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

LUCAS LUCIAN BORGES DE SOUZA

APLICAÇÃO DA FRAMEWORK NSDL PARA A DESCRIÇÃO DE CENÁRIOS DE REDES OPENFLOW

Salvador
2014

LUCAS LUCIAN BORGES DE SOUZA

APLICAÇÃO DA FRAMEWORK NSDL PARA A DESCRIÇÃO DE CENÁRIOS DE REDES OPENFLOW

Dissertação apresentada ao curso de Mestrado Profissional em Sistemas e Computação, da Universidade Salvador - UNIFACS, como requisito parcial para obtenção do título de Mestre.

Orientador: Prof. Dr. Paulo Nazareno Maia Sampaio.

Salvador
2014

FICHA CATALOGRÁFICA
Elaborada pelo Sistema de Bibliotecas da Universidade Salvador – UNIFACS

Souza, Lucas Lucian Borges de

Aplicação da framework NSDL para a descrição de cenários de redes openflow/ Lucas Lucian Borges de Souza. – Salvador, 2014.

99 f. : il.

Dissertação (Mestrado Profissional em Sistemas de Computação) - Universidade Salvador – UNIFACS.
Orientador: Prof. Dr. Paulo Nazareno Maia Sampaio.

1. Interoperabilidade. 2. OpenFlow. I. Sampaio, Paulo Nazareno Maia, orient. II. Título.

CDD: 004.6

LUCAS LUCIAN BORGES DE SOUZA

APLICAÇÃO DA FRAMEWORK NSDL PARA A DESCRIÇÃO DE CENÁRIOS DE REDES OPENFLOW

Dissertação, Mestrado Profissional em Sistemas e Computação da Universidade Salvador - UNIFACS, como requisito parcial para a obtenção do título de Mestre em Sistemas e Computação.

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

Sidney Viana _____
Doutor em Engenharia Elétrica pela Universidade de São Paulo - USP
Centro Universitário Dinâmica das Cataratas

Joberto Sérgio Barbosa Martins –_____
Doutor em Ciência da Computação, Université Paris VI
Universidade Salvador - UNIFACS

Artur Henrique Kronbauer _____
Doutor em Ciência da Computação, na área de Interação Humano-Computador pela Universidade Federal da Bahia - UFBA
Universidade Salvador – UNIFACS

Salvador, 20 de maio de 2014.

RESUMO

A constante evolução tecnológica e o desenvolvimento das telecomunicações levam ao surgimento de novos equipamentos e à proposta de novos padrões e protocolos para redes de comunicação. Daí a importância da existência de ambientes de testes mais dinâmicos e realísticos, sem que esses testes interfiram no ambiente operacional dessas redes. No entanto, a implementação desses testes pode representar uma atividade muito complexa e essa é uma das razões que motivaram a proposta do paradigma de *Redes Definidas por Software* foi proposto sendo uma de suas premissas possibilitar o experimento de novos protocolos em redes de computadores. O protocolo *OpenFlow* é uma das propostas para implementação de *Redes Definidas por Software* que visa a introdução de um modelo computacional às redes de dados através da separação do plano de controle e do plano de dados nos comutadores de redes. Como resultado, é possível a execução de um experimento sobre uma rede real sem interferir no tratamento do tráfego de produção, reduzindo assim o custo operacional da rede. Atualmente existem diferentes ferramentas e abordagens para a implementação e monitoração de testes *OpenFlow*, e nessa perspectiva, surge igualmente a necessidade de propostas que permitam uma gerência dessas redes de uma forma mais integrada. No entanto, no geral essas ferramentas não permitem a troca de informações entre elas, sendo necessária a existência de soluções que promovam a interoperabilidade entre essas ferramentas. Uma dessas soluções é a *Framework Network Scenario Description Language - NSDL* (cujo nome foi definido originalmente em inglês), que foi proposta na Universidade da Madeira (Portugal) de forma a proporcionar a interoperabilidade entre diferentes ferramentas de gestão de redes. Essa *Framework* baseia-se em uma linguagem comum (também chamada de *NSDL*) que permite a descrição dos cenários de redes e o seu intercâmbio entre diferentes ferramentas. Os cenários de redes permitem representar textualmente ou graficamente uma topologia de rede, ou seja, representa todos os componentes da rede: computadores, comutadores, protocolos e informação sobre o tráfego, etc. Inclui também uma descrição dos objetos e dos seus parâmetros de rede num determinado contexto. O objetivo deste trabalho foi a proposta de uma extensão da linguagem *NSDL* para a descrição de cenários de rede *OpenFlow*. A contribuição deste trabalho foi viabilizada através do estudo e a análise de ambientes em *Redes Definidas por Software* utilizando o paradigma *OpenFlow* o que permitiu identificar e documentar os seus principais requisitos funcionais e não-funcionais. De forma a validar a linguagem *NSDL OpenFlow* proposta, foram realizados alguns estudos de caso.

Palavras-chave: *OpenFlow*. Redes Definidas por Software. NSDL. Interoperabilidade.

ABSTRACT

The constant technological evolution and development of telecommunication lead to the proposal of new equipment, standards and protocols for communication networks. Therefore, it is important to provide more dynamic and realistic, testing environments where the current experiments are not allowed to interfere in the operational network. Nevertheless, the implementation of these experiments can be a very complex activity, for this reason, in order to allow the experimentation of new protocols in computer networks, the paradigm of programmable virtualized networks (so-called Software-Defined Networks) has been proposed. *OpenFlow* is one of the proposals for the implementation of Software-Defined Networks. The main goal of *OpenFlow* is to allow an experiment to be executed in a real network allowing reprogramming the switches so that they can handle the traffic from the experiments without interfering in the operation of the production network. Currently there are different tools and approaches for the implementation and monitoring of *OpenFlow* experiments. From this perspective, there is also the need of proposals that allow the integrated management of these networks providing the interoperability between these tools. One possible solution is the adoption of the *NSDL Framework* (*Network Scenarios Description Language*), which has been proposed at University of Madeira (Portugal) to provide interoperability between different tools for network management. This *Framework* is based on a common language (also called *NSDL*) that allows the description of the network scenarios that can be exchanged between different tools. The network *Scenarios* allow textually or graphically to represent a network topology, that is, all the network components: computers, routers, protocols, traffic information, etc. It also includes a description of the objects and their network parameters in a given context. The main goal of this work is the proposal of an extension to *NSDL* in order to support the description of *OpenFlow* scenarios. The contribution of this work was enabled by the study and analysis of Software-Defined Networks, which allowed the identification of functional and non-functional requirements. Further on, the proposed language was also validated by the implementation of some case studies.

Keywords: *OpenFlow. NSDL. Software* Defined Networks. Interoperability.

AGRADECIMENTOS

Desejo registrar os meus mais sinceros agradecimentos por tudo que foi conquistador até a conclusão deste trabalho. Primeiramente gostaria de agradecer a Deus, pois sem ele não seria possível a realização de nada em nossa vida.

Em seguida agradeço ao meu orientador Paulo Sampaio, sempre atencioso e paciente com todos os alunos, um exemplo de professor, educador e amigo.

Aos meus familiares agradeço pela paciência, cooperação, apoio e confiança que colaboram para que eu possa realizar todos os meus objetivos de vida. Um agradecimento especial para minha mãe, meu pai e minha esposa, amo muito todos vocês.

LISTAS DE ABREVIATURAS E SIGLAS

DiffServ	Differentiated Services
DNS	Domain Name System
ETH	Ethernet
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
IPv4	Internet Protocol Version 4
LTE	Long Term Evolution
NSDL	Network Scenarios Description Language
NS-2	Network Simulations Version 2
NS-3	Network Simulations Version 3
RTP	Real-time Transport Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VoIP	Voice over Internet Protocol
XML	eXtensible Markup Language
WiMAX	Worldwide Interoperability for Microwave Access

LISTAS DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são discutidos o contexto de desenvolvimento deste trabalho de mestrado, os problemas identificados e a solução proposta. Por fim, a organização do trabalho é apresentada.

1.1 CONTEXTUALIZAÇÃO

A constante evolução tecnológica e o desenvolvimento das telecomunicações levam ao surgimento de uma nova gama de equipamentos e à proposta de novos padrões e protocolos para redes de comunicação. Portanto, é importante a existência de ambientes de testes mais dinâmicos e realísticos, sem que esses testes interfiram no ambiente operacional dessas redes.

No entanto, o teste de novas tecnologias e protocolos pode representar uma atividade muito complexa, pois seria necessária uma intervenção direta em todos os equipamentos da rede para garantir o suporte a esses novos padrões. Por essa razão, considera-se que a infra-estrutura pública e privada de rede de computadores ainda não é flexível para a realização desses testes.

Uma possível solução para viabilizar o experimento de novos protocolos em redes de computadores sem afetar a rede de produção é o paradigma de *Redes Definidas por Software*. As *Redes Definidas por Software* têm como um de seus objetivos a separação do plano de dados e do plano de controle nos seus comutadores, oferecendo uma maior flexibilidade ao tratamento dos pacotes de tráfego experimental. Dessa forma, é possível viabilizar a realização de experimentos sobre uma rede real de produção sem interferir no tráfego da mesma.

Uma outra consequência da separação do plano de controle e de dados é a possibilidade dos fabricantes adicionarem as suas funcionalidades aos seus comutadores sem necessitarem expor o projeto de hardware desses equipamentos.

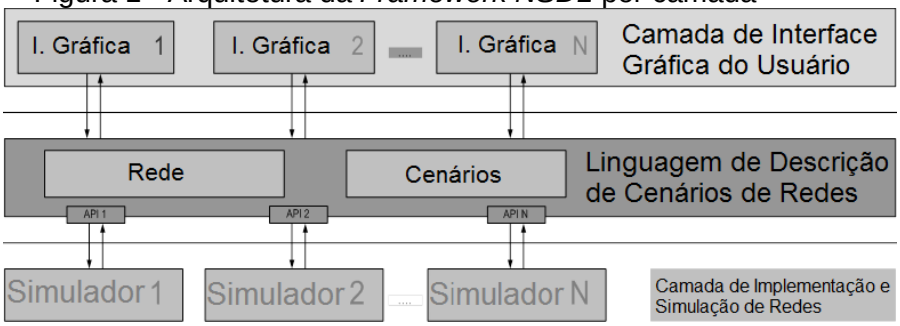
Dentre as propostas para implementação de *Redes Definidas por Software*, podemos citar o protocolo *OpenFlow* (MCKEOWN et al., 2008). O protocolo *OpenFlow* tem como objetivo ser flexível, proporcionar implementações de baixo custo e alto desempenho, entre outros requisitos (MCKEOWN et al., 2008).

Atualmente existem diferentes ferramentas e abordagens para a implementação e monitoração de redes *OpenFlow*, tais como: *OpenFlow in Europe: Linking Infrastructure and Applications* – OFELIA *(OPENFLOW IN EUROPE*, 2011*)*, *OpenFLow Operations Per Second* (OFLOPS, 2012), *OpenFlow Wireshark dissector* (*OPENFLOW* WIRESHARK DISSERCTOR, 2012), etc. Nessa perspectiva, surge igualmente a necessidade de propostas que permitam uma gerência dessas redes de uma forma mais integrada e otimizada, através da interoperabilidade entre essas ferramentas (por exemplo, pela partilha da descrição do cenário de rede a ser estudado ou monitorado, e dos resultados desse monitoramento entre ferramentas de criação dos testes de redes, monitoração, simulação, visualização e virtualização). No contexto desta dissertação, um cenário de rede descreve todos dos componentes ativos de comunicação de uma rede, suas interligações, os protocolos utilizados, assim como a origem e destino dos tráfegos de dados em um dado contexto de utilização.

No entanto, no geral essas ferramentas não são interoperáveis, ou seja, elas não comunicam entre si os resultados das suas operações. Portanto, é necessária a existência de soluções que promovam a interoperabilidade entre as ferramentas de gestão de redes. Uma dessas soluções é a *Framework Network Scenario Description Language – NSDL (originalmente em inglês)*, que foi proposta na Universidade da Madeira (Portugal) de forma a proporcionar a interoperabilidade entre diferentes ferramentas de gestão de redes (MARQUES, 2013). Essa solução foi adotada no contexto deste trabalho dada a sua simplicidade e generalidade para descrever diferentes tecnologias e domínios de rede.

A *Framework NSDL* é baseada em uma linguagem (também chamada de NSDL) que permite descrever os cenários de redes e intercambiá-los entre diferentes ferramentas. Os cenários de redes consistem na topologia de rede e na descrição de todos os componentes da rede: computadores, comutadores, protocolos e informação sobre o tráfego, etc. Eles podem também incluir uma descrição dos objetos e dos seus parâmetros de rede num determinado contexto. A *Framework NSDL* é ilustrada na Figura 1.

Figura 1 - Arquitetura da *Framework NSDL* por camada



Fonte: Marques (2013).

Na Figura 1 é possível verificar a arquitetura em camadas da *Framework NSDL*, a camada superior (*Camada de Interface Gráfica do Usuário*) comporta a utilização integrada de diferentes ferramentas de rede gráficas, tais como criação de topologia, monitoração e visualização de cenários de rede. A camada inferior (*Camada de Implementação e Simulação de Redes*) contempla as ferramentas de simulação e de virtualização. Por fim, a camada intermediária (*Linguagem de Descrição de Cenários de Redes*) é responsável por efetuar a ligação entre as camadas superiores e inferiores da arquitetura *NSDL*. Esta camada é constituída pela linguagem de descrição de redes *NSDL*, que possui dois elementos básicos: a *Rede* e os *Cenários*. Esses elementos permitem, respectivamente, a descrição da topologia do cenário de rede e informações de contexto sobre o cenário descrito.

1.2 OBJETIVOS

A presente dissertação de Mestrado visa a extensão da linguagem *NSDL* de forma a viabilizar a descrição de cenários de rede *OpenFlow*. Para isso, foram realizadas durante este trabalho a análise e estudo do domínio proposto (protocolo *OpenFlow*), a identificação dos requisitos funcionais e não-funcionais para a descrição de cenários de rede OpenFlow, a proposta da extensão da linguagem *NSDL*, chamada de *NSDL OpenFlow*, e a realização de cenários de estudo de caso para validar a linguagem proposta.

A principal motivação para a realização deste trabalho está no interesse da aplicação das arquiteturas de gestão de redes existentes às Redes Definidas por Software, e em particular ao protocolo OpenFlow. Para isso, a Framework NSDL

surge como uma solução para promover a integração de diferentes ferramentas para a gestão OpenFlow.

Na próxima seção é apresentada a estrutura desta dissertação de mestrado.

1.3 PRINCIPAIS CONTRIBUIÇÕES

Podemos citar como principais contribuições deste trabalho:

a) O levantamento de requisitos em um ambiente diferente do tradicional de *software*, por se tratar de um ambiente de rede;

b) A identificação de requisitos (funcionais e não-funcionais) de redes *OpenFlow*;

c) A proposta do perfil *NSDL OpenFlow*;

d) A validação através de cenários de estudo de caso, e;

e) A aplicação prática desses requisitos em ferramentas diversas integradas à *Framework NSDL* (por exemplo, a autoria de cenários de testes *OpenFlow* com *NSDL*).

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Este documento está organizado em sete capítulos, incluindo este capítulo, conforme apresentado:

O segundo capítulo consiste na abordagem conceitual, onde são introduzidos os conceitos e tecnologias utilizados ao longo da dissertação.

No terceiro capítulo é realizada a apresentação do protocolo *OpenFlow*, onde é discutido o funcionamento das redes através desse paradigma.

O quarto capítulo consiste na apresentação da *Framework NSDL*, onde é possível compreender os seus conceitos básicos, a sua arquitetura e a sua estrutura.

No quinto capítulo é apresentada a proposta do perfil *NSDL OpenFlow* e todos os detalhes do seu desenvolvimento, seus requisitos e ilustrações do seu código.

No sexto capítulo são ilustrados alguns cenários de estudo de caso, e sua contribuição para a validação dos requisitos identificados.

O capítulo sete é reservado às conclusões e perspectivas futuras

2 ABORDAGEM CONCEITUAL E TRABALHOS RELACIONADOS

2.1 INTRODUÇÃO

Neste capítulo é realizada uma revisão conceitual sobre os principais tópicos abordados durante este trabalho de mestrado, especificamente relacionados às redes de computadores, às ferramentas para a gerência de redes, à *Framework NSDL* e também são discutidas as principais características relacionadas à tecnologia *OpenFlow* e algumas ferramentas que auxiliam a criação e testes de redes *OpenFlow*. Da mesma forma, são apresentados e discutidos alguns dos trabalhos relacionados existentes na literatura.

2.2 REDES DE COMPUTADORES

Conceitualmente, uma rede de computadores consiste em um conjunto de computadores e outros dispositivos interligados de modo a poderem compartilhar recursos físicos e lógicos (MENDES, 2005). Dessa forma, o objetivo da rede de computadores é permitir a troca de dados entre computadores e a partilha de recursos de hardware e *software*. Para que isso ocorra, diversos padrões e protocolos são estabelecidos de forma a garantir a interoperabilidade entre essas máquinas.

De forma a viabilizar a descrição de uma redes de computadores é importante identificar os seus principais componentes, tais como os comutadores, roteadores ou qualquer outro equipamento de comunicação ativo na rede, e que são chamados genericamente de nó da rede ou simplesmente nó. Existem ainda as conexões entre os nós, genericamente chamadas de *ligações*, e que são implementadas fisicamente através de diferentes meios de transmissão (por exemplo, fibras ópticas, cabos de cobre trançados, cabos coaxiais, etc.).

Além dos componentes físicos da rede, também é importante considerarmos a sua configuração lógica, através dos protocolos utilizados nas diferentes camadas TCP/IP, além das aplicações e respectivo tráfego nessa rede.

Atualmente com a proposta de novos protocolos e tecnologias de comunicação de dados, é importante que essas novas soluções possam ser

validadas em ambientes de testes cada vez mais dinâmicos e realísticos. No entanto, para que esses testes possam ser realizados em redes operacionais, eles não devem interferir no funcionamento da rede. De forma a viabilizar, o experimento de novos protocolos em redes operacionais de computadores, as *Redes Definidas por Software* têm sido propostas.

2.3 REDES DEFINIDAS POR *SOFTWARE*

As *Redes Definidas por Software (SDNs)* são uma abordagem para a construção de redes de computadores que permite a separação dos "plano de controle" e o "plano de dados" desse sistema. O plano de controle é a parte do equipamento, ou equipamentos, responsável pela tomada de decisões sobre para onde o tráfego deve ser enviado, enquanto o plano de dados é utilizado para encaminhar o tráfego para o destino selecionado. As SDNs desacoplam o sistema que faz as decisões sobre o plano de controle do plano de dados. Essa abordagem torna a rede mais flexível possibilitando um maior controle sobre o redirecionamento do tráfego. Através da virtualização de rede é possível otimizar a utilização dos recursos reais de uma rede, possibilitando o tratamento de tráfego experimental sem prejudicar o tráfego de produção da rede (ROTHENBERG et al., 2011).

Uma SDN permite que os administradores de rede tenham o controle central programável (implementado em software) de tráfego de rede sem a necessidade de acesso físico aos dispositivos da rede de hardware (ROTHENBERG et al., 2011).

Como exemplo, pode ser citada a proposta apresentada em (GENI, 2012), que tem como premissa a atribuição de uma “fatia” virtual dos recursos da rede que consiste na utilização parcial de enlaces de rede, encaminhadores, comutadores e terminais clientes. Assim, é possível o tratamento de um tráfego experimental podendo ser configurado e personalizado de acordo com a necessidade em questão.

Da mesma forma, ainda podemos citar o protocolo *OpenFlow* (MCKEOWN et al., 2008) que permite a separação do plano de controle e do plano de dados, oferecendo mais simplicidade e otimização ao modelo de gestão do tráfego, diminuição do custo operacional (já que sua implementação pode ser baseada na virtualização de recursos reais), entre outras características.

