UNIFACS UNIVERSIDADE SALVADOR
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO
MESTRADO ACADÊMICO EM SISTEMAS E COMPUTAÇÃO

RAMON DOS REIS FONTES

AUTORIA DE CENÁRIOS OPENFLOW UTILIZANDO VISUAL NETWORK DESCRIPTION (VERSÃO SDN)

Salvador
2013

RAMON DOS REIS FONTES

AUTORIA DE CENÁRIOS OPENFLOW UTILIZANDO VISUAL NETWORK DESCRIPTION (VERSÃO SDN)

Dissertação apresentada ao Curso de Mestrado Acadêmico em Sistemas e Computação, UNIFACS Universidade Salvador, Universidade Salvador – Laureate International Universities como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. Paulo Sampaio.

Salvador
2013

FICHA CATALOGRÁFICA
Elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate Internacional Universities

Fontes, Ramon dos Reis

Autoria de cenários openflow utilizando visual network description (versão SDN)./ Ramon dos Reis Fontes. – Salvador, 2013.

144 p.: il.

Dissertação apresentada ao Curso de Mestrado em Sistemas e Computação, UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. Paulo Sampaio.

1. Redes de computadores. 2. OpenFlow. 3. Framework NSDL. 4. Network. I. Sampaio, Paulo, orient. II. Título.

CDD.004.6

RAMON DOS REIS FONTES

AUTORIA DE CENÁRIOS OPENFLOW UTILIZANDO VISUAL NETWORK DESCRIPTION (VERSÃO SDN)

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, UNIFACS Universidade Salvador, Laureate International Universities, pela seguinte banca examinadora:

Paulo Nazareno Maia Sampaio – Orientador _____
Doutor em Informatique et Telecommunications pela Université Toulouse III Paul Sabatier, UPS, França.
UNIFACS Universidade Salvador, Laureate International Universities

Gustavo Bittencourt Figueiredo _____
Doutor em Ciência da Computação pela Universidade Estadual de Campinas, UNICAMP, Brasil.
Universidade Federal da Bahia - UFBA

Glauco de Figueiredo Carneiro
Doutor em Ciência da Computação - Universidade Federal da Bahia – UFBA/ UNIFACS Universidade Salvador - Brasil.
UNIFACS Universidade Salvador, Laureate International Universities

Salvador, 19 de dezembro de 2013.

RESUMO

A demanda pelo envio, consulta e recepção de informações, tais como texto, imagens, áudio ou vídeo, vem crescendo de forma exponencial e como a princípio a Internet não foi concebida para atender a esta demanda, problemas acerca da vulnerabilidade, instabilidade, escalabilidade e incompatibilidades são cada vez mais evidentes. É inegável o seu sucesso, mas a Internet já apresenta sinais de decadência e por isso é necessário realizar mudanças em sua arquitetura, e as redes experimentais surgem como importante solução para a simulação de novas propostas. Neste contexto, foi proposto o conceito de Redes Definidas por Software, e em particular o protocolo OpenFlow, onde os softwares são responsáveis por ditar o comportamento da rede. Quando associadas às redes experimentais, as ferramentas de gestão de redes são importantes de forma possibilitar uma visão detalhada de uma rede, permitindo assim a sua otimização. Contudo, diferentes ferramentas podem ser utilizadas, como ferramentas responsáveis pela construção da topologia, monitoração, simulação e análise. No entanto, no geral essas ferramentas não são interoperáveis. Como solução a esse problema, é proposta a utilização da Framework NSDL concebida para proporcionar a interoperabilidade entre ferramentas heterogêneas, permitindo assim a comunicação entre diferentes ferramentas de gestão de redes. Ainda assim, o principal objetivo deste trabalho é a extensão da ferramenta gráfica de autoria de redes chamada Visual Network Description, inicialmente proposta para a autoria de cenários de rede tradicionais e sem fio, de forma a dar também suporte à autoria de cenários de Redes Definidas por Software. Como principal resultado, foi concebida a ferramenta gráfica Visual Network Description – VND (versão SDN). Além da autoria de cenários OpenFlow, o VND (versão SDN) também permite a geração automática do descrições NSDL e de scripts que viabilizam a sua posterior integração com outras ferramentas de simulação que suportem OpenFlow. De forma a ilustrar a autoria utilizando VND (versão SDN) e execução de cenários OpenFlow, foram realizados quatro estudos de caso. Uma das principais vantagens dessa ferramenta, além sua simplicidade, é a sua flexibilidade na customização de objetos e parâmetros que podem ser manipulados durante a criação de cenários de redes.

Palavras-chave: Redes Definidas por Software. OpenFlow. Framework NSDL. Visual Network Description.

ABSTRACT

The demand for sending, querying and receiving information, e.g. text, images, audio or video, is growing exponentially and since Internet was not designed to meet this demand, issues such as vulnerability, instability, scalability and incompatibilities are more evident as well. The success of Internet is undeniable, however some of its limitations are being unveiled, thus it is important to reconsider its architecture and main protocols. For this reason, experimental networks arise as a solution for the simulation of new proposals. In this context, the concept of Software Defined Networks was conceived, and in particular the OpenFlow protocol, where software is responsible for steering the behavior of the network. When applied to experimental networks, network management tools are also useful since they provide a detailed view of a network, thus allowing its optimization. Therefore, different tools can be applied, such as tools responsible for building the topology, monitoring, simulation and analysis. Nevertheless, these tools in general lack interoperability. As a solution to this problem, the utilization of the NSDL Framework is proposed for providing the interoperability among heterogeneous tools, allowing thus the communication between different network management tools. Nevertheless the main goal of this work is the extension of the graphical authoring tool so-called Visual Description Network, originally proposed for the authoring of traditional and wireless network scenarios, in order to also support the authoring of software defined networks scenarios. As a main result, the graphical tool Visual Network Description - VND (SDN version) was conceived. Besides the authoring of OpenFlow scenarios, the VND (SDN version) also allows the automatic generation of NSDL descriptions and scripts in order to enable their subsequent integration with other OpenFlow compliant tools. To illustrate the authoring with VND and execution of OpenFlow scenarios, four case studies were conducted. One of the main advantages of this tool, besides its simplicity, is its flexibility for the customization of objects and parameters that can be manipulated during the creation of network scenarios.

Keywords: Software Defined Networks. OpenFlow. NSDL Framework. Visual Network Description.

AGRADECIMENTOS

Devido ao ano intenso em estudos, pesquisas, vida profissional e pessoal nesses últimos dois anos, eu preciso deixar registrado os meus agradecimentos:

Ao pai celeste, nosso Deus, pela saúde, força, capacidade de entendimento e livramentos.

Ao meu orientador, o professor Dr. Paulo Sampaio, pela paciência, compromisso impecável, responsabilidade, compartilhamento de sabedoria e amizade. O êxito deste trabalho também é fruto do seu esforço. Sou eternamente grato pelas experiências que adquiri nesse período em que estive sob sua orientação.

Aos meus colegas de turma, em especial ao Rui e ao Lucas. Passamos um ano difícil e a convivência que tivemos nesse período foi fundamental para atenuar a dificuldade que todos nós passamos. Agora, na distância, ficou a amizade. Uma amizade de vida longa. Acho que agora posso dedicar um pouco de atenção para mim mesmo e você, Rui, passará longe do meu "Personal Best".

E ao meu pai, minha mãe e minha noiva, agradeço pelas orações e pela compreensão nos meus momentos de isolamento em prol do fruto deste trabalho.

LISTAS DE FIGURAS

LISTAS DE QUADROS

LISTAS DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A demanda pelo envio, consulta e recepção de informações (tais como textos, imagens, áudio ou vídeo) em redes e comunicações de dados vem crescendo de forma exponencial e como a princípio a Internet não foi concebida para atender a esta demanda, problemas acerca da vulnerabilidade, instabilidade, escalabilidade e incompatibilidades são cada vez mais evidentes. Inicialmente criada para atender a uma pequena demanda, a Internet é atualmente utilizada por centenas de milhões de pessoas e a sua arquitetura já apresenta indícios de limitações.

No cenário atual, a maior parte das novas ideias da comunidade de pesquisa em redes de computadores, incluindo a própria Internet, não é testada, levando à necessidade de simular novas soluções para que se tenha uma medida mais precisa de suas implementações em situações reais. Nesse contexto, é importante a concepção das ferramentas de autoria e gestão de redes e também as redes definidas por software ou simplesmente *SDN (Software Defined Networking)*.

As ferramentas de autoria e gestão de redes são importantes, pois permitem um melhor conhecimento do funcionamento de uma rede. Diferentes ferramentas existentes podem ser utilizadas para essa finalidade, tais como ferramentas de monitorização, criação de topologias, simulação e análise de redes. O desenvolvimento de uma ferramenta para a gestão de redes exige diferentes competências e conhecimentos, tais como as redes de computadores e os seus componentes e tecnologias, a usabilidade e a sua programação. Já as redes definidas por software são fundamentais no contexto das redes experimentais, pois permitem a separação do plano de dados do plano de controle. Em vez de concentrar as operações no hardware, o controle da rede pode ser realizado através de recursos de software presente em um dispositivo chamado controlador, que por sua vez, é independente aos protocolos implementados por fabricantes de equipamentos

1.1 PROBLEMÁTICA

Para o desenvolvimento deste trabalho, foram consideradas as seguintes problemáticas: a) Falta de interoperabilidade entre as ferramentas de gerenciamento de

redes heterogêneas, onde cada ferramenta possui arquiteturas e características peculiares e não interagem entre si; b) Complexidade na configuração e gerenciamento de redes OpenFlow, pois ainda são necessárias muitas etapas para a configuração dessas redes, se comparada às redes tradicionais e; c) Complexidade na aprendizagem acerca das redes definidas por software devido a necessidade de domínio em redes de computadores e linguagens de programação e características peculiares às redes definidas por software.

1.2 MOTIVAÇÃO

A implementação de planos de controles independentes aos protocolos implementados por fabricantes de equipamentos possibilitam a definição de uma importante solução para as redes definidas por software: o protocolo OpenFlow (MCKEOWN et al., 2008). O OpenFlow consiste em um protocolo de comunicação que implementa conceitos de redes definidas por software, abstraindo os recursos de hardware de cada dispositivo intermediário, sejam eles comutadores ou encaminhadores.

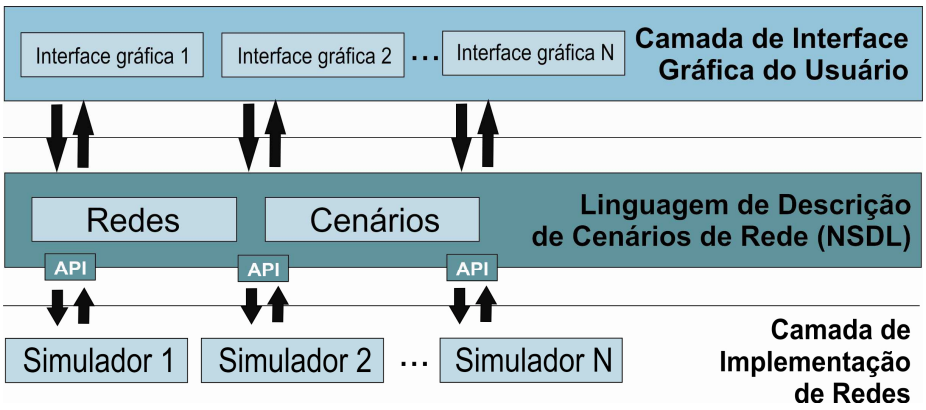
As propostas das redes definidas por software e do protocolo OpenFlow são importantes para o desenvolvimento de novas soluções voltadas para redes de computadores e aprimoramento das soluções já existentes. No entanto, duas principais limitações são identificadas neste estudo:

- As ferramentas OpenFlow (emuladores e controladores) existentes são diversificadas, heterogêneas, a maioria em código aberto, com diferentes números de funcionalidades. A variedade de ferramentas existentes valida a relevância da solução proposta e o interesse da comunidade científica, além de contribuir para a pesquisa e estudo de novas soluções, e;
- A criação e configuração de experimentos OpenFlow exige do experimentador um conhecimento diversificado, como trabalhar com linguagens de programação, compreender o funcionamento dos protocolos de redes e conhecer sistemas *Unix* e *Linux*.

De forma a proporcionar a interoperabilidade entre as ferramentas de gestão e simulação de redes, foi proposta a *Framework Network Scenario Description Language*

(NSDL - descrita originalmente em inglês) (MARQUES, 2012). O principal objetivo da *Framework* NSDL é fornecer uma abordagem de integração de ferramentas heterogêneas de redes, a fim de otimizar tarefas de gerenciamento. Esta abordagem se baseia em uma representação comum entre estas ferramentas, intitulada também de linguagem NSDL. NSDL é uma linguagem baseada em XML (XML, 2013) que fornece a descrição de uma topologia de rede, as propriedades de seus componentes e informações que podem ser úteis para essas ferramentas. A Figura 1 ilustra a estrutura da *Framework* NSDL.

Figura 1 - Estrutura em camadas da Framework NSDL



Como ilustrado na na Figura 1, a camada superior (*Camada de Interface Gráfica do Usuário*), como o próprio nome indica, é responsável pela interação com o usuário e inclui todas as interfaces gráficas do usuário (GUI). Esta camada também inclui a caracterização dos objetos de rede, monitoramento, construção da topologia, definição de agendamento de eventos e também a apresentação dos resultados da análise/simulação. A camada intermediária (*Linguagem de Descrição de Cenários de Rede – NSDL*) é representada pela linguagem NSDL que é aplicada para descrever componentes da rede e outras informações específicas às ferramentas utilizadas. Esta camada é responsável por fornecer a interoperabilidade entre as diferentes ferramentas que apoiam o desenvolvimento de redes. A última camada (*Camada de Implementação de Redes*) refere-se às bibliotecas e/ou aplicações responsáveis pela execução de testes e validação dos cenários de rede descritos, tais como as ferramentas de simulação de rede.

Dessa forma, dada a viabilidade da aplicação da *Framework* NSDL no contexto deste trabalho, e visando a diminuição da curva de aprendizagem de determinadas tecnologias e, em particular, do OpenFlow, é recomendável a utilização de uma ferramenta gráfica para a autoria de cenários de redes que seja compatível com a

linguagem NSDL. Essa ferramenta deverá ser capaz de se adaptar às características dos experimentos OpenFlow e proporcionar um ambiente sólido e eficaz para otimizar o gerenciamento de redes, minimizando principalmente a necessidade de conhecimento sobre várias linguagens de programação. O suporte à linguagem NSDL deverá ser dado a fim de garantir a interoperabilidade dessa ferramenta com as demais ferramentas OpenFlow.

Portanto, nesta dissertação de mestrado são apresentados os principais aspectos do desenvolvimento da ferramenta de autoria gráfica *Visual Network Description – VND (versão SDN*). Ferramenta que dispensa o usuário da exigência do conhecimento sobre diferentes aspectos de configuração e criação de redes OpenFlow, incluindo o domínio de linguagens de programação. O VND (versão SDN) permite a configuração realística de redes tradicionais e OpenFlow de forma a beneficiar o usuário no aprendizado das redes definidas por software, e também gera descrições NSDL e *scripts* de forma automática para garantir a interoperabilidade com ferramentas OpenFlow.

1.3 OBJETIVOS

O principal objetivo deste trabalho é a extensão da ferramenta *Visual Network Description - VND* (Azevedo, 2010), inicialmente proposta para a autoria de cenários de redes tradicionais e sem fio, de forma a dar suporte a autoria de cenários de redes definidas por software. A partir da extensão do *VND*, são considerados também os seguintes objetivos:

- Concepção da ferramenta gráfica *Visual Network Description – VND (versão SDN).*
- Autoria de cenários de redes OpenFlow;
- Geração automática do código NSDL e;
- Geração automática de *scripts* OpenFlow.

Como forma de alcançar os objetivos, inicialmente foi realizado um estado da arte aprofundado dos vários tópicos que enquadram neste trabalho, em seguida a extensão da linguagem NSDL para a descrição de redes OpenFlow e realização de estudos de casos para validação do trabalho realizado utilizando ferramentas OpenFlow.

As principais contribuições deste trabalho são discutidas na próxima seção.

1.4 PRINCIPAIS CONTRIBUIÇÕES

A principal contribuição deste trabalho passa pelo desenvolvimento de uma ferramenta (*VND – versão SDN)* com suporte a um ambiente gráfico para autoria de redes OpenFlow. Esta ferramenta poderá ser integrada às ferramentas OpenFlow já desenvolvidas e em desenvolvimento, para que os usuários possam ter uma visão mais abrangente e complementar sobre os cenários em análise. Como resultado, os usuários terão uma diminuição no tempo de aprendizagem da tecnologia adotada sobre os aspectos que envolvem o funcionamento das redes OpenFlow, o que poderá otimizar o desenvolvimento de novas soluções e aprimorar as atualmente existentes.

Portanto, dentre as demais contribuições deste trabalho, podem ser citadas:

- Proposta e implementação de uma ferramenta baseada em uma interface gráfica para a autoria de cenários SDN - *VND (versão SDN),* e;
- Realização de estudos de casos através da descrição de cenários de rede OpenFlow e validação do *VND (versão SDN).*

A próxima seção apresenta a organização desta dissertação.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

Além deste Capítulo introdutório, onde foram apresentadas as problemáticas, motivações, objetivos e principais contribuições, esta dissertação está organizada em mais 6 (seis) capítulos, conforme apresentados a seguir:

- Capítulo 2 (Abordagem Conceitual): onde são introduzidos os aspectos teóricos e conceituais utilizados ao longo da dissertação, além da identificação e estudo dos principais trabalhos relacionados existentes na literatura;
- Capítulo 3 (OpenFlow): são apresentados conceitos e características do protocolo OpenFlow;

- Capítulo 4 (*Framework* NSDL): onde é realizada a apresentação da *Framework NSDL*;
- Capítulo 5 (Autoria de cenários): são apresentados os principais resultados deste trabalho, relacionados à extensão da ferramenta *Visual Network Description (VND)* para o suporte à autoria de experimentos OpenFlow;
- Capítulo 6 (Estudos de casos): são apresentados alguns estudos de casos como forma de validar as principais funcionalidades desenvolvidas para o VND (versão SDN) e algumas conclusões acerca destes estudos;
- Capítulo 7 (Conclusões e perspectivas): reservado às conclusões e perspectivas futuras.

A seguir é apresentado um capítulo que faz uma fundamentação teórica acerca deste trabalho, apontando também alguns trabalhos relacionados.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

São realizadas neste capítulo discussões acerca da fundamentação teórica e trabalhos relacionados, Partes I e II, respectivamente, a partir de estudos realizados em literaturas.

2.1 FUNDAMENTAÇÃO TEÓRICA

Nesta Parte I são apresentados os principais assuntos que fazem referência ao desenvolvimento deste trabalho. Dentre os principais tópicos abordados estão a introdução às redes de computadores, redes definidas por software e redes virtualizadas, e uma breve introdução ao protocolo OpenFlow. Também são apresentados aspectos relacionados à autoria e simulação de redes, com destaque para algumas ferramentas responsáveis pela simulação e emulação de redes.

2.1.1 Redes de Computadores

As redes de computadores surgiram a partir da necessidade da troca de informações (e partilha de recursos, etc.), onde é possível ter acesso a um dado que está em uma localização geograficamente distante (TORRES, 2001). Forouzan (2006) definiu redes de computadores como um conjunto de dispositivos conectados por ligações, normalmente denominados de nós. Esse nó pode ser um computador, impressora ou outro dispositivo capaz de enviar ou receber dados gerados em outros nós da rede.

Redes de computadores e infraestrutura de tecnologia da informação representam uns dos principais segmentos da tecnologia de informação e uns dos que mais crescem, tanto em investimentos quanto em necessidade de mão-de-obra qualificada. O desenvolvimento de soluções na área de redes de computadores continua crescendo de forma vertiginosa. Alguns exemplos de soluções recentes podem ser citadas, como a Computação em Nuvem, Computação em Grid e Realidade Aumentada. Seguindo as tendências de evolução, novas tendências também podem ser citadas, como a Computação Ubíqua e a Robótica.

Todos os dias, milhares de pesquisadores e engenheiros procuram formas de manter o ser humano informado e conectado, resultando em uma tendência em comum: conectividade. A próxima seção introduzirá uma importante solução que pode ser base para as tendências anteriormente citadas e que certamente irá proporcionar o aparecimento de novas soluções em tecnologia da informação.

2.1.2 Redes Definidas por Software

As Redes Definidas por Software (ou originalmente em inglês *Software Defined Networking – SDN*) possibilitam atribuir as decisões lógicas a um dispositivo de rede definido por software denominado C*ontrolador* (ROTHENBERG et al., 2011). Apoiada no conceito "*Open Source*", as SDNs permitem que pesquisadores em tecnologias ou usuários interessados em propor novas alternativas/soluções possam aperfeiçoar e colaborar com o aprimoramento e desenvolvimento de novas tecnologias. Os usuários poderão definir fluxos de dados e o caminho que o fluxo irá percorrer utilizando um software, que é independente do hardware. Com isso, as SDNs proporcionam um grande avanço, pois as soluções resultantes delas não estarão vinculadas ao hardware e às soluções de determinados fabricantes.

Em SDN, a inteligência da rede fica concentrada em um local que permite ter uma visão geral da sua topologia. Em contraste com as redes atualmente existentes, não há necessidade de replicação das informações contidas nos protocolos de encaminhamento, pois todas as informações estarão concentradas em apenas um lugar. Esses protocolos de encaminhamento também impõem limitações, pois não é possível tomar decisões que não estejam previstas neles. Em SDN, o encaminhamento de pacotes é realizado a partir de protocolos que não residem em hardwares específicos. Isso acarreta na separação do plano de controle dos sistemas pré-configurados e rompe com os limites impostos por equipamentos proprietários.

2.1.3 Redes Virtualizadas

As redes virtualizadas representam redes sobrepostas a uma infraestrutura de rede física, onde os enlaces lógicos entre os elementos de comutação ou encaminhamento da rede virtualizada representam uma abstração das possíveis

conexões entre os nós de comutação ou encaminhamento da rede física (FERNANDES et al., 2010). A virtualização de redes permite particionar a capacidade dos componentes de uma rede física. Essa ideia de compartilhamento de recursos físicos, porém, foi estendido no âmbito de nós individuais para os demais elementos de uma rede de computadores, dando origem ao paradigma de virtualização de redes (CHOWDHURY; BOUTABA, 2010). Dentre as vantagens da virtualização de redes, está no fato das aplicações poderem ser carregadas em máquinas virtuais distintas, garantindo que uma não interfira diretamente na execução de outra.

Atuando em conjunto, as *Redes Definidas por Software* e as *Redes Virtualizadas* são fundamentais no contexto das redes experimentais e permitem aos provedores de serviços e conteúdo baseados na Internet a oportunidade da concentração na operação do serviço, ao invés da operação dos recursos de hardware. Em linhas gerais, servidores presentes em nós poderão funcionar como encaminhadores ou comutadores para diversos protocolos controlados por software. Solução que contrasta com o cenário atual, onde os equipamentos são responsáveis por realizar todo o plano de controle a partir de protocolos implementados em seu sistema operacional, impedindo as tomadas de decisões que não estejam previstas nesses protocolos.

Algumas das ferramentas mais importantes que permitem a virtualização de redes são: Xen (BARHAM et al., 2003) e VMWare (VMWARE, 2013).

2.1.4 OpenFlow

O OpenFlow é um protocolo desenvolvido em *opensource*, proposto pela Universidade de Stanford em conjunto com a Universidade da Califórnia, em Berkeley para atender à demanda de novas propostas de arquiteturas e protocolos de rede. A plataforma OpenFlow permite a criação de redes de teste em paralelo com a rede de produção, utilizando equipamentos de rede comerciais (MCKEOWN et al., 2008). Por ser um padrão aberto e programável, o OpenFlow dá suporte à inovação, permitindo o desenvolvimento de novos mecanismos de controle e o seu teste em ambientes reais (FERNANDES et al., 2010). Dada a ênfase necessária à apresentação do Protocolo OpenFlow no contexto deste trabalho, este protocolo é apresentado em mais detalhes no Capítulo III.

Na próxima seção são discutidas algumas ferramentas existentes para a autoria e simulação de redes.

2.1.5 Autoria e Simulação de Redes

São discutidos a seguir os principais aspectos que envolvem a autoria e simulação de redes.

2.1.5.1Autoria de Cenários de Redes

A autoria e simulação de redes são importantes para a criação de novas redes, pois através do estudo, planejamento e execução pode ser possível prever o comportamento das redes antes de sua implementação em ambientes reais. Através dela, é possível especificar aspectos importantes sobre o funcionamento de uma rede, tais como:

a) Topologia: através da qual é possível verificar a estrutura física da rede;
b) Parâmetros de rede: demonstram as características da rede, como: largura de banda, *jitter*, filas de pacotes, etc;
c) Eventos: descrevem o início e término de serviços e/ou aplicações;
d) Tipos de tráfego: utilizados na transmissão de dados entre os nós, como: VoIP, telnet, CBR, etc;
e) Nós: definem os dispositivos de comunicação responsáveis pela origem e destino dos pacotes, comutação e encaminhamento de pacotes, etc. Tais como: computadores, comutadores e encaminhadores, e;
f) Ligações: relacionados aos canais de comunicação entre os vários nós em uma rede.

Em particular, as ferramentas de autoria de cenários de rede possibilitam visualizar a conectividade dos dispositivos e a relação de comunicações existentes entre eles, possibilitando também a visualização da topologia da rede. No geral, uma ferramenta de autoria de cenários de redes deve contemplar as seguintes características (PORTNOI, 2007): flexibilidade, facilidade no aprendizado e implementação de novos recursos e documentação que permita a adaptação através do uso de somente uma

linguagem de programação ou configuração, e não de várias. Dentre as diversas ferramentas de autoria de redes, podem ser destacadas:

- Cisco Packet Tracer (CISCO, 2013): permite desenhar cenários de rede através de representações gráficas dos vários elementos da rede (computadores, encaminhadores, comutadores, *links*, etc), e o;
- NetSim (TETCOS, 2011): ferramenta muito popular no meio acadêmico pelo seu uso em disciplinas de redes e pelo fato de ser uma aplicação *opensource*, com suporte à modelação de diversas tecnologias e especial destaque para a funcionalidade de captura de pacotes. Baseia-se em codificação C/C++/Java e permite a modificação da configuração da interface gráfica integrada;

2.1.5.2 Linguagens para a descrição de redes

As linguagens para a descrição de redes permitem a descrição de componentes da topologia de uma rede (cenário de rede), assim como a descrição dos protocolos e do tráfego que poderá ser analisado sobre essa topologia. Por outro lado, essas linguagens podem ser um instrumento importante para permitir a interoperabilidade entre diferentes ferramentas para a análise e gestão de redes. Elas também são importantes, pois possibilitam uma visão ampla e detalhada de toda a estrutura de uma rede de computadores.

A maioria das ferramentas de gestão de redes utiliza uma linguagem para descrever as redes que suportam. O uso de tal linguagem pode ser transparente para os usuários dessas ferramentas se existirem ferramentas gráficas que facilitem a criação, edição e execução de comandos sobre os objetos de rede. Contudo, nem sempre isso é possível e, nessa situação, os usuários têm de conhecer a linguagem de forma a poderem descrever toda a rede ou a poderem adicionar ou editar algum dos seus objetos.

2.1.5.3 Simulação/Emulação de Redes

Os processos de simulação e emulação correspondem a dois tipos de instrumentos distintos com finalidades também distintas. Enquanto o simulador permite determinar o comportamento do sistema estudado sem a necessidade de implementá-lo

em ambiente real, o emulador tenta imitar esse comportamento, com o intuito de ser o mais fiel possível ao objeto ou processo que está sendo emulado (LI et al., 2008).

Juntas, simulação e emulação podem contribuir na construção de redes experimentais, mesmo antes de serem implementadas. Essas redes servem como base para pesquisas e análises prévias do comportamento que teria caso o sistema estudado fosse implementado em um ambiente real. Com isso, é possível, por exemplo, determinar a sua viabilidade e/ou otimizar o seu comportamento.

Em redes de computadores, a simulação é bastante útil, uma vez que o comportamento de uma rede pode ser modelado através do cálculo de interações entre diferentes dispositivos que compõem uma rede, como comutadores, encaminhadores, ligações e tráfego. Já o emulador, permite o planejamento de uma rede com o objetivo de avaliar o seu desempenho como se estivesse sendo executado em um ambiente real.

Existem diversas ferramentas responsáveis pela simulação e emulação de redes e a escolha de uma delas vai de acordo com os objetivos pertinentes à proposta do cenário de rede e o tipo de análise e resultados que são pretendidos. Algumas dessas ferramentas são: OPNET, NS2, OMNet++, JiST e o SimPy.

- O *OPNET* (OPNET, 2013) é um software que possui uma boa interface gráfica que permite o estudo de redes de computadores, os dispositivos que a compõem, protocolos e aplicações.
- O *Network Simulator 2* (NS2) (NS2, 2013) é um dos simuladores de redes mais populares. Sucessor do Network Simulator, o NS2 tem sido bastante utilizado em pesquisas para fins acadêmicos.
- O *OMNeT++* (OMNeT++, 2013) não é um simulador de rede propriamente dito, no entanto, têm sido bastante utilizado em ambientes de rede, pois possui recursos que permitem a sua utilização para esse fim.
- O *JiST* (Jist, 2013) é uma ferramenta de simulação em Java em tempo de simulação que permite a implementação de simulações de rede em Java. As simulações em JiST são compostas por elementos de redes, com eventos de simulação invocados através de métodos. A fim de executar a aplicação em tempo de simulação, o JiST utiliza uma classe dinâmica que descreve o byte code da aplicação.

- O *SimPy* (Simpy, 2013) é uma ferramenta de simulação em Python que funcionam através de eventos discretos orientados a processos. O SimPy fornece instruções para a sincronização de processos de simulação e comandos para o monitoramento de dados relativos à simulação.

3 TRABALHOS RELACIONADOS

Nesta Parte II são abordados os trabalhos existentes na literatura relacionados à autoria de cenários de redes, às linguagens voltadas para a descrição de redes e aos projetos OpenFlow.

