UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HUGO SAWADA TODA

OP3NCONTROL: *FRAMEWORK* PARA GERÊNCIA E CONTROLE DE REDES *OPENFLOW*

BELÉM-PA

Março / 2013

HUGO SAWADA TODA

OP3NCONTROL: *FRAMEWORK* PARA GERÊNCIA E CONTROLE DE REDES *OPENFLOW*

Dissertação submetida à banca julgadora na Universidade Federal do Pará como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Antônio J. G. Abelém

BELÉM-PA

Março / 2013

Dados Internacionais de Catalogação-na-Publicação (CIP)
Sistema de Bibliotecas da UFPA

Toda, Hugo Sawada, 1982-
Op3ncontrol: framework para gerência e controle de redes openflow / Hugo Sawada Toda. - 2013.

Orientador: Antônio Jorge Gomes Abelém.
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Ciências Exatas e Naturais, Programa de Pós-Graduação em Ciência da Computação, Belém, 2013.

1. Redes de computadotes. 2. Internet. 3. Redes de computadores-Gerência. I. Título.
CDD 22. ed. 004.6

HUGO SAWADA TODA

OP3NCONTROL: *FRAMEWORK* PARA GERÊNCIA E CONTROLE DE REDES *OPENFLOW*

Dissertação submetida à banca julgadora na Universidade Federal do Pará como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Prof. Dr. Antônio J. G. Abelém
Universidade Federal do Pará
Orientador

Prof. Dr. Marcos Rogério Salvador
CPqD

Prof. Dr. Agostinho Luiz da Silva Castro
Universidade Federal do Pará

Dedico este trabalho aos meus pais, Helena e Roberto. Meus heróis da vida. Dedico também ao meu avô, Teruo, um grande homem, sábio, que me ensinou que a vida deve ser vivida intensamente, do lado que pessoas que amamos e nos faz sentir bem.

Agradecimentos

A Deus por tudo.

À minha mãe Helena e meu pai Roberto, meus exemplos de vida. Agradeço por terem dedicado seus esforços para poderem incentivar meu desenvolvimento. Obrigado pelos conselhos, pela força quando estava fraco, pelas palavras de conforto quando mais precisei. Mas, sobretudo, pela dedicação, carinho e amor incalculáveis. Muito obrigado.

À minha noiva Camilla, pelo carinho, companheirismo, cumplicidade e amor. Agradeço, também, pela paciência em me aturar nos momentos de estresse e, por compreender, nos momentos de ausência, a importância da realização deste trabalho.

Às minhas irmãs Danielle e Karla, companheiras, amigas, minhas "metades". Me sinto forte e confiante apenas em saber que as tenho ao meu lado.

Ao meu avô Teruo e vovó Fumika, que estão sempre presentes, me dando força, carinho e compreendendo meus períodos de ausência, independente de onde quer que estejam.

Ao professor Abelém, pela confiança depositada e por estar sempre disposto a ajudar com sua paciente, objetiva e competente orientação.

Ao Fernando Farias, pela apoio, orientação, cobranças e amizade durante todos os momentos de meu mestrado.

Aos colegas do Grupo de Estudos em Redes de Computadores e Comunicação Multimídia (GERCOM), pelo conselhos, sugestões, conhecimentos proporcionados, além do companheirismo, descontrações e cafés, representados aqui pelos membros do grupo de IF mais antigos, Airton, Salvatti, Verônica, Fábio e Raphael

Ao Serpro, por incentivar meu crescimento profissional através do Programa de educação Pós-Graduada da UniSerpro.

A todos que contribuíram de alguma forma com a realização deste trabalho.

Resumo

Resumo da Dissertação apresentada à UFPA como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Op3nControl: *Framework* para Gerência e Controle de Redes *OpenFlow*

Orientador: Prof. Dr. Antônio J. G. Abelém
Palavras-chave: *OpenFlow*, Redes Definidas por Software, *framework*, Gerência de Redes, Controle de Recursos.