Assim como na tradicional rede de computadores, as SDNs também necessitam de gerenciamento de forma a otimizar o ciclo de vida da rede em questão, dando suporte aos aspectos de modelagem, configuração, monitoração, manutenção, validação, etc. Uma das soluções existentes para proporcionar a interoperabilidade entre diferentes ferramentas heterogêneas de gestão de redes e o gerenciamento integrado das SDNs é a Framework NSDL, introduzida na próxima seção.

2.4 FRAMEWORK NSDL

A *Framework NSDL* (*Network Scenario Description Language*) foi desenvolvida na Universidade da Madeira (Portugal) como uma solução para permitir a interoperabilidade de diferentes ferramentas para a gerência de redes (MARQUES et al., 2011). A *Framework* é baseada na Linguagem *NSDL*, que é uma abstração extensível proposta em XML (XML technology, 2012) para a descrição de cenários de redes genéricos. A *Framework NSDL* é apresentada em mais detalhes no Capítulo 4.

Através de *NSDL* é possível a descrição de qualquer tipo de rede, tais como redes *DiffServ* (MARQUES et al., 2011), sem fios (MARQUES et al., 2011), LTE (SOUSA, 2011), WiMAX (SOUSA, 2011), entre outras.

Na próxima seção são apresentados algumas contribuições disponíveis na literatura relacionadas à Framework NSDL e à temática do trabalho realizado nesta dissertação.

Alguns trabalhos relacionados ao desenvolvimento da *Framework* NSDL e à implementação de SDNs são apresentados a seguir:

a) ***NSDL* Perfil NS-2** – desenvolvido na Universidade da Madeira (Portugal), de forma a modelar um conjunto de componentes de rede através da extensão da linguagem NSDL para a criação de um Perfil NS-2 a partir de um Perfil Base. Esse Perfil NS-2 permite modelar um conjunto de objetos na ferramenta de simulação NS-2, dando suporte as diferentes a aspectos de Qualidade de Serviço (DiffServ) (MARQUES, 2013);

b) **Visual Network Description (VND)** – desenvolvido na Universidade da Madeira com o principal objetivo de proporcionar uma interface web para a criação de cenários de rede genéricos, com suporte à descrição de aspectos de Qualidade de Serviço com Diffserv (AZEVEDO, 2010);

c) **NSDL Perfil LTE/WiMAX para simulação em NS-3** – desenvolvido na Universidade da Madeira (Portugal), de forma a proporcionar a extensão da linguagem NSDL para suportar a descrição de cenários de rede sem fio utilizando as tecnologias LTE e WiMAX. Os cenários criados em NSDL podem ser exportados para um script C++ para a sua posterior simulação em NS-3 (SOUSA, 2011);

d) **NSDL Perfil Virtualização** – desenvolvido também na Universidade da Madeira (Portugal), de forma a proporcionar a extensão da linguagem NSDL para suportar a modelagem de cenários de rede genéricos, a exportação da respectiva descrição NSDL do cenário e a posterior execução do cenário através da virtualização dos recursos da rede (nós, ligações, tráfego, etc.) (ARAUJO, 2011);

e) **AUTORIA DE CENÁRIOS OPENFLOW UTILIZANDO VISUAL NETWORK DESCRIPTION (VERSÃO SDN)** – desenvolvido na Universidade Salvador (UNIFACS), sendo o principal objetivo deste trabalho 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 (FONTES, 2013);

f) **Implementação e Análise dos Planos de Controle e Encaminhamento para SDN** – o paradigma de redes definidas por software (SDN) é uma arquitetura para permitir a rápida inovação enquanto esconde muito da complexidade do projeto de rede. A intenção desse trabalho é explorar o mecanismo das SDNs e executar um comparativo entre esse modelo e o modelo atual de rede. Esse comparativo visa demonstrar as vantagens e benefícios que esse novo modelo pode trazer (RISDIANTO et al., 2013).

g) **OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes** – Este artigo faz um levantamento do estado da arte da tecnologia *OpenFlow* como quebra de paradigma de controle em software (remoto) dos equipamentos de rede, habilitando o conceito de redes definidas por software (ROTHENBERG et al., 2011).

h) **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.

i) **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.

j) **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.

Os trabalhos apresentados anteriormente estão diretamente ligados ao contexto de realização deste trabalho de mestrado. Em particular, os trabalhos

introduzidos nos itens A a E fazem parte do projeto NSDL, inicialmente desenvolvido na Universidade da Madeira através de contribuições de alunos de mestrado e doutorado, e posteriormente continuado na Universidade Salvador (UNIFACS). De forma similar aos trabalhos apresentados, o presente trabalho de mestrado visa dar continuidade ao projeto através da extensão da linguagem NSDL para a descrição de redes SDN, utilizando o protocolo. Dentre os trabalhos continuados neste projeto na Universidade Salvador podemos citar extensões da linguagem NSDL para suportar a descrição de redes MPLS/GMPLS (DE OLIVEIRA, 2014), simulação NS-3 de cenários de rede *OpenFlow* (PINHEIRO, 2014) e Autoria de cenários de rede SDN (FONTES, 2013).

2.5 CONCLUSÃO

Este capítulo descreveu os principais aspectos e abordagens conceituais que serviram como base teórica desse trabalho, tais como redes de computadores, redes definidas por *software*, *OpenFlow,* e a *Framework NSDL*.

Para compreender e viabilizar o processo de criação do perfil NSDL *OpenFlow*, os trabalhos relacionados ao projeto NSDL serviram como base para os primeiros passos no desenvolvimento de um perfil baseado em NSDL. Já o trabalho intitulado "*Implementação e Análise dos Planos de Controle e Encaminhamento para SDN"*, permitiu a compreensão, comparação e melhor visualização de quais seriam as vantagens entre o modelo das Redes Definidas por *Software* e o modelo tradicional.

Em particular, vale ressaltar os trabalhos "*Autoria de Cenários OpenFlow utilizando Visual Network Description (Versão SDN)***"** (FONTES, 2013) e "Simulação NS-3 de Cenários de Redes OpenFlow descritos em NSDL" (PINHEIRO, 2014) foram fundamentais para aplicação e validação da extensão da linguagem NSDL para *OpenFlow,* proposta nesta dissertação. Essa validação é discutida em seguida no capítulo de Estudos de Caso (capítulo 6).

3 OPENFLOW

3.1 INTRODUÇÃO

As redes de computadores são compostas por milhares de equipamentos, e até mesmo milhões, sobretudo se considerarmos o número de máquinas conectadas através da Internet. Devido a esse grande número, a proposta e teste de novos protocolos para essas redes em ambiente real se torna uma atividade muito complexa, pois seria necessária uma intervenção direta em todos os equipamentos da rede. Por essa razão, considera-se que a infra-estrutura pública e privada de rede de computadores não é flexível para a realização de experimentos de novos protocolos e tecnologias de redes (MCKEOWN et al., 2008). Além da simulação, outra alternativa possível para realizar esses experimentos seria através da utilização de redes virtualizadas programáveis. Através da virtualização, para cada experimento a ser realizado é atribuída um parcela dos recursos da rede associados aos enlaces de rede, encaminhadores, comutadores e terminais clientes. Assim, o experimento pode ser configurado e personalizado de acordo com a necessidade em questão.

As redes virtualizadas programáveis são interessantes para diminuir as barreiras entre uma nova proposta e sua experimentação. Porém, a implementação dessas redes pode ser custosa e demorada dadas as suas limitações de escalabilidade (MCKEOWN et al., 2008). Para permitir a experimentação de novas soluções para redes de computadores em uma escala menor, foram propostas as *Redes Definidas por Software*, e em particular o protocolo *OpenFlow* (MCKEOWN et al., 2008). 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.

O *OpenFlow* possibilita a reprogramação dos comutadores de forma que eles possam tratar o tráfego do testes sem interferir no funcionamento da rede de produção. Outro objetivo dessa proposta é possibilitar que os fabricantes possam adicionar as funcionalidades do *OpenFlow* aos seus comutadores sem necessitarem expor o projeto desses equipamentos. Além disso, esses equipamentos devem

possuir baixo custo e desempenho semelhante aos já utilizados em uma infra-estrutura legada. Com base no exposto, o projeto do *OpenFlow* tem como objetivo ser flexível para atender os seguintes requisitos (MCKEOWN et al., 2008):

a) Proporcionar implementações de baixo custo e alto desempenho;

b) Ter capacidade de suportar uma ampla gama de pesquisas científicas;

c) Garantir isolamento entre o tráfego de pesquisa e o tráfego de produção, e;

d) Ir ao encontro da necessidade dos fabricantes de não expor o projeto de suas plataformas.

Neste capítulo são apresentados os principais conceitos, arquitetura e componentes do protocolo *OpenFlow*.

3.2 CONCEITOS BÁSICOS

A maioria dos comutadores e encaminhadores modernos suportam a adição de diferentes funcionalidades tais como, por exemplo, a configuração de aspectos de Qualidade de Serviço (QoS), *Firewalls* e coleta de estatísticas. Para a implementação do suporte ao protocolo *OpenFlow* nesses comutadores é necessária a manipulação de suas tabelas de fluxos (MCKEOWN et al., 2008). Como as funções de um comutador e/ou encaminhador podem diferir de acordo com o fabricante, na especificação de *OpenFlow* (*Open Networking Foundation, 2012*) são identificadas algumas funções comuns à maioria dos equipamentos comerciais com o objetivo de ser suportado por um maior número de equipamentos disponíveis. A partir dessas funções, o *OpenFlow* fornece um protocolo aberto para manipular o plano de dados desses equipamentos e gerenciá-los através de um plano de controle distinto.

O *plano de dados* de um Comutador *OpenFlow* consiste em uma *tabela de fluxos* com uma ação correspondente a cada um desses fluxos, o que permite que os fluxos sejam tratados de maneira diferente. Assim, pode ser definido um fluxo de tráfego experimental e outro de tráfego de produção, possibilitando isolamento entre eles.

O plano de controle está preocupado com funções de coordenação, como roteamento e sinalização, para facilitar o deslocamento do tráfego através de toda a rede. O *OpenFlow* propõe o deslocamento de grande parte da lógica de tomada de decisão dos dispositivos de rede para controladores externos.

Em particular, o tráfego de produção dessa rede é tratado da mesma forma que seria na ausência do *OpenFlow*. Com isso, o *OpenFlow* pode ser visto com uma generalização do conceito de redes virtuais (*VLANs*), que também permitem o isolamento de tráfego mas de forma menos flexível (MCKEOWN et al., 2008).

3.3 ARQUITETURA

Como discutido, uma das funcionalidades do protocolo *OpenFlow* é permitir a criação de redes de teste sobre uma rede de produção, baseadas na utilização de equipamentos de rede comerciais (MCKEOWN et al., 2008).

Devido a essa facilidade e por ser um padrão aberto, 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). Contudo, a utilização do *OpenFlow* requer a existência de ferramentas que simplifiquem o gerenciamento das redes e facilitem o desenvolvimento de novos mecanismos de controle.

A arquitetura do *OpenFlow* define um modelo de rede no qual os elementos encaminhadores, chamados de comutadores *OpenFlow*, são simples e programáveis.

Um nó especial, chamado de *controlador*, centraliza a execução de todas as tarefas de controle, sendo também utilizado para o desenvolvimento de novos mecanismos. A contribuição mais importante do paradigma *OpenFlow* é a generalização do plano de dados. Isto é, qualquer modelo de encaminhamento de dados que se baseie na tomada de decisão fundamentada em algum valor do campo de cabeçalho dos pacotes pode ser suportada. As entradas correspondentes na tabela de fluxos podem ser interpretadas como decisões em *cache* (*hardware*) do plano de controle (*software*). Um fluxo na rede é portanto nada mais que a mínima unidade de controle para criar sistemas escaláveis usando *OpenFlow* (MCKEOWN et al., 2008).

Como já citado, o plano de dados de um comutador *OpenFlow* é formado por uma tabela de fluxos com uma ação associada a cada um desses fluxos. Permitindo

que os fluxos sejam tratados de maneira diferente. Por isso o administrador consegue manter o isolamento do tráfego do fluxo de pesquisa do fluxo de produção. O conjunto de ações de um comutador *OpenFlow* é extensível, e o seu plano de dados deve ser flexível para garantir alto desempenho e baixo custo.

O *OpenFlow* explora a tabela de fluxo que existe nos comutadores atuais, que normalmente são utilizadas para implementar serviços como NAT, *firewall* e VLANs. Portanto, o plano de dados de um comutador *OpenFlow* é constituído de uma tabela de fluxos, e um evento associado a cada entrada na tabela. Basicamente, o *OpenFlow* é composto pelas três partes:

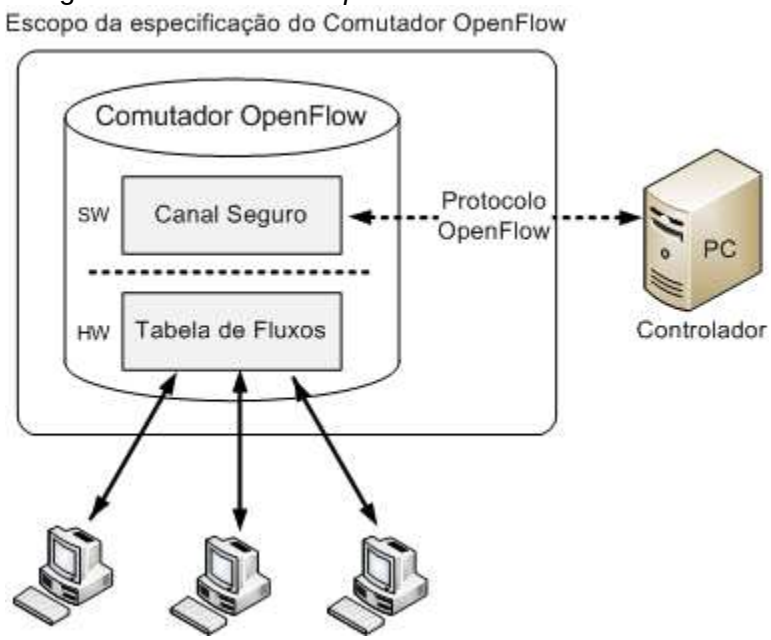
a) Tabela de Fluxos: Cada entrada na tabela de fluxos contém uma ação associada, e consiste em campos do cabeçalho (utilizado para definir um fluxo), ações (define como os pacotes devem ser processados e para onde devem ser encaminhados) e contadores (utilizados para estatísticas ou remoção de fluxos inativos);

b) Canal Seguro: Para que a rede não sofra ataques mal intencionados, o canal seguro assegura confiabilidade na troca de informações entre o comutador e o controlador. Aspectos como autenticação e criptografia são implementados através do *protocolo Secure Socket Layer (SSL)*. Essas são bem úteis em ambientes virtuais, pela simplicidade de utilização, e;

c) Protocolo *OpenFlow*: disponibiliza um mecanismo aberto para estabelecer a comunicação entre o comutador e o controlador. Ao fornecer uma interface externa que atue sobre os fluxos de um comutador, o Protocolo *OpenFlow* (OFP - *OpenFlow Protocol*) evita a necessidade de um *comutador* programável.

Com esse protocolo, é possível configurar a Tabela de Fluxos sem a necessidade de programar o comutador (MCKEOWN et al., 2008). Os principais componentes de *OpenFlow* são ilustrados na Figura 2.

Figura 2 - Comutador *OpenFlow*



Fonte: Couto (2010).

A Figura 2 ilustra o comutador *OpenFlow*. Existem dois tipos de comutadores *OpenFlow*. O primeiro chama-se comutador *OpenFlow* dedicado, que não suporta encaminhamento comum de Nível 2 e Nível 3. O segundo tipo é um comutador ou encaminhador comercial com *OpenFlow* habilitado que, além de suportar as funcionalidades do *OpenFlow*, realiza as funções tradicionais de um comutador ou encaminhador.

Um comutador *OpenFlow* dedicado apenas encaminha os pacotes entre as portas do comutador de acordo com a Tabela de Fluxos configurada remotamente. Neste contexto os fluxos podem ser definidos como todos os pacotes provenientes dos mesmos endereços MAC e IP. Além disso, alguns comutadores *OpenFlow* podem tratar pacotes que não são IPv4, verificando os campos arbitrários do cabeçalho. Isso mostra a flexibilidade do *OpenFlow* para suportar diferentes tipos de tráfego. A seguir as três ações mais básicas:

a) Encaminhamento dos pacotes do fluxo para uma determinada porta (ou portas). Isso permite que os pacotes sejam encaminhados pela rede;

b) Encapsulamento dos pacotes do fluxo e transmissão para um controlador através do canal seguro. Normalmente isso é feito com o primeiro pacote de um fluxo novo, assim o controlador pode verificar se o fluxo deve ser adicionado

à tabela de fluxo. Essa ação pode ser utilizada também, para o envio dos pacotes ao controlador para o recebimento de tratamentos adicionais, e;

c) Descarte dos pacotes do fluxo. Essa ação pode ser utilizada em casos de segurança, para impedir ataques por negação de serviço (onde um determinado servidor é sobrecarregado de requisições baixando a sua performance e impedindo o seu funcionamento normal).

Já os comutadores com *OpenFlow* habilitado são os comutadores, encaminhadores e pontos de acesso comerciais que suportam as funcionalidades do *OpenFlow* (MCKEOWN et al., 2008; COUTO, 2010). São adicionados a esses equipamentos a tabela de fluxo, o canal seguro e o protocolo *OpenFlow*.

Em uma rede de comutadores e pontos de acesso comerciais com *OpenFlow* habilitado todas as tabelas de fluxo são tratadas pelo mesmo controlador. O protocolo *OpenFlow* possui um recurso que permite que dois ou mais controladores controlem um comutador auxiliando no aumento do desempenho e da robustez.

Os equipamentos com o *OpenFlow* habilitado têm como objetivo tornar possível o isolamento do trafego experimental, que utiliza os mecanismos do *OpenFlow*, do trafego de produção que utiliza os mecanismos normais de um comutador (MCKEOWN et al., 2008; COUTO, 2010). Esse isolamento pode ser garantido através de duas opções: A primeira seria adicionar uma quarta ação, o encaminhamento dos pacotes através do tratamento normal do comutador. Assim, o fluxo que for identificado como não referente ao *OpenFlow* será processado utilizando o encaminhamento normal, e; A segunda opção para garantir que o tráfego de produção seja processado normalmente pelo comutador através do uso de VLANs, definindo conjuntos separados de VLANs para o tráfego de produção e para o tráfego experimental.

Todos os comutadores com *OpenFlow* habilitado precisam conter uma dessas duas ações, porém alguns equipamentos suportam as duas.

Da mesma forma, também é importante considerar o funcionamento do controlador, que é responsável por adicionar e remover as entradas de fluxo em uma tabela de fluxo para os testes. Por exemplo, um controlador estático é uma simples aplicação sendo executada em um computador que, durante um experimento, decide estaticamente quais fluxos pertencem ou não a um conjunto de computadores de teste (MCKEOWN et al., 2008). Para este caso, é possível

comparar os fluxos com as VLANs das redes atuais, que fornecem um mecanismo simples de isolamento de tráfego, isolando o tráfego experimental do tráfego de produção.

Alguns controladores mais sofisticados também foram concebidos de forma a adicionar e remover dinamicamente os fluxos durante um experimento (MCKEOWN et al., 2008; FERNANDES et al., 2010). Normalmente, é possível controlar toda a rede de comutadores *OpenFlow* e configurar a maneira de como os fluxos serão tratados. Esses controladores mais potentes conseguem suportar vários usuários, onde cada um possui uma conta e permissões diferentes. Os usuários podem então executar seus testes independentes em diferentes conjuntos de fluxos.

