro para a comunicação, de maneira a garantir a confidencialidade de todas as informações (COMER, 2015).

De acordo com as especificações, um comutador “Tipo 0”, termo para comutadores OpenFlow de configuração básica, dispõe de uma tabela de fluxo que implementa classificação e encaminhamento.

A parte de classificação de uma tabela de fluxo contém um conjunto de padrões que são confrontados aos pacotes, na maioria dos comutadores esta correspondência é implementada com *hardware* TCAM (do termo em inglês para memória de endereçamento de conteúdo ternário), que usa paralelismo para executar classificação, atingindo assim altas velocidades de processamento. O OpenFlow permite ao fornecedor a escolha da implementação a ser utilizada para classificação de tráfego (COMER, 2015).

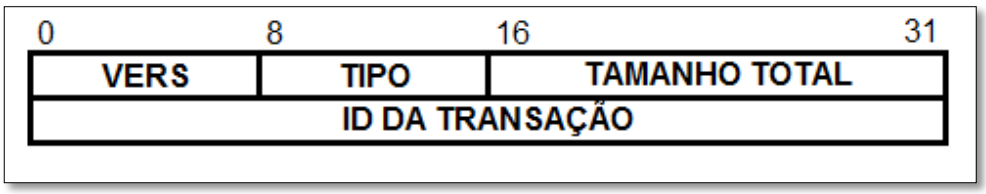

Figura 4: Cabeçalho fixo para mensagens OpenFlow.
Fonte: Comer (2015).

Uma mensagem de OpenFlow começa com um cabeçalho de tamanho fixo que compreende 16 octetos conforme apresentado na figura 4. O campo VERS refere-se ao número de versão. O OpenFlow define 24 tipos de mensagens que podem ser definidos no campo TIPO. Enquanto que em TAMANHO TOTAL, medido em octetos, determina-se o tamanho total da mensagem, incluindo o cabeçalho. O campo ID DA TRANSAÇÃO é um valor único que permite que o controlador atribua respostas a determinados pedidos (COMER, 2015).

O OpenFlow define o formato exato das mensagens e uma representação para os itens de dados; conforme especificações, estabelece um conjunto mínimo de requisitos e campos que um comutador na versão básica deve ser capaz de corresponder (COMER, 2015).

Campo	Significado
Ethersrc	Endereço de origem Thernet 48-bit
Etherdst	Endereço de destino Ethernet 48-bit
EtherType	Campo tipo Ethernet 16-bit
VLAN id	VLAN tag 12-bit no pacote
IPv4src	Endereço de origem IPv4 32 bits
IPv4dst	Endereço IPv4 de destino para IPv4 32 bits
Proto	Campo protocolo IPv4 8 bits
TCP/UDP/SCTP src	Porta origem TCP/UDP/SCTP 16 bits
TCP/UDP/SCTP dst	Porta destino TCP/UDP/SCTP 16 bits

Figura 5: Campos para classificação em comutadores OpenFlow Tipo 0. Fonte: Comer (2015).

Importante observar, na figura 5, que muitos dos campos apresentados estão relacionados a protocolos convencionais existentes, permitindo desta forma interação do tráfego SDN com redes IPv4 e *Ethernet*. Entretanto, comutadores “Tipo 0” apresentam limitações, por exemplo, como a impossibilidade de especificar o encaminhamento de tráfego ICMP (*Internet Control Message Protocol*) e distinguir entre solicitações ARP (*Address Resolution Protocol*); mas permite que pesquisadores estabeleçam regras de encaminhamento especial para o tráfego em determinada porta do comutador, admitindo assim muitos ensaios (COMER, 2015).

As limitações apresentadas pela versão básica de controladores OpenFlow são superadas pela versão 1.1 ou “avançada”, que adiciona funcionalidades consideráveis, como inserção de campos no cabeçalho de pacote e inclusão de novos protocolos como MPLS (COMER, 2015).

Nesta nova versão, em vez de uma única tabela de fluxos os controladores assumem a possibilidade de múltiplas tabelas de fluxos dispostas em uma *pipeline*, onde a primeira confrontação ocorre sempre com a primeira tabela de fluxo, na qual são definidas ações específicas para cada entrada, que podem definir que determinado pacote seja encaminhado para o próximo fluxo da tabela ou a próxima tabela de fluxo; no entanto o salto nunca poderá realizar o sentido contrário, ou seja, um pacote não poderá acessar a tabela anterior, evitando a ocorrência de *loop* (COMER, 2015).

A nova versão também apresenta novos campos; para uso na classificação de pacotes, como é o caso dos denominados “campos de correspondência”, apresentados na figura 6. Ou destinados a serem utilizados dentro da *pipeline*, como é o caso do campo “*Metadata”*. Por exemplo, um determinado estágio da *pipeline*

pode, com base no endereço IPv4, calcular o próximo estágio ao qual determinado pacote deve ser encaminhado, dentro da própria *pipeline*; entretanto, o OpenFlow não especifica o seu conteúdo, pois a *pipeline* deve ser organizada de maneira que os estágios subsequentes conheçam o conteúdo e o formato de dados recebido pelo estágio anterior.

Campo	Significado
IngressPort	Porta do comutador pela qual o pacote chegou
Metadata	Campo de 64 bits de metadados usados na pipeline
Ether src	Endereço de origem Ethernet 48-bit
Ether dst	Endereço de destino Ethernet 48-bit
Ether Type	Campo tipo Ethernet 16-bit
VLAN id	VLAN tag 12-bit no pacote
VLAN priority	Número MPLS prioritário VLAN 3-bit
MPLS label	Rótulo MPLS 20-bit
MPLS class	Classe de tráfego MPLS 3-bit
IPv4 src	Endereço de origem IPv4 32-bit
IPv4 dst	Endereço de destino IPv4 32-bit
IPv4 Proto	Campo protocolo IPv4 8-bit
ARP opcode	Opcode ARP 8-bit
IPv4 tos	Bits tipo de serviço IPv4 8-bit
TCP/UDP/SCTP src	Porta origem TCP/UDP/SCTP 16-bit
TCP/UDP/SCTP dst	Porta destino TCP/UDP/SCTP 16-bit
ICMP type	Campo tipo ICMP 9-bit
ICMP code	Campo código ICMP 8-bit

Figura 6: Campos para classificação em comutadores OpenFlow v1.1.
Fonte: Comer (2015).

Uma vez apresentada a definição do protocolo e conhecendo a estrutura das mensagens é possível apresentar o seu funcionamento e a forma como os pacotes são analisados de acordo com a regras encaminhamento instaladas.

2.6.2 Funcionamento

Ao receber um pacote, o comutador habilitado ao protocolo OpenFlow analisa o cabeçalho e compara as regras de encaminhamento definidas nas tabelas de fluxo. Os contadores são atualizados e são executadas as ações correspondentes. Em caso de não haver relação do pacote recebido com regras na tabela de fluxo o pacote é

encaminhado, por completo, ao controlador. Alternativamente, o pacote pode ser armazenado em um *buffer* no *hardware* (ROTHENBERG et al, 2010).

Basicamente o OpenFlow define três ações básicas que podem ser associadas com um padrão de classificação, enumeradas por Comer (2015) como:

1. Encaminhar o pacote para uma ou um conjunto de portas do comutador; nesta ação, quando um comutador é inicializado ele carrega regras já configuradas pelo fornecedor, desta forma todos os pacotes que chegarem serão encaminhados. A possibilidade de estabelecer regras para um determinado conjunto de portas permite a implementação de *broadcast* e *multicast*.
2. Encapsular o pacote e enviar para o controlador externo, permitindo a este, o tratamento de pacotes para os quais não foram definidas previamente as regras de encaminhamento, escolhendo um caminho para o fluxo TCP, configurando uma regra de classificação e em seguida encaminhando o pacote novamente para o comutador que, por sua vez, irá continuar o processamento do pacote.
3. Descartar o pacote sem qualquer processamento, permitindo ao OpenFlow lidar com problemas, como ataque de negação de serviços ou *broadcast* excessivo de determinado *host*.

Rothenberg et al (2010) completa, apontando o item que contempla o encaminhamento do pacote para o processamento normal do equipamento nas camadas 2 ou 3, permitindo assim que um tráfego experimental não interfira no processamento padrão do tráfego de produção. Também apresenta outra forma de garantir esta separação, através da configuração de VLANs (do termo em inglês para redes locais virtualizadas) para fins experimentais.

Essa segmentação permite a operação de equipamentos híbridos, processando o tráfego legado conforme os protocolos e as funcionalidades embarcadas no equipamento e, simultaneamente de forma isolada, obter tráfego baseado no desenvolvimento OpenFlow. Um exemplo de modelo híbrido de tráfego é mostrado na figura 7.

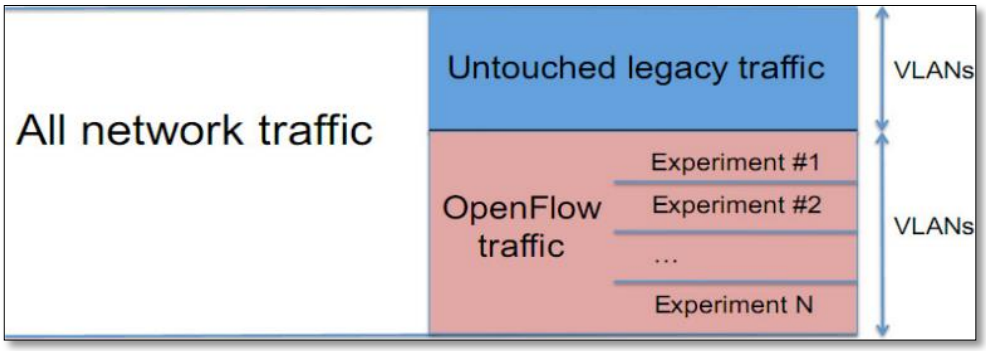

Figura 7: Modelo de operação híbrido para tráfegos legado e OpenFlow
Fonte: Rothenberg et al. (2010)

Dado que as redes definidas em *software* surgiram da necessidade do desenvolvimento e implementação de tráfegos e protocolos experimentais sem representar risco ao tráfego de produção, e observado que o protocolo OpenFlow permite a integração com redes IP, no capítulo 3 serão apresentadas as aplicações para essa nova tecnologia.

3 MERCADO E APLICAÇÕES

As redes definidas por *software* permitem uma estruturação de sistemas de rede de maneira flexível e centralizada, dessa forma podem ser amplamente utilizadas para o desenvolvimento de novas funcionalidades em um grande número de ambientes; dentre os quais alguns domínios e aplicações já foram identificados (GUEDES et al, 2012).

Conforme o IDC (International Data Corporation), empresa de consultoria na área de tecnologia da informação e telecomunicações, a tecnologia deve apresentar taxas de crescimento na ordem de 54% nos próximos anos, sendo que as categorias de *software*: virtualização e controle representarão juntos, investimentos na ordem de 5,9 bilhões de dólares (HARANAS, 2016).

A constante evolução da tecnologia e ampliação do *market share* já vem causando impacto nas decisões de compra, ou seja, clientes já têm exigido que na compra de novos equipamentos de redes os mesmos já sejam habilitados ou compatíveis com SDN. Este impacto vai atingir cada segmento do mercado, de maneira que em algum tempo nenhum cliente ou fornecedor seja imune às redes definidas por *software*.

O que não significa a extinção das redes de produção definidas em *hardware*, pois, as soluções de redes ainda irão precisar de roteadores e soluções de encaminhamento e classificação baseados na tecnologia TCAM. Entretanto, novas abordagens para o *hardware* de rede existente serão necessárias (SDXCentral, 2016).

Entre os vários cenários de rede possíveis em que a adoção da tecnologia OpenFlow traz aspectos promissores, destacam-se as seguintes aplicações:

3.1 CONTROLE DE ACESSO

Justamente pela sua natureza (das redes definidas por *software*), que possibilita a programação de fluxos, permitindo que regras de acesso sejam desenvolvidas com base em informações abrangentes e não apenas no que seria possível com o uso de um *firewall* em um enlace de rede, ou seja, regras baseadas

não apenas em protocolos e informações de destino e origem de determinado pacotes, mas no seu conjunto de forma simples (GUEDES et al, 2012).