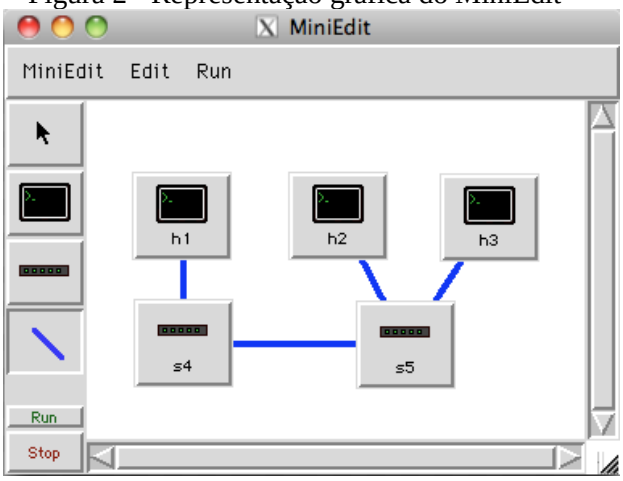
3.1 AUTORIA DE CENÁRIOS DE REDES

Como visto anteriormente, existem diversas ferramentas que são responsáveis pela autoria de cenários de redes, a exemplo do Cisco Packet Tracer (CISCO, 2013) e NetSim (TETCOS, 2011). Entretanto, são poucas as ferramentas para a autoria de cenários de redes OpenFlow. Nesta seção apresentamos duas ferramentas baseadas em ambientes de interface gráfica para a autoria de redes OpenFlow. No momento estas são as únicas ferramentas para a autoria de redes OpenFlow que possuem interface gráfica e por isso a escolha delas. O critério pela escolha de ferramentas com interfaces gráficas foi motivada pela fato da solução deste trabalho também possuir ambiente com interface gráfica.

a) MiniEdit

O MiniEdit (MINIEDIT APP, 2013), ilustrado na Figura 2, é uma aplicação desenvolvida em Python baseada em interface gráfica para o *Mininet*. É um simples editor que permite a inclusão de estações de trabalho e comutadores OpenFlow no cenário, e conectá-los para posterior utilização no Mininet. O código-fonte do MiniEdit está disponível para que usuários interessados possam aperfeiçoa-lo.

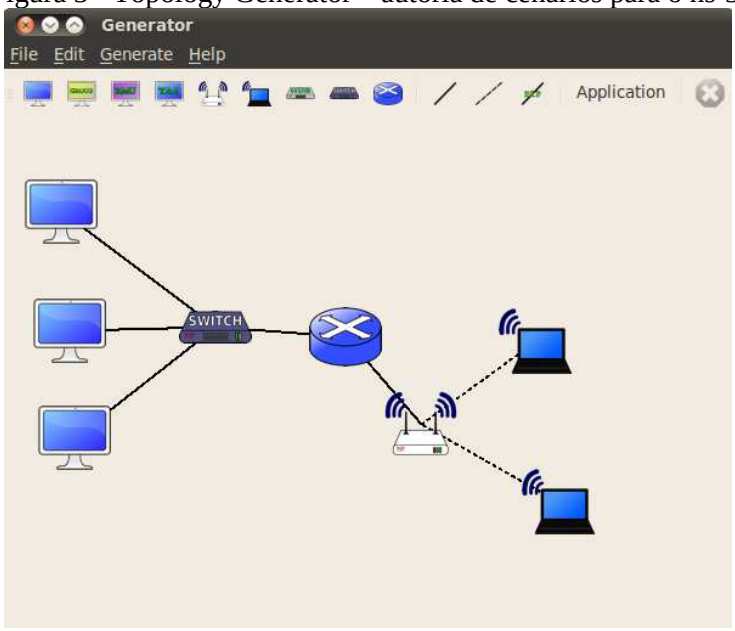
Figura 2 - Representação gráfica do MiniEdit



b) Topology Generator for NS-3

O *NS-3* (NS3, 2013) é um simulador de redes baseado em eventos especialmente desenvolvido para pesquisa e uso educacional. Com o surgimento do OpenFlow, o ns-3 passou também a dar suporte à simulação de ambientes de redes com a utilização de comutadores OpenFlow e controladores. A Figura 3 ilustra a ferramenta *Topology Generator* (TOPOLOGY GENERATOR, 2013), desenvolvida para o ns-3 para proporcionar a criação de cenários de redes de forma rápida e fácil, através de um ambiente de interface gráfica simples.

Figura 3 - Topology Generator – autoria de cenários para o ns-3



É importante destacar que as 2 (duas) ferramentas dicutidas anteriormente possuem características que diferem da ferramenta proposta neste trabalho. O *MiniEdit*, até o momento, é uma ferramenta compatível com o Mininet, mas através dele não é possível realizar configurações para controladores OpenFlow. Por exemplo, não é possível configurar fluxos para um controlador ‘X’ ou ‘Y’. Da mesma forma, o *Topology Generator* também é limitado em relação a este tipo de configuração. Em resumo, as duas ferramentas exigem do usuário um conhecimento mais aprofundado do controlador OpenFlow para a descrição mais completa de um cenário. O *MiniEdit* e o *Topology Generator* possuem as seguintes características:

- Possuem Interface Gráfica;
- Permitem a reutilização de cenários;
- Permitem a criação e configuração de topologias complexas;
- Permitem a visualização do ambiente de rede para a criação, configuração e solução de problemas;
- Suportam o protocolo OpenFlow;
- Não suportam outra ferramentas OpenFlow;
- Permitem a geração automática de *scripts*;
- Necessitam ser instaladas e;
- Executam em ambiente local.

A seguir são apresentadas algumas linguagens voltadas para a descrição de redes.

3.2 LINGUAGENS PARA DESCRIÇÃO DE REDES

As linguagens para a descrição de redes permitem a descrição dos componentes da topologia de uma rede (cenário de rede), assim como a descrição dos protocolos e do tráfego que poderá ser analisado sobre essa topologia. Por outro lado, essas linguagens podem ser um instrumento importante para permitir a interoperabilidade entre diferentes ferramenta para a análise e gestão de redes. A partir da revisão da literatura, foram encontradas algumas linguagens que foram propostas para a descrição de redes, dentre

elas estão a *Network Description Language*, *Network and Graph Markup Language e Infrastructure and Network Description Language.*

a) Network Description Language (NDL)

A *Network Description Language* (HAM et al., 2007) foi desenvolvida na Universidade de Amsterdam para a descrição de redes. NDL fornece uma semântica comum para uma aplicação, para a rede e para o prestador de serviços. Ela pode ser utilizada para criar gráficos de redes entre domínios e identifica elementos básicos de uma rede, como dispositivos, interfaces e links. O NDL também modela os domínios de redes existentes, com diferentes níveis de administração e políticas. O NDL é útil na resolução de muitos problemas no que se refere à operação de redes híbridas, permitindo a criação de mapas de rede e algoritmos para descobertas de caminho.

b) Network and Graph Markup Language (NaGML)

O *Network and Graph Markup Language* (BRADLEY, 2004) é uma linguagem desenvolvida para a descrição de redes que permite a leitura, validação, visualização e escrita de redes gráficas. A NaGML utiliza da linguagem XML para a representação da topologia de uma rede e as propriedades dos seus nós e foi concebido para apoiar a vasta e diversificada comunidade de pessoas que usam redes e gráficos para muitos propósitos e em uma variedade de contextos. O NaGML foi desenvolvido para exibir, animar e analisar dados de incidentes.

c) Infrastructure and Network Description Language (INDL)

A *Infrastructure and Network Description Language (INDL)* (GHIJSEN, 2012) tem o objetivo de fornecer descrições independentes de infraestruturas de computação. Estas descrições incluem os recursos físicos e toda a infraestrutura de rede. A INDL também fornece o vocabulário necessário para descrever a virtualização de recursos e os serviços oferecidos por esses recursos. Além disso, a linguagem pode ser facilmente estendida para descrever federações de diferentes infraestruturas de computação, tipos específicos de equipamentos

ópticos e também os aspectos comportamentais de recursos, como, por exemplo, o consumo de energia.

A Tabela 1 apresenta uma comparação entre as linguagens citadas.

Tabela 1 - Comparação entre as ferramentas para descrição de redes

	NDL	NaGML	INDL
Schema para descrição	RDF (Resource Description Framework Schema)	XML	RDF (Resource Description Framework Schema)
Suporte a redes em grande escala	Sim	Sim	Sim
Possibilidade de descrição para novas tecnologias de redes	Sim	Sim	Sim
Suporte ao OpenFlow	Não	Não	Não
Representação de redes em larga escala	Não	Sim	Não
Grau de compreensão	Difícil	Difícil	Difícil

Valem a pena serem citadas outras linguagens que também possuem a proposta de descrição de redes, como: Network Mark-up Language Working Group (NML-WG), (NML-WG, 2013), GEYSERS (ESCALONA, 2011) e NOVI (NOVI, 2013).

Na próxima seção são apresentados alguns projetos OpenFlow.

3.3 PROJETOS OPENFLOW

Nesta seção são apresentados alguns dos principais projetos que estão colaborando para a evolução do protocolo OpenFlow.

a) GENI

O projeto *Global Environment for Network Inovation (GENI)* tem sido a principal iniciativa norte americana no âmbito da Internet do Futuro. Ele abrange as mais variadas tecnologias para diferentes meios de transmissão, como: sem fio, óptico e elétrico. O objetivo do GENI é a proposta de um grande laboratório em larga escala para experimentações em redes de computadores, onde o maior propósito é validar novas possibilidades para Internet do Futuro. Para GENI, os principais conceitos relacionados à experimentação em Internet

do Futuro e que fazem parte do projeto de sua arquitetura são (GENI SYSTEM OVERVIEW, 2008):

Programabilidade: para que seja possível implementar softwares nos nós do GENI para controle de fluxo;

Virtualização e formas alternativa de compartilhamento de recursos: qualquer que seja a implementação de máquina virtual sobre um nó GENI, será permitido que múltiplos pesquisadores simultaneamente compartilhem a mesma infraestrutura. Cada experimento terá à sua disposição uma fatia isolada, com recursos fim-a-fim alocados dentro da infraestrutura do GENI;

Federação: o GENI deverá ser composto por infraestruturas próprias e por outras de apoio, operadas por organizações parceiras ao projeto, criando o conceito de uma federação de recursos e nós, que, na visão de um pesquisador, comportar-se-á como se fosse uma única infraestrutura;

Experimentos baseados em fatias: Cada experimento no GENI será realizado sobre uma plataforma composta de um conjunto de recursos reservados em diversas localizações, chamada de uma fatia de recursos. Dentro dessa fatia o pesquisador poderá remotamente descobrir, reservar, configurar, "depurar", operar e gerenciar seus experimentos e recursos disponíveis.

b) OFELIA

O projeto *OpenFlow in Europe Linking Infrastructures and Applications (OFELIA)* (FP7 OFELIA, 2013) é financiado pela União Européia e visa a implementação de uma grande plataforma para testes (*testbed*) para tráfego OpenFlow em vários locais (ou "ilhas") na Europa (KÖPSEL, 2011), incluindo Bélgica, Alemanha, Espanha, Suíça e Reino Unido. O OFELIA possibilita que pesquisadores criem um único ambiente experimental para testes e que também permite o controle da rede de forma precisa e dinâmica.

O OFELIA permitirá o teste de novos algoritmos de roteamento, tunelamento, protocolos e planos de controle. Além disso, novas aplicações poderão ser colocadas no controlador OpenFlow a qualquer momento na

infraestrutura. Também permitirá serem investigadas novas estruturas e formatos de endereçamento e modelos de encaminhamentos.

c) FIBRE

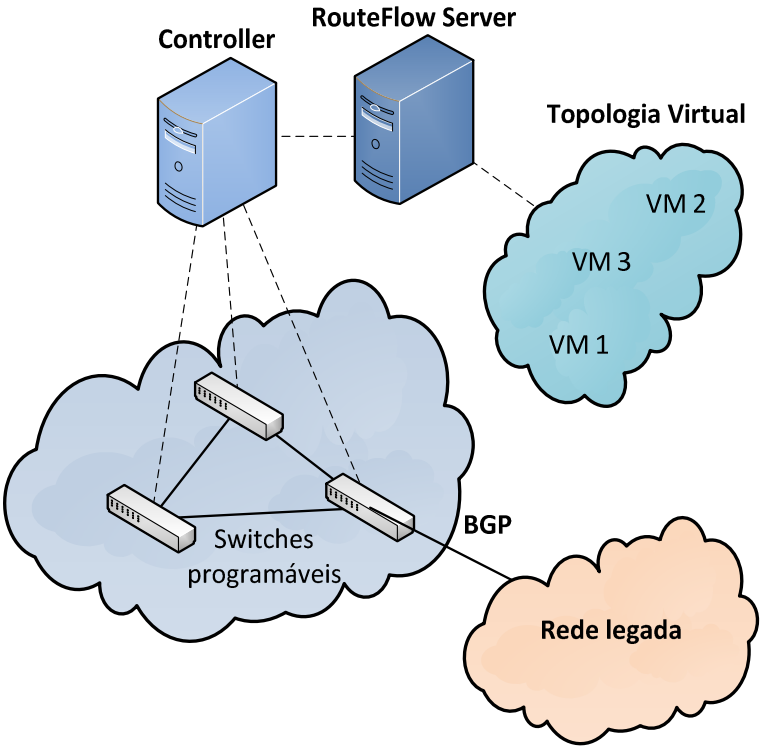
Fruto da cooperação entre pesquisadores brasileiros e europeus, o projeto FIBRE (FIBRE, 2013) pretende implementar e validar uma infraestrutura compartilhada entre institutos e universidades brasileiras e europeias. O FIBRE visa a construção de uma rede experimental em larga escala com concepções voltadas à Internet do futuro com bases no OpenFlow.

O projeto FIBRE possui características peculiares se comparado aos outros projetos atualmente existentes. Diferentemente do que acontece em outros projetos, o FIBRE possui um menor número de federações (também comumente chamada de ilhas - que podem representar uma infraestrutura de rede privada submetidas a determinadas características técnicas e também políticas), contudo, a sua abrangência é intercontinental. O fato de abranger países em continentes diferentes exige que o FIBRE utilize das mais diversas tecnologias multicamada.

d) ROUTEFLOW

O RouteFlow é um projeto em código aberto desenvolvido pelo *Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD)* situado no Brasil (ROTHENBERG et al., 2011). O RouteFlow fornece serviço virtualizado de roteamento IP sobre hardware com o OpenFlow habilitado. Ele é composto por um controlador OpenFlow, um servidor RouteFlow independente e um ambiente de rede virtual que produz a conectividade de uma infraestrutura física com serviços de roteamento (Por exemplo, Quagga). O Quagga (QUAGGA, 2013) é um pacote de serviços voltados para roteamento e suporta implementações para OSPFv2, OSPFv3, RIPv1 e RIPv2, RIPng, BGP-4 para plataformas Unix, em particular FreeBSD, Linux, Solaris e NetBSD. A Figura 4 ilustra o funcionamento do RouteFlow.

Figura 4 - Exemplo de funcionamento do RouteFlow



Na topologia virtual, cada máquina virtual executa um software responsável pelo encaminhamento, a exemplo do Quagga, anteriormente citado. Além de realizar o gerenciamento das rotas, o controlador também realiza o gerenciamento das máquinas virtuais e apesar do controle estar fisicamente centralizado, o gerenciamento da rede é logicamente distribuído. Com isso, não é necessário nenhuma alteração nos protocolos de encaminhamento atualmente existentes.

3.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi realizada uma abordagem teórica dos principais aspectos que envolvem o desenvolvimento deste trabalho, com destaque para as redes definidas por software, redes virtualizadas e o protocolo OpenFlow. Também foram apresentadas algumas ferramentas mais utilizadas e alguns dos principais orientados ao protocolo OpenFlow.

Para melhor fundamentar o desenvolvimento deste trabalho, foram levantados aspectos relacionados à autoria e simulação de redes, com a apresentação de algumas ferramentas e comparações entre elas. Apoiado nas redes definidas por software, no protocolo OpenFlow e nos aspectos que os envolvem, os assuntos abordados neste capítulo foram fundamentais para contextualizar e dar suporte à proposta de trabalho realizada nesta dissertação.

Como sequência a este trabalho, é apresentado a seguir o Protocolo OpenFlow.

4 OPENFLOW

4.1 INTRODUÇÃO

A Internet foi criada há aproximadamente 30 anos atrás, com o objetivo de atender um público restrito (cientistas e pesquisadores). Hoje ela é utilizada por centenas de milhões de pessoas, porém não houve nenhuma evolução com relação a sua estrutura básica. É inegável o sucesso da Internet, no entanto, ela apresenta limitações que tornam necessárias a realização de mudanças em sua arquitetura. Prova disso são algumas soluções pontuais, tais como o *Network Address Translation* (NAT) (EGEVANG, 1994), *IP SEC* (KENT, 1998), *filtros de spam* (KIM, 2008), *Classless Internet Domain Routing* (CIDR) (FULLER et al., 1993), dentre outros.

Os defensores do IP argumentam que o protocolo vem funcionando de forma satisfatória e que embora uma mudança na Internet seja necessária, ela deverá partir de uma evolução do próprio protocolo IP e não de uma total ruptura com ele. Já os que concordam por uma Internet reinventada e recalculada do zero, defendem a abordagem “clean-state”, que consiste em reelaborar a Internet sem os seus limites de arquitetura atual (FELDMANN, 2007).

Independentemente do método adotado, a Internet do futuro pode ser concebida através de testes em redes experimentais. No cenário atual, a maior parte das novas ideias da comunidade de pesquisa de rede não é testada, levando à crença comumente mantida de que a infraestrutura da Internet “ossificou” ou enrijeceu (PAUL et al., 2011). Como as redes experimentais permitem a simulação de novas soluções para a Internet e apesar de muitas vezes não permitirem a medida exata em situações reais, a necessidade dessas redes é não somente inegável, como também crescente, dado o número cada vez maior de aplicações para a Internet.

Neste contexto, foram concebidas as redes definidas por softwares e a virtualização de redes. Atuando em conjunto, ambas são fundamentais no contexto das redes experimentais e permitem aos provedores de serviços e conteúdo baseados na Internet a oportunidade da concentração na operação do serviço, ao invés da operação dos recursos de hardware. Em linhas gerais, servidores presentes em nós poderão funcionar como encaminhadores ou comutadores para diversos protocolos controlados

por software. Solução que contrasta com o cenário atual, onde os equipamentos são responsáveis por realizar todo o plano de controle a partir de protocolos implementados em seu sistema operacional, impedindo tomadas de decisões que não estejam previstas nesses protocolos.

A partir da necessidade de romper essa restrição e definir planos de controles independentes aos protocolos implementados por fabricantes de equipamentos, surge a tecnologia OpenFlow, proposta por (MCKEOWN et al., 2008). O OpenFlow consiste em um protocolo de comunicação que implementa conceitos de redes definidas por software, abstraindo os recursos de hardware de cada dispositivo intermediário, sejam eles comutadores ou encaminhadores.

O OpenFlow é um protocolo desenvolvido em opensource, proposto pela Universidade de Stanford em conjunto com a Universidade da Califórnia, em Berkeley para atender à demanda de novas propostas de arquiteturas e protocolos de rede. A plataforma OpenFlow permite a criação de redes de teste em paralelo com a rede de produção, utilizando equipamentos de rede comerciais (MCKEOWN et al., 2008). Por ser um padrão aberto e programável, o OpenFlow dá suporte à inovação, permitindo o desenvolvimento de novos mecanismos de controle e o seu teste em ambientes reais (FERNANDES et al., 2010).

Este capítulo foi dividido em duas partes: Parte I – Contextualização e Parte II – Ferramentas OpenFlow. Na Parte I é apresentado o protocolo OpenFlow, seus componentes e aspectos de funcionamento, e na Parte II as principais ferramentas OpenFlow, incluindo controladores e comutadores.

5 CONTEXTUALIZAÇÃO

5.1 CONCEITOS BÁSICOS

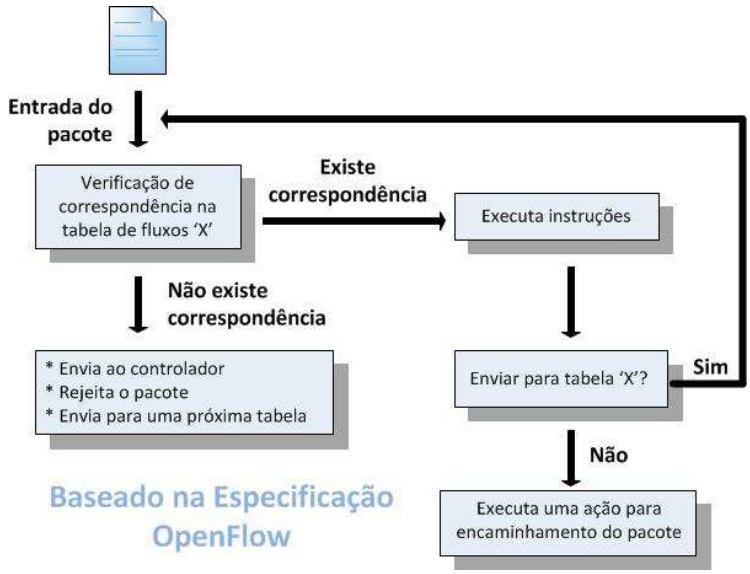
O OpenFlow, no momento, é um protocolo de implementação para redes definidas por software (SDNs) amplamente utilizado pela comunidade científica, e foi proposto na Universidade de Standford com a colaboração da Universidade da Califórnia em Berkeley (MCKEOWN et al., 2008). O protocolo OpenFlow surgiu de forma a atender a demanda para a validação da proposta de novas arquiteturas e protocolos sobre equipamentos comerciais de rede e também viabilizar a correção das deficiências das redes atualmente existentes. O OpenFlow define um protocolo padrão para determinar as ações de entrega de pacotes dentro de dispositivos de rede como, por exemplo, comutadores, encaminhadores e pontos de acesso sem fio (MCKEOWN et al., 2008). As regras e ações executadas em hardware estão sob a responsabilidade de um elemento externo, assim chamado Controlador, que pode ser implementado num servidor comum (SHERWOOD et al., 2010).

O OpenFlow fornece um padrão na manipulação das tabelas de fluxos e, consequentemente, permite as tomadas decisões de tráfegos com base no fluxos e em seus comportamentos. Neste aspecto, o controlador pode ser estendido a fim de executar tarefas complementares ou adicionais, tais como as decisões de acesso à rede e encaminhamento.

Portanto, uma rede baseada em OpenFlow consiste em equipamentos de rede habilitados e submetidos a um controlador definido por software. Este controlador centraliza todas as tarefas de controle e é responsável pela tomada de decisões acerca da adição e remoção de entradas na tabela de fluxos, de acordo com o objetivo desejado. A arquitetura OpenFlow define um modelo de rede onde os comutadores OpenFlow são totalmente programáveis, como se fossem computadores. Ele também possibilita que as instruções de encaminhamento possam ser baseadas em fluxos e não somente em terminais IP. Basicamente, o OpenFlow é constituído por três partes: tabelas de fluxos – para a associação entre ação a ser executada em relação à entrada na tabela de fluxos; canal seguro ou *secure channel* – para a garantia de troca segura de informações entre comutador e controlador e; protocolo OpenFlow – faz parte da definição de um

protocolo inteiramente aberto. A Figura ilustra o tratamento de um pacote em um comutador OpenFlow.

Figura 5 - Digrama de tratamento de um pacote em comutador OpenFlow



As características inerentes ao OpenFlow possibilitam que pesquisadores reprogramem os dispositivos OpenFlow de forma que não interfira nas configurações dos demais equipamentos de uma rede. Tendo o sistema operacional implantado em plataformas abertas, a introdução de novas funcionalidades torna-se questão de instalação de pequenos programas escritos por quem os quiser programar, seja um fabricante ou um programador.

Dentre alguns diferenciais do OpenFlow, podem ser destacados: diferentes fluxos são encaminhados por vias distintas, ambiente de desenvolvimento aberto, flexibilidade na definição dos fluxos, agregação de tráfego e sobreposição do protocolo IP. Por ainda estar em fase de padronização, a sua utilização limita-se a experimentos em equipamentos do tipo PC (Personal Computer) com GNU/Linux e um número ainda limitado de roteadores e comutadores comercias disponíveis no mercado.

5.2 COMPONENTES

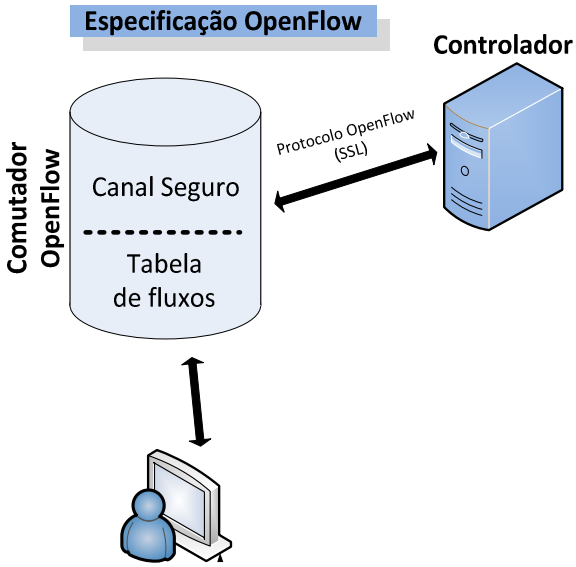
Antes de entender o funcionamento do OpenFlow é importante conhecer os seus componentes. Basicamente, uma rede OpenFlow consiste em equipamentos de rede

habilitados para que o estado das tabelas de fluxos possa ser administrado com base nas decisões de um controlador definido em software. Um comutador OpenFlow possui uma ou mais tabelas de fluxos e tabelas de grupos para o encaminhamento de pacotes e um canal responsável por realizar a comunicação com o controlador. O comutador realiza comunicação com o controlador e o controlador gerencia o comutador através do protocolo OpenFlow.

De forma resumida, os principais componentes de uma rede OpenFlow são:

- Comutadores com o OpenFlow habilitado;
- Controlador(es);
- Um banco de dados que possui um aspecto completo da topologia da rede.

Figura 6 - Principais componentes de um comutador OpenFlow



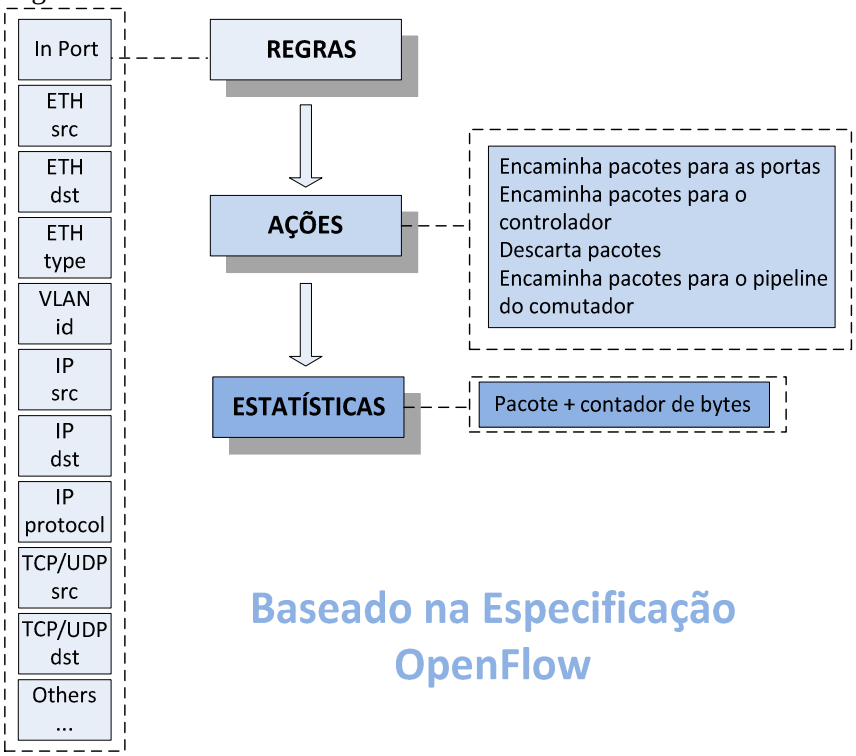
A Figura 6 ilustra os componentes de uma rede OpenFlow. A sua estrutura tem como base as especificações OpenFlow versão 1.2.3 (ONF, 2013).

5.3 FUNCIONAMENTO

Quando um pacote chega a um dispositivo com o OpenFlow habilitado, os cabeçalhos do pacote são comparados às regras das entradas das tabelas de fluxos. Os controladores logo são atualizados e o pacote é tratado conforme as decisões correspondentes. Caso não haja nenhuma correspondência entre o pacote e as entradas

da tabela de fluxos, o pacote é inteiramente encaminhado ao controlador. Como pode ser visto na Figura 7, o tratamento de fluxos é realizado de acordo com as funcionalidades de três campos: "*Regras*", "*Ações*" e "*Estatísticas*". O campo "*Regras*" é responsável por checar os campos dos cabeçalhos dos pacotes e classifica-los como parte de um fluxo. O campo "*Ações*" é responsável por definir qual o tratamento para os pacotes e o campo "*Estatísticas*" faz o registro dos pacotes para a contagem da quantidade e tamanho dos pacotes que serão reservados para cada fluxo. Ele também faz o registro dos tempos de transmissões dos pacotes para controle da tabela de fluxos.

Figura 7 - Tratamento de fluxos em uma tabela de fluxos



A Figura 8 ilustra um exemplo do funcionamento do protocolo OpenFlow. No Fluxo No. 01, os pacotes que chegam ao comutador com *DPID (Data Path Identifier), identificador único do comutador* No. 0000000000000005, são tratados da seguinte forma: os pacotes que entram neste comutador pela porta 1 com prioridade igual a 32768, devem sair pela porta 3 com a inserção de uma Tag de *Vlan ID* igual a 20. Já no Fluxo No. 02, os pacotes que chegam ao comutador com *DPID* No. 0000000000000005, são tratados da seguinte forma: os pacotes que entram neste comutador com prioridade 32768 e na Vlan 20, devem sair pela porta 1 com a desmarcação da Vlan 20.

Figura 8 - Exemplo de uma Tabela de Fluxos

Fluxo #01

```
switch1 = 0000000000000005
flow1msg = of.ofp_flow_mod()
flow1msg.cookie = 0
flow1msg.priority = 32768
flow1msg.match.in_port = 1
# ACTIONS--------------------------------
flow1vlan_id = of.ofp_action_vlan_vid (vlan_vid = 20)
flow1out = of.ofp_action_output (port = 3)
flow1msg.actions = [flow1vlan_id, flow1out]
```

Fluxo #02

```
switch3 = 0000000000000005
flow3msg = of.ofp_flow_mod()
flow3msg.cookie = 0
flow3msg.priority = 32768
flow3msg.match.dl_vlan = 20
# ACTIONS--------------------------------
flow3stripvlan = of.ofp_action_strip_vlan ()
flow3out = of.ofp_action_output (port = 1)
flow3msg.actions = [flow3stripvlan, flow3out]
```

Para processar os pacotes no OpenFlow, a forma de configuração mais simples é forçar os pacotes de um determinado fluxo passar pelo controlador e para isso o controlador não precisa adicionar novas entradas de fluxos no dispositivo OpenFlow. A vantagem neste tipo de configuração é a flexibilidade em detrimento ao desempenho. De forma alternativa, apenas o cabeçalho pode ser enviado ao controlador, mantendo o pacote armazenado no buffer do hardware.