Os fluxos que forem identificados como sendo de um determinado experimento é entregue ao chegar a um comutador e o protocolo irá traçar uma rota pelos comutadores *OpenFlow* e adicionar um fluxo de entrada em cada comutador dessa rota. Para executar o experimento em uma rede *OpenFlow* o seu respectivo tráfego não pode perturbar os outros que dependem dela. Para isso, o usuário precisa definir que um fluxo seja todo o tráfego que está entrando no *comutador OpenFlow* pela porta do *comutador* que está conectada ao seu computador e adicionar um fluxo de entrada com a ação “Encapsulamento e envio de todos os pacotes ao controlador”. Assim, quando os pacotes enviados por um experimento chegarem ao controlador, o protocolo irá traçar uma rota e adicionar um fluxo de entrada a todos os *comutadores* dessa rota. Dessa forma, os próximos pacotes que chegarem serão processados rapidamente pela tabela de fluxo (MCKEOWN et al., 2008).

Com o protocolo *OpenFlow* é possível disponibilizar um ambiente de testes em uma rede utilizando, por exemplo, a infra-estrutura de um ambiente de pesquisa. Assim ele permite os pesquisadores possam executar seus experimentos e garantir que eles sejam testados em um ambiente suficientemente semelhante ao de uma rede real. Dessa forma, é possível validar novos protocolos e tecnologias, sem a necessidade de os fornecedores exporem o projeto interior de seus produtos e sem também a necessidade do pesquisador de encomendar um equipamento com um *software* específico de controle para a realização de seus testes.

3.4 CONCLUSÃO

Neste Capítulo foi apresentada a tecnologia *OpenFlow*, que dadas as suas funcionalidades permite a utilização de uma infra-estrutura de rede já existente para a realização de experimentos de novos protocolos e tecnologias de rede, sem prejudicar o tráfego de produção desta rede. Com isso, é possível disponibilizar um ambiente de testes em uma escala maior e mais realístico para a execução de experimentos.

No entanto, as redes *OpenFlow* possuem a necessidade de atualização ou substituição dos equipamentos de rede já existentes, e apresentam uma curva maior de aprendizado para o domínio e utilização da tecnologia, que se baseia em *Redes Definidas por Software*. Por outro lado, *OpenFlow* oferece uma grande flexibilidade para a realização de testes, além de incentivar o compartilhamento e aproveitamento mais inteligente de recursos e equipamentos em escala mais ampla.

No próximo capítulo é apresentada a *Framework* NSDL, que é aplicada no contexto deste trabalho como proposta de integração para ferramentas que suportem a autoria e simulação de cenários de rede OpenFlow.

4 FRAMEWORK NSDL

4.1 INTRODUÇÃO

De forma facilitar a gerência das redes de computadores, diferentes ferramentas e ambientes integrados têm sido propostos (autoria, monitoramento, simulação, virtualização, etc.). No entanto, essas ferramentas, em geral, não são interoperáveis, dificultando a tarefa do gestor de redes quando ele necessita de uma perspectiva complementar sobre a mesma topologia.

Para propor uma gestão integrada de redes e otimizar a tarefa do gestor através de uma utilização complementar das diferentes soluções disponíveis, a *Framework Network Scenario Description Language – NSDL* (originalmente em Inglês) foi proposta na Universidade da Madeira (Portugal). O principal objetivo dessa *Framework* é a proposta de um mecanismo de representação genérica para diferentes cenários, tecnologias e protocolos de redes de computadores, que promova a interoperabilidade entre diferentes ferramentas de redes.

Este capítulo tem como objetivo proporcionar uma visão mais ampla sobre a utilização de *NSDL*. Para isso, os conceitos básicos de *NSDL* são inicialmente apresentados onde é possível compreender os princípios utilizados para a sua criação. Posteriormente, é abordada a estrutura da *Framework NSDL*. Por fim, é apresentada a estrutura de uma descrição *NSDL* especificando as suas componentes obrigatórias e opcionais.

4.2 A FRAMEWORK NSDL

Como já citado, o objetivo principal da *Framework NSDL* é propor uma estrutura onde diferentes aplicações de rede possam ser integradas e onde existam mecanismos que suportem a interoperabilidade entre essas mesmas aplicações. É importante que as diferentes aplicações possam intercambiar informações da rede tais como a topologia de rede atual, as propriedades dos seus componentes e qualquer outra informação específica sobre a rede como localização, simulação, segurança, gerenciamento entre outros.

Como já introduzido no Capítulo 1 (Introdução), 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 (MARQUES, 2013).

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.

4.3 PRINCÍPIOS BÁSICOS

A *Framework NSDL* é baseada na utilização de uma linguagem intermediária para promover a interoperabilidade entre diferentes ferramentas de gerência de redes, a linguagem *NSDL*. O propósito de *NSDL* é fornecer um vocabulário e um conjunto de regras, ambos capazes de apoiar a descrição de redes de dados com e sem fios. Além de descrever a topologia de rede com os seus objetos e características, *NSDL* introduz um conceito importante: a separação da descrição da rede a partir de várias perspectivas (por exemplo, de visualização ou simulação) que possam existir sobre essa rede (MARQUES, 2013).

Os princípios utilizados para projetar *NSDL* foram:

a) **Simplicidade** - a linguagem deve ser simples e não apenas para ser manipulada por uma ferramenta, mas também possível de ser editada por um usuário com um editor de texto simples;

b) **Definição de vários níveis de abstração** – permitindo especificar não apenas descrições simples de alto nível de um cenário de rede, mas também, se necessário, ter a possibilidade de criar uma descrição muito detalhada de todos os objetos do cenário de rede e os seus parâmetros, e;

c) **Extensibilidade** – o que implica na possibilidade de incorporar novos objetos e parâmetros em futuras descrições.

A partir da definição dos princípios apresentados, foi possível propor a linguagem NSDL, apresentada na próxima seção:

4.4 LINGUAGEM *NSDL*

Alguns elementos na linguagem *NSDL* são fundamentais para a descrição da topologia de um cenário de redes e seus componentes e outros são relacionados a outras ferramentas externas que permitem a utilização da linguagem (MARQUES et al, 2011). A linguagem *NSDL* é baseada em XML e contém dois elementos básicos: *Rede* (*Network*) e *Cenários* (*Scenarios*). O elemento *Rede* contém a descrição de uma topologia de rede com seus objetos (nós) e os seus respectivos parâmetros. O elemento *Cenário* pode conter várias descrições, cada uma referindo um uso específico de uma ferramenta, ou contexto, a essa rede (MARQUES et al, 2011).

A Figura 3 representa a estrutura da linguagem *NSDL*. O elemento *Rede* é composto por: *Modelo (Template)*, *Objeto (Object)* e *Visões (Views)*. O elemento *Objeto* contém a descrição da topologia da rede, e é composto por nós, ligações e domínios. Os elementos *Modelo* e V*isões* não são obrigatórios. O elemento M*odelo* é importante para simplificar e promover a reutilização da descrição de objetos similares. Já o elemento *Visões* é um mecanismo aplicado para descrever uma abstração de um grupo de objetos de rede a serem utilizados nos cenários.

Figura 3 - Estrutura *NSDL*

Fonte: Araujo (2011).

O elemento *Objeto* é considerado o elemento central da linguagem *NSDL* já que descreve a topologia, os nós da rede e as suas características. A descrição é feita pela identificação de todos os objetos de rede e os seus parâmetros. Para isso foram definidos três objetos base:

- **Nó (*Node*):** que representa todos os equipamentos de uma rede, como por exemplo: *comutadores*, computadores, encaminhadores, etc.;
- **Ligação *(Links)*:** é a conexão entre dois nós, que pode ser representado através de qualquer tecnologia, tais como Ethernet, ADSL, fibra óptica, etc., e;
- **Domínio (*Domain*):** é um objeto que permite abstrair uma rede completa.

Uma característica única do objeto Nó é a sua capacidade de incorporar outros objetos nele. Além dos parâmetros disponíveis do *Nó*, outros objetos que podem ser descritos no *Nó* são:

- **Interface (Interface):** este elemento representa dispositivos que permitem a comunicação entre nós de rede. É importante observar que no contexto das redes com fios a comunicação entre nós pode ser estabelecida através de uma *ligação*, no entanto nos ambientes sem fio o conceito de ligação não seria o mais correto, pois trata-se de um canal virtual e não físico, daí a

necessidade de comunicar os fluxos de dados nessas redes através de *interfaces*;

- **Protocolo (*Protocol*):** é um elemento que descreve as características de vários protocolos, e;
- **Aplicação (*Application*):** descreve o tipo de tráfego de rede gerado e recebido, por exemplo FTP, CBR, Telnet e outros.

O propósito do elemento *Cenários* é trazer informações adicionais e relevantes para a rede a partir de várias perspectivas específicas. O elemento *Cenários* é composto por dois elementos: *Visualização* e *Simulação* (MARQUES et al., 2011).

O elemento V*isualização (Visualization)* fornece informações adicionais para enriquecer a descrição da rede, tais como objetos de posicionamento espacial na interface gráfica do usuário e outras informações gráficas sobre um cenário de rede. Por outro lado, todos os parâmetros necessários para implementar uma simulação de um cenário de rede utilizando uma ferramenta de simulação genérica ou particular são definidos no elemento *Simulação (Simulation)*.

A Figura 4 ilustra um trecho de uma descrição NSDL. Neste trecho um bloco *Rede (Network)* é apresentado. O bloco *Rede* é composto por: *Modelo* (*Templates*), descrevendo o modelo de um encaminhador (*router*) simples e de uma *ligação (link)* simples. Já *Objeto* (*Objects*), descrevendo a criação dos computadores e das ligações, e por fim o elemento *Visão* (*View*), onde estão listados os objetos do servidor de rede.

Figura 4 - Trechos do código em XML do bloco Rede do *NSDL*

```
<nsdl>
  <network>
    <templates>
    <router id="T_Routers">
        <notes>Template dos Routers</notes>
        <rip id="rip01">
            <version>2</version>
            <aunds>1</aunds>
        </rip>
    </router>
    </templates>
    <objects>
        <computer id="cpu_9">
            <exponential id="aplicacao_http.cl">
                <role>client</role>
                <src.app>app.sv</src.app>
            </exponential>
        </computer>
    </objects>
    <views>
        <set id="serverNet">
            <listofobjectis>
                <objectid>sever</objectid>
                <objectid>router02</objectid>
                <objectid>serverRouter</objectid>
            </listofobjectis>
        </set>
    </views>
  </network>
  <scenarios>
      <visualization/>
      <simulation/>
      <.../>
  </scenarios>
</nsdl>
```

Fonte: Araujo (2011).

Na Figura 5, o bloco *Cenários* (*Scenarios*) do *NSDL* é ilustrado, onde é possível verificar a sua estrutura que é composta por: *Visualização* (*Visualization*), definindo a posição dos objetos e a sua cor e por fim, a *Simulação* (*Simulation*) onde está definido o tempo de simulação e os eventos.

Figura 5 - Trechos do código em XML do bloco Cenários do *NSDL*

```
<scenarios>
    <visualizations>
        <visualization id="vis01">
            <name>Visual01</name>
            <description>
                <object id="comp76">
                    <x.position>10</x.position>
                    <y.position>10</y.position>
                    <z.position>0</z.position>
                    <color>Blue</color>
                </object>
            </description>
        </visualization>
    </visualizations>
    <simulations>
        <simulation id="sim01">
            <description>
            <routing>
                <nix.vector id="nix1"/>
                <aodv id="aodvtest">
                <next.hop.wait>30</next.hop.wait>
                </aodv>
                <nix.vector id="nix1"/>
                <dsdv id="dsdvtest">
                <update.interval>4</update.interval>
                </dsdv>
                <olsr id="olsrtest">
                <willingness>always</willingness>
                </olsr>
            </routing>
            <general>
                <duration>
                    <time></time>
                </duration>
                <runs>1</runs>
                <simulator>ns2v2.34</simulator>
            </general>
            <events>
                <event objectid="telServer" time="0.2">
                    <parameter name="action" value="Start"/>
                </event>
            </events>
            <outputs>
                <output outputid="outtrace.01" type="trace">
                    <source>Linkc</source>
                    <format>tr</format>
                    <path>./</path>
                    <filename>MyFirstTrace</filename>
                </output>
            </outputs>
            </description>
        </simulation>
    </simulations>
</scenarios>
```

Fonte: Araújo (2011).

De forma a facilitar a reutilização dos elementos de NSDL e organizá-los de acordo com o domínio ou tecnologia a ser descrito, o conceito de perfil foi proposto. A próxima seção introduz o conceito de perfil para a descrição de cenários de redes.

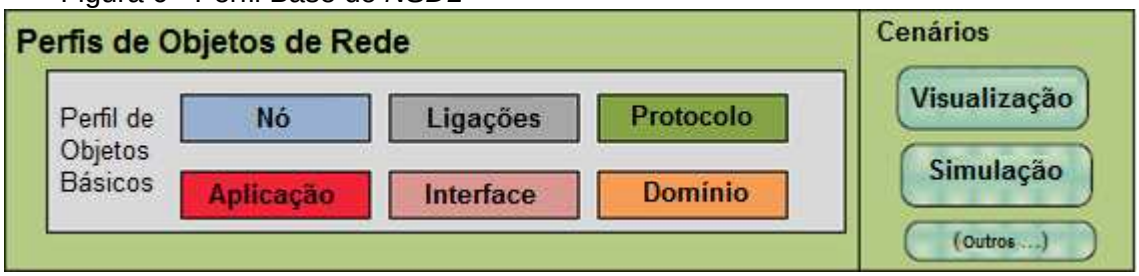
4.5 PERFIL BASE

O *NSDL* é uma linguagem baseada em XML, e por isso detém um conjunto de elementos que formam o domínio lexical permitido na validação *NSDL*. Com esses elementos, o *NSDL* permite a introdução de extensões que visam abordar a especificação de objetos existentes e/ou até de objetos novos no contexto de uma determinada plataforma de simulação final. De forma a propor uma melhor organização dos elementos ou extensões que representam um domínio ou tecnologia de redes em NSDL, foi proposta o conceito de perfil.

O perfil base para *NSDL* foi definido como um conjunto de elementos (nós, *ligações*, etc.) que permitem compor qualquer tipo de cenário de rede de forma genérica. A partir desse perfil base é possível a sua extensão de forma a caracterizar cenários mais específicos relacionados à tecnologias, domínios de redes ou mesmo ferramentas de rede utilizadas.

Uma extensão da linguagem *NSDL* inclui o *perfil* base e um conjunto de elementos novos que permitem especificar objetos que existam exclusivamente no contexto do domínio destino (*OpenFlow*, por exemplo). A Figura 6 ilustra o perfil base.

Figura 6 - Perfil Base do *NSDL*

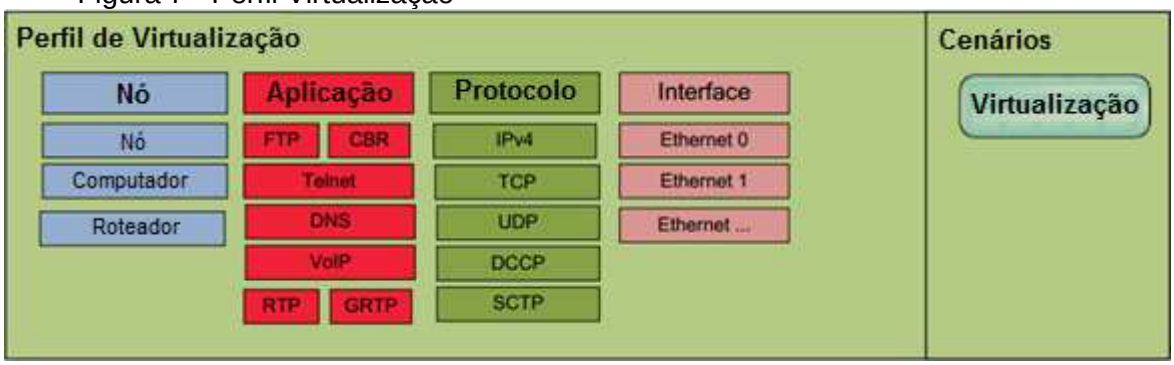


Fonte: Sousa (2011).

Além do perfil base foram propostos outros perfis NSDL para a descrição de outros domínios ou tecnologias de redes, tais como virtualização (ARAUJO, 2011), o

perfil NS-2 (DiffServ) (MARQUES et al., 2011) e NS-3 (*Wireless*, LTE, WiMax) (MARQUES et al., 2011).

Figura 7 - Perfil Virtualização



Fonte: Araujo (2011).

Na Figura 7 é ilustrado o perfil *virtualização* onde é possível verificar os objetos básicos utilizados neste perfil. Este perfil permite a descrição de: um nó, um computador ou um encaminhador. O perfil *virtualização* suporta as seguintes aplicações: FTP, CBR, Telnet, DNS, VoIP, RTP e GRTP. Em termos de protocolos o perfil suporta os seguintes: IPv4, TCP, UDP, DCCP e SCTP. O perfil ainda suporta várias interfaces, por exemplo, eth1, eth2, etc. O bloco *Cenários* está relacionado com a descrição dos aspectos de virtualização (ARAUJO, 2011).

A partir da criação e extensão de perfis pode-se alcançar qualquer objetivo dentro da descrição de cenários utilizando a *Framework NSDL*, sendo permitido desde a criação dos cenários mais complexos e específicos aos cenários não tão robustos porém que também exijam um certo grau de especificidade.

A linguagem *NSDL* possui como principal característica a sua extensibilidade, que possibilita a criação de perfis para diferentes tecnologias e domínios de redes com características totalmente peculiares e bem heterogêneas. Dessa forma, é muito evidente o quão flexível pode ser *NSDL* como recurso para descrição de cenários e interoperabilidade entre ferramentas heterogêneas de rede, o que proporciona ao usuário maior facilidade para gerenciamento de redes de qualquer tipo existente ou pertencente a uma tecnologia ainda em desenvolvimento.

4.6 CONCLUSÃO

Nesse capitulo foi apresentada a *Framework NSDL* e os detalhes de sua estrutura. Sendo uma extensão de XML, *NSDL* proporciona um alto grau de extensibilidade permitindo a criação de diferentes perfis que podem ser adotados para a descrição de vários tipos de tecnologias e domínios de redes.

Além das características discutidas da *Framerwork*, ainda podemos destacar outra vantagem de *NSDL* que é a interoperabilidade. Portanto, se um usuário desejar utilizar várias ferramentas compatíveis com o *NSDL*, ele pode facilmente analisar uma rede usando cada uma delas para obter resultados integrados e complementares. Por exemplo, o *NSDL* pode ser útil quando o usuário precisar executar duas simulações semelhantes sobre o mesmo cenário de rede mas utilizando dois simuladores diferentes (MARQUES et al, 2011).

Neste sentido a estrutura definida pelo *NSDL* e os seus parâmetros devem ser abrangentes o suficiente para descrever qualquer tipo de rede de dados e permitir a introdução na sua definição de dados para suportar objetos futuros e novas redes de dados.

No próximo capítulo é apresentada a principal contribuição deste trabalho através da proposta de uma extensão para a linguagem NSDL, a linguagem NSDL - Perfil OpenFlow.

5 EXTENSÃO À LINGUAGEM *NSDL*: PERFIL *NSDL OPENFLOW*

5.1 INTRODUÇÃO

Neste capítulo é apresentada a extensão à Linguagem *Network Scenario Description Language* (*NSDL*), e particularmente ao seu perfil base, chamada de perfil *NSDL OpenFlow*. Um perfil *NSDL* contempla a descrição de um conjunto de características que definem uma tecnologia ou domínio de redes a ser estudado, alterando ou estendendo as características base de *NSDL* (MARQUES, 2013). A partir do perfil base do *NSDL*, apresentado no capítulo 4, é possível criar vários perfis adicionando ou reutilizando objetos existentes adequados a cada perfil (ARAUJO, 2011).

A linguagem *NSDL* possui uma estrutura flexível e extensível que pode ser aplicada a qualquer cenário de redes (MARQUES et al., 2011). De forma a propor o perfil NSDL OpenFlow, foi necessário realizar o estudo do paradigma em questão para proporcionar a compreensão dos componentes e do funcionamento do protocolo OpenFlow. A partir desse estudo foi possível proceder com o levantamento dos seus requisitos funcionais e não-funcionais. Portanto, a elicitação dos requisitos e, em seguida, a proposta do perfil OpenFlow foi realizada à luz da definição dos objetivos e das principais características deste protocolo.