3.2 GERÊNCIA DE REDES

A visão global da rede, permitida pelas SDNs, simplifica as ações de configuração e gerência de rede, enquanto contadores associados aos fluxos permitem o monitoramento de fluxos específicos conforme o interesse do administrador (GUEDES et al, 2012).

3.3 REDES DOMÉSTICAS

Uma forma de aplicar os princípios de SDN em redes domésticas consiste no uso de um roteador doméstico compatível com o protocolo OpenFlow, transferindo para o provedor de acesso a responsabilidade de controlar a rede através de um controlador SDN, dessa forma seria possível programar cada roteador de acordo com as regras de acesso apropriadas para cada usuário e ter uma visão global do tráfego que atravessa o enlace de acesso de cada residência.

Assim, pode-se identificar padrões denominados tráfego licito em contraste com padrões de tráfego que podem representar riscos a rede que, por exemplo, estejam utilizando códigos maliciosos (GUEDES et al, 2012).

3.4 DATA CENTER

Em ambientes onde aplicações de vários usuários precisam coexistir, uma demanda crescente deriva do isolamento de tráfego entre diversos usuários. Uma forma de obter esse isolamento com dispositivos de rede atuais seria através da configuração de VLANs individuais para cada cliente. Contudo, trata-se de um recurso

limitado pelo tamanho dos campos (do cabeçalho) usados para identificá-las, ficando inviável a medida que o número de usuários aumenta, tornando tarefas de gerência de rede bastantes complicadas (GUEDES et al, 2012).

Uma solução para este problema pode ser diretamente construída sob o conceito de um comutador virtual distribuído, que pode ser implementada com o uso de um controlador de rede aonde são inseridas automaticamente regras de encaminhamento de tráfego entre as portas de diversos *switches* virtualizados e que são atribuídos a cada usuário específico (GUEDES et al, 2012).

Outra aplicação de SDNs em *data center* consiste em soluções de conservação de energia; através da redução da taxa de transmissão ou o desligamento completo de dispositivos de rede em sistemas subutilizados e elementos ociosos, identificados facilmente graças à visualização global da rede.

Além disso, o controle de rotas e decisões de encaminhamento permite a implementação de pontos de controle capazes de interceptar pacotes de menor importância que atingiriam equipamentos que estão operando sob o regime de "*wake-on-lan*", evitando que estes sejam ativados desnecessariamente (GUEDES et al, 2012).

3.5 SD-WAN (*SOFTWARE DEFINED – WIDE AREA NETWORK*)

A arquitetura WAN (*Wide Area Network*) é baseada num sistema de "*hub-and-spoke"* ou seja, as corporações desenvolvem suas soluções de rede baseadas em conexões com o *data Center* mais próximo, que normalmente utiliza circuitos MPLS ou outra solução fornecida pela operadora de serviços (CISCO, 2016).

Esse sistema é utilizado há décadas e por muito tempo atendeu as necessidades corporativas fornecendo conexões ponto a ponto, entretanto atualmente nem todos os negócios estão concentrados em um ambiente físico conectado a um *data center*, existem muitas empresas virtuais estabelecidas na nuvem exigindo a mudança deste paradigma (CISCO, 2015).

No ano 2000, a maior parte do tráfego corporativo era destinado a *data centers*. Entretanto, atualmente muitas corporações permitem o acesso de seus colaboradores a rede mundial, normalmente com o intuito de permitir a visualização

de conteúdo organizacional como treinamentos e informações sobre os clientes; porém o acesso descompromissado relacionado aos assuntos não profissionais acontece e, nos casos em que o bloqueio não é uma opção, há a necessidade de balanceamento do tráfego que será encaminhado ao *data center* e a *Internet* (CISCO, 2015).

Utilizando a arquitetura convencional, o tráfego destinado à I*nternet* seria roteado do ambiente corporativo em direção ao *data center*, aonde seriam necessários um ou mais saltos "*hop*" entre roteadores até atingir a *Internet*. Isso adiciona latência e reduz a capacidade, além de aumentar os custos da rede, pois quanto mais saltos, maior a largura de banda MPLS utilizada e consequentemente maior o custo do serviço contratado. Contudo, se o tráfego fosse diretamente encaminhado à *Internet* haveria apenas um salto até a rede WAN a chamada "*the dotted line*" (CISCO, 2015).

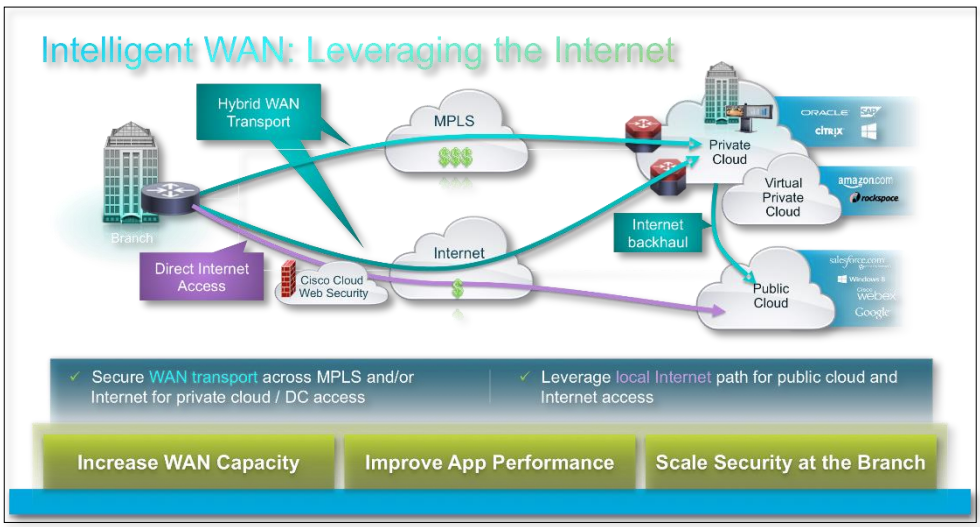

Figura 8: Topologia WAN Hibrida
Fonte: CISCO (2015)

A solução seria a implementação de uma rede híbrida, conforme mostrado na figura 8, que seria composta basicamente de duas conexões; uma convencional utilizando MPLS, ou outra tecnologia, para conectar a filial ao *data center*; que seria destinada a todo o tráfego corporativo; e outra, conectada diretamente a *Internet* permitiria além da conexão direta com rede global, a implementação de uma rede definida em *software* na nuvem.

Dessa forma, o problema do uso de banda MPLS para trafego *Internet* seria resolvido e como benefício colateral, haveria uma segunda rede em que poderia ser

implementada engenharia de tráfego de forma que o sistema pudesse definir o *link* com a menor latência (CISCO, 2015).

À vista disso, até o momento foram identificados os componentes de uma rede definida por *software*, apresentados os padrões e protocolos desenvolvidos, tal como suas aplicações. No capítulo subsequente será realizada a implementação de uma rede SDN com a finalidade de analisar o funcionamento destas redes e do protocolo OpenFlow.

4 IMPLEMENTAÇÃO

Neste capítulo detalham-se quais foram os procedimentos adotados para construção de um ambiente virtual composto por uma rede IP integrada a uma rede SDN em uma máquina física com a seguinte configuração:

- Processador Intel Core I7 2.40 GHz;
- 8GB de memória RAM;
- Disco rígido com velocidade de 5400 RPM e 500GB de capacidade;
- Sistema operacional Windows 10.

No equipamento onde os testes foram ambientados, já estavam instalados os aplicativos VirtualBox, para gerenciamento de máquinas virtuais e GNS3, um emulador amplamente utilizado que, como apontado por Oliveira et al. (2012), permite a reprodução fiel das características de diversos modelos de roteadores do fabricante Cisco Systems, possibilitando a criação de diversos cenários.

Como o princípio das redes definidas por *software* está na separação do plano de dados do plano de controle, este trabalho inicia a definição do ambiente de testes com a seleção do controlador compatível com o protocolo OpenFlow, apresentando os procedimentos para sua instalação e execução, conforme exposto na seção 4.1.

4.1 CONTROLADOR SDN

Baseados em trabalhos já desenvolvidos no segmento de SDN, aliado a portabilidade que permite a execução em sistemas operacionais Microsoft Windows e Linux sem a necessidade de configuração de um *host* virtualizado, atrelado a possiblidade de gerenciamento via interface *web*, optou-se pelo controlador OpenDaylight em sua versão Hydrogen 1.0.

A versão escolhida pode ser executada diretamente no sistema operacional sem a necessidade de uma máquina virtual, outro ponto favorável demonstra-se na interface web deste controlador, que permite ao administrador de rede se familiarizar com o controle de fluxos e visualizar a topologia SDN carregada (JACOB, 2015).

A instalação do ODL, conforme o guia de instalação disponível no site do desenvolvedor, exige atenção a certos pré-requisitos, ou seja: a instalação da máquina virtual Java e a definição de variáveis de ambiente (OPENDAYLIGHT.ORG, 2016).

Feito isso, é possível descarregar o pacote de instalação do ODL a partir da página do desenvolvedor na seção "*Release Archives*", na qual os arquivos estão dispostos na ordem de lançamento (OPENDAYLIGHT.ORG, 2016).

O arquivo obtido, encontrava-se compactado e pôde ser copiado e salvo em um diretório de fácil acesso, destinado às aplicações relacionadas a este trabalho. Em seguida, o arquivo foi descompactado em um diretório de nome atribuído OpenDaylight, e na raiz deste diretório existem dois arquivos que merecem atenção: "*run.bat*" e "*run.sh*", o primeiro pode ser diretamente executado no sistema operacional Windows e o segundo direcionado ao sistema operacional Linux.

Para o contexto aplicado foi utilizado o executável "*run.bat*", através do *prompt* de comando do Windows (modo Administrador), conforme ilustra a figura 9.

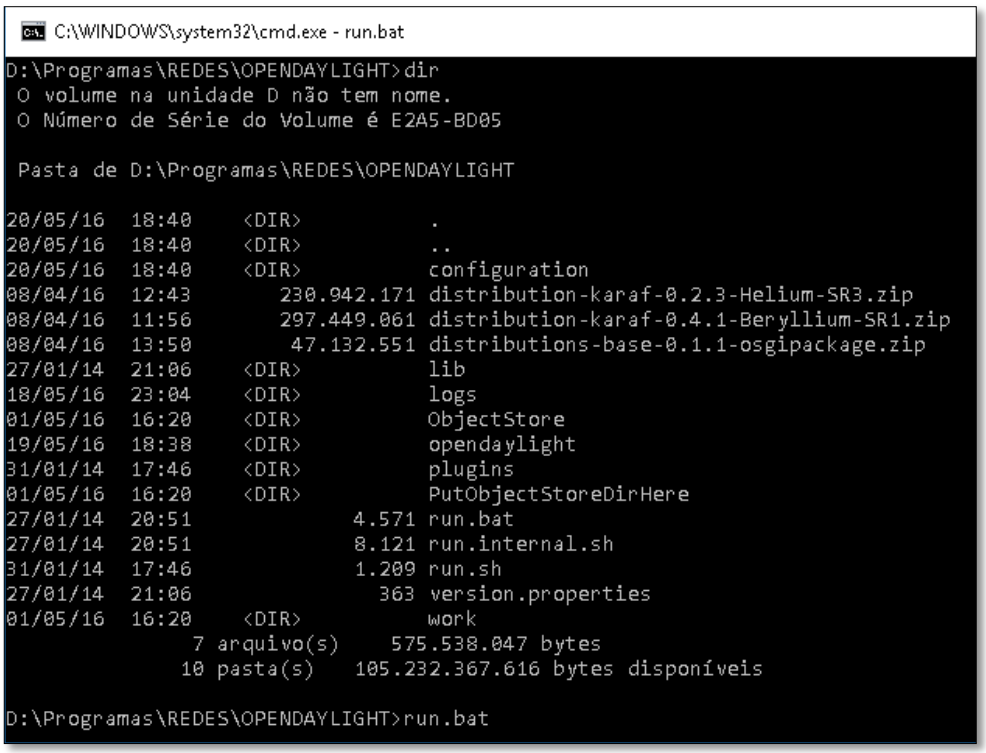

Figura 9: OpenDaylight – Diretório
Fonte: Autoria própria

Posteriormente, o carregamento do controlador ODL foi iniciado, e o processo de inicialização demorou por volta de 1 minuto para ser completado. Durante esse

processo foi observado o uso de pouco mais de 1Gb de memória, conforme mostra a figura 10.