Cada entrada de fluxo na tabela de fluxos do comutador tem uma ação associada a ela. As principais ações são: encaminhar os pacotes de um determinado fluxo para uma determinada porta (ou portas), encapsular e transmitir pacotes para um controlador utilizando um canal seguro e o descarte de pacotes de determinados fluxos. Cada entrada na tabela de fluxo, por sua vez, possui três campos: uma regra que define e classifica o fluxo, a ação que define como os pacotes devem ser processados e geração de estatísticas para ajudar na remoção de fluxos inativos.

6 FERRAMENTAS OPENFLOW

Na Parte II deste capítulo são abordadas algumas das ferramentas OpenFlow, tais como: Mininet, Nox, Pox, Floodlight, Beacon, Ryu, Flowvisor, Open vSwitch, Resonance, RouteFlow e OpenWrt. As características distintas dessas ferramentas exigem um tempo de aprendizagem natural acerca de qualquer ferramenta ou tecnologia e a experiência com mais de uma delas, em muitos casos, requer o conhecimento acerca de duas ou mais linguagens de programação. A utilização de apenas um ambiente no processo de autoria do cenário de rede interoperável com as ferramentas citadas abaixo, oportuna, sobretudo, o desenvolvimento deste trabalho.

6.1 FERRAMENTAS OPENFLOW

Com base em estudos realizados em literaturas, são apresentadas a seguir as ferramentas OpenFlow que possuem maior relevância entre usuários e pesquisadores em todo o mundo.

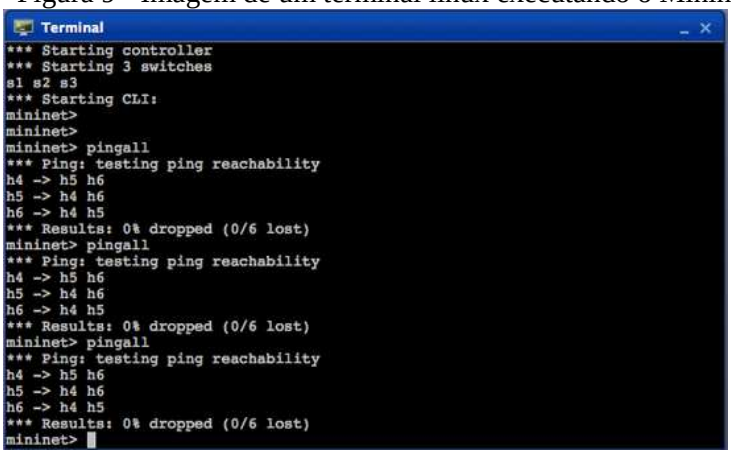
6.1.1 Mininet

O Mininet (LANTZ et al., 2010) é um emulador de rede desenvolvido em Python e permite a criação de redes experimentais virtuais, com possibilidade de comunicação com um ambiente real. No Mininet é possível definir e configurar nós, como: comutadores, controladores e estações de trabalho. No entanto, o Mininet não oferece uma interface gráfica para ilustrar a execução do cenário executado, portanto, o usuário pode definir um cenário de rede através de *scripts* ou através de comandos pré-definidos emitidos em um terminal linux, por exemplo:

```
$ sudo mn --topo tree,3 --switch ovsk --controller remote
```

Este comando cria uma topologia de árvore com profundidade 3, define um comutador OpenFlow e um controlador remoto. A forma de criação de cenários de redes através de *scripts* será apresentado nos capítulos posteriores. A execução no Mininet também será realizada através de linhas de comando. A partir de sua execução, também é possível a utilização combinada de outras ferramentas de gestão de redes (por exemplo, Wireshark) para verificar o comportamento do cenário executado.

Figura 9 - Imagem de um terminal linux executando o Mininet



A Figura 9 ilustra um terminal linux com o Mininet em execução. O Mininet permite que desenvolvedores e pesquisadores criem redes virtuais, através da execução de um núcleo (kernel) real, encaminhador e código de aplicação em uma estação (nativa, máquina virtual ou em nuvem). Os cenários de rede criados pelo Mininet são escaláveis e possibilita a realização de testes e simulações antes mesmo de sua implementação em meios físicos.

6.1.2 Nox

Desenvolvido pela Nicira Networks, o Nox (GUDE et al., 2008) foi o primeiro controlador OpenFlow a ser desenvolvido. Ele foi entregue à comunidade em 2008 e desde então têm sido bastante explorado pelos pesquisadores em redes SDN. O Nox foi desenvolvido na linguagem de programação C++ e é compatível com recentes distribuições linux, em particular: o Ubuntu 11.10 e 12.04, Debian e RHEL 6.

O Nox inclui componentes para: descoberta de topologia, aprendizagem por comutação (original em inglês: *learning switch*) e comutação em larga escala. O propósito do Nox é prover uma interface de programação de alto-nível em que aplicações de gerenciamento da rede possam ser construídas.

6.1.3 Pox

O controlador OpenFlow Pox (POX, 2013) foi desenvolvido em Python e é considerado o precursor do Nox. O Pox permite um rápido desenvolvimento e

prototipagem. O Pox inclui alguns recursos, como: componentes de amostra para a seleção de caminho, descoberta de topologia, entre outros; é compatível com Linux, Mac OS e Windows; suporta as mesmas interfaces gráficas e ferramentas de visualização compatível com o Nox; e possui melhor performance se comparado ao Nox.

6.1.4 Beacon

O controlador Beacon (BEACON, 2013) também é um controlador baseado na linguagem de programação Java que suporta operações baseadas em eventos e baseadas em múltiplos processos (threads).

O Beacon fornece uma estrutura para controlar dispositivos de rede utilizando o protocolo OpenFlow e um conjunto de aplicações que fornecem funcionalidades de plano de controle comumente necessários em uma rede OpenFlow.

6.1.5 Floodlight

O Floodlight (FLOODLIGHT, 2013) é um controlador OpenFlow desenvolvido na linguagem de programação Java que começou a ser desenvolvido por pesquisadores da Universidade de Stanford e da Universidade de Berkeley. O Floodlight foi entregue à comunidade de desenvolvedores de código aberto, à engenheiros de redes SDN e à empresa *Big Switch Networks* para evoluções e está disponível aos usuários que queiram utilizá-lo.

Licenciado pela Apache, o Floodlight pode ser utilizado tanto em comutadores virtuais quanto em comutadores físicos, desde que os comutadores sejam compatíveis com o protocolo OpenFlow.

6.1.6 Ryu

O Ryu (RYU, 2013) é um sistema operacional de rede baseado em Python e suporta vários protocolos para gerenciamento de dispositivo de rede, como o OpenFlow, Netconf, OF-config, etc. O Ryu suporta as versões OpenFlow 1.0, 1.2, 1.3 e várias extensões Nicira. Ele também funciona com *OpenStack* (*sistema operacional baseado*

na nuvem que controla o armazenamento, computação e recursos de rede em um centro de dados). Também possui código fonte aberto e tem uma grande comunidade de desenvolvedores. O Ryu tem como objetivo ser um sistema operacional para SDN e tem diversas vantagens, incluindo a integração com *OpenStack*. Por ser implementado em Python, seu desempenho é relativamente mais lento do que alguns outros controladores.

6.1.7 Flowvisor

O Flowvisor (SHERWOOD et al., 2009) é uma proposta de virtualização da camada de rede que utiliza o OpenFlow para abstração de hardware. O Flowvisor fica localizado entre o plano de controle e o plano de dados, atuando como um proxy transparente entre comutadores e controladores OpenFlow.

6.1.8 Open vSwitch

O Open vSwitch (OPEN VSWITCH, 2013) é um software que representa um comutador virtual e logicamente reside em um hypervisor ou domínio de gerenciamento (Exemplo: Dom0 no Xen (BARHAM et al., 2003)). O Open vSwitch permite a conectividade entre máquinas virtuais e interfaces físicas. Ele possibilita a implementação Ethernet com Vlans, RSPAN e configurações básicas de ACL.

O Open vSwitch difere das abordagens tradicionais na medida em que proporciona uma interface refinada de controle de encaminhamento de pacotes, que pode ser utilizado, dentre outras coisas, para apoiar Qualidade de Serviço (QoS), tunelamento, e regras de filtragem. Ele também suporta uma interface remota que permite a migração de estados de configuração, muito útil para a gestão de políticas para máquinas virtuais. Além disso, o Open vSwitch possui encaminhamento flexível, baseado em tabelas que pode ser usado para dividir de forma lógica o plano de encaminhamento. O Open vSwitch é compatível com a maioria dos ambientes de virtualização baseados em Linux, incluindo Xen, XenServer, KVM e QEMU.

6.1.9 Resonance

O Resonance (NAYAK et al., 2009) é um sistema desenvolvido para proteger redes corporativas através de controles dinâmicos baseados em informações em nível de fluxo e em tempo real. O Resonance utiliza de comutadores programáveis para manipular as camadas inferiores.

Ele fornece mecanismos para a implementação de políticas de segurança de rede diretamente em dispositivos e comutadores, deixando pouca responsabilidade para estações de trabalho ou camadas de rede superiores.

6.1.10 RouteFlow

O RouteFlow (ROTHENBERG et al., 2011) é mantido pelo CPqD e fornece serviço virtualizado de roteamento IP sobre hardware com o OpenFlow habilitado. O RouteFlow, cujo nome é o mesmo o do projeto onde foi proposto, foi discutido em mais detalhes na seção II.3.3.

6.1.11 OpenWRT

O OpenWrt (OPENWRT, 2013) é um *firmware* para sistemas embarcados normalmente utilizado em encaminhadores sem fio. O OpenWRT possui código fonte aberto e o usuário tem liberdade de realizar personalizações de acordo com as suas necessidades. O OpenFlow pode ser adicionado ao OpenWRT através da geração de uma *firmware* específica para versões e fabricantes de encaminhadores sem fio. Apesar de estar desatualizado, o site do OpenWrt traz uma extensa relação de encaminhadores sem fio que são compatíveis com *firmwares* OpenWRT.

6.1.12 Considerações finais

Devido as suas características, o protocolo OpenFlow pode ser um importante propulsor para o novo conceito de Internet e evolução das redes existentes. O OpenFlow encaixa-se perfeitamente com as redes SDN e como existem diversas comunidades que trabalham com ele, aumentam-se as chances de discussões e novas formas de pensar acerca do futuro da Internet e das redes em geral. Mesmo ainda imaturo em alguns

aspectos, principalmente no desenvolvimento de projetos de grande escala, com o OpenFlow, os caminhos para os fluxos de dados poderão ser mais facilmente determinados com base em políticas: de segurança, de engenharia de tráfego, de controle de acesso até de energia elétrica, dependendo do ambiente e natureza da organização. O OpenFlow permite criar uma rede totalmente programável, baseado no controle de rede logicamente centralizado e definindo um novo elemento na rede, o controlador. Com o OpenFlow é possível a execução de experimentos nas redes que utilizamos todos os dias (MCKEOWN et al., 2008).

No âmbito da criação de novas topologias em ambientes de testes é interessante examinar o funcionamento e desempenho do OpenFlow em máquinas virtuais distribuídas e em máquinas reais distintas. O novo modelo para a evolução ou novos conceitos de redes necessitam ser testadas em escala global, com tráfegos reais e em ambientes de testes controlados. E nestes casos, o OpenFlow juntamente com as redes experimentais e com o auxílio da virtualização, representam-se como uns dos principais recursos para a pesquisa e desenvolvimento deste novo conceito de Internet. A heterogeneidade das novas tecnologias exigem recursos que permitam a discussão de novas ideias, que visem construir soluções inteligentes e adaptativas, e o OpenFlow encaixa-se perfeitamente nessas novas exigências.

De forma a permitir a interoperabilidade entre diferentes ferramentas de autoria e gestão de redes OpenFlow, é apresentada a seguir a *Framework* NSDL.

7 FRAMEWORK NSDL

7.1 INTRODUÇÃO

Este capítulo apresenta a *Framework NSDL*, proposta inicialmente na Universidade da Madeira em Portugal (MARQUES, 2013; MARQUES; SAMPAIO, 2010) com o objetivo de integrar diferentes ferramentas de gestão, simulação, autoria e análise de redes. Ao longo deste capítulo são apresentados os principais aspectos da *Framework NSDL* e uma proposta de atualização para autoria de redes OpenFlow.

7.2 CONCEITOS BÁSICOS

O principal objetivo do *Framework NSDL* é fornecer uma abordagem de integração de ferramentas de redes heterogêneas, a fim de que essas ferramentas possam ser utilizadas de forma complementar, otimizando assim as tarefas de gerenciamento. Esta abordagem se baseia em uma representação comum entre estas ferramentas, intitulado de *Network Scenario Description Language (NSDL)*. O NSDL é uma linguagem baseada em XML que fornece a descrição de uma topologia de rede, as propriedades de seus componentes e informações específicas e que podem ser úteis para essas ferramentas, tais como: localização, simulação, segurança, gerenciamento de dispositivos entre outros.

Segundo Marques (2013), os princípios adotados para o desenvolvimento da linguagem NSDL, foram:

- *Simplicidade*: o que significa que a descrição do NSDL tem que ser simples e transparente. A sua manipulação não deve ser apenas realizada através de ferramentas de gestão de redes, mas também por usuários através de simples editores de texto;
- *Múltiplos níveis de abstração:* para especificações de simples descrições um cenário de rede e, se necessário, para uma descrição detalhada dos objetos que compõem o cenário de rede e seus respectivos parâmetros; e
- *Extensibilidade:* para possibilitar a adição de novos objetos e seus parâmetros em futuras descrições.

Na próxima seção são apresentadas as características da estrutura da *Framework* NSDL.

7.3 ESTRUTURA DA *FRAMEWORK NSDL*

Como já introduzido no Capítulo 1, a *Framework NSDL* é composta por três camadas: a *Camada de Interface Gráfica do Usuário,* a *Linguagem de Descrição de Cenários de Rede – NSDL,* e a *Camada de Implementação de Redes*.

Em particular, as Camadas *de Interface Gráfica do Usuário* e *de Implementação de Redes* agrupam um conjunto de ferramentas de gestão de redes de acordo com as suas funcionalidades, respectivamente, relacionadas aos aspectos de modelagem, visualização e monitoração de redes, e aos aspectos de simulação, validação e implementação/virtualização de redes. Já a camada intermediária (*Linguagem de Descrição de Cenários de Rede – NSDL*) é responsável pelos aspectos de integração, proporcionando a interoperabilidade horizontal e vertical entre as diferentes ferramentas compatíveis com a *Framework*.

A interoperabilidade horizontal é proporcionada entre as ferramentas cujas funcionalidades se encontram na mesma camada. Este tipo de interoperabilidade permite a realização de uma mesma operação ou então funcionalidades complementares utilizando ferramentas distintas. Já a interoperabilidade vertical envolve a utilização de ferramentas com diferentes objetivos, permitindo uma análise da rede em um novo contexto. Havendo uma descrição de um cenário de rede, é possível avaliar a rede em um novo cenário de rede acrescentando ou retirando informações pertinentes às diversas ferramentas.

É importante observar que as camadas superiores e inferiores estão relacionadas com as ferramentas de gerenciamento de redes existentes (ou que ainda serão desenvolvidas), sobretudo, orientado ao domínio de simulação de redes.

A linguagem NSDL é apresentada na próxima seção.

7.4 LINGUAGEM NSDL

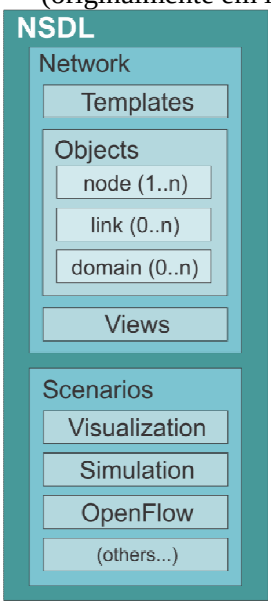
A estrutura da linguagem NSDL fornece características importantes, uma vez que apresenta uma descrição separada dos componentes (por exemplo, a topologia de rede) a partir da descrição do comportamento dos diferentes cenários de rede (por exemplo, o contexto orientado com base em características de domínios ou de ferramentas). Esta separação representa um avanço para a gestão dos cenários de rede uma vez que (i) diferentes perspectivas podem ser descritas com o mesmo cenário de rede e (ii) a descrição da rede pode ser reutilizada dentro de diferentes ferramentas de gestão de redes. Este recurso fornece resultados complementares e análises para a otimização das tarefas de gerenciamento de rede.

O principal objetivo de NSDL é proporcionar uma rica descrição dos objetos de redes e seus parâmetros e também uma descrição dos vários cenários de redes em todo o ciclo de vida da rede. Assim, a estrutura e os parâmetros definidos pelo NSDL também devem ser representativos o suficiente para descrever tipos de rede e permitir a incorporação em suas definições de dados para apoiar futuros objetos e novas definições de redes. Como uma linguagem baseada em XML, o NSDL também utiliza da representatividade e a flexibilidade desta metalinguagem. De fato, o XML fornece uma definição mais clara e possui um conjunto de ferramentas disponíveis (XML, 2013). Além disso, o XML assegura a validade dos princípios NSDL: simplicidade, abstração e extensibilidade.

A linguagem NSDL é composta de um vocabulário e um conjunto de regras capazes de suportar a descrição de redes cabeadas e sem fio, e as informações sobre o contexto (ou domínio), onde essas redes são implantadas ou avaliadas (MARQUES, 2012).

Uma representação NSDL é composta por dois elementos básicos (como representado na Figura 10): *Network* e S*cenarios* (descritos originalmente em inglês). O elemento *Network* compreende a descrição de uma topologia de rede identificando os seus objetos e os seus parâmetros. O elemento S*cenarios* pode conter várias descrições, com referências a um uso específico de uma ferramenta ou um contexto de utilização (domínio), para uma respectiva rede.

Figura 10 - Estrutura da linguagem NSDL (originalmente em inglês)



O elemento *Network* é composto de outros três sub elementos: *Templates*, *Objects* e *Views*. O elemento *Objects* representa o principal componente do NSDL, uma vez que contém a descrição da topologia da rede e de seus componentes. Este elemento é constituído por *node*, *links* e *domain*. Dentro do elemento *node* (que permite descrever um nó como uma estação de trabalho ou um comutador), existem outros três objetos: *interfaces*, *protocols* e *applications*. Esses seis últimos elementos são representativos o suficiente para representar objetos de uma rede. É também importante notar que a linguagem NSDL pode ser estendida de modo a descrever novos objetos ou domínios de rede.

Os elementos *Templates* e *Views* são importantes, mas não são obrigatórios. O elemento *Templates* é importante para simplificar a descrição de objetos semelhantes, permitindo a criação de um objeto modelo e fazendo várias referências a ele na seção de objetos, herdando todas as suas propriedades. O elemento *Views* é um mecanismo para agrupar objetos de rede e pode ser utilizado para fornecer um comportamento ou caracterização de um segmento de rede.

No elemento S*cenarios* são introduzidos outros três elementos: *Visualization*, *Simulation* e *OpenFlow*. O elemento *Visualization* fornece informações adicionais (específicas a uma ferramenta) para enriquecer a descrição da rede, como o posicionamento de objetos no ambiente. Os parâmetros necessários para implementar

uma simulação de rede utilizando uma ferramenta de simulação genérica ou específica é definida no elemento *Simulation*. Por fim, o elemento *OpenFlow* fornece informações sobre as configurações acerca do protocolo OpenFlow, como controladores, comutadores OpenFlow e tabelas de fluxos.

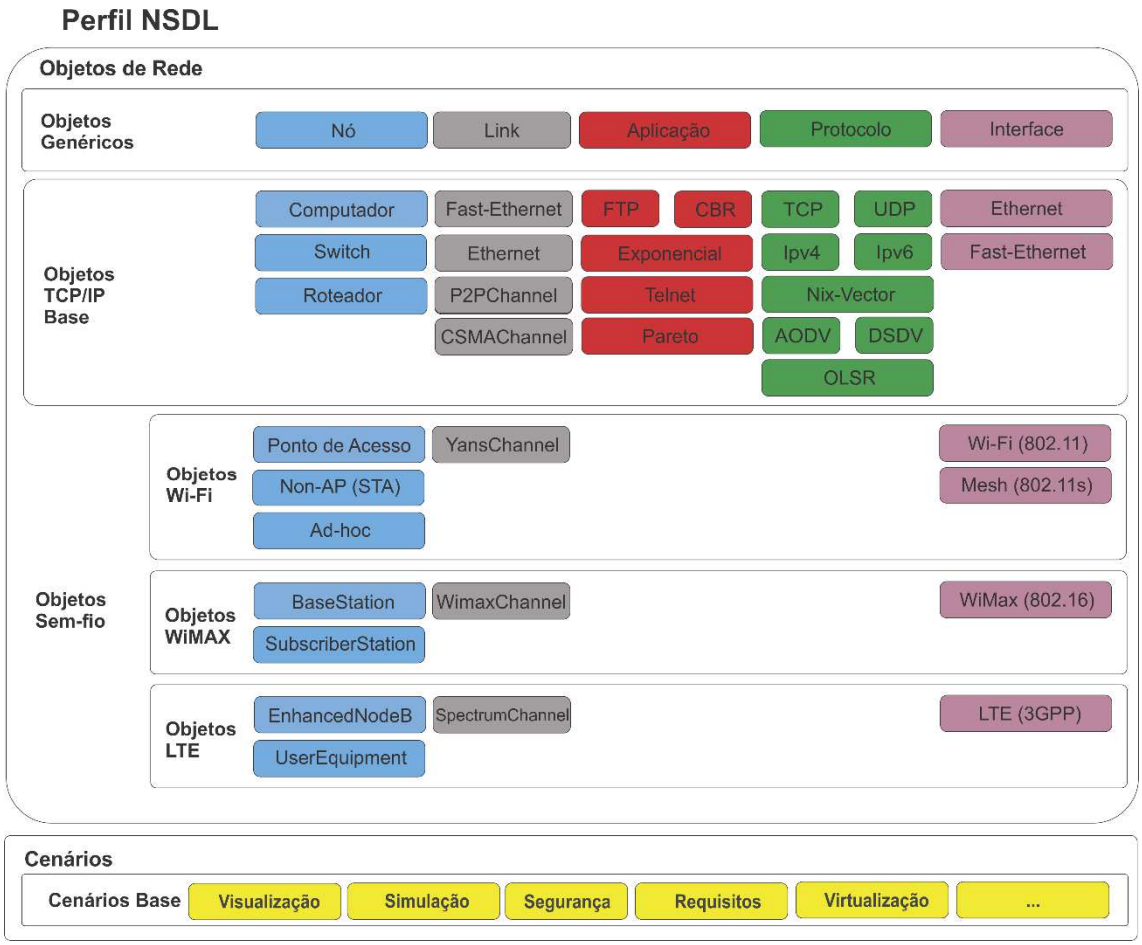
A definição de perfil base é introduzida na próxima seção.

7.5 PERFIL BASE

De forma a proporcionar uma maior organização na criação e instanciação dos objetos utilizados em uma determinada tecnologia ou domínio de redes, o conceito de perfil foi proposto. O perfil para uma determinada tecnologia ou ferramenta promove, sobretudo, a extensibilidade e a reutilização na linguagem NSDL.

Segue a seguir o perfil base do NSDL.

Figura 11 - Perfil base do NSDL

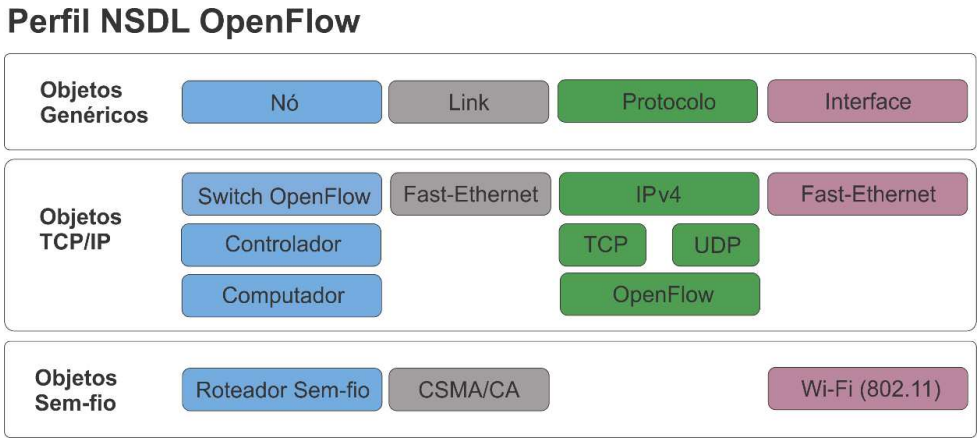


A partir do perfil base, é apresentado a seguir a proposta de perfil NSDL para o protocolo OpenFlow.

7.6 PERFIL NSDL: OPENFLOW

De forma a permitir a descrição de cenários OpenFlow, a linguagem NSDL também teve de ser estendida, através da criação de um perfil NSDL OpenFlow (Figura 12).

Figura 12 - Perfil NSDL OpenFlow



Abaixo é apresentado o perfil NSDL em XML com as adaptações necessárias para o protocolo OpenFlow. De forma a ilustrar o perfil NSDL OpenFlow, considere: a Figura 13 representa a especificação das ligações (objeto *link*); a Figura 14 representa a especificação para um computador; a Figura 15 representa a especificação para um comutador OpenFlow; a Figura16 representa a especificação para um controlador OpenFlow; a Figura 17 representa a especificação para uma tabela de fluxos com base no documento de especificações OpenFlow; a Figura 18 representa a especificação para configuração de filas; e a Figura 19 representa a especificação para configurações de QoS.

Figura 13 - Especificações de um link para o NSDL

```
1. <link id="ethernet_1">
2.     <bandwidth/>
3.     <delay/>
4.     <loss/>
5.     <maxqueue/>
6.     <htb/>
7.     <connection destination="switchOpenflow_2"
   source="computer_1"/>
8. </link>
```

Figura 14 - Especificação de um computador para o NSDL

```
1. <computer id="computer_1">
2.     <computerIPAddress/>
3.     <mask/>
4.     <computerMacAddress/>
5.     <notes/>
6. </computer>
```

Figura 15 - Especificação de um comutador OpenFlow para o NSDL

```
1. <switchOpenflow id="switchOpenflow_1">
2.     <switchMacAddress/>
3.     <notes/>
4. </switchOpenflow>
```

Figura 16 - Especificação de um controlador OpenFlow para o NSDL

```
1. <controllerOpenflow id="controllerOpenflow_1">
2.     <controllerIPAddress/>
3.     <controllerPort/>
4.     <specialController/>
5.     <openflowController/>
6. </controllerOpenflow>
```

Figura 17 - Especificação de tabelas de fluxos para o NSDL

```
1. (…)
2.  <openflow>
3.    <flowTable1 id="controllerOpenflow_1">
4.        <flowName/>
5.        <priority/>
6.        <macSwitch/>
7.        <macSource/>
8.        <macDestination/>
9.        <ingressPort/>
10.       <vlanID/>
11.       <vlanPriority/>
12.       <ethtype/>
13.       <tos/>
14.       <ipSource/>
15.       <ipDestination/>
16.       <sourcePort/>
17.       <destinationPort/>
18.       <setIPSource/>
19.       <setIPDestination/>
20.       <setMACSource/>
21.       <setMACDestination/>
22.       <setSourcePort/>
23.       <setDestinationPort />
24.       <setVlanID/>
25.       <setOutput/>
26.       <setVlanPriority/>
27.       <setStripVlan/>
28.       <setEnqueue/>
29.       <notes/>
30.    </flowTable1>
31.  </openflow>
32. (…)
```

Figura 18 - Especificação de configuração de filas para o NSDL

```
1. (…)
2.  <openflow>
3.    <queueConfiguration0 id="controllerOpenflow_1">
4.    <queueInterface/>
5.    <scenarioMaxRate/>
6.    <scenarioMinRate/>
7.    <queueName/>
8.    <linkMaxRate/>
9.    <linkMinRate/>
10.   </queueConfiguration0>
11.  </openflow>
12. (…)
```

Figura 19- Especificação de configuração de QoS para o NSDL

```
1. (…)
2.  <openflow>
3.     <qosTable0 id="controllerOpenflow_1">
4.     <qosName/>
5.     <sourceIP/>
6.     <destinationIP/>
7.     <ethType/>
8.     <protocol/>
9.     <queue/>
10.    </qosTable0>
11.  </openflow>
12. (…)
```

Como pode ser visto através das Figuras acima, a estrutura do NSDL permite a especificação completa e estruturada de uma rede OpenFlow.

7.7 CONSIDERAÇÕES FINAIS

A utilização de uma única descrição em diversos momentos do ciclo de vida de uma rede pode ser vantajosa. Se os usuários responsáveis por uma rede estiverem familiarizados com o NSDL, eles podem facilmente compreender o estado atual da rede e otimizar a sua gestão. Novas capacidades de integração também podem ser alcançadas

já que os desenvolvedores de uma ferramenta de rede são capazes de adicionar capacidades de importação e exportação para o NSDL, tornando possível a interoperabilidade com outras ferramentas de gestão de redes.

Uma das ferramentas que atualmente são compatíveis com o NSDL é justamente a ferramenta VND (versão SDN), que permite a autoria de redes OpenFlow e que será apresentada no capítulo a seguir.

8 AUTORIA DE CENÁRIOS DE REDES OPENFLOW: ABORDAGEM VND

Neste capítulo são apresentados os principais resultados da contribuição desta dissertação de mestrado, relacionados à extensão da ferramenta *Visual Network Description (VND)* (AZEVEDO, 2010) para o suporte à autoria de experimentos OpenFlow. VND foi inicialmente proposta na Universidade da Madeira (Portugal) como uma interface gráfica para a autoria rápida de cenários de rede para posterior simulação e implementação. A extensão proposta neste trabalho foi denominada de *Visual Network Description – VND (versão SDN)*.

Este capítulo está dividido em três partes: Parte I – Projeto, que traz as principais informações acerca do desenvolvimento do VND (versão SDN); Parte II – Interface Gráfica, que apresenta a interface gráfica do VND (versão SDN); e Parte III – Geração Automática de Scripts, que apresenta como se dá o processo de geração de arquivos NSDL e *scripts* para ferramentas OpenFlow. Por fim, são apresentadas algumas conclusões acerca das contribuições apresentadas ao longo deste capítulo.