Neste capítulo primeiramente, são apresentados os requisitos funcionais e não-funcionais identificados. Posteriormente, são abordados os componentes *OpenFlow* envolvidos na descrição realizada e em seguida é apresentado o perfil *NSDL OpenFlow* onde se pode verificar os objetos utilizados e necessários à descrição de testes *OpenFlow*. O formato utilizado para representar a informação dos perfis foi o XML *Schema* (XML SCHEMA, 2012).

5.2 IDENTIFICAÇÃO DOS REQUISITOS

Nesta seção são apresentados os diferentes requisitos a serem cumpridos pelo domínio a ser modelado, no caso as redes OpenFlow. Estes requisitos são classificados em (SOARES, 2005):

- Requisitos funcionais: descrevem as funcionalidades que se espera que o sistema disponibilize, de uma forma completa e consistente. É aquilo que o

usuário espera que o sistema ofereça, atendendo aos propósitos para qual o sistema será desenvolvido; e

- Requisitos não-funcionais: referem-se a aspectos não-funcionais do sistema, como restrições nas quais o sistema deve operar ou propriedades emergentes do sistema. Costumam ser divididos em Requisitos não-funcionais de: Utilidade, Confiança, Desempenho, Suporte e Escalabilidade.

Em particular, os *requisitos funcionais* estão orientados à descrição dos componentes de um cenário de rede *OpenFlow* que deverão ser contemplados no perfil *NSDL OpenFlow*. Já os *requisitos não-funcionais* estão associados aos aspectos de performance, escalabilidade, completude, etc. que a descrição *NSDL OpenFlow* deverá suportar de forma a atender aos objetivos a qual foi proposta.

Dessa forma, os **Requisitos Funcionais** identificados para o perfil NSDL OpenFlow estão relacionados à descrição dos componentes de um cenário *OpenFlow* e de seus respectivos comportamentos:

- Computadores: são os elementos ativos de comunicação relacionados à origem e destino de dados;
- Comutadores: são os equipamentos de rede ativos de comunicação responsáveis pela comutação de dados, citados nesta dissertação como não suportando o protocolo *OpenFlow*;
- Comutadores *OpenFlow*: é a extensão de um comutador (*switch*) que suporta a execução de um protocolo para programar a tabela de fluxo dos equipamentos;
- Ligações: são as conexões existentes entre os componentes de uma rede (por exemplo, ligações entre computadores e os comutadores *OpenFlow* e entre os comutadores *OpenFlow* entre si);
- Máquinas virtuais: são implementadas como uma extensão de um elemento computador, possuem as mesmas características, e seu propósito é a replicação de recursos (processamento, memória, armazenamento, etc.) de forma a viabilizar a operacionalização da rede;
- Tabelas de fluxo: são gerenciadas nos comutadores *OpenFlow* e o seu funcionamento contempla a manipulação de campos do cabeçalho de cada pacote (utilizados para definir um fluxo), implementação de ações (define como os pacotes devem ser processados e para onde devem ser

encaminhados) e contadores (utilizados para estatísticas ou remoção de fluxos inativos). Cada fluxo de um cenário pode ser controlado, através da escolha da rota e o tratamento que cada pacote deve receber;

- Controlador: é um elemento (normalmente um servidor) que tem como finalidade controlar os fluxos da rede *OpenFlow* (Plano de Controle), realizando a inserção e remoção de entradas em uma tabela de fluxo, e;
- Canal Seguro: implementa uma ligação entre o comutador OpenFlow e o controlador através do Protocolo OpenFlow, onde os aspectos de confidencialidade, autenticação e integridade são garantidos na comunicação.

Os **Requisitos Não-Funcionais** identificados para o perfil NSDL OpenFlow estão relacionados às características da linguagem que permitam a sua fácil utilização, representatividade, manutenção, extensão, etc. Dentre os requisitos estão:

- Utilidade – A linguagem deverá ser expressiva o suficiente para representar todos os componentes de um cenário *OpenFlow*, deverá também descrever a dinâmica do cenário, através da descrição do seu tráfego e da especificação do seu controle de fluxo. Também deve ser simples e estruturada de forma a proporcionar a interoperabilidade entre diferentes ferramentas;
- Facilidade de uso – Dada a sua estrutura, a linguagem deve facilitar a leitura e manipulação de dados e informações de redes *OpenFlow*;
- Desempenho – Ainda dada à simplicidade e a sua estrutura, a linguagem deve permitir um bom desempenho facilitando a leitura e manipulação rápidas dos seus elementos;
- Manutenabilidade – A linguagem deve facilitar a descrição dos elementos da linguagem, permitindo uma manutenção rápida e simplificada;
- Escalabilidade – Por se tratar de uma linguagem baseada em XML, é facilmente extensível com novos elementos da linguagem;
- Portabilidade – Ainda com base no XML, a linguagem é apropriada para ambientes heterogêneos, e;
- Confiança – O grau de confiança da linguagem proposta pode ser aferido através da realização de alguns testes como estudos de caso.

> Esses testes nos permitem concluir se a extensão *NSDL OpenFlow* responde aos demais requisitos não-funcionais listados anteriormente, tais como aspectos de completude, relevância, escalabilidade, etc.

A partir dos requisitos identificados foi possível desenvolver uma extensão à linguagem *NSDL*. Essa extensão foi desenvolvida com o nome de perfil *NSDL OpenFlow*.

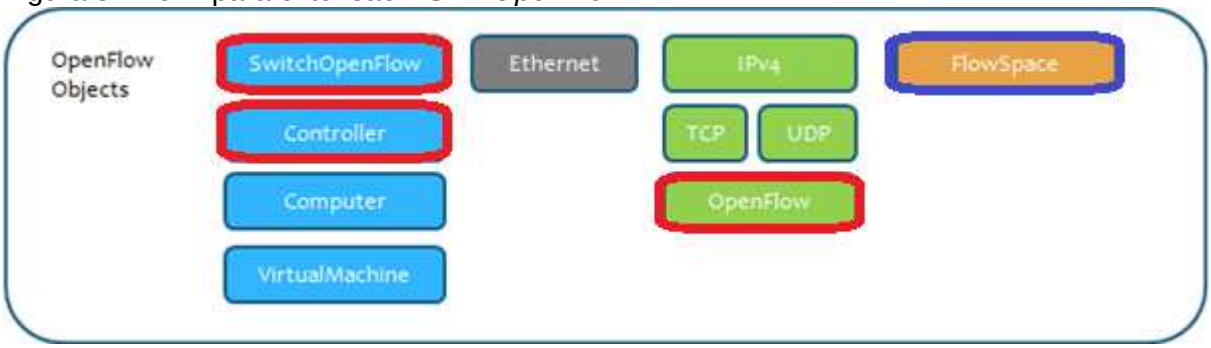
Seus componentes foram todos validados através da realização de estudos de caso baseados na descrição de testes *OpenFlow*. A seguir é apresentada a codificação de todos os componentes necessários para dar origem ao perfil proposto.

5.3 PERFIL NSDL OPENFLOW

O perfil *NSDL OpenFlow* agrega todos os objetos de redes necessários e utilizados para redes *OpenFlow*. Um Comutador *OpenFlow*, como citado no capítulo 3, é composto por três partes: Tabela de Fluxos, Canal Seguro e o Protocolo *OpenFlow*.

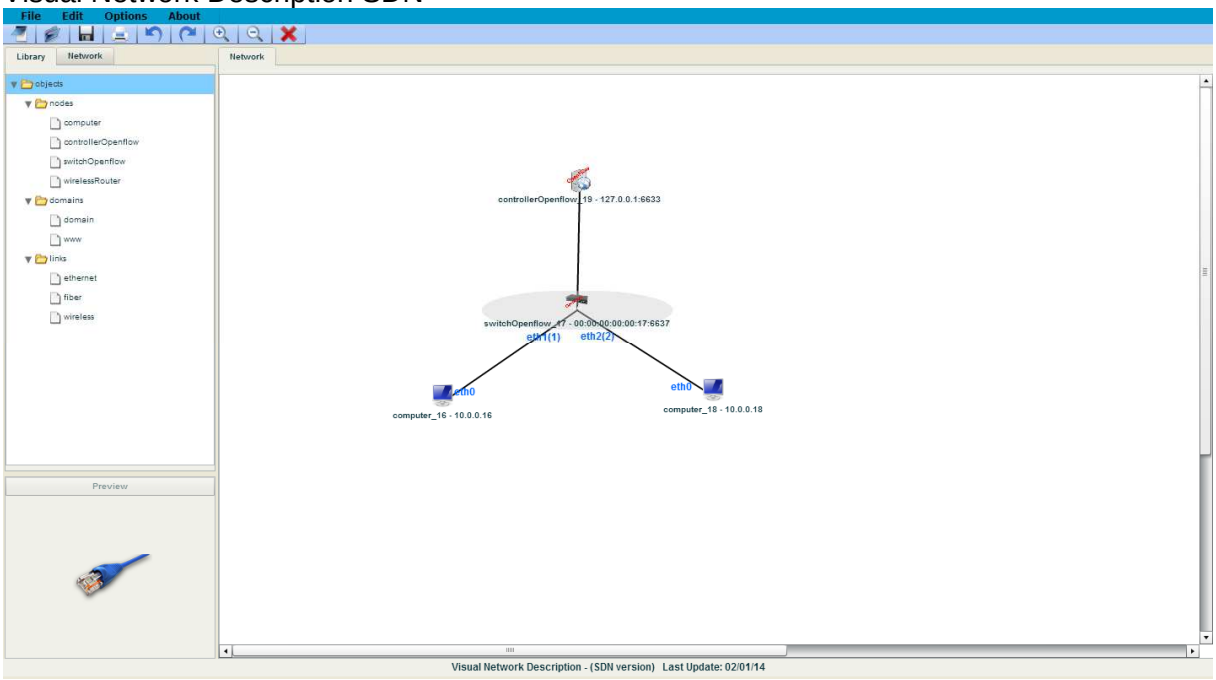
Para a criação do perfil *NSDL OpenFlow*, a tabela de fluxo é denominada de *FlowTable*, enquanto o comutador foi denominado de *switch OpenFlow* (também referido neste texto como *switchOpenFlow*). Como o canal seguro é implementado e gerenciado pelo protocolo SSL (MCKEOWN et al., 2008), ele não foi contemplado no perfil NSDL OpenFlow. Na Figura 8 é representado o perfil para extensão NSDL OpenFlow onde é possível verificar os objetos básicos utilizados neste perfil.

Figura 8 - Perfil para extensão *NSDL OpenFlow*



Como é possível verificar na Figura 8, o perfil OpenFlow foi proposto a partir da extensão dos elementos do perfil base de NSDL. Os elementos adicionados nessa proposta ao perfil OpenFlow estão circulados em vermelho. De forma a ilustrar esses elementos, consideremos o cenário apresentado na Figura 9, que foi criado utilizando a ferramenta para a autoria de cenários de rede OpenFlow *Visual Network Description (VND – SDN version)* (FONTES, 2013).

Figura 9 – Autoria gráfica do cenário OpenFlow descrito na Figura 18 utilizando a ferramenta Visual Network Description SDN



Fonte: Fontes (2013).

O cenário *OpenFlow* apresentado na Figura 9 ilustra dois terminais conectados a um *comutador OpenFlow (switchOpenFlow)*. Por sua vez, esse comutador está conectado a um objeto controlador. De forma a ilustrar a descrição em NSDL *OpenFlow* alguns dos elementos deste cenário, consideremos:

- O *SwitchOpenFlow* é baseado no elemento comutador descrito no perfil base da linguagem *NSDL*, e adaptado para a realidade do perfil proposto. Este elemento apresenta: um identificador (id); seu endereço MAC (*swittchMACAddress*) e a sua porta de comunicação (*listeningPort*) (Figura10);

Figura 10 - Código para o *SwitchOpenFlow* descrito pelo *NSDL*

```
<switchOpenflow id="switchOpenflow1">
  <switchMacAddress>00:00:00:00:00:03</switchMacAddress>
  <listeningPort>6635</listeningPort>
  <switch>ovsk</switch>
  <notes/>
</switchOpenflow>
```

- O *controlador* é descrito por um id, seu endereço ip (*controllerIPAddress*), a sua porta de comunicação (*controllerPort*) e a tabela de fluxo que está sendo controlada (*openflowController*) (Figura 11);

Figura 11 - Código para o Controlador descrito pelo *NSDL*

```
<controllerOpenflow id="controllerOpenflow1">
  <controllerIPAddress>127.0.0.1</controllerIPAddress>
  <controllerPort>6633</controllerPort>
  <openflowController>null</openflowController>
</controllerOpenflow>
```

- O *computador* possui além do id, endereço ip (*computerIPAddress*), mascára de rede (*mask*) e o endereço MAC (*computerMacAddress*), (Figura 12), e;

Figura 12 - Código para o Computador descrito pelo *NSDL*

```
<computer id="computer1">
  <computerIPAddress>10.0.0.16</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:16</computerMacAddress>
</computer>
```

- A *máquina virtual* também segue os mesmos padrões do elemento computador, além de possuir um nome (*name*), quantidade de memória (memory), a imagem de disco utilizada (*discImage*) o tipo de disco rígido (*hdSetupType*), o tipo de virtualização (*virtualizationSetupType*), o estado de funcionamento no momento (*state*) e o sistema operacional (*operatingSystem*) (Figura 13).

Figura 13 - Código para o Máquina Virtual descrito pelo *NSDL*

```
<virtualMachine id="VM1">
    <ipv4 id="ipv4VM1">
        <address>10.216.27.19</address>
        <mask>255.255.255.0</mask>
        <interface.id>interface1</interface.id>
    </ipv4>
    <name>maquinaVitual</name>
    <memory>256</memory>
    <discImage>Default</discImage>
    <hdSetupType>File Image</hdSetupType>
    <virtualizationSetupType>Paravirtualization</virtualizationSetupType>
    <state>running</state>
    <operatingSystem>GNU/Linux Debian (6.0)</operatingSystem>
</virtualMachine>
```

O perfil para extensão *NSDL OpenFlow* possui ainda as descrição para a ligação. Em termos de protocolos, o perfil suporta atualmente: IPv4, TCP, UDP e *OpenFlow*, podendo ser estendido para outros se necessário. A seguir a descrição dos protocolos é ilustrada acompanhada de trechos dos seus respectivos códigos:

IPV4 é o Protocolo de Internet, sua descrição é obtida a partir do elemento IPV4 da *NSDL*, possui endereço ip (*address*), mascára de rede (*mask*) e interface de rede a qual está relacionada (*interface.id*) (Figura14);

Figura 14 - Código para o IPV4 descrito pelo *NSDL*

```
<ipv4 id="ipv4serv1">
<address>200.123.1.51</address>
<mask>255.255.255.0</mask>
<interface.id>interface1</interface.id>
</ipv4>
```

- *TCP* representa o *Transmission Control Protocol*, baseia-se no elemento TCP e possui o ip remetente (*ipSource*), o ip de destino (*ipDestination*), um campo para verificação de erros (*checksum*), e um controle booleano (*flag*) (Figura 15);

Figura 15 - Código para o protocolo TCP descrito pelo *NSDL*

```
<tcp id="tcp">
    <ipSource>200.123.1.51</ipSource>
    <ipDestination>201.127.1.33</ipDestination>
    <checksun>15892099</checksun>
    <dados></dados>
    <flag></flag>
</tcp>
```

- *UDP*, baseado no UDP da *NSDL*, possui ip do remetente e do destinatário, além do campo para os dados que são transmitidos (Figura 16);

Figura 16 - Código para o protocolo UDP descrito pelo *NSDL*

```
<udp id="udp">
    <ipSource>200.123.1.51</ipSource>
    <ipDestination>201.127.1.33</ipDestination>
    <dados></dados>
</udp>
```

- *OpenFlow* é o protocolo que permite que o controlador se comunique com o *SwitchOpenFlow* enviando as instruções necessárias para que o controlador realize suas ações (MCKEOWN et al., 2008). Possui instâncias para o controlador (*controller*) e para o *switchOpenFlow* de destino (*switchOpenFlow*) (Figura 17).

Figura 17 - Código para o protocolo *OpenFlow* descrito pelo *NSDL*

```
<openflowprotocol id="openflowprotocol1">
   <controller>controller1</controller>
   <switchopenflow>switchopenflow1</switchopenflow>
</openflowprotocol>
```

- *Tabela de fluxo* – Contém basicamente as informações referentes à origem e destino de cada fluxo. Cada entrada na tabela de fluxo descreve o endereço mac de origem (*macSource*), endereço mac de destino (*macDestination*), endereço Ip de origem (*ipSource*), Ip de destino (*ipDestination*), e o endereço mac do comutador ao qual se destina o fluxo (*macSwitch*), ilustrado na Figura 18, e;

Figura 18 - Código para a Tabela de Fluxo descrito pelo *NSDL*

```
<openflow>
  <flowTable1 id="controllerOpenflow_4">
    <flowName>flow1</flowName>
    <priority/>
    <macSwitch>00:00:00:00:00:03</macSwitch>
    <macSource>00:00:00:00:00:01</macSource>
    <macDestination>00:00:00:00:00:02</macDestination>
    <ingressPort/>
    <vlanID/>
    <vlanPriority/>
    <ethtype/>
    <tos/>
    <protocol/>
    <ipSource>10.0.0.1</ipSource>
    <ipDestination>10.0.0.2</ipDestination>
    <sourcePort/>
    <destinationPort/>
    <setOutput/>
    <setMACSource>00:00:00:00:00:01</setMACSource>
    <setMACDestination>00:00:00:00:00:02</setMACDestination>
    <setEnqueue/>
    <setVlanID/>
    <setVLANPriority/>
    <setStripVlan/>
    <setTOS/>
    <setIPSource>10.0.0.1</setIPSource>
    <setIPDestination>10.0.0.2</setIPDestination>
    <setSourcePort/>
    <setDestinationPort/>
    <notes/>
  </flowTable1>
 </openflow>
</scenarios>
</nsdl>
```

A Figura 19 apresenta um trecho do código *NSDL* relacionado com o perfil para extensão *NSDL OpenFlow* onde é possível observar de uma forma integrada e resumida: o computador e o *switchOpenFlow*. A sintaxe completa para a descrição desses elementos é apresentada no XML Schema introduzido no Anexo A – *NSDL Schema Perfil NSDL-OpenFlow.*

Figura 19 - Trecho do código para extensão *NSDL OpenFlow*

```
<nsdl>
  <network>
    <objects>
      <link id="ethernet_1">...</link>
      <computer id="computer_1">...</computer>
      <switchOpenflow id="switchOpenflow_1">...</switchOpenflow>
      <controllerOpenflow id="controllerOpenflow_1">...</controllerOpenflow>
    </objects>
  </network>
  <scenarios>
    <visualizations>
      <visualization id="vis01">
        <description>
          <object id="computer_1">...</object>
          <object id="switchOpenflow_1">...</object>
          <object id="controllerOpenflow_1">...</object>
        </description>
      </visualization>
    </visualizations>
    <openflow>
      <flowTable1 id="flowtable1">...</flowTable1>
    </openflow>
  </scenarios>
</nsdl>
```

A partir da definição dos elementos do perfil *NSDL OpenFlow*, é possível a descrição de testes dos mais simples aos mais complexos, o que pode ser validado através de estudos de caso, apresentados no capítulo 7.

5.4 CONCLUSÃO

Este capítulo descreveu como a extensão à *Framework NSDL* e o desenvolvimento do perfil *NSDL OpenFlow* foram realizados. Esse perfil foi ilustrado através de trechos dos códigos desenvolvidos para a extensão.

O perfil *NSDL OpenFlow* foi desenvolvido através da identificação dos requisitos desse domínio de redes, da observação e da utilização de algumas ferramentas para a criação de testes *OpenFlow*, como OFELIA. A partir desse estudo, foi possível a proposta e validação da extensão realizada, chamada de *NSDL OpenFlow*.