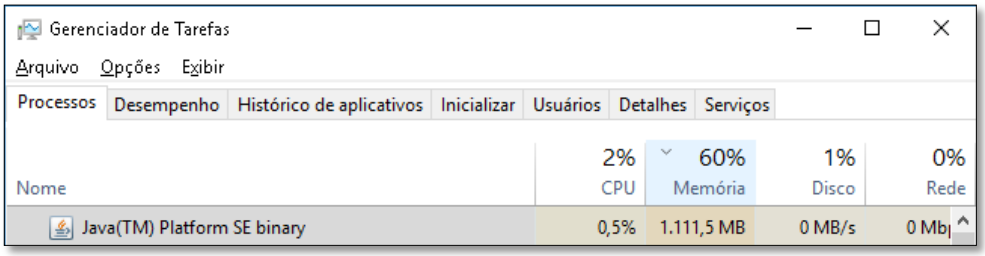

Figura 10: OpenDaylight – Uso de Memória
Fonte: Autoria própria

Em outros trabalhos, utilizando o ODL em máquinas virtuais, foi simulada a execução do controlador utilizando menos de 1GB de memória, sendo observado que a inicialização se dava de forma lenta e em alguns momentos a aplicação encerrava automaticamente, sendo necessário o reajuste das configurações do host virtualizado (JACOB, 2015).

Dessa forma, a execução da aplicação diretamente sob o sistema operacional nativo, em princípio, se mostrou mais fluída que em aplicações virtualizadas, entretanto ainda não há o carregamento de uma topologia gráfica, tampouco fluxos a serem controlados.

A inicialização completa pode ser identificada após a permissividade de inserção de comandos na plataforma, conforme exemplifica a figura 11.

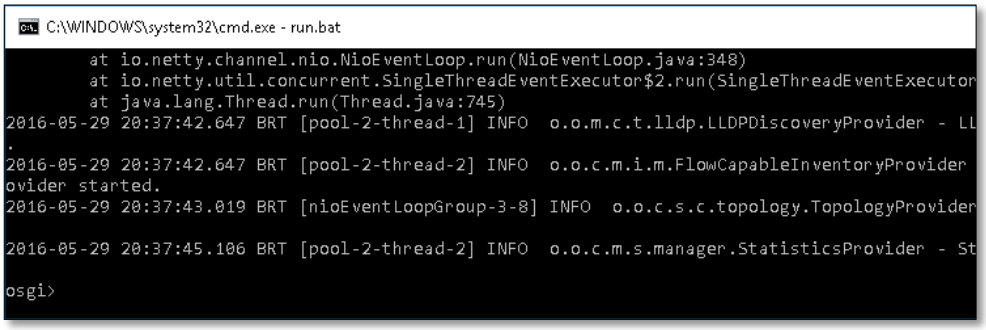

Figura 11: Carregamento do controlador OpenDaylight
Fonte: Autoria própria

A partir desta etapa é possível acessar a plataforma através de sua interface *web*, utilizando a porta 8080, após autenticação com usuário e senha *'admin'* se tem acesso a tela apresentada na figura 12 (JACOB, 2015).

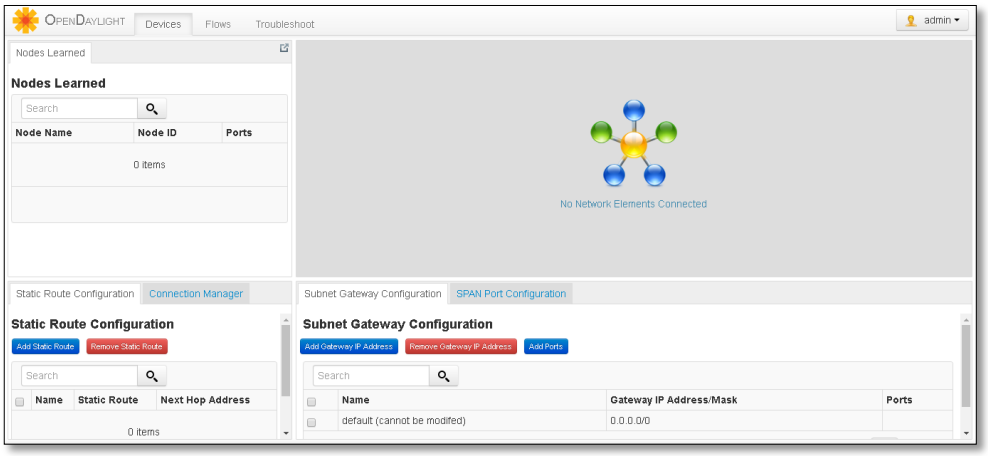

Figura 12: Tela inicial OpenDaylight – Interface Web
Fonte: Autoria própria

Nesta etapa ainda não há o reconhecimento de uma rede SDN operacional, pois ainda não existem switches OpenFlow ativos conectados ao controlador. Portanto é necessário a implementação de comutadores compatíveis com o protocolo OpenFlow, dentre as soluções disponíveis optou-se pela que provê uma rede SDN completa e que permite a virtualização de *switches*, *hosts*, e um controlador interno.

4.2 EMULADOR MININET

Foi escolhido o Mininet, um sistema que permite a rápida elaboração de grandes redes escaláveis utilizando um computador convencional baseando-se em princípios de virtualização do Sistema Operacional. Os elementos de rede emulados são executados como se fossem processos, permitindo o desenvolvimento e customização de laboratórios virtuais de Redes Definidas por *Software* de maneira rápida (GUEDES et al, 2012).

Ele possibilita a emulação de uma rede SDN completa, composta por *switches* OpenFlow (Open vSwitches), *hosts* e controladores. Além de contar com topologias, a aplicação também permite a definição de uma rede customizada utilizando a linguagem de programação Python, ou até mesmo via interface gráfica, utilizando o aplicativo "*miniedit*", sendo necessário para tanto direcionamento para um servidor X, já que o Mininet é carregado em modo texto, para reduzir o seu tamanho e uso de memória da máquina virtual (AMORIM, 2012).

Dentre as opções de instalação, disponíveis na página do desenvolvedor, foi escolhida uma máquina virtual contendo o sistema de pacotes de dependências do Mininet previamente instalada sob o sistema operacional Linux Ubuntu 14.04 (MININET.ORG, 2016). O desenvolvedor recomenda a utilização da versão i386 (32 bits) para usuários do sistema operacional Windows e do aplicativo de virtualização VirtualBox. Ambos se aplicam ao ambiente utilizado neste trabalho (MININET, 2016).

O arquivo descarregado corresponde, como já apresentado, a uma máquina virtual preparada para a aplicação Mininet, sendo necessário apenas descompactar o arquivo e importar as configurações utilizando o programa VirtualBox, conforme os passos abaixo.

Com o VirtualBox em execução, selecionando *Arquivo/Importar Appliance*, deverá ser localizado o arquivo de extensão "*.ovf*", e em seguida escolhida a opção abrir, conforme ilustrado nas figuras 13 e 14.

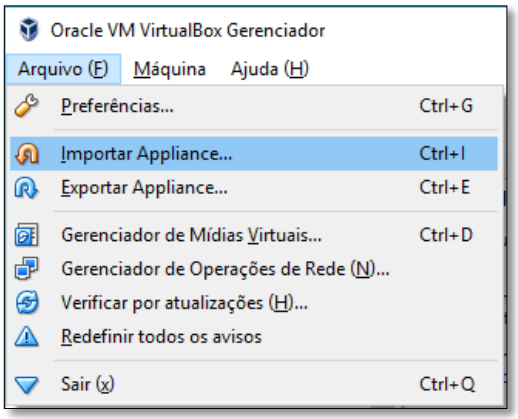

Figura 13: VirtualBox – Importar Appliance
Fonte: Autoria própria

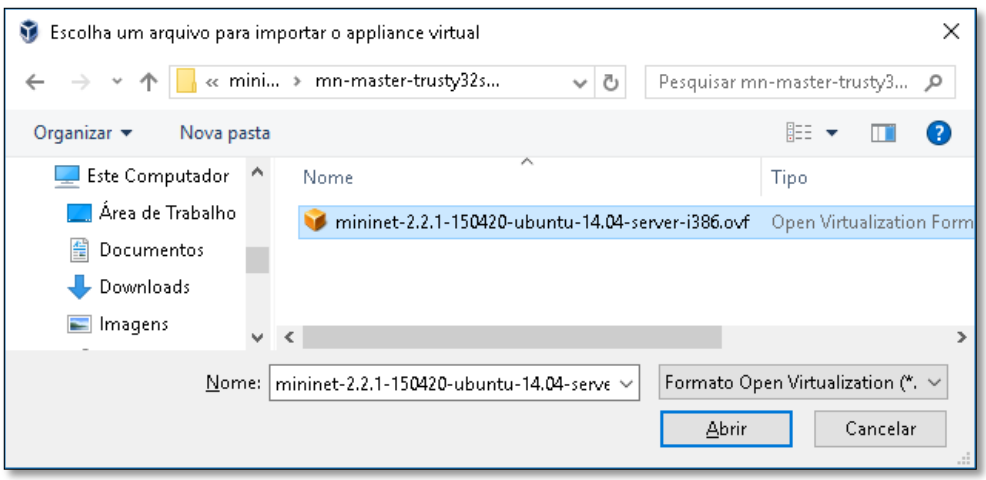

Figura 14: VirtualBox – Localizando arquivo .ovf
Fonte: Autoria própria

Em seguida, o assistente reporta as configurações sugeridas para a máquina virtual antes da importação, para os fins deste trabalho não foram alteradas, seguindo com a conclusão do processo com a seleção da opção “importar”.

Para a integração do Mininet com a topologia foram configuradas três interfaces na máquina virtual, a figura 15 mostra que na opção ‘Configurações’ no campo que corresponde a ‘Rede’ serão carregadas quatro abas correspondentes aos adaptadores de rede virtuais.

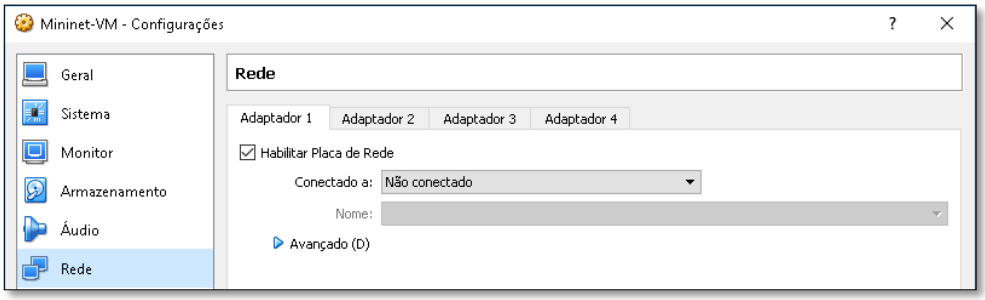

Figura 15: VirtualBox – Definição de Adaptadores de rede
Fonte: Autoria própria

De maneira que, para este trabalho, foram habilitados três adaptadores de rede sem definição de conexão, ou seja, à interface virtual não foi atribuído nenhum dispositivo de rede físico ou virtual da máquina hospedeira.

Por fim, com um duplo clique sobre o ícone “Mininet-VM”, a máquina virtual é carregada e em seguida já é possível acessá-la após a autenticação com usuário e senha: ‘mininet’, como demonstrado na figura 16.