8.1 PROJETO

Para que os sistemas tenham sucesso, é necessário que os mesmos proporcionem uma experiência de uso adequada. Para tanto, é importante entender o equilíbrio entre o conjunto de metas de usabilidade e metas de experiência com o usuário, a fim de oferecer compatibilidade nas aplicações de tais metas ao sistema, adequando características que tenham impacto positivo no seu uso (PREECE, 2005).

A engenharia de requisitos é fundamental acerca da definição do que deverá ser a experiência dos usuários do sistema, uma vez que é responsável por entender desejos e necessidades dos clientes, além de verificar a viabilidade, especificar, negociar e validar soluções (PRESSMAN, 2002). Diante desse contexto, destaca-se a elicitação de requisitos, fase crucial do processo, já que é nesse momento que tem o conhecimento do que deve ser entregue ao cliente.

A seguir são apresentados os requisitos funcionais e não funcionais do VND (versão SDN). Para a identificação desses requisitos, foram realizadas pesquisas em outras ferramentas de simulação que possuem características similares ao VND (versão

SDN) e principalmente a ferramenta a qual o VND (versão SDN) foi estendido, o VND proposto por (AZEVEDO, 2010).

8.2 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Os requisitos de software representam descrições dos serviços que devem ser fornecidos pelo sistema e as suas restrições operacionais (SOMMERVILLE, 2007). Em (PFLEEGER, 2004) o requisito de um sistema é descrito como uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar para atingir os seus objetivos, e em (ROBERTSON ; ROBERTSON, 2006) define requisito como algo que o produto tem de fazer ou uma qualidade que ele precisa apresentar.

Com base nessas definições, pode-se dizer que os requisitos de um sistema incluem as especificações dos serviços que o sistema tem de oferecer, restrições de operação e até restrições inerentes ao seu processo de desenvolvimento. Dentre os diferentes tipos de requisitos dentro da engenharia de software, os requisitos de um sistema normalmente são caracterizados como funcionais ou não funcionais (SOMMERVILLE, 2007).

a) Requisitos Funcionais

Os requisitos funcionais tratam dos serviços que o sistema deve prover, descrevendo o que o sistema deve fazer, podendo descrever, ainda, como o sistema deve reagir a entradas específicas, bem como o que o sistema não deve fazer (SOMMERVILLE, 2007).

- O sistema deverá permitir que o cenário construído possa ser salvo para posterior utilização;
- O sistema deverá permitir a criação de cenários através de uma simples ação de *arrastar e soltar*, também conhecido como recurso de *drag-and-drop;*
- Quando um usuário selecionar um objeto ele deve ter destaque através de um realce na área de desenho;
- Os objetos (por exemplo, comutadores, controladores e computadores) devem ser configuráveis;

- Em cada objeto criado, os campos *ID* e *Name* devem ser preenchidos automaticamente. O mesmo vale para outros campos de objetos específicos para que se reduza o tempo de configuração dos cenários por parte dos usuários;
- O usuário pode adicionar, remover e mover os objetos do cenário;
- O usuário pode refazer ou desfazer uma ação;
- O usuário pode aumentar ou diminuir o nível de visualização (*zoom*);
- Os objetos devem ser representados graficamente através de ícones e devidamente nomeados;
- Os objetos de links deverão ajustar-se automaticamente conforme os objetos que a compõe são movidos;
- Após a conexão entre os objetos através de links, deverá aparecer qual a porta/interface do objeto o link está associado;
- A área de configuração de cenário do sistema deve possuir dimensão suficiente para suportar a criação de cenários complexos;
- Cada vez que um usuário seleciona um objeto na lista de objetos, o ícone do objeto deve ser mostrado como forma de facilitar o entendimento do que vai ser inserido no ambiente de construção de cenário, e;
- Deve ser possível a exportação de *scripts* para ferramentas OpenFlow e arquivos NSDL;
- Um cenário salvo para posterior utilização deve ter extensão .xml.

b) Requisitos não Funcionais

Os requisitos não funcionais descrevem restrições sobre os serviços ou funções oferecidas pelo sistema, tendo origem nas necessidades dos usuários, em restrições políticas organizacionais, em restrições de orçamento, em necessidades de interoperabilidade com outros softwares ou hardwares ou em fatores externos como regulamentos e legislações (SOMMERVILLE, 2007). Abaixo são apresentados os requisitos não funcionais do VND (versão *SDN*):

- O sistema deve ser compatível com as seguintes plataformas através da utilização de um ambiente WEB: Linux, Windows e Mac;

- O sistema deve ser fácil de aprender de modo que o usuário possa rapidamente começar a fazer algum trabalho com o mesmo;
- A representação da lista de objetos deve ser feita de forma organizada e fácil de utilizar;
- O sistema deve ter alta disponibilidade. Não menos que 99% em cada mês do ano;
- Deve ser capaz de processar ao menos 100 (cem) requisições simultâneas;
- Todas as variáveis de entrada terão valores padrão e esses valores serão usados sempre que dados de entrada estiverem faltando ou inválidos;
- Deve possuir rápido tempo de resposta. O tempo de geração dos *scripts* não deve ultrapassar cinco segundos;
- O código deve ser devidamente comentado com o propósito de facilitar possíveis manutenções.

Como forma de validar os seus requisitos, incialmente o VND (versão SDN) foi divulgado através de um protótipo na Web, principalmente por dois motivos: para obter retorno dos usuários e validar a proposta acerca do desenvolvimento da ferramenta. Este protótipo estava inicialmente orientado apenas à geração de *scripts* para o Mininet. Para este protótipo foram estabelecidas as seguintes métricas: avaliação dos usuários em relação à usabilidade do software, a quantidade de erros encontrados e tempo de resposta da ferramenta. O retorno dos usuários permitiu identificar, sobretudo, equívocos em nível de programação, que acarretava em erros no funcionamento do VND (versão SDN) em sistemas operacionais *Linux*. Além disso, os *scripts* criados pelos usuários são mantidos no servidor Web onde o VND (versão SDN) é mantido e isso colaborou para também identificar erros nos *scripts* gerados, antecipando, em alguns casos, possíveis retornos dos usuários.

Após o lançamento do protótipo, novos recursos foram implementados, como a geração de arquivos NSDL e *scripts* para controladores OpenFlow, e foi percebido que a quantidade de usuários que acessavam a ferramenta aumentada à medida que os novos recursos foram implementados. Da mesma forma, a quantidade de retornos também aumentava e mesmo sendo subjetiva, os retornos dos usuários foram importantes no processo de desenvolvimento da ferramenta.

Não foram elaborados questionários, mas os usuários enviaram e-mails com elogios e sugestões. Após os e-mails foi possível validar a proposta do VND (versão SDN) e com isso determinar os seus requisitos funcionais e não funcionais, apresentados logo abaixo.

Na próxima seção a arquitetura proposta para a implementação do VND (versão SDN) é apresentada.

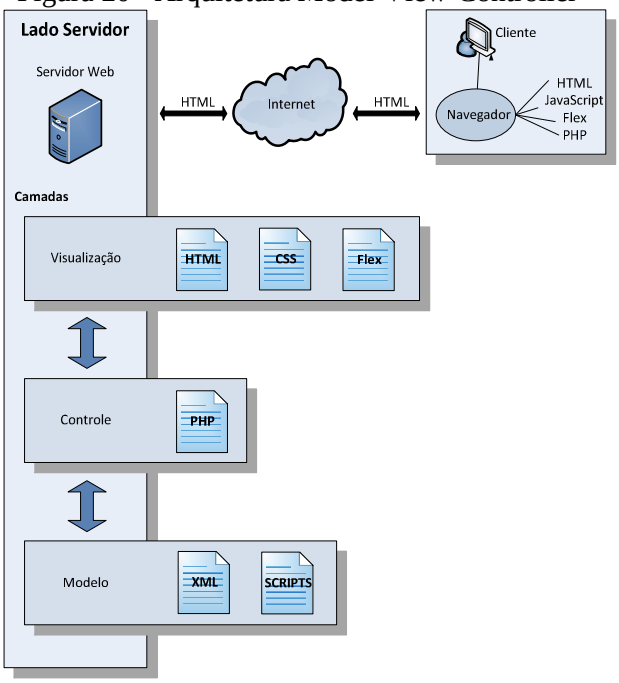
8.3 ARQUITETURA

Para o desenvolvimento do VND (versão SDN) foi adotada a arquitetura MVC – Modelo-Visualização-Controle ou *Model-view-controlller* (GAMMA et al., 1994). As principais vantagens da utilização de MVC são:

- Separação entre os dados (modelo) e a visualização (*layout*);
- Permite que alterações feitas na visualização não afetam a manipulação dos dados, que também podem ser reorganizados sem alterar a visualização;
- Torna a aplicação escalável;

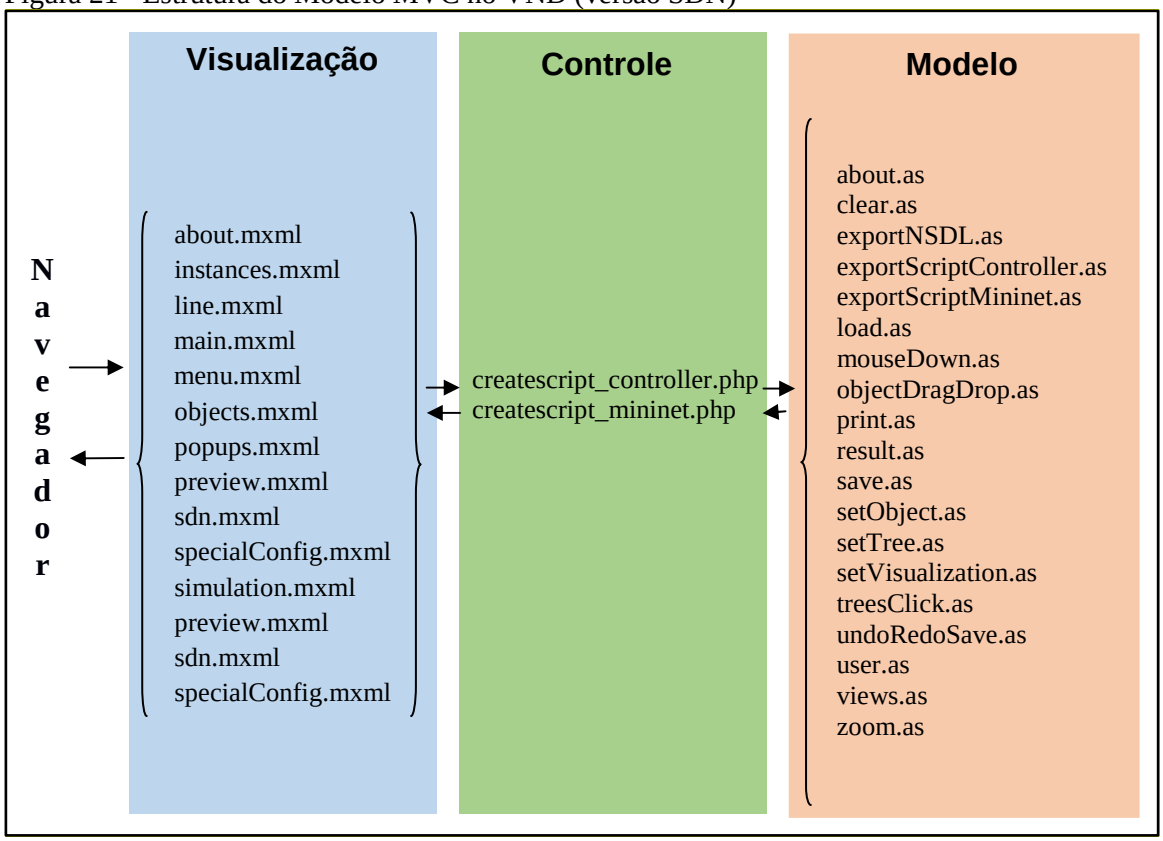
A Figura 20 ilustra a arquitetura MVC e as respectivas tecnologias utilizadas na implementação do VND (versão SDN).

Figura 20 - Arquitetura Model-View-Controller



As três camadas Modelo (em inglês *Model)*, Visualização (em inglês *View)* e Controle (em inglês *Controller)* são, respectivamente, responsáveis por: representação das informações persistentes no sistema e as regras de negócios que governam o acesso e atualizações de dados; acesso dos dados através do modelo e especifica como os dados devem ser representados. Quando o modelo é alterado, a visão é quem fica responsável por manter a consistência na sua apresentação; traduzir as interações com a visão em ações a serem realizadas pelo modelo. Com base na interação do usuário e os resultados das ações do modelo, o controlador responde ao selecionar uma visão adequada. No VND (versão SDN), o modelo MVC está estrutura conforme a Figura 21.

Figura 21 - Estrutura do Modelo MVC no VND (versão SDN)



A estrutura como os arquivos estão disponibilizados na pasta do projeto VND (versão SDN) é discutida próxima seção.

8.4 ESTRUTURA DA INFORMAÇÃO

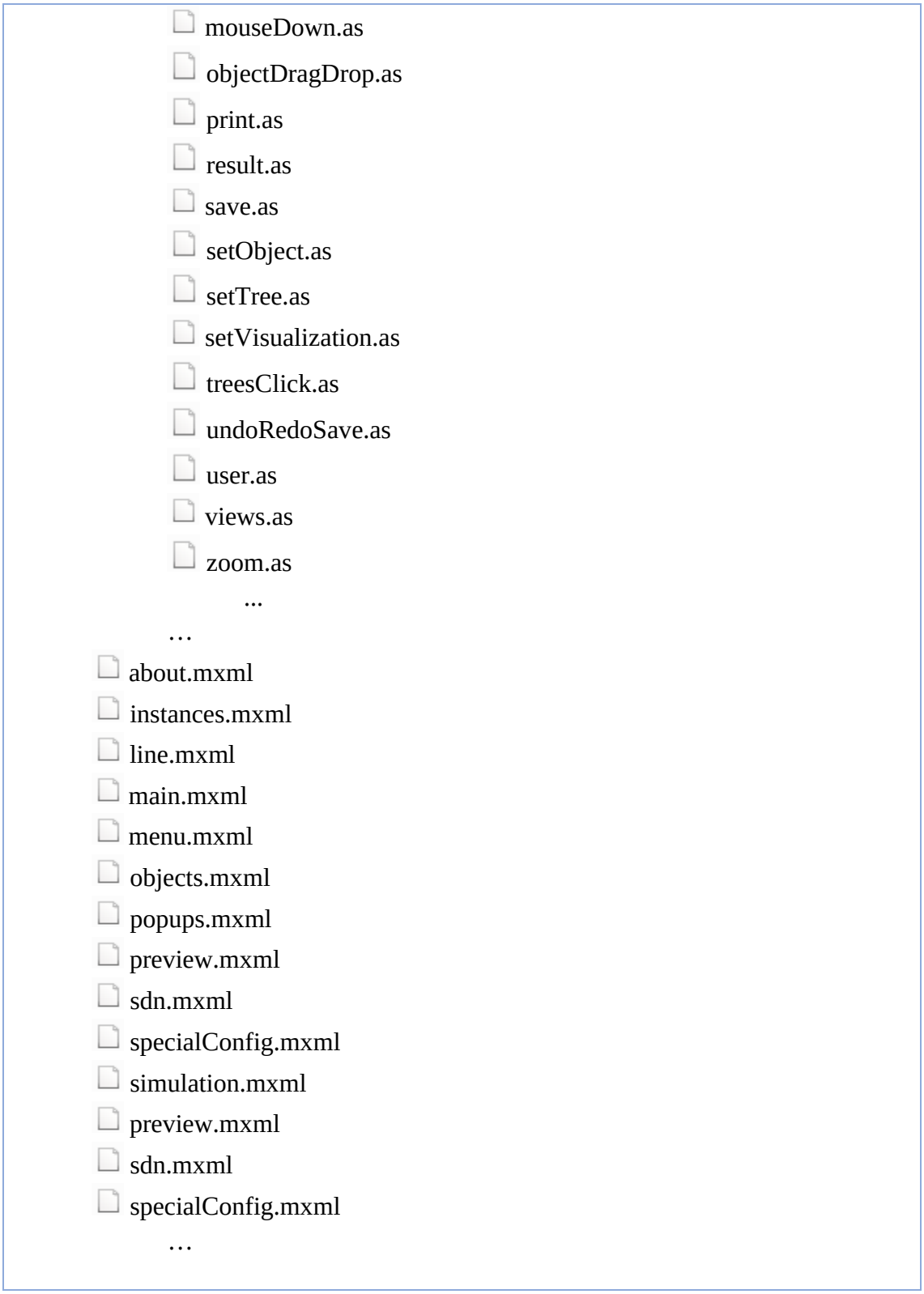
A seguir são apresentados 2 (dois) quadros: O primeiro quadro (Quadro 1) presenta a estrutura do VND (versão SDN) em ambiente de desenvolvimento e o

segundo quadro (Quadro 2) apresenta outras estruturas importantes para o funcionando do VND (versão SDN).

O primeiro quadro (Quadro 1) apresenta 3 (três) diretórios principais: alerts, actionScripts e o diretório raiz. Em alerts estão os arquivos responsáveis por apresentar mensagens popups durante a execução do VND (SDN version); em actionScripts estão uma série de arquivos responsáveis por diversas funções, como: *about* (responsável por apresentar informações), *clear* (responsável por limpar cenários), *exportNSDL* (responsável pela exportação de arquivos NSDL), *exportScriptController* (responsável por executar instruções para exportação de *scripts* para controladores OpenFlow), *exportScriptMininet* (responsável por executar instruções para exportação de *scripts* para o Mininet), *load* (responsável por carregar cenários anteriormente salvos), *mouseDown, objectDragDrop, treesClick, user, views e setObject* (responsável por eventos relacionados ao objeto escolhido pelo usuário), *print* (responsável pela impressão de cenários), *result, setTree* e *setVisualization* (responsável por iniciar o VND (versão SDN), *save* (responsável por salvar cenários), *undoRedoSave* (responsável por avançar ou retroceder uma ação executada pelo usuário) e *zoom* (responsável pelo zoom).

Quadro 1 - Estrutura para desenvolvido do VND

Estrutura para desenvolvimento
alerts ... alert.as messages.as ... actionScripts ... about.as clear.as exportNSDL.as exportScriptController.as exportScriptMininet.as load.as



E na raiz temos outros arquivos importantes e responsáveis pela programação do VND (versão SDN), como: *about* (permite configurar informações do sistema), *instances* (responsável por instâncias de objetos), *line* (responsável por configuração de

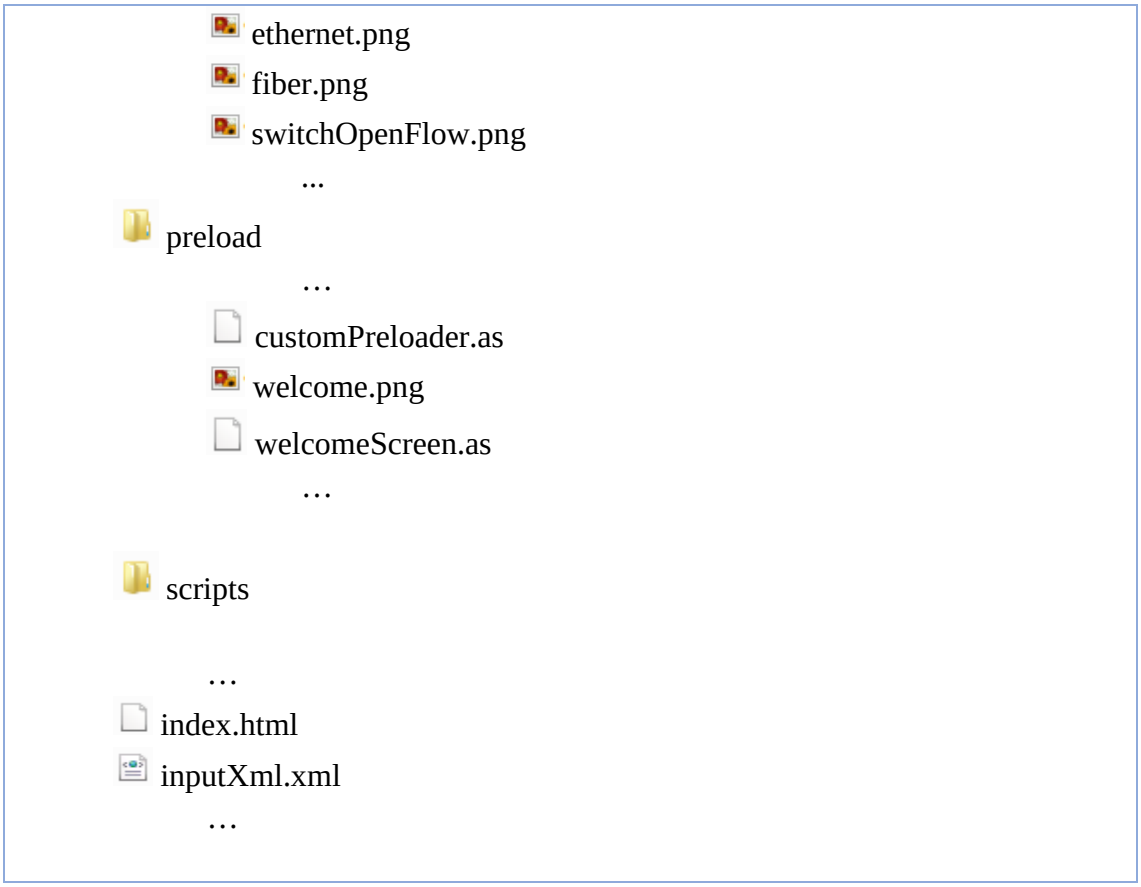
links), *main* (arquivo principal do sistema), *menu* (responsável por configuração do menu do sistema), *objects* (responsável pelas configurações dos objetos), *popups* (responsável pelas configurações de popups), *preview* (responsável por apresentar imagem dos objetos selecionados pelo usuário), *sdn* (responsável por algumas configurações relativas às SDNs) e *specialConfig* (responsável por algumas configurações especiais).

Há também os arquivos *createscript_controller.php e createscript_mininet.php*, que são responsáveis, respectivamente, por: geração de *scripts* para controladores OpenFlow e geração de *scripts* para o emulador Mininet.

Já no Quadro 2 são apresentadas outras estruturas importantes para o funcionamento do VND (SDN version), como: *css* (onde estão presentes as folhas de estilos), *history* (onde estão presentes arquivos com histórico de utilização do sistema), *images* (onde contêm os arquivos de imagem do sistema), *scripts* (onde estarão os arquivos de *scripts* criados pelos usuários), *preload* (onde contêm arquivos para um pré-carregamento do sistema), o arquivo *index.html* (arquivo html principal) e o arquivo *inputXml.xml* (onde contêm informações sobre os objetos do sistema).

Quadro 2 - Arquivos acessíveis pelos usuários

Outras estruturas importantes do sistema
css ... main.css ... history ... history.css history.js ... images ... coomputer.png controllerOpenFlow.png



Uma vez os cenários de rede criados e configurados, a ferramenta VND (versão SDN) pode exportá-los para a linguagem NSDL, visando a interoperabilidade com outras ferramentas compatíveis com essa linguagem, ou então é possível a geração automática de *scripts* para ferramentas OpenFlow para a execução de experimentos diretamente nessas ferramentas.

São apresentadas na próxima seção as linguagens e ferramentas de implementação utilizadas no desenvolvimento do VND (versão SDN).

8.5 LINGUAGENS E FERRAMENTAS DE IMPLEMENTAÇÃO

A ferramenta VND (versão SDN) foi desenvolvida utilizando as linguagens de programação Adobe Flex (FLEX, 2013), PHP (PHP, 2013), Java Script (JAVASCRIPT, 2013) e XML (W3C, 2013). O Flex é uma aplicação de código fonte aberto para o desenvolvimento de aplicações Web; o PHP (Hypertext Preprocessor) é uma linguagem de *script*, software livre, muito utilizada para o desenvolvimento de aplicações Web; o Java Script é uma linguagem de programação também baseada em *scripts* que é

executada ao lado do usuário que utiliza do serviço disponibilizado por um sistema; e o XML é uma linguagem que auxilia na integração, interoperabilidade e publicação de dados na Web.

No VND (versão SDN), o Flex é o responsável direto pela interação com o usuário; o PHP e o Java Script são responsáveis por carregar a biblioteca de objetos, geração de *scripts* e o tratamento (abrir, salvar) de arquivos; e o XML é responsável na leitura e gravação de arquivos. A leitura dos objetos que iniciarão junto ao VND (versão SDN), bem como a lista de configurações disponíveis para cada objeto e exportação para o NSDL são realizadas através do XML.

A próxima parte, a Parte II, apresenta a interface gráfica do VND (versão SDN).

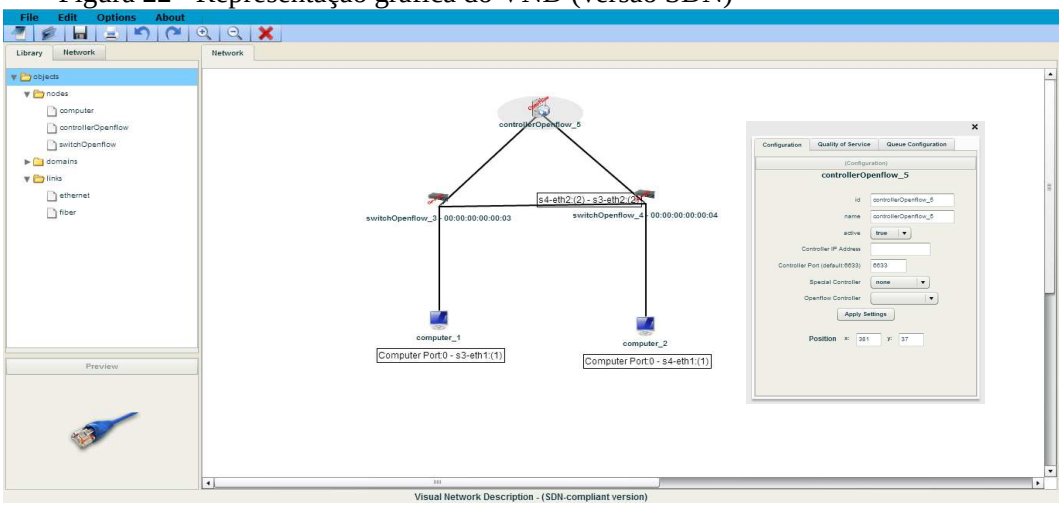
9 INTERFACE GRÁFICA

A seguir é apresentada a interface gráfica do VND (versão SDN), com destaque para as principais telas e seus detalhes de configuração.

9.1 INTERFACE GRÁFICA

A Figura 22 ilustra a interface gráfica do VND (versão SDN). A seguir são discutidas as principais características e funcionalidades introduzidas nessa interface gráfica.

Figura 22 - Representação gráfica do VND (versão SDN)



A interface do VND (versão SDN) foi concebida com uma combinação de cores suaves de forma a proporcionar um ambiente leve e áreas bem definidas para que seu uso seja de fácil entendimento. Possui também uma extensa área para a criação de cenários de rede e as configurações dos objetos que compõem o cenário são acessíveis através de um simples duplo clique.

Através do menu, o usuário têm disponíveis os seguintes recursos:

- Criar novos cenários;
- Abrir um cenário;
- Salvar um cenário;
- Exportar cenários para NSDL;

- Exportar arquivos de *scripts* para o Mininet;
- Exportar arquivos de *scripts* para controladores OpenFlow;
- Exportar arquivos de *scripts* referentes à Qualidade de Serviço;
- Imprimir um cenário;
- Retroceder;
- Avançar;
- Zoom in;
- Zoom out;
- Apagar um cenário;
- Ter informações sobre novidades, e;
- Acesso a vídeos com demonstrações sobre a utilização do VND (versão SDN).

Na Figura 23 é possível visualizar a biblioteca de objetos disponíveis e é possível perceber que a cada objeto selecionado possui uma imagem ilustrativa que é apresentada logo abaixo.

Figura 23 - Navegação da biblioteca

Uma vez os objetos são selecionados da biblioteca e utilizados no cenário de redes em construção, uma lista dos objetos utilizados é construída. A Figura 24 ilustra a lista dos objetos que já foram inseridos no cenário. Essa lista é útil para que o usuário

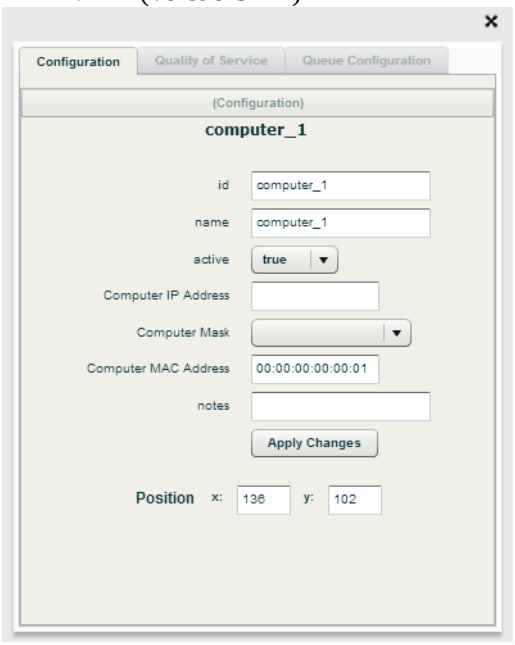
saiba quais são os objetos que irão compor o arquivo NSDL.

Figura 24 - Lista de objetos inseridos no cenário

As Figuras 25, 26, 27, 28, 29, 30 e 31 ilustram as configurações disponíveis para os objetos no VND (versão SDN). Esses objetos são: computador, comutador OpenFlow, controlador OpenFlow, tabela de fluxos, Qualidade de Serviço (QoS), filas e links.

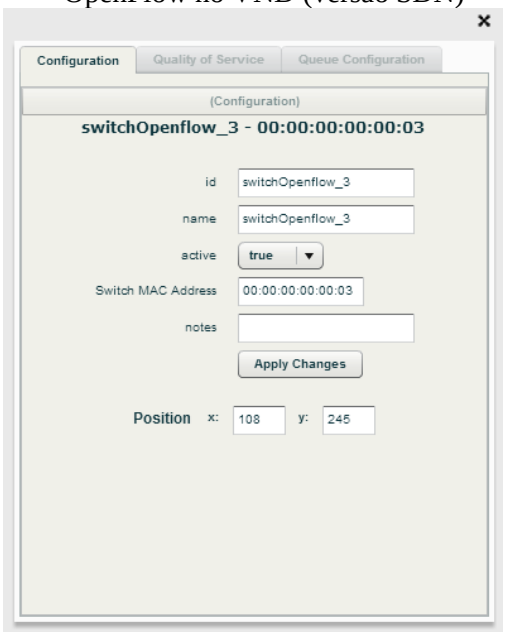
Para um **computador** são possíveis as seguintes configurações (Figura 25): **name** (para definição de nome); **active** (para habilitar/desabilitar); **Computer IP Address** (para definição do endereço IP); **Computer Mask** (para definição da máscara de subrede); **Computer MAC Address** (para definição do endereço MAC) e **notes** (para comentários). Por questões de controle, o campo **id** é bloqueado para configurações.