Primeiramente foram apresentados a base da *Framework NSDL* e a identificação dos requisitos funcionais e não-funcionais para a descrição de cenários experimentais OpenFlow. A partir do perfil base, seguindo a utilização dos componentes mais genéricos, foi possível a proposta do perfil *NSDL OpenFlow*. O perfil base permitiu a introdução dos objetos de rede genéricos que compõem um cenário básico de rede, ou seja, o nó, a ligação, o protocolo, a aplicação, a interface e o domínio. Através deste perfil é permitido criar outros perfis acrescentando ou removendo objetos de forma a se adequar ao perfil pretendido (ARAUJO, 2011).

Neste capítulo foi introduzida a proposta para a descrição do perfil *NSDL OpenFlow*. Baseada nessa proposta é possível desenvolver diferentes cenários de redes *OpenFlow* através de adaptações realizadas ao perfil base, assim como sugere a linguagem *NSDL* e os demais trabalhos que utilizaram a *Framework NSDL* (MARQUES et al., 2011).

6 ESTUDO DE CASO

Neste capítulo são apresentados três estudos de caso de forma a ilustrar a utilização da extensão de *NSDL* para a descrição de cenários OpenFlow. Para isso, foram definidos dois cenários de redes. O cenário 1 é um cenário simples constituído por 2 computadores, um comutador OpenFlow e um controlador (com a sua respectiva tabela de fluxo).

No cenário 2 é introduzido um ambiente de rede um pouco mais elaborado do que o cenário 1, constituído por duas ilhas - uma ilha é uma infra-estrutura de rede constituída para realização de testes e experimentos, possuindo computadores, comutadores e outros equipamentos de comunicação conectados a outras redes experimentais e/ou reais (FIBRE, 2012) - uma localizada em Salvador (Brasil) e outra em Sidney (Austrália). Cada ilha possui 2 servidores, 1 laptop e 1 comutador OpenFlow. As ilhas estão conectadas entre si através de 6 roteadores.

De forma a validar a utilização da descrição NSDL como fator de integração entre diferentes ferramentas para a gestão de redes, o cenário 3 foi desenvolvido. Este cenário ilustra como a descrição NSDL pode ser utilizada no ciclo de vida experimental de um cenário OpenFlow (autoria/modelagem, exportação para descrição NSDL, simulação NS-3).

Na próxima seção são apresentados os cenários de estudo de caso.

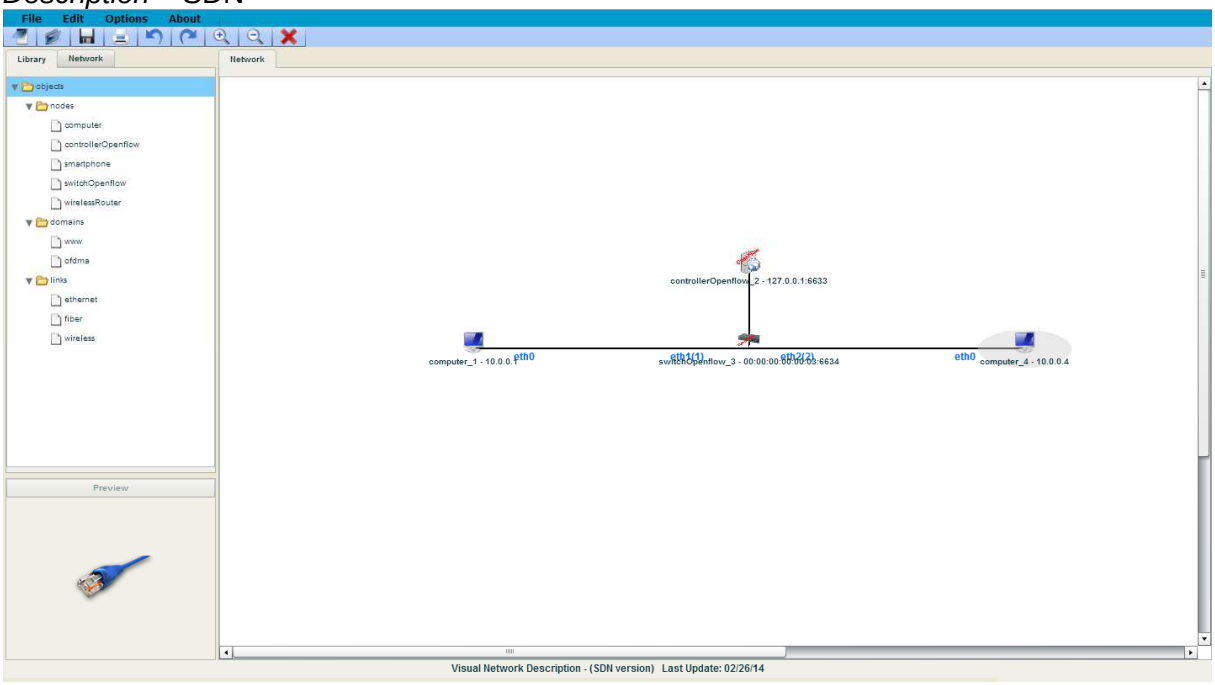
6.1 CENÁRIO 1

O cenário 1 foi proposto como uma experiência simples na descrição de um cenário *OpenFlow*. Neste primeiro cenário o objetivo foi validar a descrição dos principais componentes de um cenário OpenFlow, sua configuração e interligações. Neste momento, foi possível identificar potenciais problemas e inconsistências existentes relacionadas aos requisitos levantados.

O cenário 1 é composto por dois computadores, o servidor 1 e o servidor 2, por um *switchOpenFlow,* identificado como *switchOpenFlow* 1.

A topologia do cenário 1 é ilustrada utilizando a ferramenta *Visual Network Description* (versão SDN) (FONTES, 2013), como é possível verificar na Figura 22.

Figura 20 - Representação da topologia do cenário 1 utilizando a ferramenta *Visual Network Description* – SDN



Fonte: Fontes (2013).

A seguir, a Figura 21 apresenta o código resumido para o perfil NSDL *OpenFlow* para o cenário 1.

Figura 21 - Representação do código NSDL para o cenário 1

```
<nsdl>
  <network>
    <objects>
      <link id="ethernet_1">...</link>
      <computer id="computer_1">...</computer>
      <computer id="computer_2">...</computer>
      <switchOpenflow id="switchOpenflow_1">...</switchOpenflow>
      <controllerOpenflow id="controllerOpenflow_1">...</controllerOpenflow>
    </objects>
    <views/>
  </network>
  <scenarios>
    <visualizations>
      <visualization id="vis01">
        <description>
          <object id="computer_1">...</object>
          <object id="computer_2">...</object>
          <object id="switchOpenflow_1">...</object>
          <object id="controllerOpenflow_1">...</object>
        </description>
      </visualization>
    </visualizations>
    <openflow>
      <flowTable1 id="flowtable1">...</flowTable1>
    </openflow>
  </scenarios>
</nsdl>
```

Em seguida, apresentamos na Figura 22 o código NSDL dando ênfase ao elemento *network*, onde é possível observar os objetos pertencentes ao perfil NSDL *openFlow* tais como: *links*, computadores, o *switchOpenFlow* e o controlador.

Figura 22 - Representação do código NSDL para o cenário 1 (ênfase elemento *network*)

```
<nsdl>
  <network>
    <objects>
      <link id="ethernet_1">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="switchOpenflow_1" destination="controllerOpenflow_1"/>
      </link>
      <link id="ethernet_2">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="computer_1" destination="switchOpenflow_1"/>
      </link>
      <link id="ethernet_3">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="computer_2" destination="switchOpenflow_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>
      <controllerOpenflow id="controllerOpenflow_1">
        <controllerIPAddress>127.0.0.1</controllerIPAddress>
        <controllerPort>6633</controllerPort>
        <specialController>none</specialController>
        <openflowController>floodlight</openflowController>
      </controllerOpenflow>
      <switchOpenflow id="switchOpenflow_1">
        <switchMacAddress>00:00:00:00:00:03</switchMacAddress>
        <listeningPort>6634</listeningPort>
        <switch>ovsk</switch>
        <notes/>
      </switchOpenflow>
      <computer id="computer_2">
        <computerIPAddress>10.0.0.4</computerIPAddress>
        <mask>255.0.0.0</mask>
        <computerMacAddress>00:00:00:00:00:04</computerMacAddress>
        <notes/>
      </computer>
    </objects>
    <views/>
  </network>
  <scenarios>...</scenarios>
</nsdl>
```

A Figura 22 evidencia o código NSDL para o elemento *network* sendo possível observar os seus sub-itens: *Link*, que são as ligações entre os equipamentos, os computadores envolvidos no cenário, visualizamos também, os controladores e comutadores *openFlow* descritos. A descrição desses elementos foi realizada no capítulo 6.

A Figura 23 apresenta o código NSDL para o elemento *scenarios* referente à visualização e posicionamento dos objetos na interface gráfica do VND (versão SDN).

Figura 23 - Representação do código NSDL para o cenário 1 (ênfase elemento *scenarios*)

```
<nsdl>
  <network>...</network>
   <scenarios>
    <visualizations>
      <visualization id="vis01">
        <description>
          <object id="computer_1">
            <x.position>332</x.position>
            <y.position>463</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="controllerOpenflow_1">
            <x.position>530</x.position>
            <y.position>112</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="switchOpenflow_1">
            <x.position>518</x.position>
            <y.position>296</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="computer_2">
            <x.position>770</x.position>
            <y.position>468</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
        </description>
      </visualization>
    </visualizations>
    <openflow>
    <flowTable1 id="controllerOpenflow_1">...</flowTable1>
    </openflow>
  </scenarios>
</nsdl>
```

A Figura 23, que representa o código para o elemento *scenarios*, ilustra como sub-itens, o *visualizations*, que descreve a localização dos elementos visualizados no cenário além dos elementos que se referem ao *openFlow*, apresentados na figura seguinte.

A seguir, a Figura 24 apresenta o código NSDL para o elemento *scenarios* referente a tabela de fluxo *OpenFlow*.

Figura 24 - Representação do código NSDL para o cenário 1 (ênfase elemento *flowTable*)

```
<nsdl>
  <network>...</network>
   <scenarios>
    <visualizations>...</visualizations>
    <openflow>
      <flowTable1 id="controllerOpenflow_1">
        <flowName>flow1</flowName>
        <priority/>
        <macSwitch>00:00:00:00:00:03</macSwitch>
        <macSource>00:00:00:00:00:01</macSource>
        <macDestination>00:00:00:00:00:02</macDestination>
        <ingressPort/>
        <vlanID/>
        <vlanPriority/>
        <ethtype/>
        <tos/>
        <protocol/>
        <ipSource>10.0.0.1</ipSource>
        <ipDestination>10.0.0.2</ipDestination>
        <sourcePort/>
        <destinationPort/>
        <setOutput/>
        <setMACSource>00:00:00:00:00:01</setMACSource>
        <setMACDestination>00:00:00:00:00:02</setMACDestination>
        <setEnqueue/>
        <setVlanID/>
        <setVLANPriority/>
        <setStripVlan/>
        <setTOS/>
        <setIPSource>10.0.0.1</setIPSource>
        <setIPDestination>10.0.0.2</setIPDestination>
        <setSourcePort/>
        <setDestinationPort/>
        <notes/>
      </flowTable1>
    </openflow>
  </scenarios>
</nsdl>
```

O elemento *flowTable*, apresentado pela Figura 24, representa a tabela de fluxo do componente *openFlow.* A tabela de fluxo possui ainda elementos que iniciam-se com *set* que irão determinar os fluxos de entrada e saída em um controlador. Como exemplo ilustrado, temos o elemento *setIPDestination* que enviará o comando para que o controlador configure o IP de destino de um determinado fluxo. Os outros elementos *sets* também agem de maneira similar. Sub-itens, como *macSwitch, macSource e macDestination,* dizem respeito ao fluxo que foi inicialmente configurado pelo controlador.

O cenário 1 foi construído para servir como base para o desenvolvimento do cenário 2 e como forma de auxílio na experimentação e criação do código *NSDL OpenFlow* utilizado para a descrição do cenário 2. No entanto, apresentamos o código para o cenário 2 e outros detalhes sobre esse cenário a seguir.

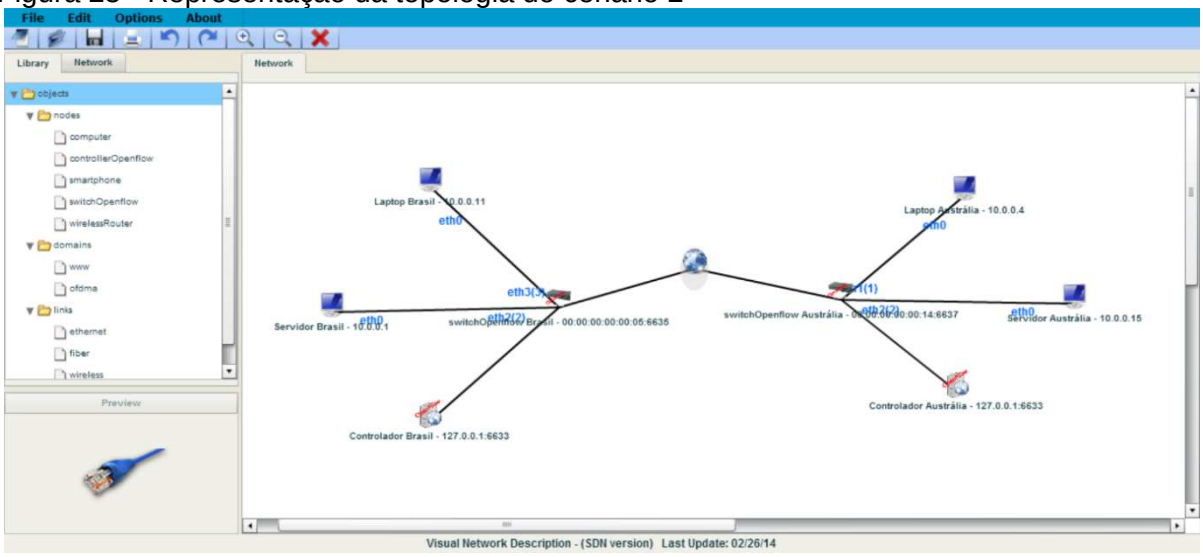
6.2 CENÁRIO 2

A realização do cenário 1 ilustra a viabilidade da utilização da exensão do perfil base de NSDL para a descrição de cenários *OpenFlow*. No cenário 2 são explorados os recursos do perfil NSDL *OpenFlow* para a descrição de um cenário mais complexo e elaborado.

Para o cenário 2 foi proposta a elaboração de um cenário de redes *OpenFlow* mais realístico. Para isso, foi realizada a adaptação de um cenário real desenvolvido no projeto FIBRE (FIBRE, 2012). O projeto FIBRE tem como objetivo principal a concepção, implementação e validação de uma plataforma para a realização de monitoração e testes de rede, apoiando a experimentação conjunta de pesquisadores europeus e brasileiros (FIBRE, 2012).

O cenário em questão é constituído por 6 computadores: 2 controladores, 2 *laptops* e 2 servidores. O cenário 2 ainda possui dois *comutadores OpenFlow*, como podem ser verificados na Figura 25

Figura 25 - Representação da topologia do cenário 2



A Figura 26 apresenta a descrição *NSDL OpenFlow* resumida para o cenário 2. Dessa forma, é possível obter uma visão geral do código gerado para o cenário em questão.

Figura 26 - Representação de uma visão geral do código NSDL para o cenário 2

```
<nsdl>
  <network>
    <objects>
      <link id="ethernet_1">...</link>
      <computer id="servidor_Brasil">...</computer>
      <computer id="laptop_Brasil">...</computer>
      <computer id="servidor_Australia">...</computer>
      <computer id="laptopr_Australia">...</computer>
      <switchOpenflow id="switchOpenflow_Brasil">...</switchOpenflow>
      <controllerOpenflow id="controllerOpenflow_Brasil">...</controllerOpenflow>
      <switchOpenflow id="switchOpenflow_Australia">...</switchOpenflow>
      <controllerOpenflow id="controllerOpenflow_Australia">...</controllerOpenflow>
    </objects>
  </network>
  <scenarios>
    <visualizations>
      <visualization id="vis01">
        <description>
          <object id="servidor_Brasil">...</object>
          <object id="laptop_Brasil">...</object>
          <object id="servidor_Australia">...</object>
          <object id="laptopr_Australia">...</object>
          <object id="switchOpenflow_Brasil">...</object>
          <object id="controllerOpenflow_Brasil">...</object>
          <object id="switchOpenflow_Australia">...</object>
          <object id="controllerOpenflow_Australia">...</object>
        </description>
      </visualization>
    </visualizations>
    <openflow>
      <flowTable1 id="controllerOpenflow_Brasil">...</flowTable1>
      <flowTable2 id="controllerOpenflow_Australia">...</flowTable2>
    </openflow>
  </scenarios>
</nsdl>
```

A Figura 27 – Representação do código NSDL do componente *network* para o cenário 2, apresenta o código gerado para o perfil *NSDL OpenFlow* para os componentes do cenário 2. Esses elementos são basicamente muito semelhantes, pois são todos descritos a partir do mesmo objeto, computador.

Figura 27 - Representação do código NSDL do componente *network* para o cenário 2

```
<nsdl>
  <network>
    <objects>
      <link id="ethernet_1">...</link>
      <computer id="servidor_Brasil">
        <computerIPAddress>10.0.0.1</computerIPAddress>
        <mask>255.0.0.0</mask>
        <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
        <notes/>
      </computer>
      <computer id="laptop_Brasil">
        <computerIPAddress>10.0.0.4</computerIPAddress>
        <mask>255.0.0.0</mask>
        <computerMacAddress>00:00:00:00:00:04</computerMacAddress>
        <notes/>
      </computer>
      <computer id="servidor_Australia">
        <computerIPAddress>10.0.0.1</computerIPAddress>
        <mask>255.0.0.0</mask>
        <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
        <notes/>
      </computer>
      <computer id="laptopr_Australia">
        <computerIPAddress>10.0.0.4</computerIPAddress>
        <mask>255.0.0.0</mask>
        <computerMacAddress>00:00:00:00:00:04</computerMacAddress>
        <notes/>
      </computer>
      <switchOpenflow id="switchOpenflow_Brasil">
        <switchMacAddress>00:00:00:00:00:05</switchMacAddress>
        <listeningPort>6635</listeningPort>
        <switch>ovsk</switch>
        <notes/>
      </switchOpenflow>
      <controllerOpenflow id="controllerOpenflow_Brasil">
        <controllerIPAddress>127.0.0.1</controllerIPAddress>
        <controllerPort>6633</controllerPort>
        <specialController>none</specialController>
        <openflowController>nox</openflowController>
      </controllerOpenflow>
      <switchOpenflow id="switchOpenflow_Australia">
        <switchMacAddress>00:00:00:00:00:14</switchMacAddress>
        <listeningPort>6637</listeningPort>
        <switch>ovsk</switch>
        <notes/>
      </switchOpenflow>
      <controllerOpenflow id="controllerOpenflow_Australia">
        <controllerIPAddress>127.0.0.1</controllerIPAddress>
        <controllerPort>6633</controllerPort>
        <specialController>none</specialController>
        <openflowController>undefined</openflowController>
      </controllerOpenflow>
    </objects>
    <views/>
  </network>
  <scenarios>...</scenarios>
</nsdl>
```

O elemento *controlador* é descrito por um id, seu endereço ip (*controllerIPAddress*), a sua porta de comunicação (*controllerPort*) e a tabela de fluxo que está sendo controlada (*openflowController*), assim como já foi explicado em seções anteriores dessa dissertação.

O *SwitchOpenFlow* apresenta: um identificador (id); seu endereço MAC (*swittchMACAddress*) e a sua porta de comunicação (*listeningPort*).

As próximas figuras ilustram o trecho de código que se refere ao componente *Scenarios* e seus subcomponentes. Neste caso, o componente *Scenarios* irá descrever alguns aspectos gerais da configuração *OpenFlow*.