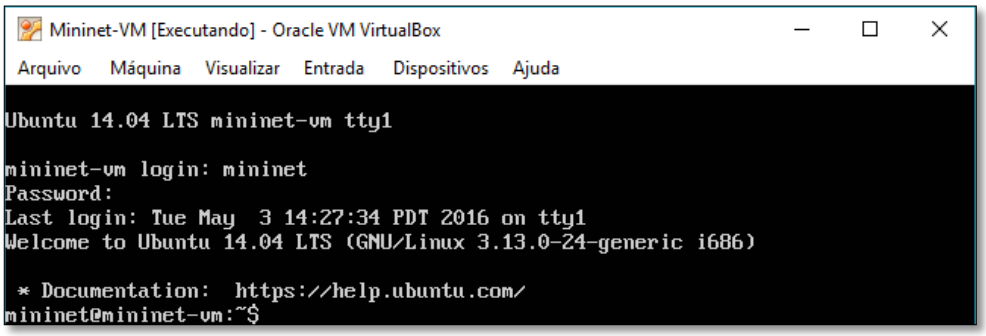

Figura 16: Mininet-VM – Inicial
Fonte: Autoria própria

Contudo, o acesso ao Mininet é feito por meio da aplicação VirtualBox e, como não foram integrados adaptadores e conexões de rede, ainda não é possível emular

uma SDN que possa se conectar ao controlador. Para tanto é necessário incluí-lo na topologia, conforme será melhor apresentado na seção 4.3 deste trabalho.

4.3 TOPOLOGIA

De maneira simplificada, a topologia híbrida de testes poderá ser observada na figura 17. Embora não reflita, em termos de dimensão e volume de tráfego, a capacidade real de uma rede híbrida, ela é suficiente para demonstração das características e funcionalidades de Switches OpenFlow e o papel do controlador em redes SDN.

Nela estão dispostos três roteadores comerciais, representando uma rede de produção IP; uma máquina virtual com a aplicação Mininet, responsável pela emulação da rede SDN e composta por um switch OpenFlow e duas instâncias de *host* virtualizadas; e um terceiro host virtual carregado diretamente no programa GNS3.

O controlador SDN OpenDaylight está instalado no sistema operacional do equipamento onde o ambiente está sendo simulado, e será acessado por meio de uma interface *Loopback*.

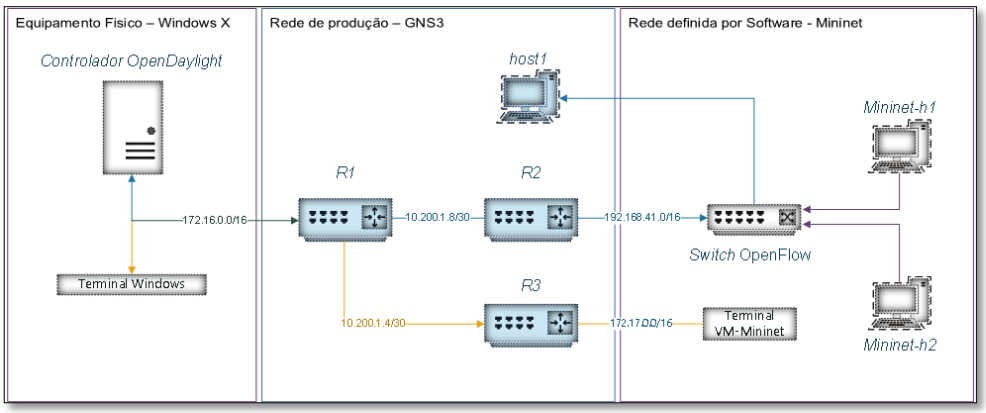

Figura 17: Topologia Hibrida IP/SDN
Fonte: Autoria própria

A máquina virtual com a aplicação Mininet foi configurada no GNS3 com três interfaces ethernet, denominadas eth0, eth1 e eth2; a primeira conectada ao roteador

R3 é dedicada aos tráfegos de gerência e controle, as demais foram integradas ao *switch* OpenFlow emulado e conectadas ao roteador R2 e ao *host*1, respectivamente.

O roteador R1 está conectado aos roteadores R2 e R3, bem como à interface *loopback* do host físico, ficando responsável pela interligação das redes de gerência, controle e produção.

O *host1,* foi carregado através de uma imagem QEMU[1] e configurado para obter o endereçamento IP via servidor DHCP[2] (*Dynamic Host Configuration Protocol*), sendo utilizados endereços reservados[3] na faixa 192.168.41.0/24. Ao roteador R1 foi atribuída a função de servidor DHCP, enquanto o roteador R2 foi configurado como gateway desta rede.

Importante destacar que, por definição, os roteadores não propagam *broadcast*, utilizado pelos hosts para identificar o servidor DHCP, assim é necessário "forçar" o roteador a encaminhar mensagens *broadcast*. Isso se dá, em roteadores Cisco, através do comando *ip helper*, configurado na interface local (FILIPPETTI, 2009). As configurações dos roteadores estão disponíveis no Apêndice A.

Elemento de Rede	Interface	Endereçamento/Rede
R1	Fastethernet 0/1	172.17.0.1/16
	Serial 1/1	10.200.1.5/30
	Serial 1/0	10.200.1.9/30
R2	Fastethernet 0/0	192.168.41.1/24
	Serial 1/0	10.200.1.10/30
R3	Fastethernet 0/0	172.17.0.1/16
	Serial 1/1	10.200.1.6/30
Host Físico	Loopback	172.16.0.2/16
Mininet	Eth0	172.17.0.2/16

Quadro 1 – Endereçamento IP dos elementos de rede
Fonte: Autoria própria

A escolha pelo endereçamento automático se deu pela flexibilidade de implementação de mais hosts no ambiente de testes; o endereçamento dos demais dispositivos de rede poderá ser verificado conforme a tabela 1.

[1] QEMU é um emulador desenvolvido em código aberto para virtualização de sistemas operacionais (QEMU, 2016).

[2] Protocolo utilizado para obter informações de configuração, incluindo endereçamento IP, a partir de um roteador ou de um servidor de domínio (COMER, 2015).

[3] A RFC (*Request for Comments*) 1918 determina um intervalo de endereços para uso interno, ou seja, não roteáveis na *Internet* (FILIPPETTI, 2009).

Uma vez construída a topologia e ativos todos os elementos de rede é possível iniciar a execução do ambiente de testes.

4.4 TESTES DE CONECTIVIDADE

O primeiro passo consiste em validar a conectividade do emulador Mininet com o sistema rodando o controlador – neste momento o controlador ainda não foi iniciado – via console é possível executar comandos para validação das configurações das interfaces do mesmo, através do comando *ifconfig*, ilustrado na figura 18.

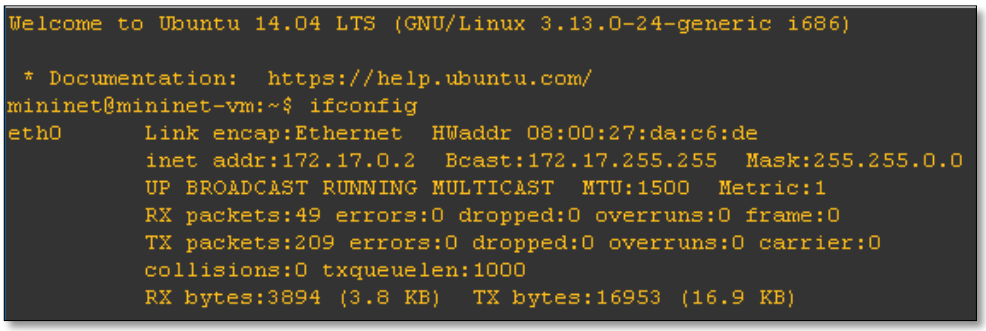

Figura 18: Mininet-VM – Verificando interfaces com "*ifconfig*".
Fonte: Autoria própria

Observa-se que as interfaces eth1 e eth2 ainda não estão listadas, isso se deve ao fato que elas ainda não foram iniciadas, o que pode ser resolvido executando os comandos:

```
sudo ifconfig eth1 up
sudo ifconfig eth2 up
```

Como verificado na tabela 1, o endereço de IP atribuído a interface *Loopback* do sistema e, consequentemente, ao controlador foi 172.16.0.2; para validação da conectividade da máquina virtual, em que será executado o Mininet, com o sistema será realizado um simples teste de *ping*, apresentado na figura 19.

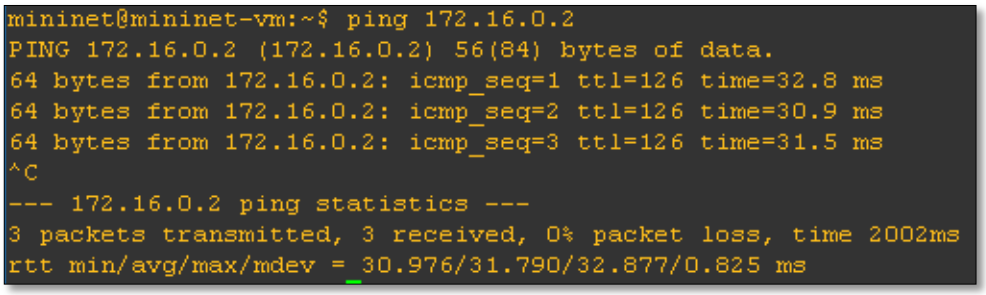

Figura 19: Mininet-VM – Teste de conectividade
Fonte: Autoria própria

Todavia é necessária a validação a partir do host físico, e o processo obedece ao mesmo princípio. Entretanto, o sistema operacional Windows por padrão define como rota *default* o *gateway* da interface de rede externa LAN (*Local Area Network*), quando ativa; neste caso todas as solicitações direcionadas ao IP 172.17.0.1 (Mininet-VM) eram direcionadas ao roteador da rede LAN e encaminhados a rede WAN, como pode ser visto na figura 20.

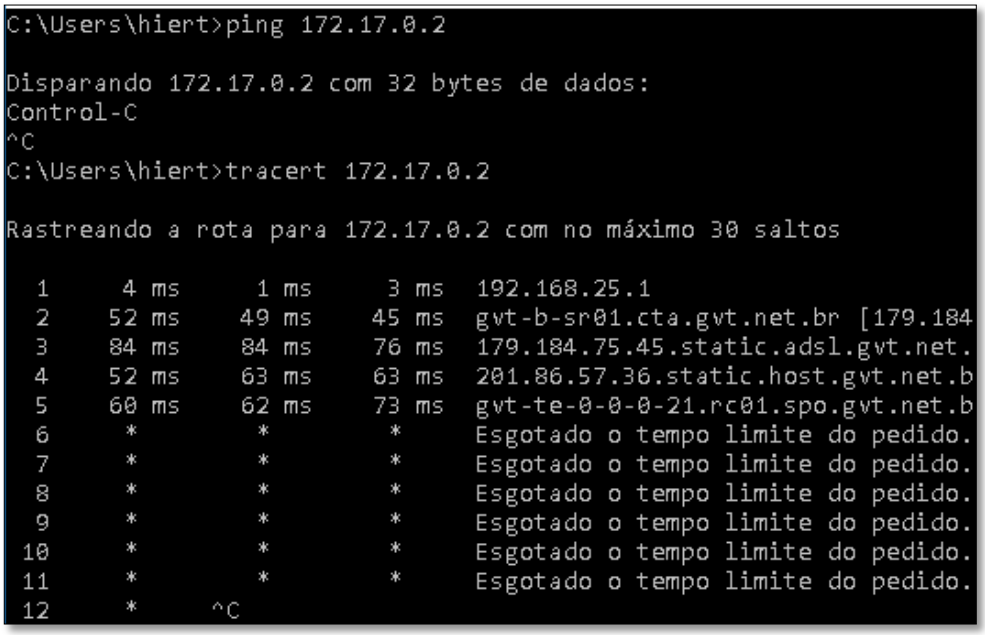

Figura 20: Host hospedeiro – Teste de conectividade
Fonte: Autoria própria

Dessa forma, necessita-se configurar o roteamento estático no sistema, designando o roteador R1 como *gateway* das redes do ambiente de teste. Na figura 21 poderão ser observados um exemplo desta configuração, bem como do teste de conectividade com o sistema Mininet.