Figura 25 - Ambiente de configuração de um computador no VND (versão SDN)



Para um comutador OpenFlow são possíveis as seguintes configurações (Figura 26): **name** (para definição de nome); **active** (para habilitar/desabilitar); **Switch MAC Address** (para definição de endereço MAC) e **notes** (para comentários). Por questões de controle, o campo id é bloqueado para configurações.

Figura 26 - Ambiente de configuração de um comutador OpenFlow no VND (versão SDN)



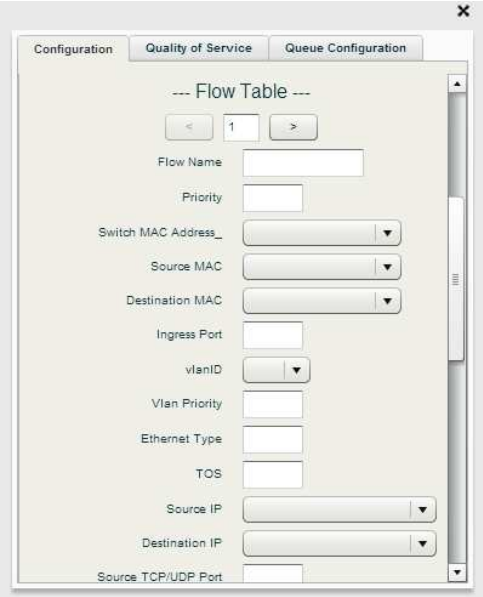
Para um **controlador OpenFlow** são possíveis as seguintes configurações (Figura 27): **name** (para definição de nome); **active** (para habilitar/desabilitar);

Controller IP Address (para definição do endereço IP do controlador); **Controller Port** (para definição da porta lógica do controlador); **Special Controller** (para definição/ou não de um controlador especial) e **OpenFlow Controller** (para definição do controlador OpenFlow). Por questões de controle o campo id é bloqueado para configurações.

Figura 27 - Ambiente de configuração de um controlador OpenFlow no VND
(versão SDN)

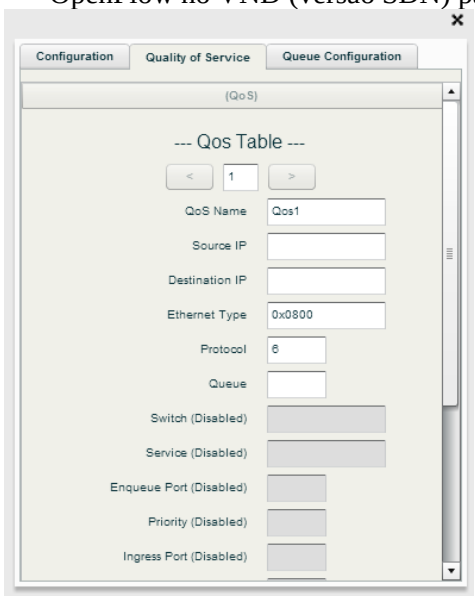
Para a configuração da **tabela de fluxos** estão disponíveis as seguintes configurações (Figura 28): **Flow Name** (para definição de nome); **Priority** (para definição de prioridades); **Swich MAC Address** (para definição de endereço MAC de um comutador); **Source MAC** (para definição de endereço MAC de origem); **Destination MAC** (para definição de endereço MAC de destino); **Ingress Port** (para definição de uma porta de entrada no comutador); **vlanID** (para definição de um Vlan ID); **Vlan Priority** (para definição de prioridades para Vlan); **Ethernet Type** (para definição do ethernet type do pacote OpenFlow payload após uma Vlan tag); **TOS** (Type of Service – para definição do tipo de serviço); **Source IP** (para definição de endereço IP de origem); **Destination IP** (para definição de endereço IP de destino); **Source TCP/UDP Port** (para definição de porta lógica de origem) e **Destination TCP/UDP Port** (para definição de porta lógica de destino). Os demais campos de configurações são similares aos anteriormente mencionados. Com exceção dos campos referentes às ações que dizem respeito ao tratamento dos pacotes de entrada, ou seja, a ação que será dada ao pacote que entrar no comutador.

Figura 28 - Ambiente de configuração de um controlador OpenFlow no VND (versão SDN) para configuração de tabelas de fluxo



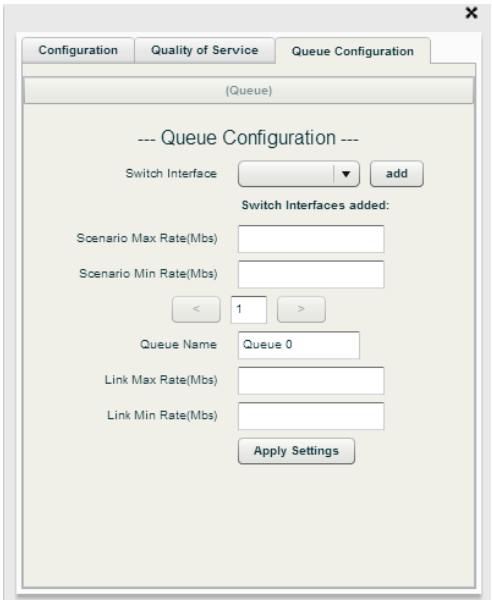
Para a configuração de **QoS** no controlador OpenFlow é possível definir as características dos seguintes parâmetros (Figura 29): **QoS Name** (para definição de nome); **Source IP** (para definição de endereço IP de origem); **Destination IP** (para definição de endereço IP de destino); **Ethernet Type** (para definição do ethernet type do pacote OpenFlow payload após uma Vlan tag); **Protocol** (para definição de protocolo IPv4 ou IPv6) e **Queue** (para definição da fila). As demais configurações estarão sendo disponibilizadas de acordo com as especificações dos controladores ao qual o VND (SDN version) estiver sendo adaptado.

Figura 29 - Ambiente de configuração de um controlador OpenFlow no VND (versão SDN) para configurações de QoS



Para a configuração de **filas** no controlador é possível definir os seguintes parâmetros (Figura 30): **Switch Interface** (para escolha das interfaces do comutador que farão parte da fila a ser configurada); **Scenario Max Rate** (necessário para definição de largura de banda máxima do cenário); **Scenario Min Rate** (necessário para definição de largura de banda mínima do cenário); **Link Max Rate** (para definição de largura de banda máxima do link em fila específica) e **Link Min Rate** (para definição de largura de banda mínima do link em fila específica).

Figura 30 - Ambiente de configuração de um controlador OpenFlow no VND (versão SDN) para configuração de filas



Para configuração de **links** os seguintes parâmetros podem ser definidos (Figura 31): **name** (para definição de nome); **source** (identificação do objeto de origem); **destination** (identificação do objeto de destino); **type** (para definição de link simplex ou duplex); **Bandwidth** (para definição de largura de banda); **Delay** (para definição de atraso); **Loss** (para definição de perdas); **Max Queue Size** (para definição do tamanho máximo da fila) e **Hierarchical Token Bucket** (para ativar ou não o Token Bucket Hierarchical). Por questões de controle o campo **id** é bloqueado para configurações.

Figura 31 - Ambiente de configuração de um link no VND (versão SDN)

A seguir, na Parte III, são apresentados os principais aspectos da geração de arquivos NSDL e scripts para o Mininet e controladores OpenFlow.

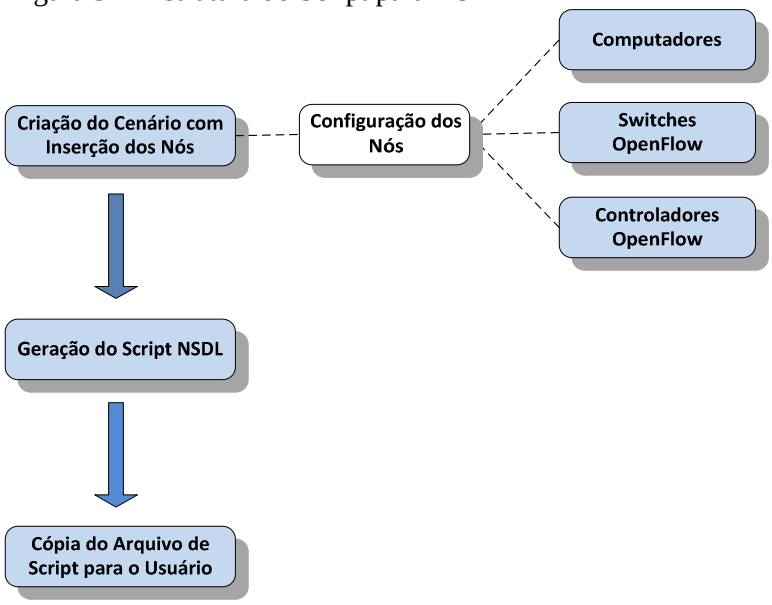
10 GERAÇÃO DE SCRIPTS

A seguir são apresentados os aspectos de configuração para a geração automática de arquivos NSDL, scripts para o emulador Mininet e scripts para controladores OpenFlow.

10.1 GERAÇÃO AUTOMÁTICA DE NSDL

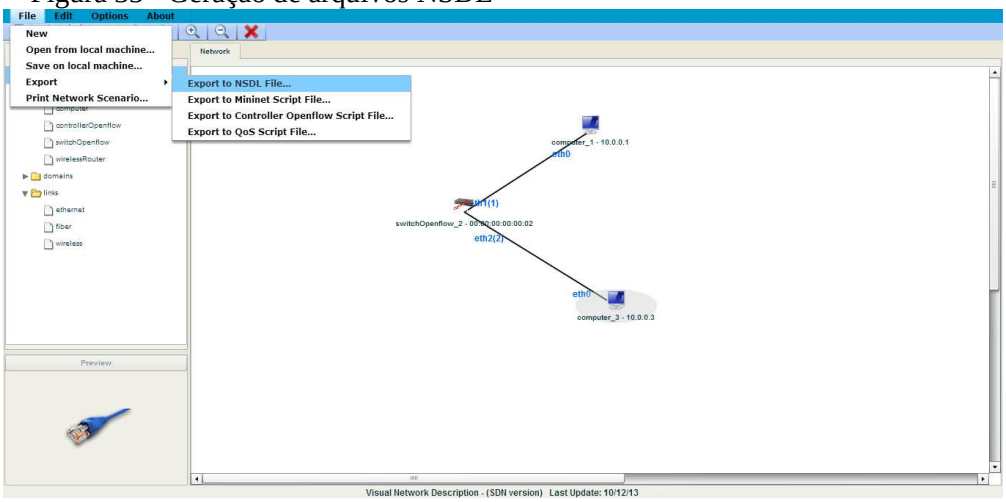
A Figura 32 ilustra os principais passos para a geração de arquivos NSDL. A sua geração inicia a partir da *Criação do Cenário com Inserção do Nós* e neste passo se faz necessária a configuração destes nós. Após esse passo, o usuário deve solicitar a *Geração do Script NSDL* para finalmente obter uma *Cópia do Arquivo de Script*.

Figura 32 - Estrutura do Script para NSDL



A geração automática de arquivos NSDL através do VND (versão SDN) pode ser realizada logo após a criação e configuração dos cenários. Para tanto, basta clicar no menu *File*, *Export* e em seguida *Export to NSDL File*, conforme ilustrado na Figura 33.

Figura 33 - Geração de arquivos NSDL



O conteúdo de um arquivo NSDL gerado pode ser visto logo a seguir, na Figura 34. Através de sua descrição, é possível perceber a presença dos seguintes objetos: duas estações de trabalho (*computer_1* e *computer_3*), um comutador OpenFlow (*switchOpenFlow_2*) e duas conexões Ethernet (*ethernet_4* e *ethernet_5*). Além da presença dos objetos, também é possível perceber que existem algumas configurações, como: definição de endereço MAC no comutador; endereçamento IP e endereço MAC nas estações de trabalho e a distribuição de conexão entre os dispositivos de rede através das conexões Ethernet (exemplo: o cabo *ethernet_5* conecta o *computer_3* ao *switchOpenFlow_2*). Por fim, no campo *visualizations* é possível perceber as localizações *x* e *y* dos objetos no ambiente de configuração de cenários do VND (versão SDN).

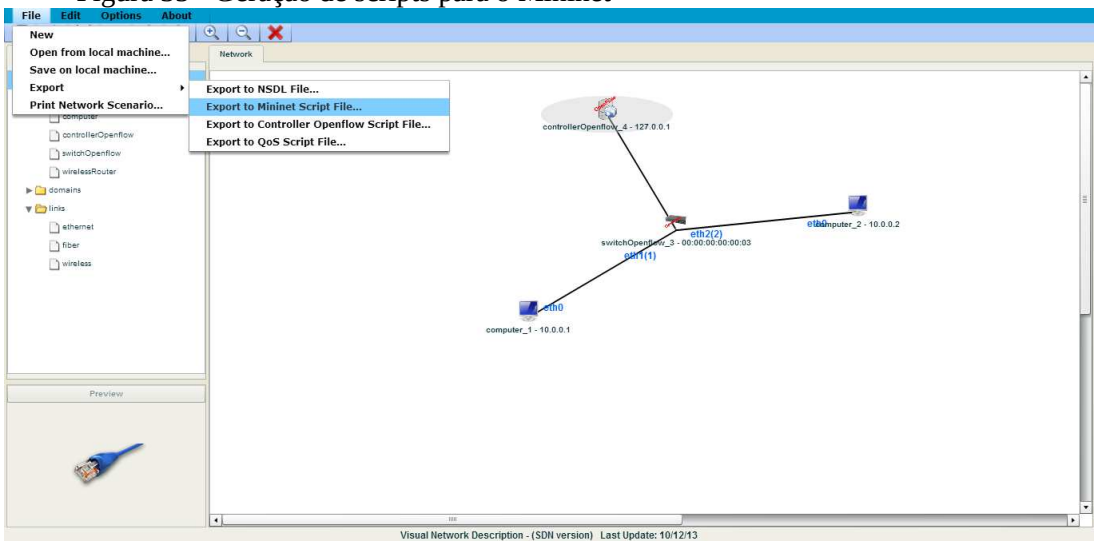
Figura 34 - Trecho de uma descrição NSDL para um cenário criado no VND

```
<?xml version="1.0"?>
- <nsdl>
  - <network>
      <templates/>
    - <objects>
      - <link id="ethernet_5">
          <delay/>
          <loss/>
          <maxqueue/>
          <connection destination="computer_3" source="switchOpenflow_2"/>
        </link>
      - <link id="ethernet_4">
          <delay/>
          <loss/>
          <maxqueue/>
          <connection destination="switchOpenflow_2" source="computer_1"/>
        </link>
      - <computer id="computer_1">
          <computerIPAddress>10.0.0.1</computerIPAddress>
          <mask>255.0.0.0</mask>
          <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
          <notes/>
        </computer>
      - <switchOpenflow id="switchOpenflow_2">
          <switchMacAddress>00:00:00:00:00:02</switchMacAddress>
          <notes/>
        </switchOpenflow>
      - <computer id="computer_3">
          <computerIPAddress>10.0.0.3</computerIPAddress>
          <mask>255.0.0.0</mask>
          <computerMacAddress>00:00:00:00:00:03</computerMacAddress>
          <notes/>
        </computer>
      </objects>
      <views/>
    </network>
  - <scenarios>
    - <visualizations>
      - <visualization id="vis01">
        - <description>
          - <object id="computer_1">
              <x.position>152</x.position>
              <y.position>46</y.position>
              <z.position>0</z.position>
              <color/>
            </object>
          - <object id="switchOpenflow_2">
              <x.position>221</x.position>
              <y.position>124</y.position>
              <z.position>0</z.position>
              <color/>
            </object>
          - <object id="computer_3">
              <x.position>348</x.position>
              <y.position>274</y.position>
              <z.position>0</z.position>
              <color/>
            </object>
          </description>
        </visualization>
      </visualizations>
      <simulations/>
      <openflow/>
    </scenarios>
</nsdl>
```

10.2 GERAÇÃO DE SCRIPTS MININET

A geração de *scripts* para o Mininet é uma tarefa muito simples na interface do VND. Como ilustrado na Figura 35, após a criação do cenário, basta o usuário escolher o menu *File*, *Export* e em seguida *Export to Mininet Script File*.

Figura 35 - Geração de scripts para o Mininet



Para a geração de *scripts* para o Mininet, foi necessário construir uma estrutura "*template*" de uma função em Python. Essa função é alimentada de acordo com o cenário que o usuário cria e exporta no VND (SDN version). A seguir é apresentado um exemplo dessa função com alguns comentários.

```
# -- Executa o Python --
#!/usr/bin/python

# -- Comentários --
"""
Script criado pelo VND - Visual Network Description
"""

# -- Carrega módulos responsáveis por executar o Mininet --
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
```

```
from mininet.link import Link, TCLink
# -- Define uma função chamada 'topology' --
def topology():

    # -- Comentários --
    "Criando a rede..."

    # -- Declara uma variável 'net' --
    net = Mininet(controller=Controller, link=TCLink, switch=OVSKernelSwitch)

    # -- Comentários --
    print "*** Criando nós..."

    # -- Definindo os hosts e seus atributos --
    h1 = net.addHost( 'h1', mac='00:00:00:00:00:01', ip='10.0.0.1/8' )
    h2 = net.addHost( 'h2', mac='00:00:00:00:00:02', ip='10.0.0.2/8' )

    # -- Definindo o comutador e seus atributos --
    s3 = net.addSwitch( 's3', mac='00:00:00:00:00:03')

    # -- Definindo o controlador e seus atributos --
    c4 = net.addController( 'c4', controller=RemoteController, defaultIP="127.0.0.1", port=6633)

    # -- Imprime informação na tela --
    print "*** Criando links..."

    # -- Adiciona o link entre os objetos. A primeira coluna (ex: s3) refere-se ao objeto de origem; a segunda coluna (ex: h2) refere-se ao objeto de destino; a terceira coluna (ex: 2) refere-se à interface do objeto de origem; a quarta coluna (ex: bw=10) refere-se à largura de banda do link; a quinta coluna (ex: delay='5ms') refere-se ao delay do link; a sexta coluna (ex: max_queue_size=1000) refere-se ao tamanho máximo da fila no link; a sétima coluna (ex: loss=10 – em porcentagem) refere-se à quantidade de perdas aceitáveis no link; e a última coluna (ex: use_htb=True) refere-se à utilização ou não de Token Bucket hierárquico –
    net.addLink(s3, h2, 2, 0, bw=10, delay='5ms', max_queue_size=1000, loss=10, use_htb=True)
    net.addLink(s3, h1, 1, 0)

    # -- Imprime informação na tela --
    print "*** Iniciando a rede"

    # -- Constrói o cenário --
```

```
    net.build()

    # -- Vincula o comutador s3 ao controlador c4 --
    s3.start( [c4] )

    # -- Inicia o controlador c4 --
    c4.start()

    # -- Imprime informação na tela --
    print "*** Executando CLI"

    # -- Execução das instruções na CLI --
    CLI( net )

    # -- Imprime informação na tela --
    print "*** Parando a rede"

    # -- Interrompe a construção do cenário --
    net.stop()

# -- Responsável por chamar a função topology() --
if __name__ == '__main__':
    setLogLevel( 'info' )
    topology()
```

10.3 GERAÇÃO DE SCRIPTS PARA CONTROLADORES OPENFLOW

A geração de *scripts* para controladores OpenFlow também é possível graças à definição de outro "*template*" de código pré-definido em Python e que é alimentado de acordo com o cenário criado pelo usuário. A seguir são apresentados 2 (dois) trechos de *scripts*: a) um para o controlador **Pox**; b) e outro para o controlador **Floodlight**.

a) Script – Controlador Pox

```
# -- Executa o Python --
#!/usr/bin/python

# -- Comentários --
#Script criado pelo VND - Visual Network Description

# -- Carrega módulos/bibliotecas responsáveis por executar o controlador POX
from pox.core import core
```

```
from pox.lib.addresses import IPAddr
from pox.lib.addresses import EthAddr
import pox.openflow.libopenflow_01 as of
log = core.getLogger()

# flow0:
# -- Definição de DPID --
switch0 = 0000000000000005
# -- Carrega módulo em flow0msg --
flow0msg = of.ofp_flow_mod()
# -- Definição de cookie --
flow0msg.cookie = 0
# -- Definição de prioridade --
flow0msg.priority = 32768
# -- Definição de porta de entrada --
flow0msg.match.in_port = 1
# ACTIONS--------------------------------
# -- Definição de Vlan ID --
flow0vlan_id = of.ofp_action_vlan_vid (vlan_vid = 10)
# -- Definição de porta de saída --
flow0out = of.ofp_action_output (port = 3)
# -- Carrega as ações previamente definidas --
flow0msg.actions = [flow0vlan_id, flow0out]

(…)

# -- Função para instalação dos fluxos --
def install_flows():
  log.info("    *** Instalando fluxos estáticos.. ***")
  # Instalando fluxos nos comutadores
  core.openflow.sendToDPID(switch0, flow0msg)
  (…)
  log.info("    *** Fluxos estáticos instalados. ***")

# -- Função para iniciar script --
def launch ():
  log.info("*** Iniciando... ***")
  core.callDelayed (15, install_flows)
  log.info("*** Esperando a conexão dos comutadores.. ***")
```

b) Script – Controlador Floodlight

```
# -- Importação de bibliotecas --
```

```
import httplib
import json

# -- Classe para instalação dos fluxos --
class StaticFlowPusher(object):

    def __init__(self, server):
        self.server = server

    def get(self, data):
        ret = self.rest_call({}, 'GET')
        return json.loads(ret[2])

    def set(self, data):
        ret = self.rest_call(data, 'POST')
        return ret[0] == 200

    def remove(self, objtype, data):
        ret = self.rest_call(data, 'DELETE')
        return ret[0] == 200

    def rest_call(self, data, action):
        path = '/wm/staticflowentrypusher/json'
        headers = {
            'Content-type': 'application/json',
            'Accept': 'application/json',
            }
        body = json.dumps(data)
        conn = httplib.HTTPConnection(self.server, 8080)
        conn.request(action, path, body, headers)
        response = conn.getresponse()
        ret = (response.status, response.reason, response.read())
        print ret
        conn.close()
        return ret

# -- Definição de endereço do controlador --
pusher = StaticFlowPusher('192.168.1.66')

# -- Definição de um fluxo --
flow1 = {
    # -- DPID do comutador --
    'switch':"00:00:00:00:00:00:00:03",
    # -- Nome do fluxo --
```

```
"name":"flow-1",
# -- Cookie --

"cookie":"0",
# -- Prioridade --
"priority":"32768",
# -- Interface de entrada no comutador --
"ingress-port":"1",
# -- Ativar/ou não um fluxo --
"active":"true",
# -- Ações executadas em relação aos fluxos de entrada --
"actions":"set-vlan-id=10,output=3"
}

(…)

# -- Inicia um fluxo --
pusher.set(flow1)
(…)
```

10.4 CONSIDERAÇÕES FINAIS

Neste capítulo foram abordados os principais aspectos sobre a implementação do VND (versão SDN), os seus requisitos funcionais e não-funcionais, a sua arquitetura, interface gráfica e seus principais componentes, sua organização estrutural e informações importantes acerca da geração automática de arquivos NSDL e *scripts* para ferramentas OpenFlow, como o *Mininet* e *Controladores*.

A geração automática de arquivos NSDL através do VND (versão SDN) possibilita a interoperabilidade com ferramentas de simulação ou gestão de redes que sejam capazes de interpretar a linguagem NSDL. Alguns projetos já foram desenvolvidos no âmbito da *Framework* NSDL, a exemplo do Visual Network Simulator – VNS (PLÁCIDO, 2010), NSDL perfil NS-2 (MARQUES, 2010) e a Virtualização Automática de Cenários de Rede (ARAÚJO, 2011). O VND (versão SDN), por sua vez, dispensa o usuário da exigência do conhecimento sobre diversas linguagens de programação para a configuração de cenários OpenFlow, e uma vez que esta ferramenta oferece um ambiente de configuração próximo às redes tradicionais, o VND (versão SDN) também é considerado como um ambiente de aprendizado para os usuários sem experiência em SDN.

Para atender às expectativas dos usuários, o VND (versão SDN) tem passado por contínuas atualizações de forma a melhorar a experiência dos usuários e a sua usabilidade. Alguns dos grandes desafios do VND (versão SDN) é acompanhar as evoluções que as ferramentas OpenFlow vêm passando e proporcionar suporte a outras ferramentas OpenFlow citadas ao longo deste trabalho, além das atualmente suportadas.

Como forma de elucidar as características do VND (versão SDN) que foram apresentadas ao longo deste capítulo, o próximo capítulo traz alguns estudos de casos apresentando, sobretudo, o funcionamento do VND (versão SDN) e as etapas necessárias para a autoria de cenários de redes OpenFlow.

11 ESTUDO DE CASO

11.1 INTRODUÇÃO

Neste capítulo são apresentados alguns estudos de casos que permitem ilustrar o funcionamento do VND (versão SDN) na autoria de cenários OpenFlow e sua respectiva geração automática de *scripts* e também de descrições NSDL. Para facilitar o entendimento sobre o funcionamento do VND (versão SDN) e as características do protocolo OpenFlow, o nível de dificuldade dos cenários apresentados neste capítulo foi acrescido de forma gradativa. Como forma de expor as potencialidades do VND (versão SDN) e também do protocolo OpenFlow, os cenários também foram constituídos de forma que atendessem a diferentes aspectos, tais como: simples configuração de fluxos para o encaminhamento de pacotes; segmentação da rede através de Vlans; diferenciação de fluxos através de Qualidade de Serviço (QoS); e configuração de rede sem fio.

A demonstração de estudos de casos com implementações de diferentes aspectos acerca do protocolo OpenFlow é relevante para a validação da proposta do VND (versão SDN). Entende-se nesse trabalho que essa é a melhor alternativa para demonstrar as suas potencialidades e justificá-las através do êxito nas suas implementações. Os estudos de casos permitirão através da utilização de diferentes controladores OpenFlow, sobretudo, perceber que o usuário não terá que conhecer as linguagens de programação desses controladores. Neste sentido, caso o usuário desejar trabalhar com mais de uma ferramenta OpenFlow (incluindo controladores e emuladores), ele não terá que constituir/configurar o cenário de rede nessas diferentes ferramentas. Uma vez constituído o cenário no VND (versão SDN), o usuário poderá escolher qual ferramenta deseja trabalhar, permitindo concentrar seus esforços apenas no protocolo OpenFlow e não nas linguagens de programação dessas ferramentas.

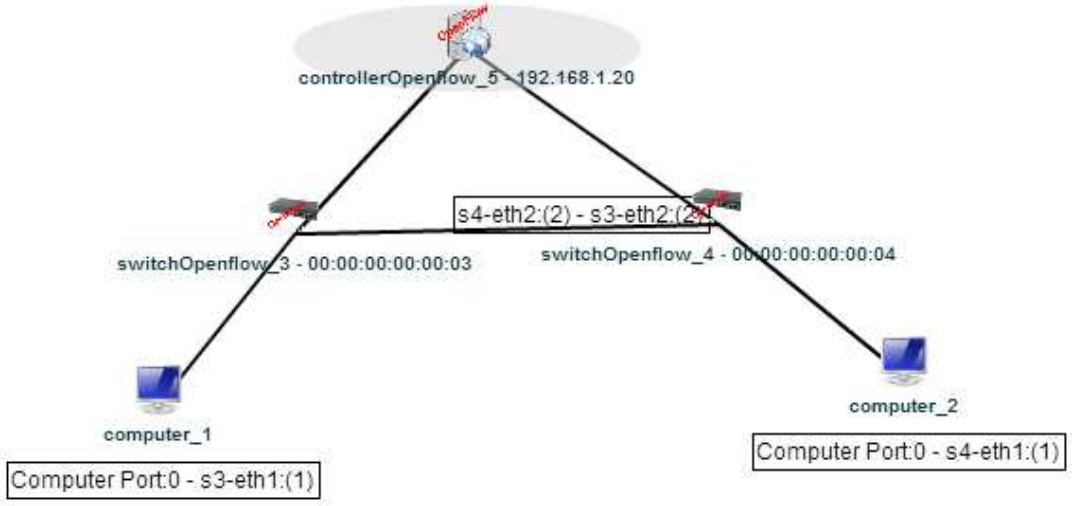
Desta forma, nas próximas seções são apresentados quatro estudos de casos ilustrando algumas das características principais do OpenFlow.

11.2 CENÁRIO NO. 01 – INICIANDO NA UTILIZAÇÃO DO VND (VERSÃO SDN)

O cenário No. 01 têm o objetivo de ilustrar o início de utilização do VND (versão SDN) e apresentar algumas características acerca do funcionamento do protocolo OpenFlow. Neste cenário, o VND (versão SDN) é utilizado para a criação do cenário e para a geração de *script* a ser executado no Mininet. Através dos fluxos de pacotes gerados pelo Mininet será possível visualizar algumas informações contidas nos cabeçalhos desses pacotes e verificar a presença do protocolo OpenFlow na rede.

Como é possível verificar na Figura 36, o cenário No. 01 é composto por dois comutadores Openflow (*Open vSwitch*), um controlador Openflow (*Floodlight*) e duas estações de trabalho (*hosts*).

Figura 36 - Topologia do cenário No. 01 criado no VND (SDN version)



A criação da topologia no VND (versão SDN) é uma tarefa muito simples. Basta o usuário escolher e arrastar os dispositivos para o ambiente de configuração de cenários e conecta-los através dos meios físicos (cabos) disponíveis. Após a criação da topologia, o usuário deve configurar os dispositivos de acordo com as suas necessidades. De acordo com a topologia do cenário No. 01, o usuário pode seguir os seguintes passos para a sua configuração:

1) Inserir os dispositivos de rede no cenário (não é relevante a ordem ao qual os dispositivos são inseridos);

2) Conectá-los através dos cabos disponíveis;

3) Endereçar o computador e o controlador (para este último também se faz necessária a definição do software responsável pelo controle da rede);

4) Para a configuração de dispositivos de rede, basta que o usuário selecione e clique duas vezes sobre o dispositivo que deseja configurar.

Para maiores detalhes da configuração de cada objeto do cenário, o usuário deverá fazer referência às Figuras 25, 26, 27, 28, 29, 30 e 31 do Capítulo V.

Para a validação do cenário No. 01, o *script* a seguir foi gerado automaticamente através do VND (versão SDN) para ser executado no Mininet.