Figura 28 - Representação do código do componente *Scenarios* para o cenário 2

```
<nsdl>
  <network>...</network>
  <scenarios>
    <visualizations>
      <visualization id="vis01">
        <description>
          <object id="servidor_Brasil">
            <x.position>77</x.position>
            <y.position>220</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="laptop_Brasil">
            <x.position>748</x.position>
            <y.position>96</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="switchOpenflow_Brasil">
            <x.position>263</x.position>
            <y.position>215</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="controllerOpenflow_Brasil">
            <x.position>158</x.position>
            <y.position>336</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="servidor_Australia">
            <x.position>184</x.position>
            <y.position>87</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="switchOpenflow_Australia">
            <x.position>556</x.position>
            <y.position>207</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="laptopr_Australia">
            <x.position>857</x.position>
            <y.position>211</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="www">
            <x.position>512</x.position>
            <y.position>174</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
          <object id="controllerOpenflow_Australia">
            <x.position>705</x.position>
            <y.position>332</y.position>
            <z.position>0</z.position>
            <color/>
          </object>
        </description>
      </visualization>
    </visualizations>
     <openflow>
      <flowTable1 id="controllerOpenflow_Brasil">...</flowTable1>
      <flowTable2 id="controllerOpenflow_Australia">...</flowTable2>
    </openflow>
  </scenarios>
</nsdl>
```

Na Figura 28 é possível observar os objetos responsáveis pelo posicionamento dos elementos que compõe o cenário descrito, os sub-elementos pertencentes ao elemento *visualizations* dizem respeito a posicionamento, cor e outras características inerentes a visualização. É possível ainda, observar os sub-elementos pertencentes a descrição do *openFlow*, como a tabela de fluxo.

As Figuras 29 e 30, apresentam as tabelas de fluxo para o cenário proposto.

Figura 29 - Representação do código NSDL do componente *Scenarios* para o cenário 2

```
<nsdl>
  <network>...</network>
  <scenarios>
    <visualizations>...</visualizations>
     <openflow>
      <flowTable1 id="controllerOpenflow_Brasil">
        <flowName>flow1</flowName>
        <priority/>
        <macSwitch>00:00:00:00:00:05</macSwitch>
        <macSource>00:00:00:00:00:11</macSource>
        <macDestination>00:00:00:00:00:04</macDestination>
        <ingressPort/>
        <vlanID/>
        <vlanPriority/>
        <ethtype/>
        <tos/>
        <protocol/>
        <ipSource>10.0.0.11</ipSource>
        <ipDestination>10.0.0.4</ipDestination>
        <sourcePort/>
        <destinationPort/>
        <setOutput/>
        <setMACSource>00:00:00:00:00:11</setMACSource>
        <setMACDestination>00:00:00:00:00:04</setMACDestination>
        <setEnqueue/>
        <setVlanID/>
        <setVLANPriority/>
        <setStripVlan>null</setStripVlan>
        <setTOS/>
        <setIPSource>10.0.0.11</setIPSource>
        <setIPDestination>10.0.0.4</setIPDestination>
        <setSourcePort/>
        <setDestinationPort/>
        <notes/>
      </flowTable1>
      <flowTable2 id="controllerOpenflow_Australia">...</flowTable2>
     </openflow>
  </scenarios>
</nsdl>
```

Figura 30 - Representação do código do componente *Scenarios* para o cenário 2

```
<nsdl>
 <network>...</network>
 <scenarios>
    <visualizations>...</visualizations>
     <openflow>
      <flowTable1 id="controllerOpenflow_Brasil">...</flowTable1>
      <flowTable2 id="controllerOpenflow_Australia">
        <flowName>flow2</flowName>
        <priority/>
        <macSwitch>00:00:00:00:00:14</macSwitch>
        <macSource>00:00:00:00:00:04</macSource>
        <macDestination>00:00:00:00:00:11</macDestination>
        <ingressPort/>
        <vlanID/>
        <vlanPriority/>
        <ethtype/>
        <tos/>
        <protocol/>
        <ipSource>10.0.0.4</ipSource>
        <ipDestination>10.0.0.11</ipDestination>
        <sourcePort/>
        <destinationPort/>
        <setOutput/>
        <setMACSource>00:00:00:00:00:04</setMACSource>
        <setMACDestination>00:00:00:00:00:11</setMACDestination>
        <setEnqueue/>
        <setVlanID/>
        <setVLANPriority/>
        <setStripVlan/>
        <setTOS/>
        <setIPSource>10.0.0.4</setIPSource>
        <setIPDestination>10.0.0.11</setIPDestination>
        <setSourcePort/>
        <setDestinationPort/>
        <notes/>
      </flowTable2>
     </openflow>
 </scenarios>
</nsdl>
```

Nas Figuras 29 e 30 são descritos, respectivamente, os elementos *FlowTable* da ilha localizada no Brasil e da ilha localizada na Austrália. O elemento *FlowTable* é a tabela de fluxo descrita em seções anteriores dessa dissertação e contém basicamente as informações referentes à origem e destino de cada fluxo. Cada entrada na tabela de fluxo descreve o endereço mac de origem (*macSource*), endereço mac de destino (*macDestination*), endereço Ip de origem (*ipSource*), Ip de

destino (*ipDestination*), e o endereço mac do comutador ao qual se destina o fluxo (*macSwitch*).

De acordo com as figuras apresentadas, podemos observar o código *NSDL* para o cenário *OpenFlow* proposto. Nessas figuras foram apresentados os componentes *network* e *scenarios*, além de seus respectivos sub-componentes. A partir da realização deste cenário, foi possível constatar a viabilidade e representatividade de NSDL para a descrição de cenário mais complexos.

6.3 CENÁRIO 3

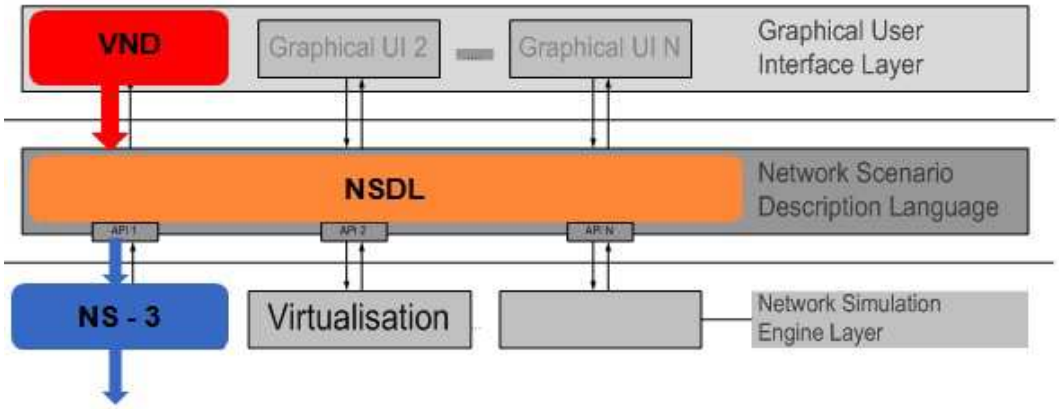
De forma a validar a utilização da descrição NSDL como fator de integração entre diferentes ferramentas para a gestão de redes, foi desenvolvido um terceiro estudo de caso. Este estudo de caso ilustra como a descrição NSDL pode ser utilizada no ciclo de vida experimental de um cenário OpenFlow (autoria/modelagem, exportação para descrição NSDL, simulação NS-3).

Como já citado nessa dissertação, o VND (versão *SDN*) é utilizado para a autoria de cenários *OpenFlow.* Depois da criação desse cenário, é possível gerar automaticamente uma descrição em NSDL equivalente. De forma a permitir a integração com outras ferramentas *OpenFlow* e, nesse caso, com o simulador NS-3, foi proposto o desenvolvimento de um script para o mapeamento automático entre NSDL e NS-3 (C++) (FONTES et al., 2014).

A estrutura do *script* é baseada em um código comum em C++ estruturado para a simulação de cenários *OpenFlow* assim como ilustrado em (SOUSA, 2011), (FONTES et al., 2014).

A Figura 31 apresenta o ciclo de vida de um cenário *OpenFLow* retratado através da *Framework NSDL,* começando com sua criação utilizando o VND (Versão SDN) como uma ferramenta gráfica aplicado a camada topo da framework NSDL, a subsequente geração automática da sua respectiva descrição NSDL (camada intermediária), e o mapeamento final em sua respectiva descrição NS-3 (camada inferior) (FONTES et al., 2014).

Figura 31 - Ciclo de vida de um cenário *OpenFlow* baseado na Framework NSDL aplicando as ferramentas VND e NS-3

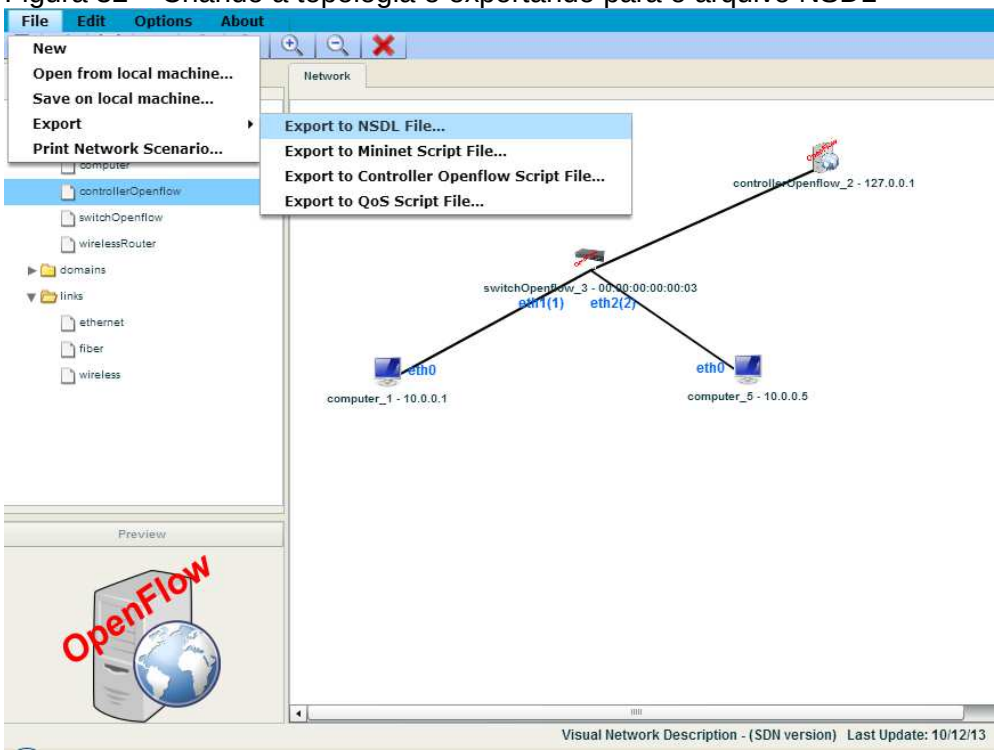


Fonte: Fontes et al (2014).

Conforme ilustrado na Figura 31, inicialmente a topologia do cenário foi criada usando VND (versão SDN). Durante a autoria do cenário é necessário incluir os nós que compõe o cenários (Servidores, Comutadores *OpenFlow,* Controlador e links) e configurar seus respectivos atributos para cada componente. Todos os componentes do cenário podem ser configurados usando algumas janelas e quadros interativos (FONTES et al., 2014).

Uma vez que o cenário foi criado e configurado, a sua descrição pode ser exportada para NSDL. A funcionalidade da exportação pode ser acessada usando o menu principal, como ilustrado na Figura 30 (FONTES et al., 2014).

Figura 32 – Criando a topologia e exportando para o arquivo NSDL



Fonte: Fontes et al (2014).

A Figura 33, expõe o código NSDL resumindo os elementos do cenário 3.

Figura 33 – Código NSDL resumido para o Cenário 3

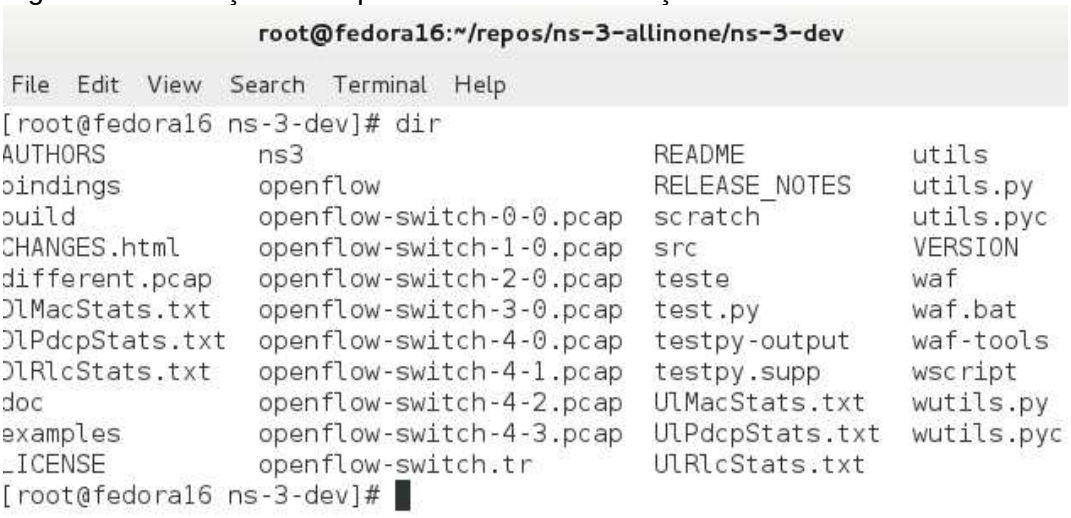
```
<nsdl>
  <network>
    <objects>
      <link id="ethernet_1">...</link>
      <computer id="computer_1">...</computer>
      <computer id="computer_2">...</computer>
      <switchOpenflow id="switchOpenflow_1">...</switchOpenflow>
      <controllerOpenflow id="controllerOpenflow_1">...</controllerOpenflow>
    </objects>
    <views/>
  </network>
  <scenarios>
    <visualizations>
      <visualization id="vis01">
        <description>
          <object id="computer_1">...</object>
          <object id="computer_2">...</object>
          <object id="switchOpenflow_1">...</object>
          <object id="controllerOpenflow_1">...</object>
        </description>
      </visualization>
    </visualizations>
    <openflow>
      <flowTable1 id="controllerOpenflow_1">...</flowTable1>
      <qosTable0 id="controllerOpenflow_1">...</qosTable0>
      <queueConfiguration0 id="controllerOpenflow_1">...</queueConfiguration0>
    </openflow>
  </scenarios>
</nsdl>
```

Assim como apresentado para os cenários anteriores, o código NSDL para o cenário 3, apresenta um resumo dos componentes descritos, posicionamento e a sua tabela de fluxo. A descrição NSDL completa para o cenário 3 é apresentada no Anexo D.

Após a geração do código NSDL, é possível mapear automaticamente essa descrição para qualquer código utilizado por uma ferramenta de simulação que suporte o protocolo *OpenFlow*. Nesse caso, foi utilizado um *script* desenvolvido para proporcionar o mapeamento automático entre NSDL e NS-3 (C++) (FONTES et al., 2014).

Uma vez o código C++ para o respectivo cenário foi gerado, a simulação do cenário pode ser executada utilizando NS-3. Durante a execução, NS-3 gera um conjunto de arquivos correspondente com a saída esperada para a simulação. Alguns desses arquivos são o traço da simulação e o pcaps, arquivos com os quais é possível analisar o comportamento da rede durante a simulação.

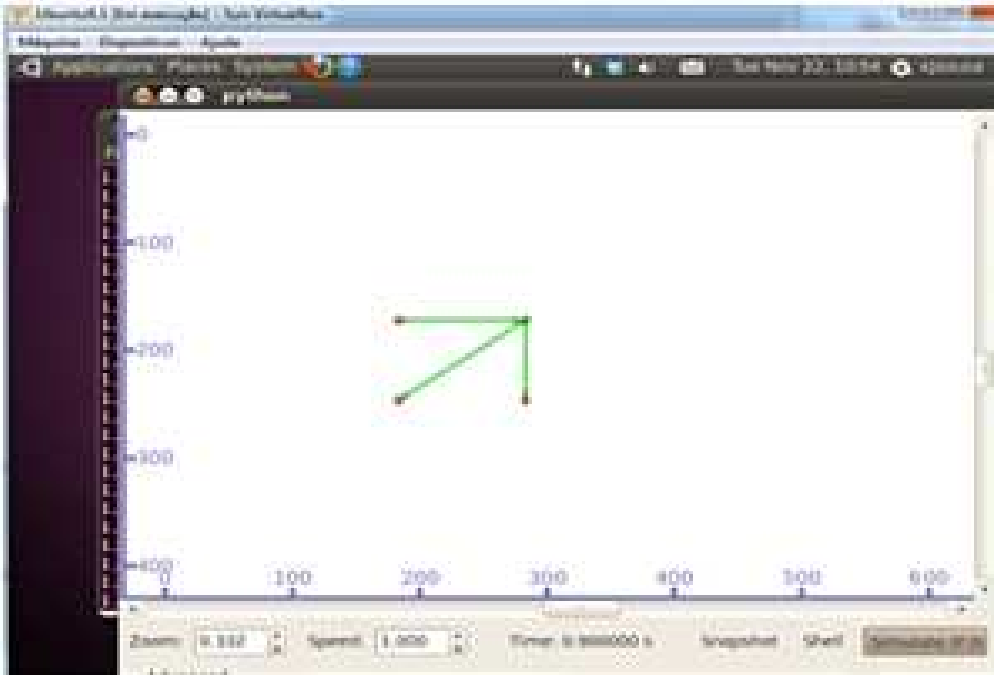
Figura 34 – Geração de arquivos durante a simulação NS-3



Fonte: Fontes et al (2014).

Baseado também nos arquivos de saída em uma simulação NS-3, é possível visualizar uma animação gráfica para o cenário correspondente. Em particular, a Figura 35 ilustra a animação de uma simulação para o cenário retratado na Figura 30 utilizando a ferramenta PyViz (FONTES et al., 2014).

Figura 35 – Visualização gráfica da simulação NS-3 com PyViz



Fonte: Fontes et al (2014).

Através da utilização da descrição NSDL *openFlow*, é possível validar e testar cenários em diferentes ferramentas existentes, por exemplo, com o uso do simulador de NS - 3.

6.4 DISCUSSÃO

A partir da realização dos cenários de estudo de caso, foi possível verificar a viabilidade da utilização da extensão de NSDL para a descrição de cenários *OpenFlow*. Os estudos de caso realizados visaram validar a utilização do perfil NSDL *OpenFlow* através da descrição de cenário simples e mais complexos. Nos cenários simples foi possível verificar a conectividade e funcionamento coordenado dos componentes de um experimento *OpenFlow*. Os cenários mais complexos permitiram a compreensão e descrição do tráfego experimental, determinando assim como esse tráfego seria isolado em uma rede operacional.

Também foi realizado um terceiro estudo de caso ilustrando a aplicação da descrição NSDL de um cenário *OpenFlow* no ciclo de vida experimental dessas redes (modelagem e configuração, geração da respectiva descrição NSDL, mapeamento e simulação NS-3). Esse estudo de caso nos permitiu validar a

utilidade da descrição NSDL como fator de integração de diferentes ferramentas para a gestão de redes. Essa contribuição permite claramente a otimização no projeto e implementação de cenários de rede.

De uma forma geral, a realização dos cenários de estudo de caso permitiram também a validação dos requisitos funcionais e não funcionais inicialmente identificados neste trabalho. A validação demonstrou que a proposta da extensão de NSDL para *OpenFlow* refletiu os requisitos pretendidos.