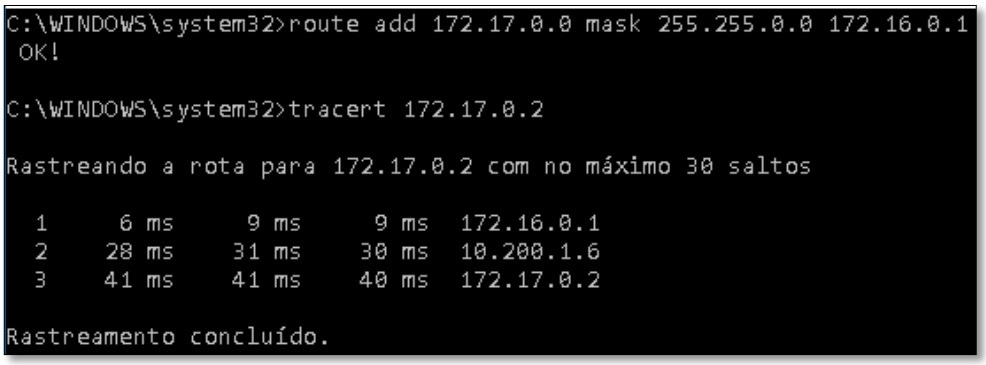

Figura 21: Host hospedeiro – Configuração de rota estática.
Fonte: Autoria própria

A partir deste ponto, todos os nós da rede estão conectados entre si e permitem a conexão com qualquer elemento do ambiente de testes, exceto o *host1*; isso porque ainda depende da atribuição de um IP pelo R1, e para que esta conexão seja estabelecida é necessário que a rede SDN esteja operacional.

4.5 EMULANDO SWITCHES OPENFLOW

Com a finalidade de demonstração do funcionamento básico de uma SDN, após a inicialização do controlador ODL, será simulada uma rede composta por um switch OpenFlow e dois *hosts*. Esta topologia pode ser carregada, no console da máquina virtual, através do comando:

```
sudo mn –controller=remote,ip=172.16.0.2
```

Em que "*sudo mn*" chama a aplicação e consequentemente o Mininet é iniciado, com base nos parâmetros inseridos, ou seja, a SDN será gerenciada por um controlador externo de IP 172.16.0.2, e uma vez que não foi definido o parâmetro referente a topologia, o sistema define uma topologia padrão, formada por um *switch* e dois *hosts*.

Durante o processo de carregamento, o Mininet apresenta em tela as etapas da criação da topologia, um exemplo pode ser visto na figura 22.

```
mininet@mininet-vm:~$ sudo mn --controller=remote,ip=172.16.0.2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Figura 22: Mininet – Etapas de carregamento.
Fonte: Autoria própria

Então já é possível observar, por meio da interface web do controlador, a exibição dos elementos de rede, neste caso o Switch OpenFlow.

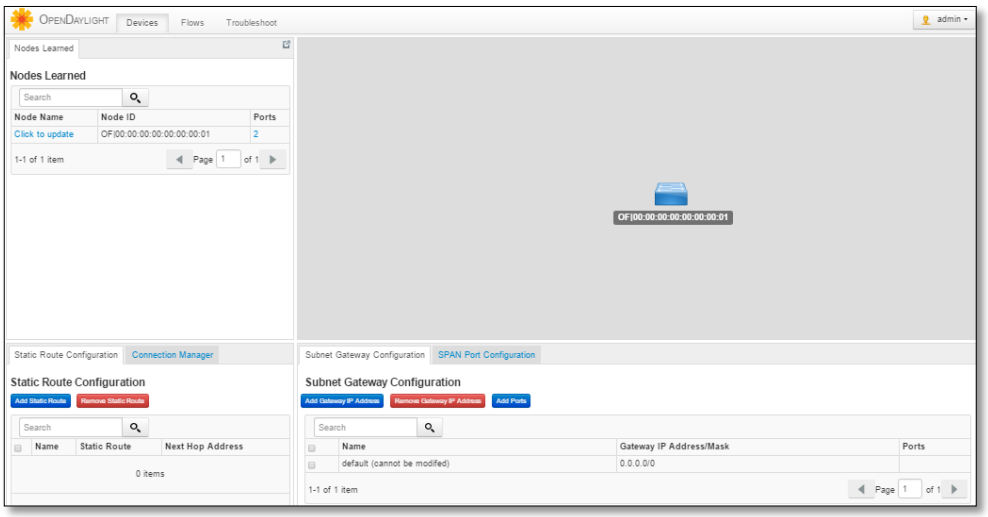

Figura 23: OpenDaylight – Visualização do OVS via Interface Web.
Fonte: Autoria própria

Ainda assim, não há configurações de fluxo atribuídas e o único elemento reconhecido neste ambiente é propriamente o *switch* emulado “s1”, identificado neste controlador pelo ID 00:00:00:00:00:00:00:01, que representa o seu endereço MAC (Media Access Control).

Para que os demais elementos de rede sejam interpretados é necessário que haja conectividade entre eles e, portanto, as interfaces do *switch* devem ser configuradas.

Como a máquina virtual não dispõe de ambiente gráfico (KDE, Gnome) para redução do seu tamanho, as configurações dos elementos da rede SDN poderiam ser realizados a partir de acesso SSH e redirecionamento do display gráfico a um servidor X remoto (AMORIM, 2012). Ou, como o emulador utiliza-se de instancias do OVS, instaladas no sistema operacional e rodam no espaço *Kernel* da máquina virtual, é possível, através de uma nova sessão SSH (*Secure Shell*), realizar configurações necessárias, sem a necessidade de instalação de servidores X.

Nesse contexto, o próximo passo consiste na definição das interfaces externas *eth1* e *eth2* da máquina virtual como interfaces do *switch* OpenFlow denominado “s1”; este processo é facilmente implementado através dos comandos:

```
sudo ovs-vsctl add-port s1 eth1
sudo ovs-vsctl add-port s1 eth2
```

Após esta ação, é possível validar as conexões e obter informações específicas através do comando “*sudo ovs-vsctl show”*, que estabelece comunicação com os *switches* que suportam *OpenFlow* (JACOB, 2015). O resultado deste comando pode observado abaixo.

```
mininet> s1  ovs-vsctl show
aaa4c93f-aa5a-4e41-b565-47c6f100c291
    Bridge "s1"
        Controller "tcp:172.16.0.2:6633"
            is_connected: true
        Controller "ptcp:6634"
        fail_mode: secure
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1"
            Interface "s1"
                type: internal
        Port "eth2"
            Interface "eth2"
        Port "eth1"
            Interface "eth1"
        Port "s1-eth2"
            Interface "s1-eth2"
    ovs_version: "2.0.2"
```

Exibe-se, neste caso, um único *switch* definido como “s1”, seguido do controlador e das interfaces internas e externas definidas. Ao final é possível ainda observar a versão do OVS utilizada (JACOB, 2015).

Conforme apontado por Amorim (2012), a maioria dos *switches* OpenFlow inicia com uma porta de escuta passiva na qual é possível interagir com o mesmo sem a necessidade de ação por meio do controlador, na configuração atual é designada a porta 6634.

Assim, é possível conectar-se ao switch e descarregar o estado e capacidade de suas portas, bem como os fluxos ativos, através do utilitário DPCTL, que pode ser chamado diretamente no console do Mininet, conforme a figura 24; ou em uma nova seção SSH, como mostra a figura 25.

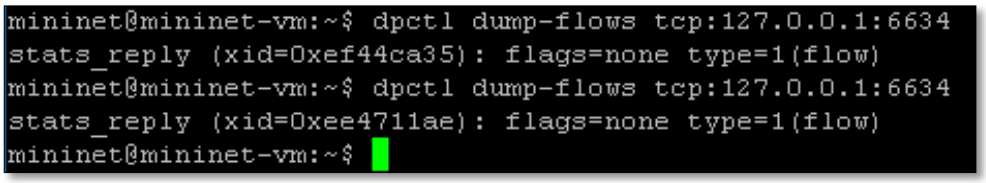

Figura 24: SSH – Tabela de Fluxo vazia.
Fonte: Autoria própria

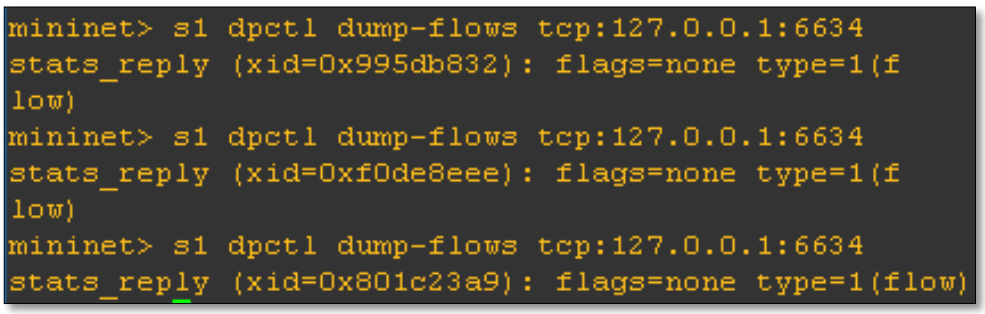

Figura 25: Console Mininet – Tabela de Fluxo vazia.
Fonte: Autoria própria

Ambos modos possuem a mesma função e irão apresentar a mesma resposta: descarregar a tabela de fluxo do switch OVS em execução no momento; a diferença reside no fato de que na aplicação Mininet deverá ser indicado o dispositivo ao qual o comando irá se aplicar, enquanto na sessão SSH o comando será executado diretamente no sistema operacional e irá retornar o fluxo dos OVS carregados.

Conforme observado, não há carregamento na tabela de fluxos do switch SDN. Também ainda não é possível conectar-se remotamente ao *host1*, dessa forma deve-se atribuir um fluxo a tabela do *switch*, e esta ação pode ser feita de duas maneiras: via utilitário DPCTL ou através do controlador OpenDaylight, apresentados nos itens 4.5.1 e 4.5.2.

4.5.1 Gerenciamento de Fluxos com DPCTL

A primeira, consiste no uso do utilitário DPCTL, configurando diretamente no *switch* o encaminhamento dos fluxos indicando a porta de entrada, a ação e a porta de saída.

No comando abaixo, é possível identificar os pontos já observados, como o uso do “dpctl” seguido de uma instrução, que será encaminhada a porta de escuta do *switch* com a regra: todo o tráfego recebido na porta 2 – definida a ação – deverá ser encaminhado para a porta de saída 1, e o inverso também se aplica.

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

Entretanto, o OVS por definição já conta com duas interfaces, que são utilizadas para a conexão com os *hosts* criados, automaticamente, na topologia SDN, com a adição das portas externas – da máquina virtual - ao OVS, é necessário atentar-se a configuração de fluxo nas portas 3 e 4 conforme abaixo:

```
sudo dpctl add-flow tcp:127.0.0.1:6634 in_port=4,actions=output:3
sudo dpctl add-flow tcp:127.0.0.1:6634 in_port=3,actions=output:4
```

Agora, executando o comando de consulta a tabela de fluxos já é possível observá-los, conforme ilustra a figura 26:

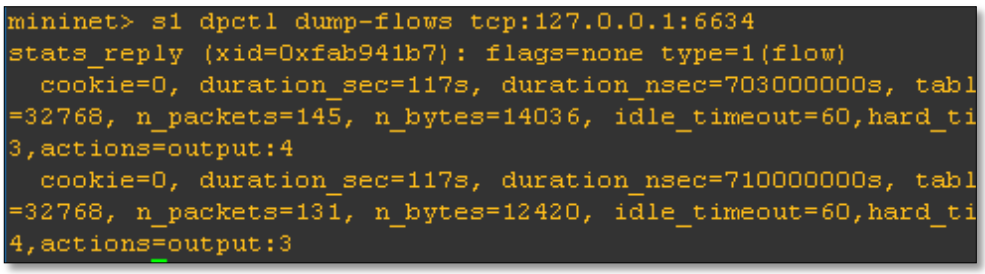

Figura 26: Tabela de fluxo do OVS.
Fonte: Autoria própria

Neste momento, também é possível verificar que o *host1* já está acessível através dos outros elementos de rede, bem como a partir dele poderão ser realizados testes de conectividade. Na figura 27 é possível verificar o endereço de IP atribuído

pelo servidor DHCP em R1, bem como o roteamento até o sistema local (interface loopback).

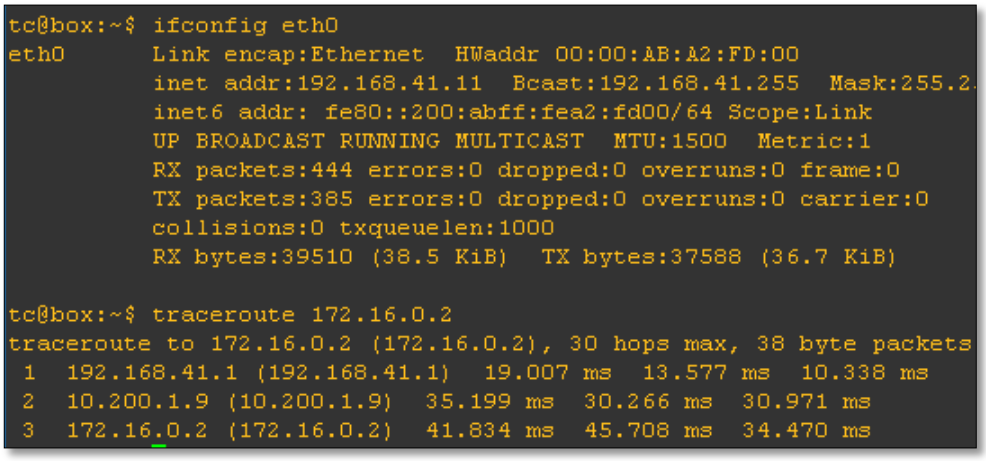

Figura 27: *Host1* – Teste de conectividade.
Fonte: Autoria própria

Também é possível através da interface web do controlador visualizar que o *host1* foi carregado na topologia, como o exemplo da figura 28.

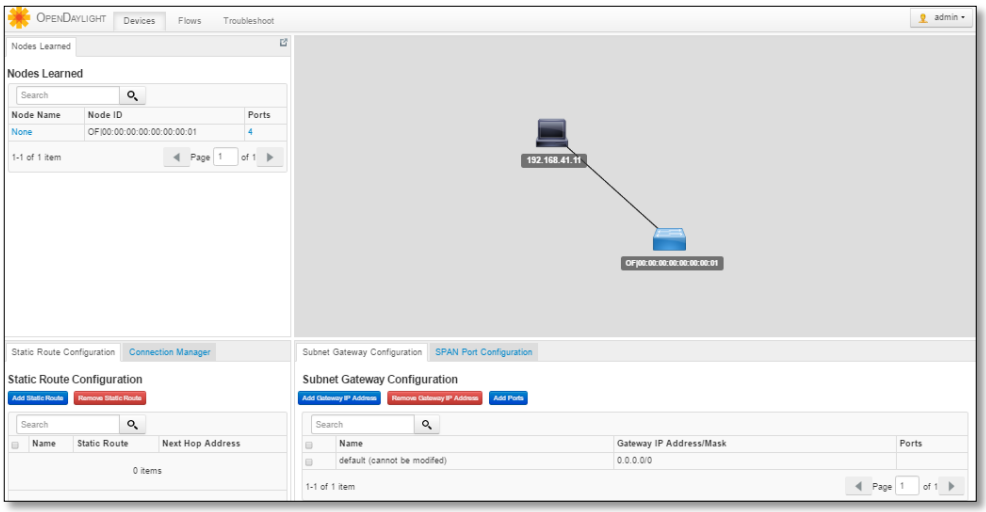

Figura 28: OpenDaylight – Carregamento da topologia SDN.
Fonte: Autoria própria

A segunda maneira de criar o fluxo entre as interfaces do *switch* OpenFlow é através do controlador ODN, mais precisamente pela sua interface web, conforme descreve Jacob (2015), ao acessar o controlador é possível identificar o menu "*Flows*", pelo qual podem ser incluídos e excluídos os fluxos.

4.5.2 Gerenciamento De Fluxos Via Interface Web

Para o fim específico deste trabalho foi configurada a regra de fluxo "*eth3_to_eth4"* que corresponde ao encaminhamento do fluxo de entrada na porta *eth3* com saída pela interface *eth4*, os passos para a configuração do fluxo podem ser observados a seguir:

No menu "*Flows"* do controlador, deve-se selecionar a opção *"Add Flow Entry"*; em seguida, o sistema ira iniciar uma janela na qual, após a atribuição do nome da regra, deve ser selecionado o nó de rede na caixa de seleção "*node*"; neste caso o ID 00:00:00:00:00:00:00:01, referente ao *switch* "s1".

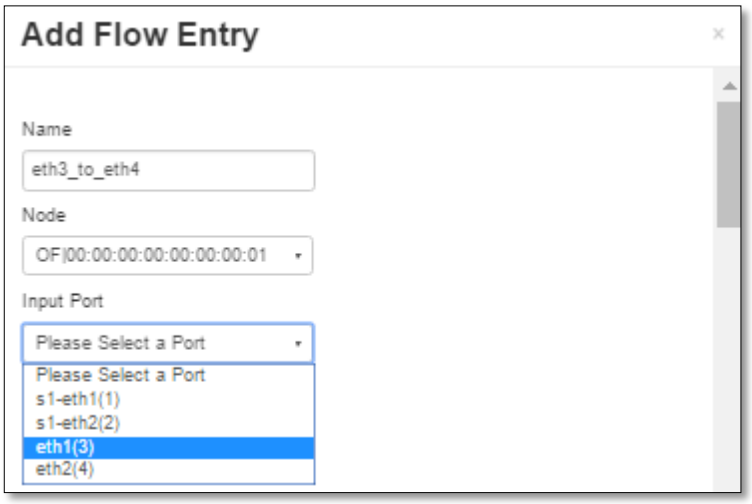

Figura 29: Regras de Fluxo, atribuição da porta de entrada.
Fonte: Autoria própria

No campo "Input Port" pode ser selecionada, dentre as opções, a porta de entrada do fluxo, conforme ilustrado na figura 29.

Como, neste caso, o objetivo é o encaminhamento do tráfego de uma porta de entrada para a outra de saída; o próximo passo consiste na atribuição da ação da regra: ao final da janela de configuração, no campo "*Actions*" é selecionada a opção "*Add Output Port"*, consequentemente o sistema irá reportar as interfaces de saída disponíveis, ilustra a figura 30.

Figura 30: Regras de Fluxo, atribuição da porta de saída.
Fonte: Autoria própria

Uma vez finalizada a configuração, a regra deverá ser instalada, através da opção "*Install"* para que seja colocada em funcionamento. Porém, para que a conexão com o *host1* possa ser estabelecida é necessário que seja criada a regra para o fluxo oposto, ou seja, entrada na porta eth4 com saída pela interface eth3, uma vez que os fluxos são unilaterais.

Um teste de conectividade foi executado utilizando o comando "*traceroute"* a partir do *host1* e apontado para o endereço da interface *Loopback – 172.16.0.2* - conforme apresentado na figura 31:

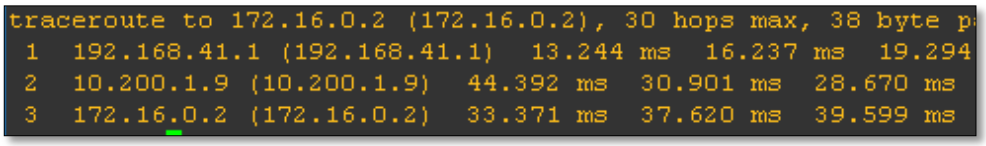

Figura 31: Teste de conectividade a entre *host1* e interface *Loopback.*
Fonte: Autoria própria

Abaixo, a figura 32 mostra a topologia final carregada pelo controlador e nela estão compreendidos os *hosts* criados pelo Mininet, o *host1* e o Open vSwitch (OVS) emulado.

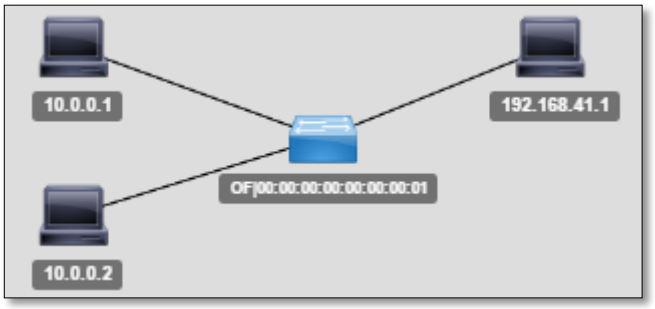

Figura 32: Carregamento de todos os hosts no ODL.
Fonte: Autoria própria

Com o intuito de validar que a rede SDN é de fato funcional e que a troca de informações entre o OVS e o ODL está sendo feita utilizando o protocolo OpenFlow, foi realizada a captura do tráfego entre o roteador R3 e a máquina virtual Mininet, mais detalhes sobre esta operação podem ser acompanhados na próxima seção.

4.6 ANALISANDO O PROTOCOLO OPENFLOW

O Wireshark analisador de redes que apresenta o conteúdo de pacotes capturados em um determinado *link* (WIRESHARK.ORG, 2016); o GNS3 possui uma extensão Wireshark que permite a captura do tráfego em uma determinada interface. Foi selecionado o link de interesse, entre Mininet-VM e R3, e escolhida a opção "*start capture*" o programa é automaticamente iniciado.

Para demonstrar as requisições do protocolo OpenFlow, a rede SDN foi interrompida e todas as configurações atribuídas ao programa OVS foram removidas utilizando o comando "*sudo mn –c*" no console da máquina virtual; foi executado um teste de *ping* da máquina virtual para a interface *Loopback*; e o tráfego capturado não implica em nenhuma solicitação do protocolo OpenFlow, conforme mostra a figura 33.

	Time	Source	Destination	Protocol	Length	Info
79	268.939041	ca:01:09:b4:00:08	CDP/VTP/DTP/PAgP/UD…	CDP	340	Device ID: R3 Port ID: FastEthernet0/0
80	269.987958	ca:01:09:b4:00:08	ca:01:09:b4:00:08	LOOP	60	Reply
81	279.973604	ca:01:09:b4:00:08	ca:01:09:b4:00:08	LOOP	60	Reply
82	289.976198	ca:01:09:b4:00:08	ca:01:09:b4:00:08	LOOP	60	Reply
83	292.612933	172.17.0.2	172.16.0.2	ICMP	98	Echo (ping) request id=0x05bc, seq=1/25
84	292.653338	172.16.0.2	172.17.0.2	ICMP	98	Echo (ping) reply id=0x05bc, seq=1/25
85	293.615436	172.17.0.2	172.16.0.2	ICMP	98	Echo (ping) request id=0x05bc, seq=2/51
86	293.651546	172.16.0.2	172.17.0.2	ICMP	98	Echo (ping) reply id=0x05bc, seq=2/51
87	294.616758	172.17.0.2	172.16.0.2	ICMP	98	Echo (ping) request id=0x05bc, seq=3/76
88	294.657788	172.16.0.2	172.17.0.2	ICMP	98	Echo (ping) reply id=0x05bc, seq=3/76
89	295.618703	172.17.0.2	172.16.0.2	ICMP	98	Echo (ping) request id=0x05bc, seq=4/10
90	295.656345	172.16.0.2	172.17.0.2	ICMP	98	Echo (ping) reply id=0x05bc, seq=4/10
91	296.621155	172.17.0.2	172.16.0.2	ICMP	98	Echo (ping) request id=0x05bc, seq=5/12
92	296.655918	172.16.0.2	172.17.0.2	ICMP	98	Echo (ping) reply id=0x05bc, seq=5/12
93	297.621844	172.17.0.2	172.16.0.2	ICMP	98	Echo (ping) request id=0x05bc, seq=6/15
94	297.664814	172.16.0.2	172.17.0.2	ICMP	98	Echo (ping) reply id=0x05bc, seq=6/15
95	298.624595	172.17.0.2	172.16.0.2	ICMP	98	Echo (ping) request id=0x05bc, seq=7/17
96	298.659708	172.16.0.2	172.17.0.2	ICMP	98	Echo (ping) reply id=0x05bc, seq=7/17
97	299.984364	ca:01:09:b4:00:08	ca:01:09:b4:00:08	LOOP	60	Reply

Figura 33: Captura de tráfego com SDN inativa.
Fonte: Autoria própria

Iniciada novamente a SDN Mininet, é possível observar na figura 34, a conexão do circuito virtual sendo formada pela análise das requisições do protocolo TCP; primeiro é enviado o segmento de sincronismo SYN (do termo em inglês *Synchronizing*), para sincronizar os lados do circuito TCP, seguido do segmento de

reconhecimento ACK (do termo em inglês *Acknowledgement*) do sincronismo SYN, ou simplesmente SYN, ACK (COMER, 2015).

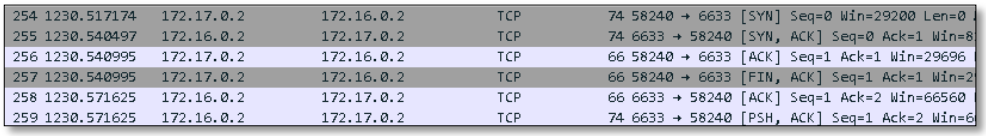

Figura 34: Requisições TCP entre ODL e Mininet
Fonte: Autoria própria

Então iniciarão as mensagens do protocolo OpenFlow, a primeira delas o *Hello*, que pode ser verificada na figura 35, que é trocada entre s*witch* e controlador durante o estabelecimento da conexão, para determinar a versão do protocolo OpenFlow suportada, em caso de esta negociação falhar será enviada a mensagem do tipo *HelloFailed* (FLOWGRAMMABLE.ORG).

Figura 35: Protocolo OpenFlow – Mensagem *Hello*
Fonte: Autoria própria

Quando um canal é estabelecido entre o *switch* e o controlador, a primeira atividade é a determinação das características do equipamento. O controlador irá enviar uma requisição "*FEATURE_REQUEST*". Então o *switch* reporta estas informações através de uma mensagem "*FEATURE_REPLY*" (FLOWGRAMMABLE.ORG).

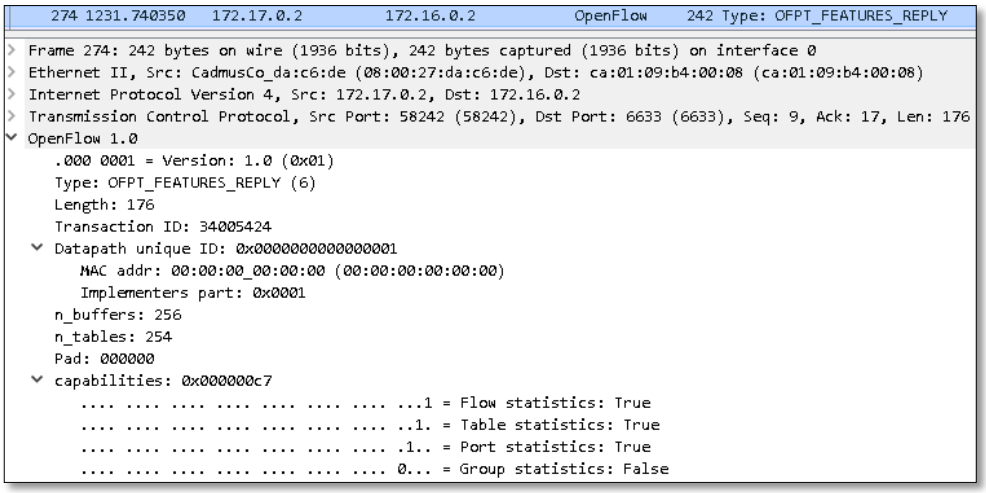

Figura 36: Protocolo OpenFlow – Mensagem "FEATURE_REPPLY"
Fonte: Autoria própria

Conforme ilustrado na figura 36, é possível observar um exemplo da estrutura da mensagem, com os campos relativos a versão, tipo, tamanho total e ID da requisição, além das informações específicas reportadas, neste caso as características do equipamento.

Já o controle dos fluxos ativos instalados é feito por meio requisição "*STATS_REQUEST*" encaminhada ao *switch* que retorna a mensagem "*STATS_REPLY*" contendo as informações dos fluxos ativos e, apesar de os pacotes, contendo essas mensagens, não terem sido "dissecados" pelo Wireshark, outro de grande importância que pode ser avaliado contem a mensagem "*FLOW_MOD*", que permite ao controlador alterar e criar um novo estado de fluxo (FLOWGRAMMABLE.ORG).

Para facilitar o entendimento foi reinstalado o fluxo Eth4_to_Eth3 já utilizado anteriormente e que consiste no redirecionamento do tráfego de entrada da porta Eth4 do "*s1* para a interface Eth3. Na figura 37 é possível observar o conteúdo do pacote contendo a mensagem de modificação - com as informações da configuração do fluxo - no campo "*in port*" está designada a porta 4, enquanto no campo "*Command*" consta a informação "*new flow"*, ou seja, a criação de um novo fluxo.

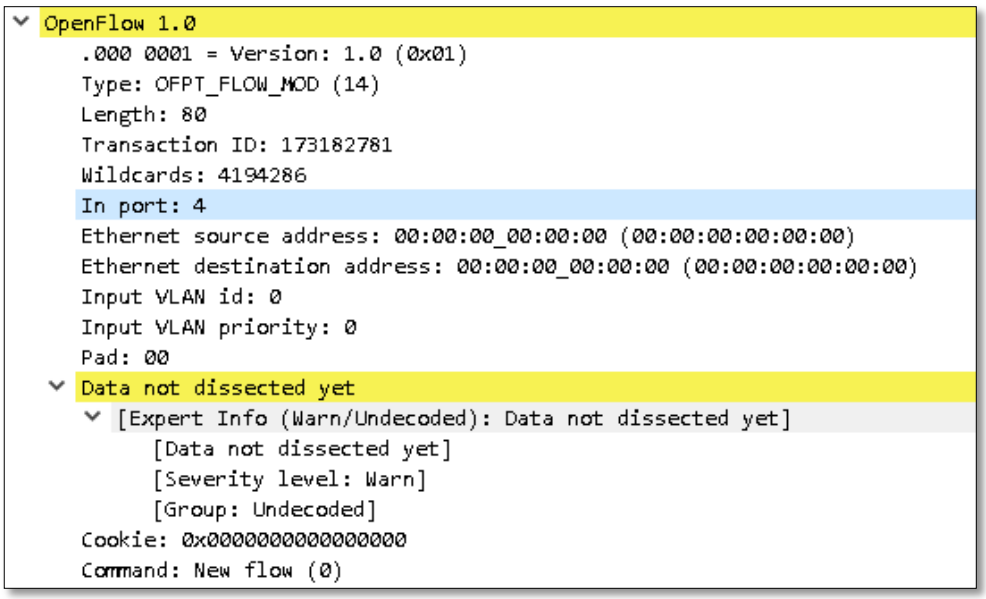

Figura 37: Protocolo OpenFlow – Mensagem *FLOW_MOD*
Fonte: Autoria própria

Após observados os pacotes do protocolo OpenFlow, foi iniciada a avaliação desempenho do circuito híbrido SDN/IP que para um maior detalhamento será apresentada na seção 4.7 desse trabalho.

4.7 TESTES DE DESEMPENHO

Conforme a pesquisa realizada por Amorim (2012), a máquina virtual do sistema Mininet já dispõe, em sua instalação, de duas opções de *switches*: um rodando em espaço usuário e outro, OVS, que é executado diretamente no espaço do *kernel* do sistema; testes comprovaram que o *switch* executado em modo *kernel* era 7,5 vezes mais rápido, devido os pacotes permanecerem o tempo todo alocado na memória do sistema, enquanto o fluxo passava pelo *switch*; ao passo que no modo usuário, cada pacote precisa cruzar o espaço de memória do usuário para o espaço kernel e voltar a cada salto ("*hop*").

Todas as atividades realizadas até o momento utilizaram um switch OVS e *hosts* externos para demonstrar a interoperabilidade de redes híbridas IP/SDN. Entretanto, para esta etapa de testes serão utilizados os *hosts* carregados pela topologia padrão do Mininet, assim como os fluxos criados serão removidos para evitar interferências da rede de produção nos resultados. Nesse contexto, primeiramente será avaliado funcionamento do comutador em modo usuário, em uma topologia composta por um *switch* e três *hosts* que será criada através do comando:

```
sudo mn --topo single,3 --mac --switch user --controller=remote,ip=172.16.0.2
```

Após o carregamento, é possível verificar através da interface *web* do ODL, a reprodução da topologia de testes, como apresenta a figura 38.

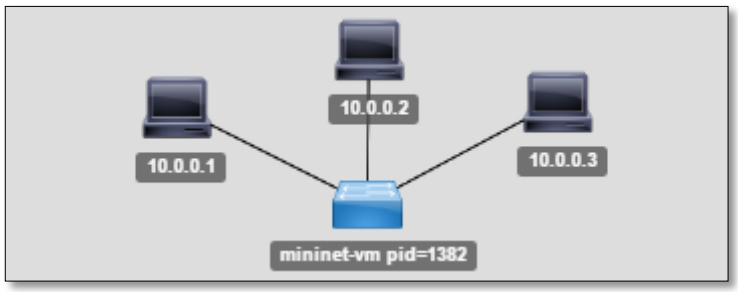

Figura 38: Topologia SDN para testes de desempenho.
Fonte: Autoria própria

Agora, no terminal do Mininet poderá ser iniciada a ferramenta “Iperf”[4], que atribui a um dos *hosts* a função de servidor de tráfego, enquanto outro assume o papel de cliente. Uma vez estabelecido o canal, são geradas rajadas de pacotes entre os *endpoins*. Um exemplo do comando para início da aplicação e o resultado aplicado a um switch emulado no modo usuário pode ser observado na figura 39.

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['1.19 Mbits/sec', '1.31 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['1.19 Mbits/sec', '1.31 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['1.19 Mbits/sec', '1.31 Mbits/sec']
```

Figura 39: Teste de desempenho em espaço usuário
Fonte: Autoria própria

Posteriormente, após a limpeza do cache das configurações do primeiro teste foi novamente carregada a topologia composta por três *hosts*, desta vez, utilizando o OVS (carregado diretamente no espaço *Kernel*), utilizando os comandos:

```
sudo mn -c
sudo mn --topo single,3 --mac --switch ovsk --controller=remote,ip=172.16.0.2
```

Na figura 40 é mostrado o resultado do teste com “Iperf”, que comprova os resultados apontados por Amorim (2012).