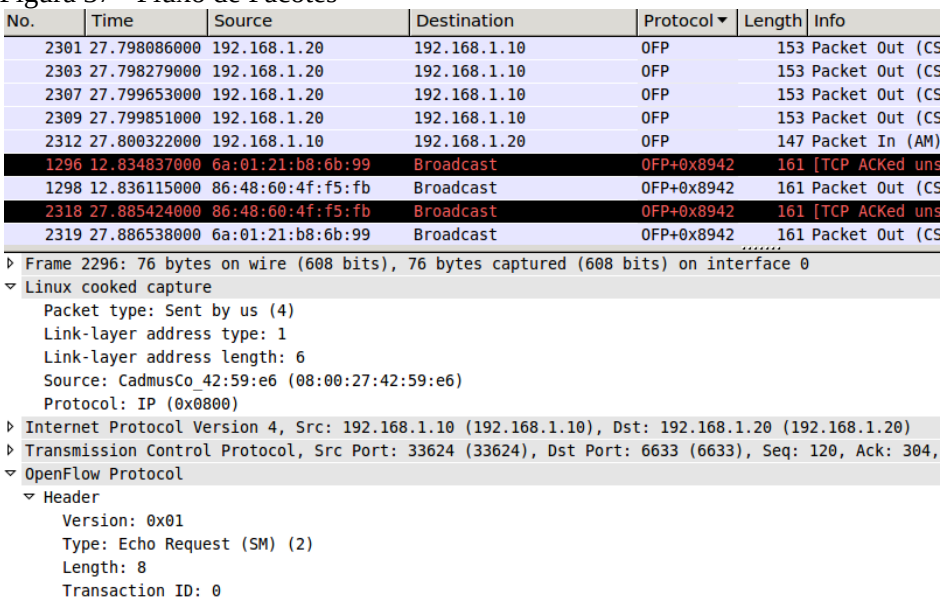
```
#!/usr/bin/python
"""
Script criado pelo VND - Visual Network Description (versão SDN)
"""
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink

def topology():
    "Criando a rede."
    net = Mininet( controller=Controller, link=TCLink,
switch=OVSKernelSwitch )

    print "*** Criando os nós"
    h1 = net.addHost( 'h1', mac='00:00:00:00:00:01', ip='10.0.0.10/8' )
    h2 = net.addHost( 'h2', mac='00:00:00:00:00:02', ip='10.0.0.20/8' )
    s3 = net.addSwitch( 's3', mac='00:00:00:00:00:03')
    s4 = net.addSwitch( 's4', mac='00:00:00:00:00:04')
    c5 = net.addController( 'c5', controller=RemoteController,
defaultIP="192.168.1.20", port=6633)

    print "*** Criando os links entre os nós"
    net.addLink(s3, s4, 2, 2)
    net.addLink(s4, h2,1, 0)
    net.addLink(s3, h1, 1, 0)

    print "*** Iniciando a rede"
    net.build()
```

```
    s3.start( [c5] )
    s4.start( [c5] )
    c5.start()

    print "*** Executando CLI"
    CLI( net )

    print "*** Parando a rede"
    net.stop()

if __name__ == '__main__':
   setLogLevel( 'info' )
   topology()
```

Após a execução do *script* no Mininet, as estações de trabalho ainda não poderão se comunicar, pois os comutadores (*switches*) não saberão o que fazer com os pacotes que receberem. Por isso, em um ambiente OpenFlow faz-se necessária a presença de um controlador para ditar o comportamento do comutador em relação aos pacotes que o mesmo receber.

Para o cenário No. 01, será utilizado o controlador *Floodlight* (FLOODLIGHT, 2013) através da execução do comando *‘java -jar target/floodlight.jar’*. Este comando deve ser inserido em um terminal linux e instrui ao comutador *OpenFlow* a atualizar a sua tabela CAM (*Content Addressable Memory*) - também chamada de tabela de endereços MAC ou tabela de comutação - à medida em que os pacotes trafegam por ele, possibilitando a comunicação entre as estações de trabalho.

Figura 37 - Fluxo de Pacotes



Utilizando a ferramenta Wireshark (WIRESHARK, 2013) é possível verificar a presença do protocolo OFP (Protocolo OpenFlow) nas comunicações realizadas entre o comutador OpenFlow e o Controlador. O Wireshark é um analisador de protocolos que permite a captura de pacotes em tempo real, a Figura 37 ilustra alguns desses pacotes. Nesse cenário, o Mininet permite a emulação de uma estação de trabalho em uma máquina virtual com endereço IP 192.168.1.10 e a execução do controlador em outra máquina virtual com endereço IP 192.168.1.20.

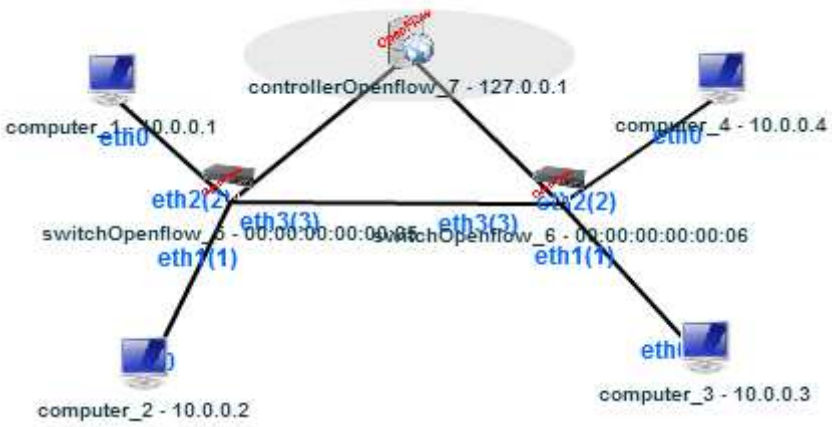
O desenvolvimento deste estudo de caso mostrou os passos necessários para sua configuração no VND (versão SDN), além de algumas informações acerca do funcionamento do protocolo OpenFlow. Para a criação de uma cenário de redes com o protocolo OpenFlow o usuário pode utilizar o VND (versão SDN), posteriormente ele pode dispor da geração automática de *scripts* do VND para que o cenário criado possa ser executado no emulador Mininet. Após a sua execução, através de um teste de conectividade entre as estações, foi possível perceber a presença do protocolo OpenFlow na rede. À medida que as estações se comunicavam, os comutadores trocavam informações com o controlador para obterem informações em prol do devido tratamento dos pacotes que recebiam. Neste cenário, o controlador atuou apenas como um simples dispositivo de camada 2, atualizando a sua tabela de comutação à medida que os pacotes trafegavam na rede.

A descrição NSDL deste cenário No. 01 é apresentado no Anexo A.

11.3 CENÁRIO NO. 02 – SEGMENTAÇÃO EM VLANS

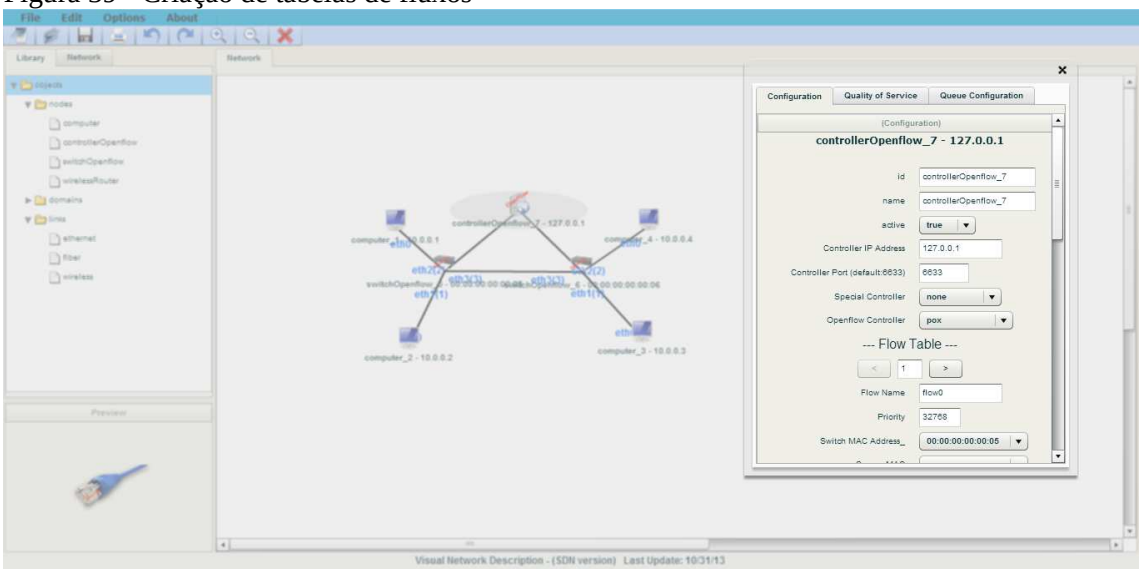
O cenário No. 02 visa ilustrar como pode ser realizada a configuração de redes virtuais locais (Vlans) em uma rede OpenFlow. A proposta deste cenário é segmentar a rede através de Vlans, onde as estações *computer_1* e *computer_3* estarão em uma Vlan ID 10 e as estações *computer_2* e *computer_4* em uma Vlan ID 20. Como não há configuração de encaminhamento entre Vlans, não haverá comunicação entre estações de trabalho de Vlans distintas. O cenário No. 02 é composto por quatro estações de trabalho, um controlador e dois comutadores OpenFlow (Figura 38).

Figura 38 - Topologia com separação de Vlans criado através do VND (versão SDN)



Para a configuração de Vlans é importante criar tabelas de fluxos no *controlador* para determinar o que será feito com os pacotes que entrarem e saírem por determinadas portas de comunicação no *comutador* OpenFlow. A Figura 39 ilustra a janela de configuração responsável por criar as tabelas de fluxos no *controlador*.

Figura 39 - Criação de tabelas de fluxos



Após a configuração da topologia do cenário No. 02, também é possível a geração automática de *script* através do VND (versão SDN) a ser executado no Mininet. Segue abaixo:

```
#!/usr/bin/python
"""
Script criado pelo VND - Visual Network Description (versão SDN)
"""
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink

def topology():
    "Criando a rede."
    net = Mininet( controller=Controller, link=TCLink,
switch=OVSKernelSwitch )

    print "*** Criando nós"
    h1 = net.addHost( 'h1', mac='00:00:00:00:00:01', ip='10.0.0.10/8' )
    h2 = net.addHost( 'h2', mac='00:00:00:00:00:02', ip='10.0.0.20/8' )
    h3 = net.addHost( 'h3', mac='00:00:00:00:00:03', ip='10.0.0.30/8' )
    h4 = net.addHost( 'h4', mac='00:00:00:00:00:04', ip='10.0.0.40/8' )
    s5 = net.addSwitch( 's5', mac='00:00:00:00:00:05')
    s6 = net.addSwitch( 's6', mac='00:00:00:00:00:06')
    c7 = net.addController( 'c7', controller=RemoteController,
defaultIP="127.0.0.1", port=6633)
```

```
    print "*** Criando Links"
    net.addLink(s5, s6, 3, 3)
    net.addLink(s6, h4, 2, 0)
    net.addLink(s6, h3, 1, 0)
    net.addLink(s5, h1, 2, 0)
    net.addLink(h2, s5, 0, 1)

    print "*** Iniciando a rede."
    net.build()
    s6.start( [c7] )
    s5.start( [c7] )
    c7.start()

    print "*** Executando CLI"
    CLI( net )

    print "*** Parando a rede"
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    topology()
```

Uma vez criado o cenário ou ambiente de simulação, falta apenas gerar o *script* para o controlador de preferência do usuário e executá-lo. O trecho de *script* a seguir foi gerado para o controlador Pox (POX, 2013).

```
#!/usr/bin/python
#Script criado pelo VND - Visual Network Description (versão SDN)

from pox.core import core
from pox.lib.addresses import IPAddr
from pox.lib.addresses import EthAddr
import pox.openflow.libopenflow_01 as of

log = core.getLogger()
```

O fluxo *flow0* instrui ao controlador a definir os pacotes que entram pela porta 2 do comutador com ID ‘0000000000000005’, a serem marcados com Vlan No. 10 ao saírem pela porta 3.

```
# flow0:
```

```
switch0 = 0000000000000005
flow0msg = of.ofp_flow_mod()
flow0msg.cookie = 0
flow0msg.priority = 32768
flow0msg.match.in_port = 2
# ACTIONS---------------------------------
flow0vlan_id = of.ofp_action_vlan_vid (vlan_vid = 10)
flow0out = of.ofp_action_output (port = 3)
flow0msg.actions = [flow0vlan_id, flow0out]
```

O fluxo *flow1* instrui ao controlador a definir os pacotes que entram pela porta 1 do comutador com ID ‘0000000000000005’, a serem marcados com Vlan No. 20 ao saírem pela porta 3.

```
# flow1:
switch1 = 0000000000000005
flow1msg = of.ofp_flow_mod()
flow1msg.cookie = 0
flow1msg.priority = 32768
flow1msg.match.in_port = 1
# ACTIONS---------------------------------
flow1vlan_id = of.ofp_action_vlan_vid (vlan_vid = 20)
flow1out = of.ofp_action_output (port = 3)
flow1msg.actions = [flow1vlan_id, flow1out]

...

# flow7:
```

A função *install_flows()* é responsável pela aplicação dos fluxos previamente configurados.

```
def install_flows():
  log.info("    *** Instalando fluxos estáticos... ***")
  # Instalando fluxos nos comutadores
  core.openflow.sendToDPID(switch0, flow0msg)
  core.openflow.sendToDPID(switch1, flow1msg)
  …
  log.info("    *** Fluxos estáticos instalados. ***")
```

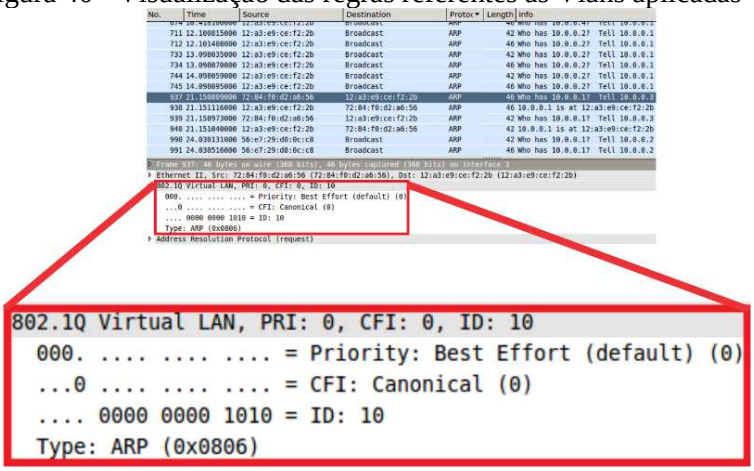
A função *launch()* é responsável por chamar a função install_flows.

```
def launch():
  log.info("*** Iniciando... ***")
  core.callDelayed (15, install_flows)
        log.info("*** Esperando pela conexão aos comutadores.. ***")
```

No *script* acima só estão descritos dois fluxos que definem os pacotes nas Vlans 10 e 20 em apenas uma interface para cada Vlan. Esse mesmo *script* está descrito em sua íntegra no ANEXO E.

Após a execução dos *scripts* e realização do teste de conectividade entre as estações de trabalho, é possível visualizar os cabeçalhos dos pacotes em tráfego através do Wireshark. A Figura 40 ilustra parte do cabeçalho de um pacote onde evidencia que, de fato, as configurações realizadas foram atribuídas com êxito.

Figura 40 - Visualização das regras referentes às Vlans aplicadas



É importante ressaltar que apesar das estações de trabalho estarem na mesma sub-rede, como o controlador as separa por Vlans, estas só poderão comunicar-se se estiverem na mesma Vlan. A Figura 40 traz parte do fluxo de pacotes no momento em que a estação *computer_3* realizou comunicação com a estação *computer_1*. É possível perceber em um dos tráfegos a presença do protocolo 802.1q com o quadro marcado (*frame tagged*) na Vlan 10.

A Tabela 2 ilustra alguns comandos para o *Floodlight* que podem ser úteis em alguns casos.

Tabela 2 - Alguns comandos úteis no Floodlight

Apagar um fluxo
curl -X DELETE -d '{"name":"flow-1"}' http://<192.168.1.20>:8080/wm/staticflowentrypusher/json
Listar todos os fluxos
curl http://192.168.1.20:8080/wm/staticflowentrypusher/list/00:00:00:00:00:00:00:03/json
Apagar todos os fluxos
curl http://192.168.1.66:8080/wm/staticflowentrypusher/clear/00:00:00:00:00:00:00:03/json

Este estudo de caso demonstrou que para segregar fluxos, por exemplo, através de Vlans, é necessário criar tabelas de fluxos no controlador para que este dite as regras acerca dos pacotes que trafegam na rede. O *script* gerado para o controlador, no caso o Pox, traz uma informação importante: para cada fluxo de entrada, existe uma ação a ser realizada. Em cada fluxo criado existe uma definição de interface de saída e/ou definição de atribuição de Vlan ID na tabela de fluxos.

A geração de *scripts* por parte do VND (versão SDN) demanda atenção do usuário, pois é importante a escolha do controlador no campo *OpenFlow Controller* para que seja gerado o *script* correto para o controlador em questão. Por outro lado, o VND (versão SDN) foi desenvolvido para criar tabelas de fluxos com base nas especificações OpenFlow, portanto, caso o usuário tenha noções sobre controles de fluxos baseados no protocolo OpenFlow, ele não terá dificuldades em gerar *scripts* para o controlador que deseja trabalhar.

A descrição NSDL deste cenário No. 02 é apresentado no ANEXO B.

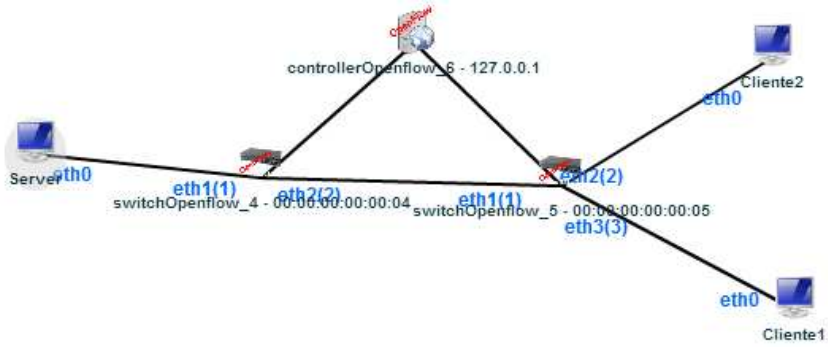
11.4 CENÁRIO NO. 03 – IMPLEMENTAÇÃO DE QUALIDADE DE SERVIÇO

A proposta deste cenário é atribuir diferentes prioridades para os fluxos entre clientes e servidor. Com isso, também mostrar como é possível utilizar o VND (versão SDN) para implementação de *Qualidade de Serviço (QoS)* com o protocolo OpenFlow.

O cenário No. 03 é composto por três estações de trabalho, um controlador e dois comutadores OpenFlow. O *cliente1* e o *cliente2* terão prioridades iguais a *100* e *1*, respectivamente (quanto maior o número de prioridade, maior será a prioridade do pacote em relação aos demais). Neste cenário, é possível fazer uma analogia a um sistema de IPTV – Internet Protocol Television, método de transmissão de conteúdos televisivos distribuídos via Internet, onde usuários possuem qualidade de imagem

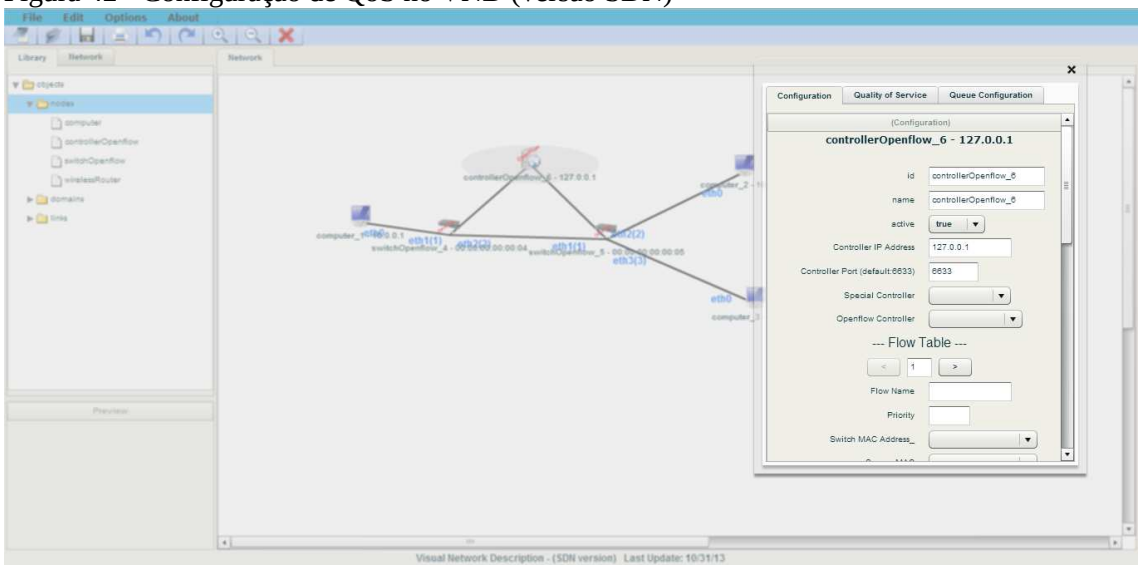
diretamente proporcional ao pacote de serviços que contratam das operadoras de telecomunicações.

Figura 41 - Cenário de rede com QoS criado através do VND (versão SDN)



Para a criação deste cenário no VND (versão SDN), além da inclusão dos dispositivos de rede no ambiente de configuração de cenários, também é necessário realizar algumas configurações no controlador. É necessário que sejam criadas filas e sejam definidas as configurações dessas filas conforme o objetivo do cenário. Nos trechos de *scripts* abaixo é possível perceber que foi criada uma fila *q0* com prioridade igual a 100 e outra fila *q1* com prioridade 1. Ambas possuem largura de banda disponível de 5000000 kbps ou 5Mbps.

Figura 42 - Configuração de QoS no VND (versão SDN)



Para a configuração de QoS no VND (versão SDN), o usuário deverá selecionar o controlador OpenFlow, criar as filas e atribuir as regras necessárias para as filas criadas. Esses passos devem ser realizados nos menus *Quality of Service* e *Queue*

Configuration (em português, respectivamente, *Qualidade de Serviço e Configuração de Filas*).

Após a criação do cenário da Figura 41 no VND (versão SDN), o seguinte *script* para o Mininet poderá ser gerado automaticamente:

```
#!/usr/bin/python
"""
Script criado pelo VND - Visual Network Description (versão SDN)
"""
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink

def topology():
   "Criando a rede."
   net = Mininet( controller=Controller, link=TCLink,
switch=OVSKernelSwitch )

   print "*** Criando nós"
   cliente1 = net.addHost( 'client1', mac='00:00:00:00:00:01', ip='172.16.0.2/16'
)
   cliente2 = net.addHost( 'client2', mac='00:00:00:00:00:02', ip='172.16.0.3/16'
)
   server = net.addHost( 'server', mac='00:00:00:00:00:03', ip='172.16.0.1/16' )
   s4 = net.addSwitch( 's4', mac='00:00:00:00:00:04')
   s5 = net.addSwitch( 's5', mac='00:00:00:00:00:05')
   c6 = net.addController( 'c6', controller=RemoteController,
defaultIP="127.0.0.1", port=6633)

   print "*** Criando links"
   net.addLink(s4, s5, 2, 1)
   net.addLink(s5, client1, 2, 0)
   net.addLink(s5, client2, 3, 0)
   net.addLink(server, s4, 0, 1)

   print "*** Iniciando a rede"
   net.build()
   s4.start( [c6] )
   s5.start( [c6] )
   c6.start()
```

```
    print "*** Executando CLI"
    CLI( net )

    print "*** Parando a rede"
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    topology()
```

O trecho de *script* abaixo ilustra as configurações de filas e QoS definidas no cenário.

```
ovs-vsctl -- set Port s4-eth1 qos=@newqos \
-- set Port s4-eth2 qos=@newqos \
-- set Port s5-eth2 qos=@newqos \
-- set Port s5-eth1 qos=@newqos \
-- set Port s5-eth3 qos=@newqos \
-- --id=@newqos create QoS type=linux-htb other-config:max-rate=1000000000
queues=0=@q0,1=@q1 \
-- --id=@q0 create Queue other-config:max-rate=5000000 \
-- --id=@q1 create Queue other-config:max-rate=5000000
```

Por fim, também é necessário executar o trecho de *script* a seguir gerado também pelo VND. Como forma de mostrar a adaptabilidade do VND (versão SDN) a diversas ferramentas OpenFlow, o trecho de *script* a seguir foi gerado para o controlador *Floodlight*.

```
./qospath2.py --add --name Qos1 -S 172.16.0.2 -D 172.16.0.1 --json '{"eth-
type":"0x0800","priority":"100","protocol":"6","queue":"0"}' -c 127.0.0.1 -p
8080

./qospath2.py --add --name Qos2 -S 172.16.0.3 -D 172.16.0.1 --json '{"eth-
type":"0x0800","priority":"1","protocol":"6","queue":"1"}' -c 127.0.0.1 -p 8080
```

A partir da realização de testes de comunicações realizadas de forma simultânea dos clientes para o servidor, é possível observar através das Figuras 43 e 44 que existe uma grande diferença no tempo de resposta dos pacotes enviados. O envio simultâneo dos pacotes ocorreu a partir do envio do quarto pacote e prolongou-se durante 10 segundos. Além do aumento do tempo de resposta ocorrido no *cliente2* em relação ao *cliente1* em cada pacote enviado durante o período de simultaneidade, é possível

observar também que o tempo médio de resposta dos pacotes originados no *cliente2* foi de 102.741ms, enquanto o *cliente1* teve tempo médio de resposta igual a 2.760ms.

Figura 43 - Teste de comunicação realizada para o servidor - cliente 1

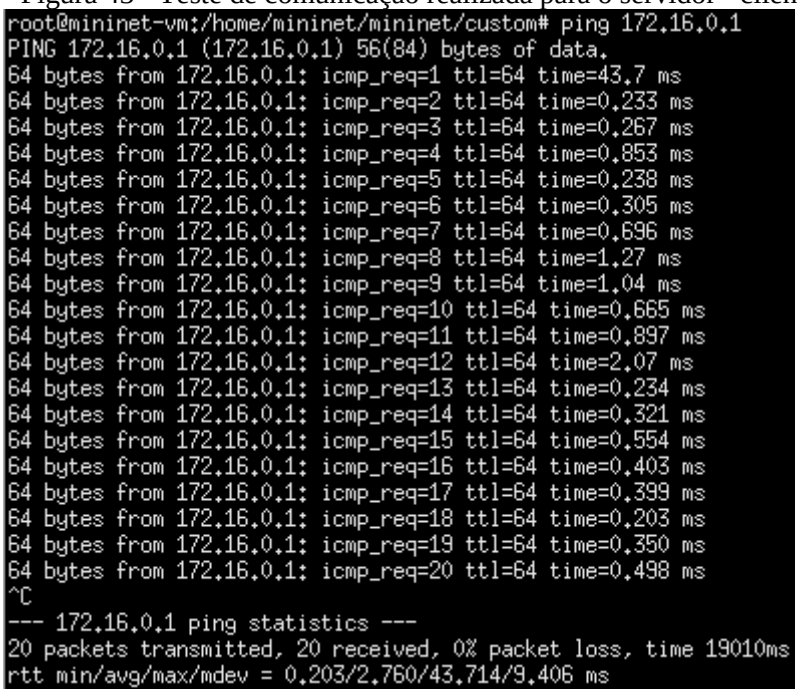
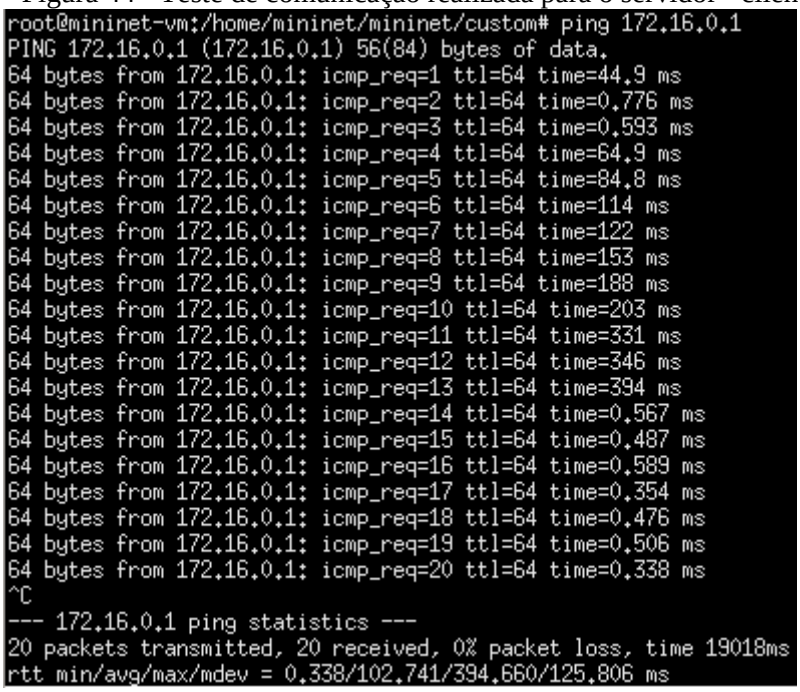


Figura 44 - Teste de comunicação realizada para o servidor - cliente 2



O *script* a seguir irá estabelecer a comunicação entre as estações de trabalho e irá criar a seguinte tabela de configuração:

```
QoS applied to switch 00:00:00:00:00:00:00:05 for circuit 1Mbps
00:00:00:00:00:00:00:05: in:3 out:1
Adding Queueing Rule
{
        “ip-src”: “172.16.0.3”,
        “protocol”: “6”,
        “name”: “q1.00:00:00:00:00:00:00:05”,
        “ip-dst”: “172.16.0.1”,
        “sw”: “00:00:00:00:00:00:00:05”,
        “priority”: “1”,
        “queue”: “1”,
        “enqueue-port”:” “1”,
        “eth-type”:” “0x0800”
}
Connection Successful
…
QoS applied to switch 00:00:00:00:00:00:00:04 for circuit 1Mbps
00:00:00:00:00:00:00:04: in:2 out:1
Adding Queueing Rule
{
        “ip-src”: “172.16.0.3”,
        “protocol”: “6”,
        “name”: “q1.00:00:00:00:00:00:00:04”,
        “ip-dst”: “172.16.0.1”,
        “sw”: “00:00:00:00:00:00:00:04”,
        “priority”: “1”,
        “queue”: “1”,
        “enqueue-port”:” “1”,
        “eth-type”:” “0x0800”
}
Connection Successful
...
QoS applied to switch 00:00:00:00:00:00:00:05 for circuit 5Mbps
00:00:00:00:00:00:00:05: in:2 out:1
Adding Queueing Rule
{
        “ip-src”: “172.16.0.2”,
        “protocol”: “6”,
        “name”: “q0.00:00:00:00:00:00:00:05”,
        “ip-dst”: “172.16.0.1”,
        “sw”: “00:00:00:00:00:00:00:05”,
        “priority”: “100”,
```

```
        “queue”: “0”,
        “enqueue-port”:” “1”,
        “eth-type”:” “0x0800”
}
Connection Successful
…
QoS applied to switch 00:00:00:00:00:00:00:04 for circuit 5Mbps
00:00:00:00:00:00:00:04: in:2 out:1
Adding Queueing Rule
{
        “ip-src”: “172.16.0.2”,
        “protocol”: “6”,
        “name”: “q0.00:00:00:00:00:00:00:04”,
        “ip-dst”: “172.16.0.1”,
        “sw”: “00:00:00:00:00:00:00:04”,
        “priority”: “100”,
        “queue”: “0”,
        “enqueue-port”:” “1”,
        “eth-type”:” “0x0800”
}
Connection Successful
```

O desenvolvimento deste estudo de caso mostrou alguns passos importantes para as configurações de *QoS* em ambientes OpenFlow. Antes de aplicar regras de *QoS*, é fundamental a criação de filas para os fluxos e nessas filas devem conter informações das interfaces físicas dos comutadores que irão compor os caminhos dos fluxos, além de também ser necessário determinar qual a largura de banda de cada fila. Apesar dos *scripts* referentes ao *QoS* estarem ilustrados de forma separadas, acima, o VND (versão SDN) cria apenas um *script* contendo as informações de filas e *QoS* para o controlador que o usuário deseja trabalhar.