Dessa forma, os **Requisitos Funcionais** identificados para o perfil NSDL *OpenFlow* estão relacionados à descrição dos componentes de um cenário *OpenFlow* e dessa forma os cenários apresentados nos estudos de caso serviram para validar os seguintes requisitos:

- Computadores: são os elementos ativos de comunicação relacionados à origem e destino de dados, o que pode ser constatado através do primeiro estudo de caso, o cenário 1;
- Comutadores *OpenFlow*: é a extensão de um comutador (*switch*) que suporta a execução de um protocolo para programar a tabela de fluxo dos equipamentos, também validado pelo cenário 1;
- Ligações: são as conexões existentes entre os componentes de uma rede (por exemplo, ligações entre computadores e os comutadores *OpenFlow* e entre os comutadores *OpenFlow* entre si);
- Máquinas virtuais: são implementadas como uma extensão de um elemento computador, possuem as mesmas características, e seu propósito é a replicação de recursos (processamento, memória, armazenamento, etc.) de forma a viabilizar a operacionalização da rede apesar de não ser apresentado nos cenários existentes, também foi validado através do elemento computador por se tratar do mesmo componente;
- Tabelas de fluxo: são gerenciadas nos comutadores *OpenFlow* e o seu funcionamento contempla a manipulação de campos do cabeçalho de cada pacote (utilizados para definir um fluxo), implementação de ações (define como os pacotes devem ser processados e para onde devem ser encaminhados) e contadores (utilizados para estatísticas ou remoção de fluxos inativos). Cada fluxo de um cenário pode ser controlado, através

da escolha da rota e o tratamento que cada pacote deve receber, essas características foram validadas pelo primeiro cenário e pelos demais cenários;

- Controlador: é um elemento (normalmente um servidor) que tem como finalidade controlar os fluxos da rede *OpenFlow* (Plano de Controle), realizando a inserção e remoção de entradas em uma tabela de fluxo, como pode ser observado pelos estudos de caso apresentados. O controle da rede é realizado através de recursos de software presente em um dispositivo, que por sua vez, é independente aos protocolos implementados por fabricantes de equipamentos As regras e ações executadas em hardware estão sob a sua responsabilidade.

Em relação ao **requisitos não-funcionais**, a validação dos seguintes critérios pode ser observada:

- Utilidade: Os requisitos identificados permitiram a representação completa de cenários *OpenFlow*. A partir da realização do primeiro estudo de caso, foi possível verificar a usabilidade da linguagem proposta (*NSDL OpenFlow*) e a sua validade para a descrição de cenários genéricos;
- Facilidade de Uso: O estudo de caso apresentado serviu também para que os requisitos identificados pudessem ser compreendidos de forma inequívoca pelo grupo de trabalho;
- Desempenho: Através da modelagem dos cenários e da sua descrição NSDL é possível verificar a representatividade da linguagem através da validação e simulação realizada;
- Manutenabilidade: Os requisitos identificados permitiram a descrição de cenários de forma simples e sem apresentar ambigüidade nas descrições, o que facilitou a gestão dos cenários pela equipe do projeto;
- Escalabilidade: A descrição NSDL demonstrou ser facilmente escalável, a partir da criação dos cenários mais simples aos mais complexos, e guardando ainda, as propriedades acima citadas;
- Portabilidade: Através da realização de estudos de caso, foi possível aplicar a linguagem NSDL a diferentes ferramentas de gestão e sistemas operacionais (por exemplo, VND, MiniNet, NS-3), e;

- Confiança: Com os estudos de caso propostos foi possível observar a aplicação de todas as funcionalidades pretendidas para a especificação do perfil *NSDL OpenFlow* proposto. A partir da realização dos estudos de caso e da utilização de diferentes ferramentas, na geração e no mapeamento para a linguagem *NSDL*, foi possível constatar a eficácia da linguagem na descrição de cenários *OpenFLow* e na interoperabilidade entre as ferramentas.

A proposta apresentada para o desenvolvimento do perfil NSDL *openFlow,* demonstra, a partir de estudos de casos válidos, e do uso da linguagem, que a mesma pode ser validada. É possível observar também os aspectos inerentes ao NSDL de interoperabilidade, principalmente no estudo de caso 3, em que as três camadas da *Framework* NSDL foram aplicadas.

6.6 CONCLUSÃO

Com os objetivos apresentados, fica evidente que o estudo de caso serviu como apoio e validação para a proposta apresentada nos capítulos 3 e 6 e em todo o trabalho realizado nessa dissertação. O cenário 1 foi apresentado de forma a ilustrar como a descrição NSDL pode ser aplicada para a descrição dos componentes de um cenário OpenFlow e serviu de apoio ao cenário 2 o qual foi utilizado como objeto de estudo nesse capítulo.

Como resultado, foi obtido o código esperado a partir dos requisitos que foram inicialmente identificados. Tais requisitos foram coletados a partir dos estudos realizados sobre as *Redes Definidas por Software* aplicados ao protocolo *OpenFlow,* ao serem descritos através da linguagem *NSDL*. Através dos estudos de caso apresentados foi possível validar a completude e expressividade da linguagem proposta.

Neste trabalho ainda podemos destacar como principais vantagens o uso de uma linguagem nova e fácil de ser desenvolvida e implementada, a extensão *NSDL OpenFlow*. Além disso, é possível perceber na prática a flexibilidade existente da *Framework NSDL* no sentido de intermediar a interoperabilidade de ferramentas heterogêneas, desenvolvidas e que permitirão o projeto integrado de redes *OpenFlow*.

O próximo capítulo apresenta algumas conclusões e perspectivas para continuação deste trabalho.

7 CONCLUSÕES E PERSPECTIVAS FUTURAS

7.1 CONCLUSÕES

Este trabalho de mestrado foi realizado visando o estudo e a análise de um ambiente de redes definido por *software* (ROTHENBERG et al, 2011) utilizando o paradigma *OpenFlow (OPEN NETWORKING FOUNDATION*, 2011). O principal objetivo foi identificar e realizar o levantamento de requisitos sobre o paradigma *OpenFlow* afim de viabilizar a proposta de uma extensão da linguagem *NSDL (Network Scenarios Description Language)* (MARQUES et al., 2011) para o suporte à descrição de cenários *OpenFlow*.

A principal motivação para a utilização da *Framework NSDL no contexto deste trabalho é dada a* interoperabilidade promovida entre diferentes ferramentas de gestão de redes, ou seja, os cenários de rede descritos através de NSDL podem ser utilizados em qualquer ferramenta de autoria, gestão, simulação e virtualização de redes compatíveis com esta linguagem. Dessa forma, o ciclo de vida do projeto de redes pode ser otimizado (modelagem, simulação, análise, implementação e testes).

Para o desenvolvimento deste trabalho, a metodologia adotada contemplou o estudo dos principais aspectos relacionados, tais como das *Redes Definidas por Software*, do protocolo *OpenFlow* e da *Framework NSDL*. Outros aspectos contemplados na metodologia adotada estão a implementação de cenários para a experimentação *OpenFlow* em algumas plataformas, como OFELIA (*OPENFLOW IN EUROPE: LINKING INFRASTRUCTURE AND APPLICATIONS*, 2011). Esse estudo proporcionou inicialmente uma maior compreensão sobre o funcionamento do protocolo OpenFlow e de seus componentes, para o posterior levantamento de requisitos, a proposta de extensão de *NSDL* para a descrição de testes *OpenFlow*, e a validação da proposta realizada através de estudos de caso.

Durante do desenvolvimento deste trabalho, através dos requisitos funcionais foi possível definir como seria o desenvolvimento do perfil *NSDL OpenFlow*. Os requisitos não funcionais ajudaram a perceber as características qualitativas para o desenvolvimento do perfil desejado.

Dificuldades compre open analisese alc obje propor atenden requis

Podemos citar como principais contribuições deste trabalho:

- O levantamento de requisitos em um ambiente diferente do tradicional de *software*, por se tratar de um ambiente de rede;
- A identificação de requisitos (funcionais e não-funcionais) de redes *OpenFlow*;
- A proposta do perfil *NSDL OpenFlow*;
- A validação através de cenários de estudo de caso, e;
- A aplicação prática desses requisitos em ferramentas diversas integradas à *Framework NSDL* (por exemplo, a autoria de cenários de testes *OpenFlow* com *NSDL*).

Como já citado, é importante enfatizar a relevância da proposta da extensão NSDL OpenFlow como fator de integração entre diferentes ferramentas heterogêneas para o projeto e a gestão de redes. A partir da proposta de NSDL OpenFlow foi possível integrar a utilização da ferramenta de autoria *Visual Network Description (VND – SDN version)* (RAMON, 2013) e da simulação NS-3 de cenários OpenFlow descritos em NSDL (TALITA, 2014; TALITA et al, 2013). Ambos trabalhos apresentados como resultados de dissertação de mestrado no Programa de Pós-Graduação em Sistemas e Computação da UNIFACS.

Por fim, é também importante citar os esforços existentes por parte da comunidade acadêmica e científica que está trabalhando para o desenvolvimento do *OpenFlow*, de *NSDL* ou de outros paradigmas. Uma preocupação sempre constante é fazer com que tecnologias que foram pensadas para propósitos diferentes acabem se inter-relacionando e corroborando cada vez mais para o amadurecimento dos novos protocolos emergentes e das redes de comunicação.

7.2 PERSPECTIVAS FUTURAS

Como perspectivas futuras do trabalho apresentado nesta dissertação, podemos considerar:

- A utilização da linguagem proposta, *NSDL OpenFlow*, para proporcionar de forma mais ampla a interoperabilidade entre diferentes ferramentas heterogêneas de gestão de redes, e;

- A simulação e implementação de testes *OpenFlow* a partir da descrição de cenários mais complexos com o *NSDL OpenFlow*.

REFERÊNCIAS

ARAÚJO, J. **Virtualização automática de cenários de rede**. 101 p. Dissertação (Mestrado em Engenharia Informática) - Universidade da Madeira, Funchal, Portugal, 2011.

AZEVEDO, J. ***Virtual Network Description*****:** suporte à simulação de redes através da autoria gráfica de cenários virtuais. 69 p. 2010. Dissertação (Mestrado em Engenharia Informática) - Universidade da Madeira, Funchal – Portugal, 2010.

XML SCHEMA. Disponível em*:* < http://www.w3.org/standards/xml/schema >. Acesso em: 2 set. 2012.

COUTO, R. S. ***OpenFlow***. Rio de Janeiro. Disponível em: http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2010_2/rodrigo/index.html. Acesso em: set. 2012.

DE OLIVEIRA, L. A. **Virtualização de Redes GMPLS com Engenharia de Tráfego baseada em cenários NSDL**. 2014. Dissertação (Mestrado)- Universidade Salvador (UNIFACS), Salvador2014.

FERNANDES, N. C. t al. Virtual networks: isolation, performance and trends. In: ANNALS OF TELECOMMUNICATIONS, **Annals…** Rio de Janeiro, 2010.

FIBRE. **Future internet testbeds experimentations between Brasil and Europe**. Disponível em: <http://www.fibre-ict.eu/>. Acesso em: 2 jun. 2012.

FONTES, R. **Autoria de cenários openflow utilizando visual network description (versão SDN)**. 2013. Dissertação (Mestrado Acadêmico em Sistemas e Computação) - UNIFACS – Universidade Salvador, Salvador, 2013.

FONTES, R.; PINHEIRO, T.; 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. **Proceedings…** 2013.

GENI. **Global environment for network innovations**. Disponível em: http://www.geni.net/. Acesso em: 10 set. 2012.

MARQUES, E. M. D; SOUSA, J. J. F; SAMPAIO, P. N. M. Modeling and Simulation of DiffServ Scenarios with the NSDL Framework. In: International ACM SIGCOMM Conference on Network and Service Management - CNSM 2011, 7., 2011, Paris. **Proceedings…** Paris, France, October, 2011. (Short-Paper)

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

MARQUES, E.M.D. **Interoperabilidade e Otimização da Gestão de Redes com a Framework NSDL**. 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. et al. OpenFlow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review**, v. 38, n. 2, p. 69–74, 2008.

MENDES, D. R. **Redes de computadores:** teoria e prática. Curitiba: Novatec, 2005.

NS-3. Disponível em: <http://www.nsnam.org/>. Acesso em: 2 ago. 2012.

OFLOPS. Disponível em: <http://www.*OpenFlow*.org/wk/index.php/Oflops>. Acesso em: 2 ago. 2012.

OPENFLOW in Europe: Linking Infrastructure and Applications. Disponível em: <http://www.fp7-ofelia.eu/>. Acesso em: 10 maio 2012.

OPEN Networking Foundation. Disponível em: <https://www.opennetworking.org/>. Acesso em: 10 maio 2012.

OPENFLOW Wireshark Dissector. Disponível em: <http://www.*OpenFlow*.org/wk/index.php/*OpenFlow*_Wireshark_Dissector>. Acesso em: 4 ago. 2012.

PINHEIRO, T. R. **Simulação NS-3 de Redes OpenFlow baseada em descrições NSDL**. 2014. Dissertação (Mestrado)- Programa de Pós-Graduação em Sistemas e Computação. Universidade Salvador (UNIFACS). Salvador, Brasil, 2014.

RISDIANTO, A. C. ; MULYANA, E. Implementation and analysis of control and forwarding plane for SDN. In: TELECOMMUNICATION SYSTEMS, SERVICES, AND APPLICATIONS (TSSA). INTERNATIONAL CONFERENCE ON, 7., 2012. **Proceedings…** 2012.

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.

SOARES, A. L. **Introdução, identificação e análise em engenharia de requisitos**. Porto Alegre: SBC, 2005.

SOUSA, J. **Autoria e Simulação de Cenários de Redes em NS-3**. 101 p. 2011. Dissertação (Mestrado em Engenharia Informática) - Universidade da Madeira, Funchal – Portugal, 2011.

XML Technology. Disponível em: < http://www.w3.org/standards/xml/>. Acesso em: 2 set. 2012.

ANEXO A – Schema XML completo para o perfil NSDL OpenFlow

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <xs:attribute name="id" type="nsdlID"/>
   <xs:attribute name="object" type="baseObjects"/> <!-- 15/Mar/2011 Update -->
   <xs:element name="name" type="xs:string"/>
   <!-- should be added a restriction to 255 characters -->
   <xs:element name="notes" type="xs:string"/>
   <!-- should be added a restriction to 255 characters -->
   <xs:element name="active" type="xs:boolean" default="true"/>

   <xs:simpleType name="nsdlID">
      <xs:annotation>
         <xs:documentation>Main type to identify all objects</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:ID">
         <xs:minLength value="2" />
         <xs:maxLength value="30" />
         <xs:pattern value="[a-zA-Z][a-zA-Z0-9_]+"></xs:pattern>
      </xs:restriction>
   </xs:simpleType>

   <xs:simpleType name="baseObjects">
      <xs:annotation>
         <xs:documentation>List of basic available objects in NSDL</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
         <xs:enumeration value="node"/>
         <xs:enumeration value="link"/>
         <xs:enumeration value="domain"/>
         <xs:enumeration value="application"/>
         <xs:enumeration value="protocol"/>
         <xs:enumeration value="interface"/>
      </xs:restriction>
   </xs:simpleType>


   <xs:simpleType name="linkType">
      <xs:annotation>
         <xs:documentation>The link could be simplex (source to destination) and duplex
(communication both ways).</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
         <xs:enumeration value="simplex"/>
         <xs:enumeration value="duplex"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <!-- Files included -->
   <xs:include schemaLocation="nsdlOpenFlowObjects.xsd"/>

   <!-- Elements declaration -->
   <xs:element name="ethernet" type="type_ethernet"/>
   <xs:element name="fastethernet" type="type_fastethernet"/>
   <xs:element name="adsl" type="type_adsl"/>

   <!-- Complex Types definition -->
   <xs:complexType name="type_ethernet">
      <xs:complexContent>
         <xs:extension base="type_link">
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="type_fastethernet">
      <xs:complexContent>
         <xs:extension base="type_link">
            <xs:sequence> </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="type_adsl">
      <xs:complexContent>
         <xs:extension base="type_link">
            <xs:sequence> </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <!-- Files included -->
   <xs:include schemaLocation="nsdlOpenFlowObjects.xsd"/>
   <xs:include schemaLocation="nsdlOpenFlowDatatypes.xsd"/>

   <!-- Elements declaration -->
   <xs:element name="computer" type="type_computer"/>
   <xs:element name="switch" type="type_switchOpenFlow"/>
   <xs:element name="virtualMachine" type="type_virtualMachine"/>

   <!-- Complex Types definition -->
   <xs:complexType name="type_computer">
      <xs:complexContent>
         <xs:extension base="type_node">
```

```
            <xs:sequence minOccurs="0">
               <xs:element name="role" minOccurs="0"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="type_virtualMachine">
      <xs:complexContent>
         <xs:extension base="type_node">
            <xs:sequence minOccurs="0">
               <xs:element name="role" minOccurs="0"/>
               <xs:element name="memory"/>
               <xs:element name="discImage"/>
               <xs:element name="hdSetupType"/>
               <xs:element name="virtualizationSetupType"/>
               <xs:element name="state"/>
               <xs:element name="operatingSystem"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="type_switchOpenFlow">
      <xs:complexContent>
         <xs:extension base="type_node">
            <xs:sequence minOccurs="0">
               <xs:element minOccurs="0" name="interfaces"/>
               <xs:element name="datapathId"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">


   <xs:include schemaLocation="nsdlOpenFlowDatatypes.xsd"/>
   <xs:include schemaLocation="nsdlOpenFlowNodes.xsd"/>
   <xs:include schemaLocation="nsdlOpenFlowLinks.xsd"/>
   <xs:include schemaLocation="nsdlOpenFlowProtocols.xsd"/>


   <!-- Declaration of objects for the 'objects' and 'templates' -->
   <xs:element name="node" type="type_node"/>
   <xs:element name="link" type="type_link"/>
   <xs:element name="protocol" type="type_protocol"/>
```

```
  <!-- Definition of the general object -->
  <xs:element name="metadata" type="type_metadata"/>

  <!-- This is something that all objects in nsdl should have -->
  <xs:complexType name="type_objectOLD">
    <xs:sequence minOccurs="0">
      <xs:element ref="active" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="name" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="notes" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="metadata" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute ref="id" use="required"/>
    <xs:attribute ref="object"/>

    <!-- use="required" TO DO in the future (?) -->
    <xs:attribute name="template">
      <xs:annotation>
        <xs:documentation>To use only at objects, not at template</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:restriction base="xs:string"/>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="type_nsdlobject">
    <xs:sequence minOccurs="0">
      <xs:element ref="active" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="name" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="notes" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="metadata" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute ref="id" use="required"/>
  </xs:complexType>

  <!-- Types for the ... -->
  <xs:complexType name="type_object">
    <xs:complexContent>
      <xs:extension base="type_nsdlobject">
        <xs:attribute ref="object"/>
        <!-- use="required" TO DO in the future (?) -->
        <xs:attribute name="template">
          <xs:annotation>
            <xs:documentation>To use only at objects, not at
template</xs:documentation>
          </xs:annotation>
          <xs:simpleType>
            <xs:restriction base="xs:string"/>
          </xs:simpleType>
```

```
            </xs:attribute>
          </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="type_metadata">
        <xs:sequence minOccurs="0">
            <xs:element name="title" type="xs:string"/>
            <xs:element minOccurs="0" name="description" type="xs:string"/>
            <xs:element name="author" maxOccurs="1" type="xs:string"/>
            <xs:element minOccurs="0" name="created" type="xs:dateTime"/>
            <xs:element minOccurs="0" name="copyright"/>
        </xs:sequence>
    </xs:complexType>


    <!-- Types for the views -->
    <xs:element name="set" type="type_set"/>

    <xs:complexType name="type_set">
        <xs:complexContent>
            <xs:extension base="type_nsdlobject">
                <xs:sequence>
                    <xs:element minOccurs="1" name="listobjects">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="objectid" maxOccurs="unbounded"
type="xs:IDREF"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Types for the objects and templates -->
    <!-- Definition of each type -->
    <xs:complexType name="type_node">
        <xs:complexContent>
            <xs:extension base="type_object">
                <xs:choice maxOccurs="unbounded" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>List of possible applications</xs:documentation>
                        <xs:documentation>List of possible protocols</xs:documentation>
                        <xs:documentation>List of possible interfaces</xs:documentation>
                    </xs:annotation>
                    <xs:element ref="application" minOccurs="0"/>
                    <xs:element ref="http" minOccurs="0"/>
                    <xs:element ref="ftp" minOccurs="0"/>
```