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['22.3 Gbits/sec', '22.3 Gbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['22.9 Gbits/sec', '23.0 Gbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['22.4 Gbits/sec', '22.4 Gbits/sec']
```

Figura 40: Teste de desempenho em espaço *kernel*
Fonte: Autoria própria

[4] O Iperf é um *software* livre, do tipo cliente servidor capaz de realizar medições de máxima largura de banda alcançável em redes IP (IPERF.FR, 2016)

Porém neste caso, os resultados do *switch* rodando em modo *Kernel* foram, em média, 22.400 vezes mais rápidos que os executados em modo usuário. A fim de determinar se a execução da máquina virtual sofria interferência do ambiente GNS3, os testes foram novamente realizados diretamente no aplicativo VirtualBox; os resultados não sofreram alterações.

Assim, pode-se afirmar que a perda de desempenho quando escolhido o "*switch user*" ocorre em decorrência da necessidade de interação constante entre o espaço usuário e o espaço kernel.

Os testes foram realizados em um *laptop* com processador Intel Core I7 5500 2,4ghz (apenas um núcleo foi alocado para a máquina virtual) e 8GB de RAM (1024MB alocados para a máquina virtual) usando o controlador OpenDaylight.

5 CONCLUSÃO

O desenvolvimento deste trabalho permitiu uma abordagem dos conceitos sobre a arquitetura das redes definidas por *software* e o protocolo OpenFlow, atrelado aos resultados dos testes realizados em ambiente virtual. Foi possível caracterizá-las como uma tecnologia que tem como fundamento a separação das camadas de controle e de dados nos elementos de rede.

Esta abordagem permite ao administrador da rede um maior controle sobre os fluxos nas tabelas de encaminhamento dos comutadores e maior versatilidade na aplicação de soluções de segurança e redundância dos serviços de rede.

O controlador OpenDaylight atendeu a proposta deste trabalho quanto a sua integração com o sistema operacional nativo da máquina física, sem a necessidade de virtualização e conseguinte alocação de memória e processamento necessários para a execução de todo o ambiente proposto. Sua interface gráfica *web* permitiu a visualização da topologia controlada, bem como as ferramentas de adição e controle de fluxo possibilitaram um maior entendimento a respeito do funcionamento do protocolo OpenFlow.

A composição da rede híbrida, integrando os emuladores GNS3 e Mininet se deu como esperado e oportunizou a validação dos conceitos aprendidos no que diz respeito a integração dos protocolos TCP/IP e OpenFlow, bem como possibilitou uma melhor percepção do comportamento e das configurações dos *switches* OVS.

As redes definidas por *softwares* têm aos poucos ganhado espaço no mercado e no meio acadêmico, o que pôde ser percebido durante o desenvolvimento desse trabalho, pois alguns dos referenciais utilizados foram recentemente publicados, o que é característica de uma tecnologia que está em constante evolução.

Como proposta para trabalhos futuros, propõe-se a aplicação de conceitos de engenharia de tráfego em redes definidas por *software,* através da implementação de um ambiente composto por *switches* OpenFlow e roteadores comerciais emulados; de forma a possibilitar a configuração de regras de balanceamento de tráfego e resiliência.

6 REFERÊNCIAS

AMORIM, Roberto José de. **O Uso Do Protocolo OpenFlow Em Redes Definidas Por *Software***. 2012. 41 f. Monografia (Especialização em Configuração e Gerenciamento de Servidores e Equipamentos de Redes) – Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná, Curitiba, 2012. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/1831/1/CT_GESER_II_2012_09.pdf>. Acesso em: 17 jul. 2015.

COMER Douglas E. **Interligação de Redes com TCP/IP**: Princípios, protocolos e arquiteturas. 6. Ed. Tradução: Tássia Fernanda Alvarenga. Rio de Janeiro: Elsevier, 1015. 399-412p.

FILIPPETTI, Marco Aurélio. **CCNA 4.1**: Guia Completo de Estudo. Florianópolis: Visual Books, 2008.

FLOWGRAMMABLE.ORG. **OpenFlow Messages**. 2016. Disponível em: <http://flowgrammable.org/sdn/openflow/message-layer>. Acesso em: 22 mai. 2016.

GUEDES, Dorgival O. et al. Redes Definidas por *Software*: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. In: MINICURSOS DO XXX SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, 30, 2012, Ouro Preto, **Livro Texto**, Ouro Preto: Editora Sociedade Brasileira de Computação, 2012. P. 161-207. Disponível em <http://homepages.dcc.ufmg.br/~mmvieira/cc/papers/minicurso-sdn.pdf>. Acesso em: 17 Jul. 2015

HARANAS, MARK. Top 10 SDN Market Leaders In The Data Center And Enterprise In 2016. **CRN**, 09 fev. 2016. Disponível em: <http://www.crn.com/slide-shows/networking/300079644/top-10-sdn-market-leaders-in-the-data-center-and-enterprise-in-2016.htm>. Acesso em 24 mai. 2016.

IETF, **Interface to The Internet Routing System**. 2015. Disponível em: <https://www.ietf.org/mailman/listinfo/i2rs>. Acesso em: 17 jul. 2015.

IPERF.FR. **iPerf - The network bandwidth measurement tool**. 2016. Disponível em <https://iperf.fr/>. Acesso em: 22 mai. 2016.

JACOB, Jefferson Wanderley. **Proposta de Alta Disponibilidade para Redes Definidas por *Software* através do Controlador OpenDaylight**. 2015. 45 f. Monografia (Especialização em Teleinformática e Redes de Computadores) – Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná, Curitiba, 2015.

JUNIPER NETWORKS, **Network Operating System Evolution Juniper Networks Junos OS:** Architectural Choices at the Forefront of Networking. Disponível em: < http://www.globalknowledge.nl/content/files/documents/White-Papers/Juniper-White-Paper-Network-Operating-System>. Acesso em: 17 jul. 2015.

MATTOS, D. M. F., Lopez, M. E. A., Ferraz, L. H. G. e Duarte, O. C M. B. (2015). **Controlador resiliente com distribuição eficiente para redes definidas por *software***. In: XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC ' 2015. Disponível em: <http://sbrc2015.ufes.br/wp-content/uploads/138820.1.pdf>. Acesso em: 17 jul. 2015.

MININET.ORG. **Get Started With Mininet.** 2015. Disponível em: <http://mininet.org/download/>. Acesso em: 18 mai. 2016.

_____. **Mininet Overview.** 2015. Disponível em: <http://mininet.org/overview/>. Acesso em: 18 mai. 2016.

NOBRE, Tito Sérgio Martins Pereira. **SDN em rede de transporte ópticas**. 2014. 107 f. Dissertação (Mestrado em Engenharia Informática) - Faculdade de Ciências, Universidade de Lisboa, 2014. Disponível em: <http://docs.di.fc.ul.pt/bitstream/10451/15936/1/ulfc112524_tm_Tito_Nobre.pdf>. Acesso em: 10 jul. 2015>. Acesso em: 10 jul. 2015

OLIVEIRA, José M.; LINS, Rafael D.; MENDONÇA, Roberto. **Redes MPLS**: Fundamentos e Aplicações. Rio de Janeiro: Brasport, 2012.

OPENDAYLIGHT.ORG. **Instructions:** Getting Started Guide. 2016. Disponível em: <https://www.opendaylight.org/introduction-getting-started-guide>. Acesso em: 18 mai. 2016.

_____. **OpenDaylight Release Archives**. 2016. Disponível em: <https://www.opendaylight.org/software/release-archives?page=1>. Acesso em: 22 mai. 2016.

_____. **OpenDaylight Performance**: A Practical, Empirical Guide. 2016. Disponível em : <https://www.opendaylight.org/resources/odl-performance>. Acesso em: 22 mai. 2016.

QEMU. **QEMU**: Opensource Processor Emulator. 2016. Disponível em: <http://wiki.qemu.org/Main_Page>. Acesso em: 22 mai. 2016.

RECHIA, Felipe Stall. **Estudo de Redes Definidas por *Software* e sua Implantação em Redes Corporativas**. 2014. 110 f. Monografia (Especialização em Teleinformática e Redes de Computadores) – Pró Reitoria de Pesquisa de Pós-Graduação, Universidade Tecnológica Federal do Paraná, Curitiba, 2014. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/3864/1/CT_TELEINFO_2013_1_04.pdf>. Acesso em: 18 mai. 2016.

ROTHENBERG, Christian Esteves et al. OpenFlow e redes definidas por *software*: um novo paradigma de controle e inovação em redes de pacotes. **Cad. CPqD Tecnologia**, Campinas, v. 7, n. 1, jul. 2010. Disponível em: <http://www.cpqd.com.br/cadernosdetecnologia/Vol7_N1_jul2010_jun2011/pdf/artigo6.pdf >. Acesso em: 17 jul. 2015.

SDXCENTRAL. SDN Market Size Report. **Sdxcentral**. Abr. 2013. Disponível em: < https://www.sdxcentral.com/reports/sdn-market-size-report-2013>. Acesso em 25 mai. 2016.

SILVA, Edna Lúcia da; MENEZES, Estela Muszkat. **Metodologia da Pesquisa e Elaboração de Dissertação**. 4. ed. Florianópolis: UFSC, 2005. 20-21 p.

WIRESHARK.ORG. **Wireshark User's Guide**: Chapter 1. Introduction. 2016. Disponível em: <https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs>. Acesso em 22 mai. 2016

APENDICE A – CONFIGURAÇÕES DOS ROTEADORES

São aqui apresentadas as configurações finais dos roteadores (R1, R2 e R3) da rede de produção construída no ambiente GNS3.

Ao roteador R1 foi atribuída a função de servidor DHCP da rede 192.168.41.0/24, onde está instalado o *host1*, para que os endereços sejam atribuídos corretamente, são configurados além da rede a faixa de IPs que não devem ser utilizados pelos *hosts*, *gateway* e o servidor DNS (*Domain Name Service*) preferencial.

Após a definição das interfaces e seus endereçamentos é necessário a definição do protocolo de roteamento, neste caso o OSPF (*Open Shortes Path First*), em que foi atribuído o número de processo 41.Para que o endereço de rede da conexão com a interface *Loopback* fosse roteado pela rede, foi necessário executar o comando "*redistribute connected*", responsável por redistribuir a rede diretamente conectadas ao roteador R1 (FILIPPETTI, 2008). Abaixo podem ser consultadas todas as configurações atribuídas ao roteador R1.

```
hostname R1

no ip dhcp use vrf connected
ip dhcp excluded-address 192.168.41.1 192.168.41.10

ip dhcp pool HOSTS
   network 192.168.41.0 255.255.255.0
   default-router 192.168.41.1
   dns-server 8.8.8.8

interface FastEthernet0/1
 description Conn_to_Loopback
 ip address 172.16.0.1 255.255.0.0
 duplex auto
 speed auto

interface Serial1/0
 description Conn_to_R2-s1/0
 ip address 10.200.1.9 255.255.255.252
 serial restart-delay 0
 clock rate 128000

interface Serial1/1
 description Conn_to_R3-s1/1
 ip address 10.200.1.5 255.255.255.252
 serial restart-delay 0
 clock rate 56000

router ospf 41
 log-adjacency-changes
```

```
 redistribute connected
 redistribute static
 network 10.200.1.4 0.0.0.3 area 0
 network 10.200.1.8 0.0.0.3 area 0
```

Em seguida foi configurado o roteador R2, responsável pela conexão com o *switch* OpenFlow e redistribuição a atribuição do endereçamento via DHCP ao *host1* adjacente ao *switch.* Entretanto, como abordado na seção 4.2 os roteadores não propagam *broadcast*, utilizado pelos hosts para identificar o servidor DHCP, dessa forma é necessário "forçar" o roteador a encaminhar mensagens *broadcast*, isso se dá, em roteadores Cisco, através do comando "*ip helper"* visível nas configurações da interface *FastEthernet0/0* nas configurações abaixo (FILIPPETTI, 2008).

```
hostname R2

interface FastEthernet0/0
 description Conn_to_SDN_s1
 ip address 192.168.41.1 255.255.255.0
 ip helper-address 10.200.1.9
 duplex auto
 speed auto

interface Serial1/0
 description Conn_to_R1-S1/0
 ip address 10.200.1.10 255.255.255.0
 serial restart-delay 0
router ospf 41
 log-adjacency-changes
 redistribute connected
 redistribute static
 network 10.200.1.8 0.0.0.3 area 0
 network 192.168.41.0 0.0.0.255 area 0
```

Por fim, foi configurado o roteador R3, responsável pela conexão e roteamento da rede de gerência e de controle. Ele também foi configurado como servidor DHCP do emulador Mininet, devido nos primeiros testes a máquina virtual perder as suas configurações de IP a cada vez que era reiniciada; mais tarde identificou-se que se tratava de a mesma (máquina virtual carregada no GNS3) não estar vinculada à máquina virtual salva no equipamento físico. A configurações atribuídas neste trabalho ao roteador R3 podem ser visualizadas a seguir:

```
hostname R3

no ip dhcp use vrf connected
ip dhcp excluded-address 172.17.0.1
```

```
ip dhcp pool SDN-MGMT
   network 172.17.0.0 255.255.0.0
   default-router 172.17.0.1
   dns-server 8.8.8.8

interface FastEthernet0/0
 description Conn_to_Mininet-VM
 ip address 172.17.0.1 255.255.0.0
 duplex auto
 speed auto

interface Serial1/1
 description Conn_to_R1-s1/1
 ip address 10.200.1.6 255.255.255.252
 serial restart-delay 0

router ospf 41
 log-adjacency-changes
 redistribute connected
 redistribute static
 network 10.200.1.4 0.0.0.3 area 0
```