A descrição NSDL deste cenário No. 03 é apresentado no ANEXO C.

11.5 CENÁRIO NO. 04 – AMBIENTE DE REDE SEM FIO

Este cenário demonstra a utilização do OpenFlow em um cenário de rede sem fio. A proposta deste cenário é criar fluxos que permitam a comunicação entre uma

estação de trabalho conectada ao comutador por um meio cabeado e uma outra estação de trabalho que está conectada ao mesmo comutador através de um meio sem fio.

O cenário No. 04 tem uma proposta um pouco diferente das anteriormente apresentadas, onde o Mininet foi utilizado na simulação de estações de trabalho e o *Open vSwitch* foi responsável pela simulação de comutadores *OpenFlow*. Neste cenário, são utilizados estações de trabalho reais e um encaminhador sem fio com o *OpenWRT* instalado e esse funcionará como um comutador OpenFlow. A versão do *firmware* do *OpenWRT* inclui o protocolo *OpenFlow*.

Antes de abordar o cenário propriamente dito, é importante realizar algumas observações sobre a instalação do *OpenWRT* e as dificuldades encontradas neste processo. O encaminhador sem fio utilizado neste cenário foi um TPLINK-WR1043ND v11, contudo, existem diversos encaminhadores de diversos fabricantes que também suportam o *OpenWRT* e o protocolo OpenFlow. O site apresentado em (OPENWRT, 2013) possui uma relação desses encaminhadores.

Para isso, é necessária a instalação adequada, devendo-se retirar a *firmware* padrão do encaminhador e instalar o *OpenWRT*. Contudo, caso seja utilizada uma *firmware* errada ou um encaminhador que não suporta *OpenWRT*, existem grandes possibilidades de ocorrer um problema chamado de “*brick*”. O “*brick*” é um erro que afeta a estrutura lógica do sistema e inviabiliza o acesso ao ambiente de configuração do encaminhador. Existem outras possibilidades de ocorrer um “*brick*” no encaminhador e caso o usuário queira pesquisar/utilizar o OpenFlow através do *OpenWRT*, provavelmente esse problema ocorrerá. Na implementação deste cenário o problema ocorreu e a Figura 45 ilustra parte do processo necessário para corrigir o problema e recuperar o acesso ao encaminhador.

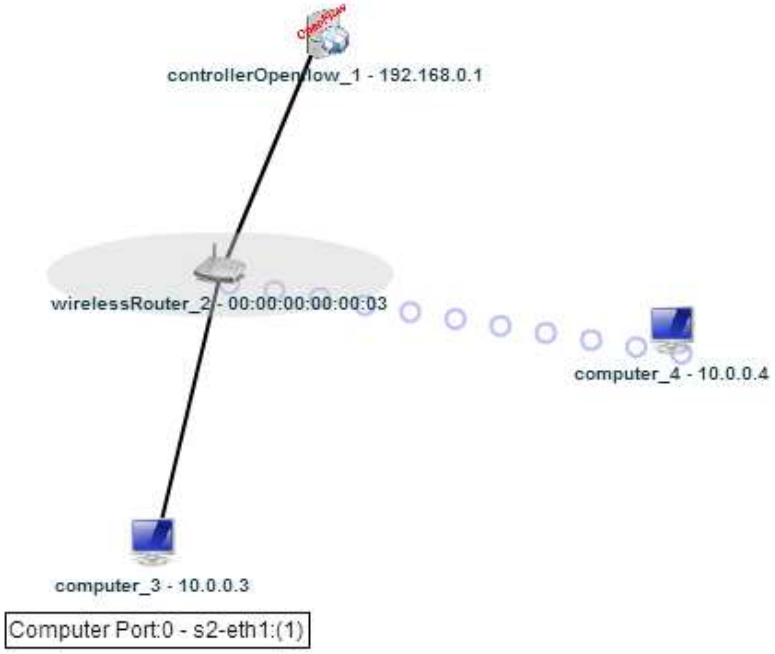
Figura 45 - Realizando “unbrick” no encaminhador

O “*unbrick*”, nome dado ao método de recuperação do encaminhador, pode ser realizado através de uma conexão serial feita entre encaminhador e computador. Neste caso, o encaminhador foi conectado a um *raspberry pi* (pequeno computador que possui todo o conjunto de hardware integrado em uma única e pequena placa) e o acesso ao *raspbeery pi* foi realizado através da rede ethernet.

Uma vez recuperado o acesso, o *OpenWRT* foi instalado com a versão 1.3 do protocolo OpenFlow, desenvolvida pelo Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD) (OPENWRT, 2013). No entanto, ainda existem poucas ferramentas OpenFlow, mais precisamente controladores, que suportam o OpenFlow 1.3. Os controladores *Pox* e *Floodlight*, utilizados nos cenários anteriores, por exemplo, ainda não suportam. Por isso, é necessário utilizar uma *firmware* com o OpenFlow 1.0 habilitado.

Uma vez que o OpenWRT e OpenFlow estão devidamente e corretamente instalados, o cenário pode ser constituído. O cenário No. 04 (Figura 46) é constituído de 2 (duas) estações de trabalho, um comutador OpenFlow e o controlador Pox sendo executado em um *raspberry pi*. Neste cenário, a configuração de fluxos permitirá a comunicação da estação 4, que utilizará o meio sem fio para comunicação, com a estação 3, que está conectado ao comutador OpenFlow através de um cabo UTP.

Figura 46 - Cenário #04 criado através do VND (versão SDN)



Para a criação do cenário No. 04 no VND (versão SDN), basta incluir os dispositivos de rede no ambiente de configuração de cenários e conecta-los através das suas respectivas ligações. A diferença deste cenário para os demais apresentados é a presença de um dispositivo sem fio. Para a comunicação entre o dispositivo sem fio e uma estação de trabalho, é necessário orientar a comunicação para a estação de trabalho através de uma simples ação de *arrastar e soltar* utilizando o mouse.

A geração de *script* para o controlador neste cenário inclui a importância de se definir a interface de entrada e saída dos fluxos e qual endereço MAC de origem será o responsável por permitir o encaminhamento dos fluxos na tabela de fluxos. No VND (versão SDN), o usuário terá que criar as tabelas de fluxos nos campos apropriados para a posterior geração do *script*.

O *script* gerado pelo VND (versão SDN) para o controlador Pox está descrito abaixo.

```
#!/usr/bin/python
Script criado pelo VND - Visual Network Description (versão SDN)
from pox.core import core
from pox.lib.addresses import IPAddr
from pox.lib.addresses import EthAddr
import pox.openflow.libopenflow_01 as of
```

```
log = core.getLogger()
```

O fluxo *flow0* é responsável por encaminhar o tráfego de pacotes que entrar pela porta 1 com endereço MAC de origem 2C:76:8A:BB:47:9D, a sair pela porta 5. O endereço MAC de origem corresponde ao endereço MAC da estação *computer_3*.

```
# flow0:
switch0 = 0000000000000001
flow0msg = of.ofp_flow_mod()
flow0msg.cookie = 0
flow0msg.priority = 32768
flow0msg.match.in_port = 1
flow0msg.match.dl_src=EthAddr("2C:76:8A:BB:47:9D")
# ACTIONS---------------------------------
flow0out = of.ofp_action_output (port = 5)
flow0msg.actions = [flow0out]
```

O fluxo *flow1* representa o caminho inverso percorrido pelo tráfego do fluxo *flow0*. Ele é responsável por encaminhar o tráfego de pacotes que entrar pela porta 5 e tiver MAC de origem D8:D3:85:3E:30:65, a sair pela porta 1. O endereço MAC de origem corresponde à estação de trabalho *computer_4*.

```
# flow1:
switch1 = 0000000000000001
flow1msg = of.ofp_flow_mod()
flow1msg.cookie = 0
flow1msg.priority = 32768
flow1msg.match.in_port = 5
flow1msg.match.dl_src=EthAddr("D8:D3:85:3E:30:65")
# ACTIONS---------------------------------
flow1out = of.ofp_action_output (port = 1)
flow1msg.actions = [flow1out]

def install_flows():
  log.info("    *** Instalando fluxos estáticos... ***")
  # Instalando fluxos nos comutadores
  core.openflow.sendToDPID(switch0, flow0msg)
  core.openflow.sendToDPID(switch1, flow1msg)
  log.info("    *** Fluxos estáticos instalados. ***")

def launch ():
```

```
log.info("*** Iniciando... ***")
core.callDelayed (15, install_flows)
log.info("*** Esperando pela conexão aos comutadores.. ***")
```

A seguir é apresentado um *script* que representa em parte como o cenário No. 04 seria executado no Mininet.

```
#!/usr/bin/python

"""
Script criado pelo VND - Visual Network Description (versão SDN)
"""
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink

def topology():
   "Criando a rede."
   net = Mininet( controller=RemoteController, link=TCLink, switch=OVSKernelSwitch )

   print "*** Criando nós"
   c1 = net.addController( 'c1', controller=RemoteController, ip='192.168.1.100', port=6633)
   s2 = net.addSwitch( 's2', mac='00:00:00:00:00:02')
   h3 = net.addHost( 'h3', mac='2C:76:8A:BB:47:9D', ip='10.0.0.3/8' )
   h4 = net.addHost( 'h4', mac='D8:D3:85:3E:30:65', ip='10.0.0.4/8' )

   print "*** Criando links"
   net.addLink(h3, s2, 0, 1)

   print "*** Iniciando a rede"
   net.build()
   s2.start( [c1] )
   c1.start()

   print "*** Executando CLI"
   CLI( net )

   print "*** Parando a rede"
```

```
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    topology()
```

Como é possível perceber, não há nenhuma representação de conexão para a estação de trabalho *computer_4*. Infelizmente, o Mininet ainda não permite a experimentação de cenários sem fio e por este motivo este enlace não pode ser implementado. O que o Mininet faz, na verdade, é permitir a configuração de largura de banda, atrasos e perdas em links, mas não a possibilidade real de experimentação de cenários sem fio.

Diante do exposto neste cenário conclui-se que também é possível fazer implementações de redes OpenFlow para ambientes sem fio. É certo que podem haver dificuldades nessas implementações, principalmente se o equipamento não vier com o OpenFlow habilitado de forma nativa, mas isso só evidencia a flexibilidade e adaptabilidade que existe no protocolo OpenFlow. Afinal de contas, foi possível instalar o OpenFlow em um equipamento proprietário e utilizar os recursos de hardware deste equipamento para a implementação de uma rede OpenFlow.

Por fim, também conclui-se que através do VND (versão SDN) é possível construir cenários de redes sem fio e a tendência é que novos recursos sejam implementados de forma a representar diversas infraestruturas de rede.

A descrição NSDL deste cenário No. 04 é apresentado no ANEXO D.

11.6 CONSIDERAÇÕES FINAIS

Diante dos estudos de casos expostos neste capítulo, é possível, primeiramente, observar algumas das características e forma de funcionamento do protocolo OpenFlow. Por seguinte, o êxito na configuração dos cenários OpenFlow pelo VND (versão SDN) demonstra que essa ferramenta está apta a proporcionar a autoria de cenários OpenFlow.

Os estudos de caso mostraram a diversidade nos *scripts* gerados. Da mesma forma, a heterogeneidade das ferramentas OpenFlow pode fazer do processo de

compreensão um processo árduo. Sendo baseado na especificação da tecnologia OpenFlow (MCKEOWN, 2008), o VND (versão SDN) simplifica o processo de autoria e testes de cenários OpenFlow, otimizando o seu ciclo de vida.

A geração automática de *scripts* para os cenários realizados neste trabalho poupou esforços a nível de programação. A forma como os *scripts* gerados são organizados também facilita o posterior entendimento sobre a linguagem de programação com a qual o *script* foi gerado. As execuções desses *scripts*, entretanto, são realizadas de acordo com as especificações da ferramenta OpenFlow com a qual o usuário deseja trabalhar. Conforme citado anteriormente, o VND (versão SDN) foi submetido à apreciação dos usuários e através dos seus retornos, foi possível validar a proposta deste trabalho.

A seleção dos estudos de casos ilustrados neste capítulo foi baseada de forma a proporcionar a autoria de cenários em diferentes aspectos de configuração de redes. Aspectos que envolvem a segmentação da uma infraestrutura de rede, implementação em qualidade de serviço e redes cabeadas e sem fio.

O próximo capítulo encerra todo o ciclo de informações apresentadas neste trabalho através de conclusões e perspectivas futuras.

12 CONCLUSÕES E PERSPECTIVAS FUTURAS

12.1 CONCLUSÕES

A variedade de contribuições orientadas ao desenvolvimento de experimentos OpenFlow demonstram o interesse e os esforços da comunidade científica na pesquisa e evolução em SDN. No entanto, a configuração dos cenários OpenFlow muitas vezes exige a utilização de linguagens de programação, e essa configuração é realizada através de interfaces baseadas em texto. Por essas razões, a utilização desse paradigma ainda é limitada dada a baixa produtividade oferecida pelos ambientes de autoria existentes.

Dessa forma, é importante a utilização de uma ferramenta com interface gráfica que permita otimizar o ciclo de vida desses experimentos (especificação, validação e implementação). A adoção de uma ferramenta como o *Visual Network Description - VND (versão SDN)* permite facilitar a criação dos cenários OpenFlow através de uma interface de simples utilização e configuração. O VND (versão SDN) também é uma ferramenta interoperável com ferramentas de gestão de redes (por exemplo, autoria, simulação, etc.) que suportam a linguagem NSDL ou através da geração automática de *scripts* para as diversas ferramentas existentes. Na Tabela 3 é apresentada uma comparação entre as ferramentas para a autoria de redes OpenFlow apresentadas em Trabalhos Relacionados e o VND (versão SDN).

Tabela 3 - Comparação entre as ferramentas para a autoria de redes OpenFlow

	MiniEdit	Topology Generator	VND (versão SDN)
Interface Gráfica	Possui	Possui	Possui
Reutilização de cenários	Permite	Permite	Permite
Redes escaláveis	Permite	Permite	Permite
Visualização do ambiente de rede para a criação, configuração e solução de problemas	Permite	Permite	Permite
Suporte ao OpenFlow	Sim	Sim	Sim
Suporte a outras ferramentas OpenFlow	Não	Não	Sim
Geração automática de scripts	Sim	Sim	Sim
Necessita de instalação	Sim	Sim	Não
Ambiente	Local	Local	Web

Como é possível observar, o VND (versão SDN) possui todas as características do *MiniEdit* e do *Topology Generator*, diferenciando, porém, em algumas delas. O

VND (versão SDN) não necessita de instalação, está acessível para qualquer usuário através da Internet em ambiente Web e diferentemente do *MiniEdit* e do *Topology Generator*, suporta outras ferramentas OpenFlow.

Durante o desenvolvimento do VND (versão SDN), no entanto, foi necessário a compreensão do modelo de funcionamento de redes OpenFlow e também a utilização de algumas ferramentas Openflow existentes. É natural que durante a proposta e o amadurecimento de uma tecnologia, as diferentes ferramentas existentes que suportem essa tecnologia sofram constantes mudanças e atualizações. Por essa razão, as diversas atualizações que as ferramentas OpenFlow passam, muitas vezes alteram completamente a sua forma de funcionamento e isso implica na dificuldade da gestão dessas ferramentas por partes dos usuários, assim como no desenvolvimento de ferramentas como o VND (versão SDN). Apesar das diversas comunidades existentes na web com o propósito de ajudar os usuários esclarecendo dúvidas, as constantes atualizações fazem com que as páginas web fiquem desatualizadas e em vez de ajudar, acabam por complicar o entendimento sobre as redes SDN e OpenFlow.

Por outro lado, vale a pena destacar, o nível de competência exigido de um profissional ao trabalhar com SDNs. Naturalmente, profissionais em redes de computadores sem conhecimento em desenvolvimento de software vão ter dificuldades em trabalhar com o OpenFlow, bem como desenvolvedores de software sem conhecimento em redes de computadores. Um engenheiro de software, por exemplo, pode realizar todo o trabalho de programação de forma rápida e eficiente, contudo, certamente estará perdido sobre o comportamento de um controlador na rede e terá dificuldades para escalona-lo.

Apesar da identificação desses problemas, o documento de especificações OpenFlow (*OpenFlow Specification*) (ONF, 2013) permite o esclarecimento de muitos aspectos no desenvolvimento de novas soluções OpenFlow. Dado que o protocolo OpenFlow está em constante de aperfeiçoamento, espera-se que o VND (versão SDN) possa ser continuamente aprimorado para estar adaptado a uma maior quantidade de ferramentas OpenFlow e também para abranger uma maior quantidade de usuários.

Vale a pena ressaltar que durante o desenvolvimento do VND (versão SDN), houveram muitas oportunidades de trocas de experiências sobre o projeto. A partir do contato com os responsáveis pela página web da comunidade de desenvolvimento da

ferramenta Mininet (MININET, 2013), foi motivado o interesse pela publicação e divulgação da ferramenta VND (versão SDN) neste site. Como resultado, houveram diferentes sugestões de melhorias no processo de criação de cenários OpenFlow utilizando o VND (versão SDN).

Na Figura 47 a área pintada representa os países que fizeram acessos ao VND (versão SDN) e pode-se perceber que é considerável a atenção dada acerca das redes SDN pela comunidade internacional.

Figura 47 - Utilização do VND (versão SDN) pelo mundo

Como forma de retorno dos usuários para validação das funcionalidades do VND (versão SDN), a ferramenta foi divulgada em comunidades internacionais e desde então diversos usuários têm utilizado e colaborado para o aprimoramento do VND (versão SDN). A Figura 47 foi extraída do *Google Analytics* - serviço gratuito da Google que permite a exibição de estatísticas de visitas.

A partir do acesso constatado e também dos contatos realizados através de mensagens eletrônicas, podemos verificar o grande interesse e utilização do VND (versão SDN) para estudos sobre o OpenFlow. Desde usuários iniciantes a usuários que já possuem experiências com o protocolo OpenFlow e que têm dado relatos positivos acerca da sua usabilidade e aplicabilidade. O VND (versão SDN) também foi citado como uma ferramenta para o desenvolvimento de experimentos OpenFlow no livro intitulado de *"SDN and OpenFlow for beginners with hands on labs"* do autor *Vivek Tiwari* (TIWARI, 2013).

A partir dos esforços de disseminação no site da comunidade para o Mininet e dos contatos obtidos, assim como através das publicações científicas realizadas, podemos afirmar que houve êxito no desenvolvimento do VND (versão SDN), e isso ajuda a justificar os recursos e esforços que foram implementados para esta ferramenta. O VND (versão SDN) mostrou ser uma ferramenta de fácil adaptação, flexível na criação de cenários OpenFlow, e na geração de arquivos NSDL e *scripts* para ferramentas OpenFlow.

Diferentemente das outras ferramentas de autoria mencionadas ao longo deste trabalho, o VND (versão SDN) permite suporte a outras ferramentas OpenFlow, o que inclui emulador e controladores. Através de um único ambiente o usuário pode constituir um cenário OpenFlow e através da geração automática de descrições NSDL e *scripts*, ele pode realizar a integração com as ferramentas OpenFlow que deseja trabalhar. É importante destacar, contudo, que o VND (versão SDN) está adaptado às ferramentas OpenFlow conforme elas são hoje e caso haja alguma mudança na estrutura de programação dessas ferramentas, pode impactar na geração dos *scripts* gerados e exigir uma atualização no VND (versão SDN) para acompanhar esse processo de mudanças.

12.2 PERSPECTIVAS FUTURAS

Como perspectivas futuras para continuação do trabalho apresentado nesta dissertação, é possível propor:

- ✓ Desenvolvimento de cenários OpenFlow mais complexos com o VND (versão SDN);
- ✓ Implementação de novos recursos no VND (versão SDN) de forma que seja compatível com o maior número possível de ferramentas OpenFlow, exemplo: *desenvolvimento de novos módulos para outros controladores OpenFlow, além do Pox e do Floodlight*;
- ✓ Validação de cenários OpenFlow criados pelo VND (versão SDN) com outras ferramentas de simulação/emulação de Redes, exemplo: ns3, que já foi apresentada ao longo deste trabalho.

REFERÊNCIAS

ARAÚJO, J. *Virtualização automática de cenários de rede*". 161 p.2011. Dissertação (Mestrado)- Programa de Mestrado em Engenharia Informática, Universidade da Madeira, Funchal, Portugal, 2011.

AZEVEDO, Jesuíno da C. S. de. *Virtual network description:* suporte à simulação de redes através da autoria gráfica de cenários virtuais. 139 p. 2010. Dissertação (Mestrado)- Programa de Mestrado em Engenharia Informática, Universidade da Madeira, Funchal, Portugal, 2010.

BARHAM, P. et al.. Xen and the Art of Virtualization. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES - SOSP '03, 19., 2003. Bolton Landing, NY USA. *Proceedings* … 2003.

BEACON. Disponível em: <https://openflow.stanford.edu/display/Beacon/Home>. Acesso em: 15 jul. 2013.

BRADLEY, G. *Network and graph markup language (NaGML)-data file formats*. Technical Report NPS-OR-04-007. Monterey, CA, USA: Department of Operations Research, Naval Postgraduate School, 2004.

CHOWDHURY, N. M. K.; BOUTABA, R. A survey of network virtualization. *Computer Networks*, n.54, p.862–876, 2010.

CISCO. *Cisco packet tracer*. Disponível em: <https://www.netacad.com/pt/web/about-us/cisco-packet-tracer/>. Acesso em: 20 out. 2013.

EGEVANG, K. ; FRANCIS, P. *The IP network address translator (NAT).* RFC 1631, 1994.

ESCALONA, E. et al. GEYSERS: a novel architecture for virtualization and co-provisioning of dynamic optical networks and IT services. In: FUTURE NETWORK AND MOBILE SUMMIT, 2011. Warsaw, Poland. *Proceedings...* 2011.

FERNANDES, N. et al. O. Virtual networks: isolation, performance, and trends. In: ANNALS OF TELECOMMUNICATIONS. *Proceedings...* 2010.

FELDMANN, A. Internet clean-slate design: what and why? In: *SIGCOMM Computer Communication Review*, v.37, n.3., jul. 2007.

FIBRE. *Future Internet testbeds experimental between Brazil and Europe*. Disponível em <http://www.fibre-ict.eu/>. Acesso em: 2 ago. 2013.

FLEX. Disponível em: <http://www.adobe.com/br/products/flex.html>. Acesso em: 5 out. 2013.

FLOODLIGHt. Disponível em: <http://www.projectfloodlight.org/floodlight/>. Acesso em: 16 jul. 2013.

FOROUZAN, B. *Comunicação de dados e redes de computadores*. 3. ed. Porto alegre. [s.n.], 2010.

FP7 OFELIA. Disponível em: <http://www.fp7-ofelia.eu>. Acesso em: 2 ago. 2013.

FULLER, V.; LI, T. ; VARADHAN, K. *Classless Inter-Domain Routing (CIDR*): an address assignment and aggregation strategy. [S.l.]: [s.n.], 1993.

GAMMa, E. et al. *Design patterns*: elements of object-oriented software. [S.l.]: Addison-Wesley Professional, 1994.

GHIJSEN, M.; et al. Towards an infrastructure description language for modeling computing infrastructures. In: PARALLEL AND DISTRIBUTED PROCESSING WITH APPLICATIONS (ISPA), IEEE 10TH INTERNATIONAL SYMPOSIUM, 2012. Madrid, Spain. 2012.

GENI. *Exploring Networks of the Future*. Disponível em: <http://www.geni.net/>. Acesso em: 17 jul. 2013.

GENI SYSTEM OVERVIEW. *Global Environment for Nework Innovations*. Geni system overview. 2008. Disponível em: <http://groups.geni.net/geni/attachment/wiki/GeniSysOvrvw/GENISysOvrvw092908.pdf>. Acesso em: 17 jul. 2013.

GUDE, N. et al. NOX: towards an operating system for networks. *SIGCOMM Computer Communication Review*, v.38, n.3, jul. 2008.

HAM, J. V. D. et al. Using the network description language in optical networks. In: INTEGRATED NETWORK MANAGEMENT 2007(IM '07.10TH IFIP/IEEE INTERNATIONAL SYMPOSIUM). 2007. Munich, Germany. *Proceedings...* 2007.

IPERF. *TCP and UDP bandwidth performance measurement tool*. Disponível em: <https://code.google.com/p/iperf>. Acesso em:1 nov. 2013.

JAVASCRIPT. *JavaScript Tutorial*. Disponível em: <http://www.w3schools.com/js/>. Acesso em: 4 nov. 2013.

JIST. *Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator*. Disponível em: <http://jist.ece.cornell.edu/>. Acesso em:3 out. 2013.

KENT, S. *Security architecture for the internet protocol*. [S.l.]: [s.n.], 1998.

KIM, J. ; CHOI, H. Spam traffic characterization. In: TECHNICAL CONFERENCE ON CIRCUITS/SYSTEMS, COMPUTERS AND COMMUNICATIONS, 1., Shimonoseki City, Japan, 2008. *Proceedings...* 2008.

KÖPSEL A.; WOESNER, H. Ofelia – Pan-European Test Facility for OpenFlow Experimentation. *Lecture Notes in Computer Science*, v. 6994/2011, 2011.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: rapid prototyping for software-defined networks. In: ACM SIGCOMM WORKSHOP ON HOT TOPICS

IN NETWORKS, HOTNETS ’10, 9., New York. *Proceedings*… New York: ACM, 2010.

MARQUES, E. M. D.; SAMPAIO, P. N. M. A Framework for the Integration of Network Modelling and Simulation Tools. In: EUROSIS - THE EUROPEAN MULTIDISCIPLINARY SOCIETY FOR MODELLING AND SIMULATION TECHNOLOGY (ESM´ 2010). 2010. Hasselt, Belgium. *Proceedings.*…2010.

MARQUES, E. M. D.; SAMPAIO, P. N. M. NSDL: an integration framework for the network modeling and simulation. *International Journal of Modeling and Optimization*, v. 2, n. 3, p. 304-308, jun. 2012.

MARQUES, E. M. D. *Interoperabilidade e otimização da gestão de redes com a framework NSDL*. Funchal: Universidade da Madeira. 229 p. 2013. Tese (Doutorado) - Programa de Doutorado em Engenharia Informática na Especialidade Sistemas Distribuídos e Centrados em Redes, Universidade da Madeira, Funchal, Portugal, 2013.

MCKEOWN, N. Anderson et al. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, n.38, p.69–74, 2008.

MINIEDIT APP. Disponível em: <https://github.com/mininet/mininet/wiki/Mini-Edit-App>. Acesso em: 04 set. 2013.

MININET. *Mininet community extensions and packages*. Disponível em: <https://github.com/mininet/mininet/wiki/Mininet-Community-Extensions-and-Packages>. Acesso em: 17 jul. 2013.

NASCIMENTO, Marcelo R. et al. RouteFlow: roteamento commodity sobre redes programáveis. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES - SBRC'2011, 29., 2011. Campo Grande. *Proceedings.*… 2011.

NAYAK, A. K. et al. Resonance: dynamic access control for enterprise networks. In: ACM WORKSHOP ON RESEARCH ON ENTERPRISE NETWORKING, 1., 2009. Espanha. *Proceedings*... 2009.

NML-WG. *Network Markup Language Working Group*. Disponível em: http://www.ogf.org/gf/group_info/view.php?group=nml--wg < >. Acesso em: 4 set. 2013.

NOVI. *Networking innovations over virtualized infrastructures*. Disponível em: <http://www.fp7-novi.eu/>. Acesso em: 3 set. 2013.

NOX. Disponível em: <http://www.noxrepo.org/nox/about-nox/>. Acesso em: 16 jul. 2013.

NS2. *The Network Simulator – NS2*. Disponível em: <http://www.isi.edu/nsnam/ns>. Acesso em 20 jul. 2013.

NS3. Disponível em: <http://www.nsnam.org/>. Acesso em: 4 out. 2013.

OMNeT++. Disponível em: <http://www.omnetpp.org>. Acesso em: 20 jul. 2013.

ONF. *OpenFlow Switch Specification. Version 1.3.2 (Wire Protocol 0x04), 2013*. Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.2.pdf>. Acesso em: 15 jul. 2013.

OPEN VSWITCH. *An Open Virtual Switch*. Disponível em: <http://openvswitch.org/>. Acesso em: 17 jul. 2013.

OPENFLOW Reference. Disponível em: <http://www.openflow.org/wp/downloads/>. Acesso em: 17 jul. 2013.

OpenWRT. *OpenFlow 1.3 for OpenWRT*. Disponível em: <https://github.com/CPqD/ofsoftswitch13/wiki/OpenFlow-1.3-for-OpenWRT>. Acesso em: 20 ago. 2013.

OPNET. *Application and Network Performance*. Disponível em: <http://www.opnet.com>. Acesso em: 20 jul. 2013.

Ovs-controller. Disponível em: <http://openvswitch.org/cgi-bin/ovsman.cgi?page=utilities%2Fovs-controller.8/>. Acesso em: 17 mar. 2013.