```
                <xs:element ref="telnet" minOccurs="0"/>
                <xs:element ref="cbr" minOccurs="0"/>
                <xs:element ref="exponential" minOccurs="0"/>
                <xs:element ref="pareto" minOccurs="0"/>
                <xs:element ref="trace" minOccurs="0"/>

                <xs:element ref="protocol" minOccurs="0"/>
                <xs:element ref="tcp" minOccurs="0"/>
                <xs:element ref="udp" minOccurs="0"/>
                <xs:element ref="ipv4" minOccurs="0"/>
                <xs:element ref="rsvp" minOccurs="0"/>
                <xs:element ref="rtp" minOccurs="0"/>
                <xs:element ref="rip" minOccurs="0"/>

                <xs:element ref="interface" minOccurs="0"/>
                <xs:element ref="lan802.3" minOccurs="0"/>
                <xs:element ref="wlan802.11" minOccurs="0"/>
                <xs:element ref="wlan802.16" minOccurs="0"/>
            </xs:choice>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="type_link">
    <xs:complexContent>
        <xs:extension base="type_object">
            <xs:sequence minOccurs="0">
                <xs:element name="connection" minOccurs="0">
                    <xs:complexType>
                        <xs:attribute name="source" type="xs:IDREF"/>
                        <xs:attribute name="destination" type="xs:IDREF"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="type" type="linkType" minOccurs="0"/>
                <xs:element name="bandwidth" minOccurs="0"/>
                <xs:element name="delay" minOccurs="0"/>
                <xs:element name="loss" minOccurs="0"/>
                <xs:element minOccurs="0" name="queue-limit">
                    <xs:annotation>
                        <xs:documentation>Queue limit for NS2</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element name="queue-type" minOccurs="0" type="queueType"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <!-- Files included -->
   <xs:include schemaLocation="nsdlOpenFlowObjects.xsd"/>

   <!-- Elements declaration -->
   <!-- Transpor Protocols -->
   <xs:element name="tcp" type="type_tcp"/>
   <xs:element name="udp" type="type_udp"/>
   <xs:element name="rtp" type="type_rtp"/>
   <!-- Network Protocols -->
   <xs:element name="ipv4" type="type_ipv4"/>
   <xs:element name="rip" type="type_rip"/>
   <!-- Signaling Protocols -->
   <xs:element name="rsvp" type="type_rsvp"/>
   <xs:element name="ldp" type="type_ldp"/>

   <!-- Complex Types definition -->
   <xs:complexType name="type_tcp">
      <xs:complexContent>
         <xs:extension base="type_protocol">
            <xs:sequence minOccurs="0">
               <xs:element minOccurs="0" name="version" type="tcpType"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="type_udp">
      <xs:complexContent>
         <xs:extension base="type_protocol">
            <xs:sequence minOccurs="0">
               <xs:element minOccurs="0" name="version" type="udpType"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="type_rtp">
      <xs:complexContent>
         <xs:extension base="type_protocol">
            <xs:sequence minOccurs="0">
               <xs:element minOccurs="0" name="version" type="rtpType"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="type_ipv4">
      <xs:complexContent>
         <xs:extension base="type_protocol">
```

```
        <xs:sequence minOccurs="0">
          <xs:element name="address" minOccurs="0"/>
          <xs:element name="mask" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="type_rsvp">
    <xs:complexContent>
      <xs:extension base="type_protocol">
        <xs:sequence minOccurs="0">
          <xs:element minOccurs="0" name="tspec">
            <xs:complexType>
              <xs:sequence minOccurs="0">
                <xs:element minOccurs="0" name="PRate"/>
                <xs:element minOccurs="0" name="MinPolUnit"/>
                <xs:element minOccurs="0" name="MaxPackSize"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element minOccurs="0" name="rspec"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="type_rip">
    <xs:complexContent>
      <xs:extension base="type_protocol">
        <xs:sequence minOccurs="0">
          <xs:element name="version" minOccurs="0"/>
          <xs:element name="admdst" minOccurs="0"/>
          <xs:element name="updint" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="type_ldp">
    <xs:complexContent>
      <xs:extension base="type_protocol">
        <xs:sequence minOccurs="0">
          <xs:element name="label.creation" minOccurs="0"/>
          <!-- Control or data driven -->
          <xs:element name="routing" minOccurs="0">
            <xs:complexType>
              <xs:choice>
                <xs:element name="explicit">
                  <xs:complexType>
```

```
                        <xs:sequence>
                           <xs:element maxOccurs="unbounded" name="lsrouter">
                              <xs:complexType>
                                 <xs:attribute name="order"/>
                              </xs:complexType>
                           </xs:element>
                        </xs:sequence>
                        <xs:attribute name="lspid"/>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="constrained"/>
               </xs:choice>
            </xs:complexType>
         </xs:element>
         <xs:element name="updint" minOccurs="0"/>
      </xs:sequence>
   </xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <!-- Elements declaration -->
   <xs:element name="openFlow" type="type_openflow"/>
   <xs:element name="slice" type="type_slice"/>
   <xs:element name="flowSpace" type="type_flowTable"/>
   <xs:element name="controller" type="type_controller"/>

   <!-- Complex Types definition -->
   <xs:complexType name="type_openFlow">
      <xs:complexContent>
         <xs:sequence minOccurs="0">
               <xs:element name="slice" minOccurs="0" type="type_slice"/>
               <xs:element name="switchPort" minOccurs="0" type="type_switchPort"/>
               <xs:element name="flowSpace" minOccurs="0" type="type_flowTable"/>
               <xs:element name="controller" minOccurs="0" type="type_controller"/>
         </xs:sequence>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="type_slice">
      <xs:sequence minOccurs="0">
         <xs:element name="name" type="xs:string"/>
         <xs:element name="description" type="xs:string"/>
         <xs:element name="aggregate"  />
      </xs:sequence>
   </xs:complexType>
```

```
    <xs:complexType name="type_flowTable">
       <xs:sequence minOccurs="0">
          <xs:element name="macSource" />
          <xs:element name="macDestination" />
          <xs:element name="ethernetType"  />
          <xs:element name="vlan" />
          <xs:element name="ipSource" />
          <xs:element name="ipDestination"  />
          <xs:element name="ipProtocol" />
          <xs:element name="tcpUdpSource" />
          <xs:element name="tcpUdpDestination"  />
          <xs:element name="switchPort"  />
       </xs:sequence>
    </xs:complexType>

    <xs:complexType name="controller">
       <xs:sequence minOccurs="0">
          <xs:element name="url" type="xs:string"/>
          <xs:element name="password" type="xs:string"/>
          <xs:element name="switchOpenFlow"  />
          <xs:element name="flowTable"  />
       </xs:sequence>
    </xs:complexType>
</xs:schema>
```

ANEXO B – Descrição NSDL cenário 1

```
<nsdl>
 <network>
  <objects>
   <link id="ethernet_1">
    <delay/>
    <loss/>
    <maxqueue/>
    <connection source="switchOpenflow_1" destination="controllerOpenflow_1"/>
   </link>
   <link id="ethernet_2">
    <delay/>
    <loss/>
    <maxqueue/>
    <connection source="computer_1" destination="switchOpenflow_1"/>
   </link>
   <link id="ethernet_3">
    <delay/>
    <loss/>
    <maxqueue/>
    <connection source="computer_2" destination="switchOpenflow_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>
   <controllerOpenflow id="controllerOpenflow_1">
    <controllerIPAddress>127.0.0.1</controllerIPAddress>
    <controllerPort>6633</controllerPort>
    <specialController>none</specialController>
    <openflowController>null</openflowController>
   </controllerOpenflow>
   <switchOpenflow id="switchOpenflow_1">
    <switchMacAddress>00:00:00:00:00:03</switchMacAddress>
    <listeningPort>6634</listeningPort>
    <switch>ovsk</switch>
    <notes/>
   </switchOpenflow>
   <computer id="computer_2">
    <computerIPAddress>10.0.0.4</computerIPAddress>
    <mask>255.0.0.0</mask>
    <computerMacAddress>00:00:00:00:00:04</computerMacAddress>
    <notes/>
   </computer>
  </objects>
  <views/>
 </network>
```

```
<scenarios>
 <visualizations>
  <visualization id="vis01">
   <description>
    <object id="computer_1">
     <x.position>332</x.position>
     <y.position>463</y.position>
     <z.position>0</z.position>
     <color/>
    </object>
    <object id="controllerOpenflow_1">
     <x.position>530</x.position>
     <y.position>112</y.position>
     <z.position>0</z.position>
     <color/>
    </object>
    <object id="switchOpenflow_1">
     <x.position>518</x.position>
     <y.position>296</y.position>
     <z.position>0</z.position>
     <color/>
    </object>
    <object id="computer_2">
     <x.position>770</x.position>
     <y.position>468</y.position>
     <z.position>0</z.position>
     <color/>
    </object>
   </description>
  </visualization>
 </visualizations>
 <openflow>
 <flowTable1 id="controllerOpenflow_1">
   <flowName>flow1</flowName>
   <priority/>
   <macSwitch>00:00:00:00:00:03</macSwitch>
   <macSource>00:00:00:00:00:01</macSource>
   <macDestination>00:00:00:00:00:02</macDestination>
   <ingressPort/>
   <vlanID/>
   <vlanPriority/>
   <ethtype/>
   <tos/>
   <protocol/>
   <ipSource>10.0.0.1</ipSource>
   <ipDestination>10.0.0.2</ipDestination>
   <sourcePort/>
   <destinationPort/>
   <setOutput/>
   <setMACSource>00:00:00:00:00:01</setMACSource>
```

```
        <setMACDestination>00:00:00:00:00:02</setMACDestination>
        <setEnqueue/>
        <setVlanID/>
        <setVLANPriority/>
        <setStripVlan/>
        <setTOS/>
        <setIPSource>10.0.0.1</setIPSource>
        <setIPDestination>10.0.0.2</setIPDestination>
        <setSourcePort/>
        <setDestinationPort/>
        <notes/>
      </flowTable1>
    </openflow>
  </scenarios>
</nsdl>
```

ANEXO C – Descrição NSDL cenário 2

```
<nsdl>
  <network>
    <objects>
      <link id="ethernet">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="switchOpenflow_Brasil"
destination="controllerOpenflow_Brasil"/>
      </link>
      <link id="ethernet_21">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="switchOpenflow_14" destination="computer_15"/>
      </link>
      <link id="ethernet_20">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="switchOpenflow_14" destination="computer_4"/>
      </link>
      <link id="ethernet_19">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="www_16" destination="switchOpenflow_14"/>
      </link>
      <link id="ethernet_18">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="switchOpenflow_5" destination="www_16"/>
      </link>
      <link id="ethernet_13">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="controllerOpenflow_7" destination="switchOpenflow_5"/>
      </link>
      <link id="ethernet_12">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="computer_11" destination="switchOpenflow_5"/>
      </link>
      <link id="ethernet_9">
        <delay/>
        <loss/>
```

```
  <maxqueue/>
  <connection source="computer_1" destination="switchOpenflow_5"/>
</link>
<computer id="servidor_Brasil">
  <computerIPAddress>10.0.0.1</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
  <notes/>
</computer>
<computer id="laptop_Brasil">
  <computerIPAddress>10.0.0.4</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:04</computerMacAddress>
  <notes/>
</computer>
<computer id="servidor_Australia">
  <computerIPAddress>10.0.0.1</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:01</computerMacAddress>
  <notes/>
</computer>
<computer id="laptopr_Australia">
  <computerIPAddress>10.0.0.4</computerIPAddress>
  <mask>255.0.0.0</mask>
  <computerMacAddress>00:00:00:00:00:04</computerMacAddress>
  <notes/>
</computer>
<switchOpenflow id="switchOpenflow_Brasil">
  <switchMacAddress>00:00:00:00:00:05</switchMacAddress>
  <listeningPort>6635</listeningPort>
  <switch>ovsk</switch>
  <notes/>
</switchOpenflow>
<controllerOpenflow id="controllerOpenflow_Brasil">
  <controllerIPAddress>127.0.0.1</controllerIPAddress>
  <controllerPort>6633</controllerPort>
  <specialController>none</specialController>
  <openflowController>nox</openflowController>
</controllerOpenflow>
<switchOpenflow id="switchOpenflow_Australia">
  <switchMacAddress>00:00:00:00:00:14</switchMacAddress>
  <listeningPort>6637</listeningPort>
  <switch>ovsk</switch>
  <notes/>
</switchOpenflow>
<controllerOpenflow id="controllerOpenflow_Australia">
  <controllerIPAddress>127.0.0.1</controllerIPAddress>
  <controllerPort>6633</controllerPort>
  <specialController>none</specialController>
  <openflowController>undefined</openflowController>
```

```
    </controllerOpenflow>
  </objects>
  <views/>
</network>
<scenarios>
  <visualizations>
    <visualization id="vis01">
      <description>
        <object id="servidor_Brasil">
          <x.position>77</x.position>
          <y.position>220</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="laptop_Brasil">
          <x.position>748</x.position>
          <y.position>96</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="switchOpenflow_Brasil">
          <x.position>263</x.position>
          <y.position>215</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="controllerOpenflow_Brasil">
          <x.position>158</x.position>
          <y.position>336</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="servidor_Australia">
          <x.position>184</x.position>
          <y.position>87</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="switchOpenflow_Australia">
          <x.position>556</x.position>
          <y.position>207</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
        <object id="laptopr_Australia">
          <x.position>857</x.position>
          <y.position>211</y.position>
          <z.position>0</z.position>
          <color/>
        </object>
```

```
      <object id="www">
        <x.position>512</x.position>
        <y.position>174</y.position>
        <z.position>0</z.position>
        <color/>
      </object>
      <object id="controllerOpenflow_Australia">
        <x.position>705</x.position>
        <y.position>332</y.position>
        <z.position>0</z.position>
        <color/>
      </object>
    </description>
  </visualization>
</visualizations>
 <openflow>
  <flowTable1 id="controllerOpenflow_Brasil">
    <flowName>flow1</flowName>
    <priority/>
    <macSwitch>00:00:00:00:00:05</macSwitch>
    <macSource>00:00:00:00:00:11</macSource>
    <macDestination>00:00:00:00:00:04</macDestination>
    <ingressPort/>
    <vlanID/>
    <vlanPriority/>
    <ethtype/>
    <tos/>
    <protocol/>
    <ipSource>10.0.0.11</ipSource>
    <ipDestination>10.0.0.4</ipDestination>
    <sourcePort/>
    <destinationPort/>
    <setOutput/>
    <setMACSource>00:00:00:00:00:11</setMACSource>
    <setMACDestination>00:00:00:00:00:04</setMACDestination>
    <setEnqueue/>
    <setVlanID/>
    <setVLANPriority/>
    <setStripVlan>null</setStripVlan>
    <setTOS/>
    <setIPSource>10.0.0.11</setIPSource>
    <setIPDestination>10.0.0.4</setIPDestination>
    <setSourcePort/>
    <setDestinationPort/>
    <notes/>
  </flowTable1>
  <flowTable2 id="controllerOpenflow_Australia">
    <flowName>flow2</flowName>
    <priority/>
    <macSwitch>00:00:00:00:00:14</macSwitch>
```

```
        <macSource>00:00:00:00:00:04</macSource>
        <macDestination>00:00:00:00:00:11</macDestination>
        <ingressPort/>
        <vlanID/>
        <vlanPriority/>
        <ethtype/>
        <tos/>
        <protocol/>
        <ipSource>10.0.0.4</ipSource>
        <ipDestination>10.0.0.11</ipDestination>
        <sourcePort/>
        <destinationPort/>
        <setOutput/>
        <setMACSource>00:00:00:00:00:04</setMACSource>
        <setMACDestination>00:00:00:00:00:11</setMACDestination>
        <setEnqueue/>
        <setVlanID/>
        <setVLANPriority/>
        <setStripVlan/>
        <setTOS/>
        <setIPSource>10.0.0.4</setIPSource>
        <setIPDestination>10.0.0.11</setIPDestination>
        <setSourcePort/>
        <setDestinationPort/>
        <notes/>
      </flowTable2>
    </openflow>
  </scenarios>
</nsdl>
```

ANEXO D – Descrição NSDL cenário 3

```
<nsdl>
  <network>
    <objects>
      <link id="ethernet_1">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="switchOpenflow_1" destination="controllerOpenflow_1"/>
      </link>
      <link id="ethernet_2">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="computer_2" destination="switchOpenflow_1"/>
      </link>
      <link id="ethernet_3">
        <delay/>
        <loss/>
        <maxqueue/>
        <connection source="computer_1" destination="switchOpenflow_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>
      <computer id="computer_2">
        <computerIPAddress>10.0.0.2</computerIPAddress>
        <mask>255.0.0.0</mask>
        <computerMacAddress>00:00:00:00:00:02</computerMacAddress>
        <notes/>
      </computer>
      <switchOpenflow id="switchOpenflow_1">
        <switchMacAddress>00:00:00:00:00:03</switchMacAddress>
        <listeningPort>6634</listeningPort>
        <switch>ovsk</switch>
        <notes/>
      </switchOpenflow>
      <controllerOpenflow id="controllerOpenflow_1">
        <controllerIPAddress>127.0.0.1</controllerIPAddress>
        <controllerPort>6633</controllerPort>
        <specialController>none</specialController>
        <openflowController>floodlight</openflowController>
      </controllerOpenflow>
    </objects>
    <views/>
```

```
</network>
<scenarios>
 <visualizations>
  <visualization id="vis01">
   <description>
    <object id="computer_1">
     <x.position>408</x.position>
     <y.position>392</y.position>
     <z.position>0</z.position>
     <color/>
    </object>
    <object id="computer_2">
     <x.position>787</x.position>
     <y.position>392</y.position>
     <z.position>0</z.position>
     <color/>
    </object>
    <object id="switchOpenflow_1">
     <x.position>563</x.position>
     <y.position>283</y.position>
     <z.position>0</z.position>
     <color/>
    </object>
    <object id="controllerOpenflow_1">
     <x.position>854</x.position>
     <y.position>163</y.position>
     <z.position>0</z.position>
     <color/>
    </object>
   </description>
  </visualization>
 </visualizations>
 <openflow>
  <flowTable1 id="controllerOpenflow_1">
   <flowName>Flow Cen3</flowName>
   <priority/>
   <macSwitch>00:00:00:00:00:03</macSwitch>
   <macSource>00:00:00:00:00:01</macSource>
   <macDestination>00:00:00:00:00:02</macDestination>
   <ingressPort/>
   <vlanID>none</vlanID>
   <vlanPriority/>
   <ethtype/>
   <tos/>
   <protocol/>
   <ipSource>10.0.0.1</ipSource>
   <ipDestination>10.0.0.2</ipDestination>
   <sourcePort>3251</sourcePort>
   <destinationPort>3251</destinationPort>
   <setOutput/>
```

```
      <setMACSource>00:00:00:00:00:01</setMACSource>
      <setMACDestination>00:00:00:00:00:02</setMACDestination>
      <setEnqueue/>
      <setVlanID>none</setVlanID>
      <setVLANPriority/>
      <setStripVlan>no</setStripVlan>
      <setTOS/>
      <setIPSource>10.0.0.1</setIPSource>
      <setIPDestination>10.0.0.2</setIPDestination>
      <setSourcePort>3251</setSourcePort>
      <setDestinationPort>3251</setDestinationPort>
      <notes/>
    </flowTable1>
    <qosTable0 id="controllerOpenflow_1">
      <qosName>Qos1</qosName>
      <sourceIP>10.0.0.1</sourceIP>
      <destinationIP>10.0.0.2</destinationIP>
      <ethType>0x0800</ethType>
      <protocol>http</protocol>
      <queue>2</queue>
    </qosTable0>
    <queueConfiguration0 id="controllerOpenflow_1">
      <scenarioMaxRate>5</scenarioMaxRate>
      <scenarioMinRate>5</scenarioMinRate>
      <queueName>Queue 0</queueName>
      <linkmaxRate/>
      <linkminRate/>
    </queueConfiguration0>
  </openflow>
 </scenarios>
</nsdl>
```