PAUL, S.; PAN, J.; JAIN, R. Architectures for the future networks and the next generation internet: A survey. *Comput. Commun*., n.34, p.2–42, 2011.

PHP. Disponível em: <http://php.net/>. Acesso em: 4 out. 2013.

PLÁCIDO, R. "*Simulação de Redes com Multicasting e garantias de Qualidade de Serviço*", Dissertação de Mestrado em Engenharia em Telecomunicações e Redes. Universidade da Madeira, 2010.

PORTNOI, M. *Um Protótipo de simulador de redes de computadores para aplicações específicas baseadas no protocolo MPLS.* 2007. Dissertação (Mestrado)- Universidade de Salvador, 2007.

POX. *Open source control platform.* Disponível em: <http://www.noxrepo.org/pox/about-pox/>. Acesso em: 16 jul. 2013.

PREECE, J.; ROGERS, Y.; SHARP, H. *Design* de interação: além da interação homem-computador. 1. ed. Porto Alegre: Bookman, 2005.

PRESSMAN, R. S. *Engenharia de software*. 5.ed. São Paulo: Mac Graw Hill, 2002.

QUAGGA. *Quagga routing software Suite.* Disponível em: <http://www.nongnu.org/quagga/>. Acesso em: 1 ago. 2013.

ROTHENBERG, C. E. 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, p. 65-76, jul. 2010/jun. 2011.

RYU. *A component-based software-defined networking framework.* Disponível em: <http://osrg.github.io/ryu/>. Acesso em: 20 jul. 2013.

SHERWOOD, R. et al. Carving Research Slices Out of Your Production Networks with Openflow. SIGCOMM Comput. Commun. Rev., n.40, p.129–130, 2010.

SHERWOOD, R. et al. *FlowVisor:* a network virtualization layer. Technical Report Openflow-tr-2009-1, Stanford University, 2009.

SIMPY. SimPy Simulation Package. Disponível em: <http://simpy.sourceforge.net/>. Acesso em: 4 out. 2013.

TETCOS. *NetSimTM Academic Ver 2.1*. No. 2089, 1267 Lake Side Avenue, Sunnyvale, California, 94085 USA, 2011.

TIWARI, V. *SDN and OpenFlow for beginners with hands on labs*. [S.l]: [s.n.], 2013.

TOPOLOGY GENERATOR. Disponível em: <http://www.nsnam.org/wiki/index.php/Topology_Generator>. Acesso em: 4 set. 2013.

TORRES, G. *Redes de computadores curso completo.* [S.l]: Axcel Books do Brasil, 2001.

VMWARE. *Virtualization Software*. Disponível em: <http://www.vmware.com.br/>. Acesso em: 17 jul. 2013

VND. Disponível em: <http://apus.uma.pt/vnd/>. Acesso em: 17 jul. 2013.

WIRESHARK. Disponível em: <http://www.wireshark.org/>. Acesso em: 9 jul. 2013.

W3C. *Exntesible Markup Language (XML).* Disponível em: <http://www.w3.org/XML/>. Acesso em 13 jul. 2013.

APÊNDICE A - Publicações do autor

FONTES, R. R.; SAMPAIO, P. N. M. . Visual network description: a customizable gui for the creation of software defined network simulations. In: EUROPEAN SIMULATION AND MODELLING CONFERENCE, 2013. Lancaster, UK. *Proceedings.*… October, 2013.

FONTES, R. R.; FIGUEIRA, R. A. R. B.; SAMPAIO, P. N. M. Visual network description: facilitando o aprendizado sobre as redes definidas por software. In: CONGRESSO NORTE NORDESTE DE PESQUISA E INOVAÇÃO (CONNEPI), 2103. Salvador. *Anais*... 2013.

PINHEIRO, T.; FONTES, R. R.; SAMPAIO, P. N. M. Authoring and simulation of openflow scenarios based on NSDL and NS-3. In: INTERNATIONAL WORKSHOP ON ADVANCES IN ICT INFRASTRUCTURES AND SERVICES, 2013. Valença/BA, Brazil. *Proceedings.*… 2013.

PINHEIRO, T. et al.. *Heterogeneous authoring and simulation of openflow scenarios.* 2013. No prelo.

ANEXO A – Arquivo em NSDL (CENÁRIO NO. 01)

```
<nsdl>
  <network>
    <templates/>
    <objects>
      <link id="ethernet_10">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="switchOpenflow_4"
destination="controllerOpenflow_5"/>
      </link>
      <link id="ethernet_9">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="switchOpenflow_3"
destination="controllerOpenflow_5"/>
      </link>
      <link id="ethernet_8">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="switchOpenflow_3" destination="switchOpenflow_4"/>
      </link>
      <link id="ethernet_7">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="computer_2" destination="switchOpenflow_4"/>
      </link>
      <link id="ethernet_6">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="computer_1" destination="switchOpenflow_3"/>
      </link>
      <computer id="computer_1">
        <computerIPAddress>10.0.0.10</computerIPAddress>
```

```
      <mask>255.0.0.0</mask>
      <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
      <notes/>
    </computer>
    <computer id="computer_2">
      <computerIPAddress>10.0.0.20</computerIPAddress>
      <mask>255.0.0.0</mask>
      <computerMacAddress>00:00:00:00:00:02</computerMacAddress>
      <notes/>
    </computer>
    <switchOpenflow id="switchOpenflow_3">
      <switchMacAddress>00:00:00:00:00:03</switchMacAddress>
      <notes/>
    </switchOpenflow>
    <switchOpenflow id="switchOpenflow_4">
      <switchMacAddress>00:00:00:00:00:04</switchMacAddress>
      <notes/>
    </switchOpenflow>
    <controllerOpenflow id="controllerOpenflow_5">
      <controllerIPAddress>192.168.1.20</controllerIPAddress>
      <controllerPort>6633</controllerPort>
    </controllerOpenflow>
  </objects>
  <views/>
</network>
<scenarios>
  <visualizations>
    <visualization id="vis01">
      <description>
        <object id="computer_1">
          <x.position>258</x.position>
          <y.position>394</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="computer_2">
          <x.position>584</x.position>
          <y.position>411</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="switchOpenflow_3">
          <x.position>304</x.position>
          <y.position>264</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="switchOpenflow_4">
          <x.position>612</x.position>
          <y.position>266</y.position>
```

```
          <z.position>0</z.position>
          <color/>
         </object>
         <object id="controllerOpenflow_5">
          <x.position>500</x.position>
          <y.position>174</y.position>
          <z.position>0</z.position>
          <color/>
         </object>
        </description>
      </visualization>
    </visualizations>
    <simulations/>
    <openflow/>
  </scenarios>
</nsdl>
```

ANEXO B– Arquivo em NSDL (CENÁRIO NO. 02)

```
<nsdl>
  <network>
    <templates/>
    <objects>
      <link id="ethernet_14">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="switchOpenflow_6"
destination="controllerOpenflow_7"/>
      </link>
      <link id="ethernet_13">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="switchOpenflow_5"
destination="controllerOpenflow_7"/>
      </link>
      <link id="ethernet_12">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="switchOpenflow_5" destination="switchOpenflow_6"/>
      </link>
      <link id="ethernet_11">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="switchOpenflow_6" destination="computer_4"/>
      </link>
      <link id="ethernet_10">
        <bandwidth>undefined</bandwidth>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb>undefined</htb>
        <connection source="switchOpenflow_6" destination="computer_3"/>
      </link>
      <link id="ethernet_9">
```

```
  <bandwidth>undefined</bandwidth>
  <delay/>
  <loss/>
  <maxqueue/>
  <htb>undefined</htb>
  <connection source="switchOpenflow_5" destination="computer_1"/>
</link>
<link id="ethernet_8">
  <bandwidth>undefined</bandwidth>
  <delay/>
  <loss/>
  <maxqueue/>
  <htb>undefined</htb>
  <connection source="computer_2" destination="switchOpenflow_5"/>
</link>
<computer id="computer_1">
  <computerIPAddress>10.0.0.10</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
  <notes/>
</computer>
<computer id="computer_2">
  <computerIPAddress>10.0.0.20</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:02</computerMacAddress>
  <notes/>
</computer>
<computer id="computer_3">
  <computerIPAddress>10.0.0.30</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:03</computerMacAddress>
  <notes/>
</computer>
<computer id="computer_4">
  <computerIPAddress>10.0.0.40</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:04</computerMacAddress>
  <notes/>
</computer>
<switchOpenflow id="switchOpenflow_5">
  <switchMacAddress>00:00:00:00:00:05</switchMacAddress>
  <notes/>
</switchOpenflow>
<switchOpenflow id="switchOpenflow_6">
  <switchMacAddress>00:00:00:00:00:06</switchMacAddress>
  <notes/>
</switchOpenflow>
<controllerOpenflow id="controllerOpenflow_7">
  <controllerIPAddress>127.0.0.1</controllerIPAddress>
  <controllerPort>6633</controllerPort>
```

```
    </controllerOpenflow>
  </objects>
  <views/>
</network>
<scenarios>
  <visualizations>
    <visualization id="vis01">
      <description>
        <object id="computer_1">
          <x.position>99</x.position>
          <y.position>36</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="computer_2">
          <x.position>97</x.position>
          <y.position>216</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="computer_3">
          <x.position>644</x.position>
          <y.position>211</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="computer_4">
          <x.position>669</x.position>
          <y.position>40</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="switchOpenflow_5">
          <x.position>197</x.position>
          <y.position>124</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="switchOpenflow_6">
          <x.position>440</x.position>
          <y.position>118</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="controllerOpenflow_7">
          <x.position>349</x.position>
          <y.position>19</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
```

```
    </description>
  </visualization>
</visualizations>
<simulations/>
<openflow>
  <flowTable1 id="controllerOpenflow_7">
    <flowName>flow0</flowName>
    <priority>32768</priority>
    <macSwitch>00:00:00:00:00:05</macSwitch>
    <macSource/>
    <macDestination/>
    <ingressPort>2</ingressPort>
    <vlanID/>
    <vlanPriority/>
    <ethtype/>
    <tos/>
    <ipSource/>
    <ipDestination/>
    <sourcePort/>
    <destinationPort/>
    <setIPSource/>
    <setIPDestination/>
    <setMACSource/>
    <setMACDestination/>
    <setSourcePort/>
    <setDestinationPort/>
    <setVlanID>10</setVlanID>
    <setOutput>3</setOutput>
    <setVLANPriority/>
    <setStripVlan/>
    <setEnqueue/>
    <notes/>
  </flowTable1>
  <flowTable2 id="controllerOpenflow_7">
    <flowName>flow1</flowName>
    <priority>32768</priority>
    <macSwitch>00:00:00:00:00:05</macSwitch>
    <macSource/>
    <macDestination/>
    <ingressPort>1</ingressPort>
    <vlanID/>
    <vlanPriority/>
    <ethtype/>
    <tos/>
    <ipSource/>
    <ipDestination/>
    <sourcePort/>
    <destinationPort/>
    <setIPSource/>
    <setIPDestination/>
```

```
  <setMACSource/>
  <setMACDestination/>
  <setSourcePort/>
  <setDestinationPort/>
  <setVlanID>20</setVlanID>
  <setOutput>3</setOutput>
  <setVLANPriority/>
  <setStripVlan/>
  <setEnqueue/>
  <notes/>
</flowTable2>
<flowTable3 id="controllerOpenflow_7">
  <flowName>flow2</flowName>
  <priority>32768</priority>
  <macSwitch>00:00:00:00:00:05</macSwitch>
  <macSource/>
  <macDestination/>
  <ingressPort/>
  <vlanID>10</vlanID>
  <vlanPriority/>
  <ethtype/>
  <tos/>
  <ipSource/>
  <ipDestination/>
  <sourcePort/>
  <destinationPort/>
  <setIPSource/>
  <setIPDestination/>
  <setMACSource/>
  <setMACDestination/>
  <setSourcePort/>
  <setDestinationPort/>
  <setVlanID/>
  <setOutput>2</setOutput>
  <setVLANPriority/>
  <setStripVlan>yes</setStripVlan>
  <setEnqueue/>
  <notes/>
</flowTable3>
<flowTable4 id="controllerOpenflow_7">
  <flowName>flow3</flowName>
  <priority>32768</priority>
  <macSwitch>00:00:00:00:00:05</macSwitch>
  <macSource/>
  <macDestination/>
  <ingressPort/>
  <vlanID>20</vlanID>
  <vlanPriority/>
  <ethtype/>
  <tos/>
```

```
  <ipSource/>
  <ipDestination/>
  <sourcePort/>
  <destinationPort/>
  <setIPSource/>
  <setIPDestination/>
  <setMACSource/>
  <setMACDestination/>
  <setSourcePort/>
  <setDestinationPort/>
  <setVlanID/>
  <setOutput>1</setOutput>
  <setVLANPriority/>
  <setStripVlan>yes</setStripVlan>
  <setEnqueue/>
  <notes/>
 </flowTable4>
 <flowTable5 id="controllerOpenflow_7">
  <flowName>flow4</flowName>
  <priority>32768</priority>
  <macSwitch>00:00:00:00:00:06</macSwitch>
  <macSource/>
  <macDestination/>
  <ingressPort>1</ingressPort>
  <vlanID/>
  <vlanPriority/>
  <ethtype/>
  <tos/>
  <ipSource/>
  <ipDestination/>
  <sourcePort/>
  <destinationPort/>
  <setIPSource/>
  <setIPDestination/>
  <setMACSource/>
  <setMACDestination/>
  <setSourcePort/>
  <setDestinationPort/>
  <setVlanID>10</setVlanID>
  <setOutput>3</setOutput>
  <setVLANPriority/>
  <setStripVlan/>
  <setEnqueue/>
  <notes/>
 </flowTable5>
 <flowTable6 id="controllerOpenflow_7">
  <flowName>flow5</flowName>
  <priority>32768</priority>
  <macSwitch>00:00:00:00:00:06</macSwitch>
  <macSource/>
```

```
  <macDestination/>
  <ingressPort>2</ingressPort>
  <vlanID/>
  <vlanPriority/>
  <ethtype/>
  <tos/>
  <ipSource/>
  <ipDestination/>
  <sourcePort/>
  <destinationPort/>
  <setIPSource/>
  <setIPDestination/>
  <setMACSource/>
  <setMACDestination/>
  <setSourcePort/>
  <setDestinationPort/>
  <setVlanID>20</setVlanID>
  <setOutput>3</setOutput>
  <setVLANPriority/>
  <setStripVlan/>
  <setEnqueue/>
  <notes/>
</flowTable6>
<flowTable7 id="controllerOpenflow_7">
  <flowName>flow6</flowName>
  <priority>32768</priority>
  <macSwitch>00:00:00:00:00:06</macSwitch>
  <macSource/>
  <macDestination/>
  <ingressPort/>
  <vlanID>10</vlanID>
  <vlanPriority/>
  <ethtype/>
  <tos/>
  <ipSource/>
  <ipDestination/>
  <sourcePort/>
  <destinationPort/>
  <setIPSource/>
  <setIPDestination/>
  <setMACSource/>
  <setMACDestination/>
  <setSourcePort/>
  <setDestinationPort/>
  <setVlanID/>
  <setOutput>1</setOutput>
  <setVLANPriority/>
  <setStripVlan>yes</setStripVlan>
  <setEnqueue/>
  <notes/>
```

```
      </flowTable7>
      <flowTable8 id="controllerOpenflow_7">
        <flowName>flow7</flowName>
        <priority>32768</priority>
        <macSwitch>00:00:00:00:00:06</macSwitch>
        <macSource/>
        <macDestination/>
        <ingressPort/>
        <vlanID>20</vlanID>
        <vlanPriority/>
        <ethtype/>
        <tos/>
        <ipSource/>
        <ipDestination/>
        <sourcePort/>
        <destinationPort/>
        <setIPSource/>
        <setIPDestination/>
        <setMACSource/>
        <setMACDestination/>
        <setSourcePort/>
        <setDestinationPort/>
        <setVlanID/>
        <setOutput>2</setOutput>
        <setVLANPriority/>
        <setStripVlan>yes</setStripVlan>
        <setEnqueue/>
        <notes/>
      </flowTable8>
    </openflow>
  </scenarios>
</nsdl>
```

ANEXO C – Arquivo em NSDL (CENÁRIO NO. 03)

```
<nsdl>
  <network>
    <templates/>
    <objects>
      <link id="ethernet_10">
        <bandwidth/>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb/>
        <connection source="switchOpenflow_4"
destination="controllerOpenflow_5"/>
      </link>
      <link id="ethernet_9">
        <bandwidth/>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb/>
        <connection source="switchOpenflow_3"
destination="controllerOpenflow_5"/>
      </link>
      <link id="ethernet_8">
        <bandwidth/>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb/>
        <connection source="switchOpenflow_3" destination="switchOpenflow_4"/>
      </link>
      <link id="ethernet_7">
        <bandwidth/>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb/>
        <connection source="computer_2" destination="switchOpenflow_4"/>
      </link>
      <link id="ethernet_6">
        <bandwidth/>
        <delay/>
        <loss/>
        <maxqueue/>
        <htb/>
        <connection source="computer_1" destination="switchOpenflow_3"/>
      </link>
      <computer id="computer_1">
        <computerIPAddress>10.0.0.10</computerIPAddress>
```

```
      <mask>255.0.0.0</mask>
      <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
      <notes/>
    </computer>
    <computer id="computer_2">
      <computerIPAddress>10.0.0.20</computerIPAddress>
      <mask>255.0.0.0</mask>
      <computerMacAddress>00:00:00:00:00:02</computerMacAddress>
      <notes/>
    </computer>
    <switchOpenflow id="switchOpenflow_3">
      <switchMacAddress>00:00:00:00:00:03</switchMacAddress>
      <notes/>
    </switchOpenflow>
    <switchOpenflow id="switchOpenflow_4">
      <switchMacAddress>00:00:00:00:00:04</switchMacAddress>
      <notes/>
    </switchOpenflow>
    <controllerOpenflow id="controllerOpenflow_5">
      <controllerIPAddress>192.168.1.20</controllerIPAddress>
      <controllerPort>6633</controllerPort>
    </controllerOpenflow>
  </objects>
  <views/>
</network>
<scenarios>
  <visualizations>
    <visualization id="vis01">
      <description>
        <object id="computer_1">
          <x.position>258</x.position>
          <y.position>394</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="computer_2">
          <x.position>584</x.position>
          <y.position>411</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="switchOpenflow_3">
          <x.position>304</x.position>
          <y.position>264</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="switchOpenflow_4">
          <x.position>612</x.position>
          <y.position>266</y.position>
```

```
        <z.position>0</z.position>
        <color/>
       </object>
       <object id="controllerOpenflow_5">
        <x.position>500</x.position>
        <y.position>174</y.position>
        <z.position>0</z.position>
        <color/>
       </object>
     </description>
   </visualization>
</visualizations>
<simulations/>
<openflow>
  <qosTable0 id="controllerOpenflow_5">
    <qosName>Qos1</qosName>
    <sourceIP/>
    <destinationIP/>
    <ethType>0x0800</ethType>
    <protocol>6</protocol>
    <queue>0</queue>
  </qosTable0>
  <qosTable1 id="controllerOpenflow_5">
    <qosName>Qos2</qosName>
    <sourceIP/>
    <destinationIP/>
    <ethType>0x0800</ethType>
    <protocol>6</protocol>
    <queue>1</queue>
  </qosTable1>
  <queueConfiguration0 id="controllerOpenflow_5">
    <qosInterface>s3-eth1</qosInterface>
    <qosInterface>s3-eth2</qosInterface>
    <qosInterface>s4-eth1</qosInterface>
    <qosInterface>s4-eth2</qosInterface>
    <scenarioMaxRate>1000000000</scenarioMaxRate>
    <scenarioMinRate/>
    <queueName>Queue 0</queueName>
    <linkmaxRate>5000000</linkmaxRate>
    <linkminRate/>
  </queueConfiguration0>
  <queueConfiguration1 id="controllerOpenflow_5">
    <qosInterface>s3-eth1</qosInterface>
    <qosInterface>s3-eth2</qosInterface>
    <qosInterface>s4-eth1</qosInterface>
    <qosInterface>s4-eth2</qosInterface>
    <scenarioMaxRate>1000000000</scenarioMaxRate>
    <scenarioMinRate/>
    <queueName>Queue 1</queueName>
    <linkmaxRate>1000000</linkmaxRate>
```

```
        <linkminRate/>
      </queueConfiguration1>
    </openflow>
  </scenarios>
</nsdl>
```

ANEXO D – Arquivo em NSDL (CENÁRIO NO. 04)

```
<nsdl>
 <network>
  <templates/>
  <objects>
   <link id="ethernet_7">
    <delay/>
    <loss/>
    <maxqueue/>
    <connection source="wirelessRouter_2" destination="controllerOpenflow_1"/>
   </link>
   <link id="wireless_6">
    <delay/>
    <loss/>
    <maxqueue/>
    <connection source="wirelessRouter_2" destination="computer_4"/>
   </link>
   <link id="ethernet_5">
    <delay/>
    <loss/>
    <maxqueue/>
    <connection source="computer_3" destination="wirelessRouter_2"/>
   </link>
   <controllerOpenflow id="controllerOpenflow_1">
    <controllerIPAddress>192.168.1.100</controllerIPAddress>
    <controllerPort>6633</controllerPort>
    <specialController>none</specialController>
    <openflowController>pox</openflowController>
   </controllerOpenflow>
   <wirelessRouter id="wirelessRouter_2">
    <switchMacAddress>00:00:00:00:00:02</switchMacAddress>
    <notes/>
   </wirelessRouter>
   <computer id="computer_3">
    <computerIPAddress>10.0.0.3</computerIPAddress>
    <mask>255.0.0.0</mask>
    <computerMacAddress>2C:76:8A:BB:47:9D</computerMacAddress>
    <notes/>
   </computer>
   <computer id="computer_4">
    <computerIPAddress>10.0.0.4</computerIPAddress>
    <mask>255.0.0.0</mask>
    <computerMacAddress> D8:D3:85:3E:30:65</computerMacAddress>
    <notes/>
   </computer>
  </objects>
  <views/>
 </network>
 <scenarios>
```

```
<visualizations>
  <visualization id="vis01">
    <description>
      <object id="controllerOpenflow_1">
        <x.position>295</x.position>
        <y.position>39</y.position>
        <z.position>0</z.position>
        <color/>
      </object>
      <object id="wirelessRouter_2">
        <x.position>316</x.position>
        <y.position>179</y.position>
        <z.position>0</z.position>
        <color/>
      </object>
      <object id="computer_3">
        <x.position>229</x.position>
        <y.position>336</y.position>
        <z.position>0</z.position>
        <color/>
      </object>
      <object id="computer_4">
        <x.position>663</x.position>
        <y.position>232</y.position>
        <z.position>0</z.position>
        <color/>
      </object>
    </description>
  </visualization>
</visualizations>
<simulations/>
<openflow>
  <flowTable1 id="controllerOpenflow_1">
    <flowName>flow0</flowName>
    <priority>32768</priority>
    <macSwitch>00:00:00:00:00:02</macSwitch>
    <macSource>2C:76:8A:BB:47:9D</macSource>
    <macDestination/>
    <ingressPort>1</ingressPort>
    <vlanID/>
    <vlanPriority/>
    <ethtype/>
    <tos/>
    <ipSource/>
    <ipDestination/>
    <sourcePort/>
    <destinationPort/>
    <setIPSource/>
    <setIPDestination/>
    <setMACSource/>
```

```
        <setMACDestination/>
        <setSourcePort/>
        <setDestinationPort/>
        <setVlanID/>
        <setOutput>5</setOutput>
        <setVLANPriority/>
        <setStripVlan/>
        <setEnqueue/>
        <notes/>
      </flowTable1>
      <flowTable2 id="controllerOpenflow_1">
        <flowName>flow2</flowName>
        <priority>32768</priority>
        <macSwitch>00:00:00:00:00:02</macSwitch>
        <macSource>D8:D3:85:3E:30:65</macSource>
        <macDestination/>
        <ingressPort>5</ingressPort>
        <vlanID/>
        <vlanPriority/>
        <ethtype/>
        <tos/>
        <ipSource/>
        <ipDestination/>
        <sourcePort/>
        <destinationPort/>
        <setIPSource/>
        <setIPDestination/>
        <setMACSource/>
        <setMACDestination/>
        <setSourcePort/>
        <setDestinationPort/>
        <setVlanID/>
        <setOutput>1</setOutput>
        <setVLANPriority/>
        <setStripVlan/>
        <setEnqueue/>
        <notes/>
      </flowTable2>
    </openflow>
  </scenarios>
</nsdl>
```

ANEXO E – Script para o Controlador POX (CENÁRIO NO. 02)

```
#!/usr/bin/python
#Script criado pelo VND - Visual Network Description (versão SDN)

from pox.core import core
from pox.lib.addresses import IPAddr
from pox.lib.addresses import EthAddr
import pox.openflow.libopenflow_01 as of

log = core.getLogger()
```

O flow0 instrui ao controlador a definir os pacotes que entram pela porta 2 do comutador com ID ‘0000000000000005’, a serem marcados com Vlan tag 10 ao saírem pela porta 3.

```
# flow0:
switch0 = 0000000000000005
flow0msg = of.ofp_flow_mod()
flow0msg.cookie = 0
flow0msg.priority = 32768
flow0msg.match.in_port = 2
# ACTIONS---------------------------------
flow0vlan_id = of.ofp_action_vlan_vid (vlan_vid = 10)
flow0out = of.ofp_action_output (port = 3)
flow0msg.actions = [flow0vlan_id, flow0out]
```

O flow1 instrui ao controlador a definir os pacotes que entram pela porta 1 do comutador com ID ‘0000000000000005’, a serem marcados com Vlan tag 20 ao saírem pela porta 3.

```
# flow1:
switch1 = 0000000000000005
flow1msg = of.ofp_flow_mod()
flow1msg.cookie = 0
flow1msg.priority = 32768
flow1msg.match.in_port = 1
# ACTIONS---------------------------------
flow1vlan_id = of.ofp_action_vlan_vid (vlan_vid = 20)
flow1out = of.ofp_action_output (port = 3)
flow1msg.actions = [flow1vlan_id, flow1out]
```

O flow2 instrui ao controlador a definir os pacotes que entram com Vlan ID 10 no comutador com ID ‘0000000000000005’, a saírem pela porta 2 com a retirada da Vlan tag 10.

```
# flow2:
switch2 = 0000000000000005
flow2msg = of.ofp_flow_mod()
flow2msg.cookie = 0
flow2msg.priority = 32768
flow2msg.match.dl_vlan = 10
# ACTIONS---------------------------------
flow2stripvlan = of.ofp_action_strip_vlan ()
flow2out = of.ofp_action_output (port = 2)
flow2msg.actions = [flow2stripvlan, flow2out]
```

O flow3 instrui ao controlador a definir os pacotes que entram com Vlan ID 20 no comutador com ID ‘0000000000000005’, a saírem pela porta 1 com a retirada da Vlan tag 10.

```
# flow3:
switch3 = 0000000000000005
flow3msg = of.ofp_flow_mod()
flow3msg.cookie = 0
flow3msg.priority = 32768
flow3msg.match.dl_vlan = 20
# ACTIONS---------------------------------
flow3stripvlan = of.ofp_action_strip_vlan ()
flow3out = of.ofp_action_output (port = 1)
flow3msg.actions = [flow3stripvlan, flow3out]
```

O flow4 instrui ao controlador a definir os pacotes que entram pela porta 1 do comutador com ID ‘0000000000000006’, a serem marcados com Vlan tag 10 ao saírem pela porta 3.

```
# flow4:
switch4 = 0000000000000006
flow4msg = of.ofp_flow_mod()
flow4msg.cookie = 0
flow4msg.priority = 32768
flow4msg.match.in_port = 1
# ACTIONS---------------------------------
flow4vlan_id = of.ofp_action_vlan_vid (vlan_vid = 10)
flow4out = of.ofp_action_output (port = 3)
flow4msg.actions = [flow4vlan_id, flow4out]
```

O flow5 instrui ao controlador a definir os pacotes que entram pela porta 2 do comutador com ID ‘0000000000000006’, a serem marcados com Vlan tag 20 ao saírem pela porta 3.

```
# flow5:
switch5 = 0000000000000006
flow5msg = of.ofp_flow_mod()
flow5msg.cookie = 0
flow5msg.priority = 32768
flow5msg.match.in_port = 2
# ACTIONS---------------------------------
flow5vlan_id = of.ofp_action_vlan_vid (vlan_vid = 20)
flow5out = of.ofp_action_output (port = 3)
flow5msg.actions = [flow5vlan_id, flow5out]
```

O flow6 instrui ao controlador a definir os pacotes que entram com Vlan ID 10 no comutador com ID ‘0000000000000006’, a saírem pela porta 1 com a retirada da Vlan tag 10.

```
# flow6:
Switch6 = 0000000000000006
flow6msg = of.ofp_flow_mod()
flow6msg.cookie = 0
flow6msg.priority = 32768
flow6msg.match.dl_vlan = 10
# ACTIONS---------------------------------
flow6stripvlan = of.ofp_action_strip_vlan ()
flow6out = of.ofp_action_output (port = 1)
flow6msg.actions = [flow6stripvlan, flow6out]
```

O flow7 instrui ao controlador a definir os pacotes que entram com Vlan ID 20 no comutador com ID ‘0000000000000006’, a saírem pela porta 2 com a retirada da Vlan tag 10.

```
# flow7:
switch7 = 0000000000000006
flow7msg = of.ofp_flow_mod()
flow7msg.cookie = 0
flow7msg.priority = 32768
flow7msg.match.dl_vlan = 20
# ACTIONS---------------------------------
flow7stripvlan = of.ofp_action_strip_vlan ()
```

```
flow7out = of.ofp_action_output (port = 2)
flow7msg.actions = [flow7stripvlan, flow7out]
```

A função *install_flows()* é responsável pela aplicação dos fluxos previamente configurados.

```
def install_flows():
  log.info("    *** Instalando fluxos estáticos... ***")
  # Instalando fluxos nos switches
  core.openflow.sendToDPID(switch0, flow0msg)
  core.openflow.sendToDPID(switch1, flow1msg)
  core.openflow.sendToDPID(switch2, flow2msg)
  core.openflow.sendToDPID(switch3, flow3msg)
  core.openflow.sendToDPID(switch4, flow4msg)
  core.openflow.sendToDPID(switch5, flow5msg)
  core.openflow.sendToDPID(switch6, flow6msg)
  core.openflow.sendToDPID(switch7, flow7msg)
  log.info("    *** Fluxos estáticos instalados. ***")
```

A função *launch()* é responsável por chamar a função install_flows.

```
def launch():
  log.info("*** Iniciando... ***")
  core.callDelayed (15, install_flows)
        log.info("*** Esperando pela conexão aos comutadores.. ***")
```