Redes Definidas por Software e *OpenFlow* apontam como alternativas para a potencialização dos ambientes experimentais (*testbeds*), estes que são necessários para a validação de resultados de pesquisas de novos protocolos e arquitetura de redes. Devido suas características de flexibilidade e robustez, padrões SDN/*OpenFlow* também ganham atenção do mercado, em especial em ações relacionadas ao gerenciamento e controle de redes. Este trabalho apresenta o *Op3nControl*, uma proposta de *framework* para controle e gerência de redes *OpenFlow*, entre suas principais funcionalidades estão: (i) coleta de estatísticas dos elementos de rede em tempo real; (ii) módulo para desenvolvimento de algoritmos de encaminhamento; (iii) alocação de recursos, de forma dinâmica, através da criação de circuito virtual sobre a infraestrutura de redes; (iv) tratamento de QoS; e (v) integração de equipamentos de rede legados ao cenário *OpenFlow*. Com objetivo de avaliar a proposta, é realizada a implementação do Op3nControl, assim como é disponibilizado um *testbed* real, composto por equipamentos habilitados com *OpenFlow*. A aplicação implementada foi submetida a diversos cenários de experimentação a fim de validar seus aspectos funcionais e de desempenho. Por fim, são apresentados os resultados que atestam baixos tempos de resposta das ações executadas pelo Op3nControl quando comparado a solução com características similares, os quais embasam sua disponibilização em ambientes de redes mais complexos, podendo, desta forma contribuir, de modo mais efetivo, tanto para fins de pesquisa acadêmica quanto para o mercado.

Abstract

Abstract of Dissertation presented to UFPA as a partial fulfillment of the requirements for the degree of Master in Computer Science.

Op3nControl: *Framework* para Gerência e Controle de Redes *OpenFlow*

Advisor: Prof. Dr. Antônio J. G. Abelém
Key words: *OpenFlow*, Software-Defined Networking, *framework*, Network Management, Resource Control.

Software-Defined Networking and OpenFlow arise as interesting alternatives to enhance the researches on experimental network environments (testbeds). These testbeds are relevant to validate innovate network proposals, such as new protocols or network architectures. Deploying a open protocol, SDN/OpenFlow specification get market attention, especially in activities about managing and controlling networks. This work presents Op3nControl, a proposal about a framework to manage and control OpenFlow networks. Among its main functionalities, Op3nControl can (i) collect network elements statistics in real time; (ii) deploy an environment to develop graph algorithm (routing); (iii) control resources through the establishment of virtual circuits (VC) over network infrastructure; (iv) offer QoS treatment; and (v) integrate legacy equipment to OpenFlow environment. A Op3nControl instance was implemented, thus, this software was executed over a real OpenFlow testbed, among several experimental scenarios, looking for assure that Op3nControl objectives and performance measurements were achieved. The results show that Op3nControl got great performance metrics in its executions, that underlies its deployment in more a complex network environment. Finaly, by exposing Op3nControl architecture and its functionalities, this work intend to improve both academic and market researches in SDN area.

Sumário

Lista de Abreviaturas

TIC	Tecnologia da Informação e Comunicações
VoIP	Voz sobre IP
P2P	Peer-to-Peer
MIT	Massachusetts Institute of Technology
ARPA	Advanced Research Project Agency
WWW	World Wide Web
NAT	Network Address Translation
CIDR	Classless Inter-Domain Routing
IPSec	Internet Protocol Security
GENI	Global Environment for Network Innovations
FIRE	Future Internet Research & Experimentation
API	Application Programming Interface
CV	Circuito Virtual
VoD	Video on Demand
QoS	Quality of Service
SDN	Software Defined Networking
TCAM	Ternary Content Addressable Memory
SSL	Secure Sockets Layer
VLANs	Virtual Local Area Network
MAC	Media Access Control
SOs	Sistemas Operacionais
ICMP	Internet Control Message Protocol
MPLS	Multiprotocol Label Switching
OXM	OpenFlow Extensible Match
BoS	Bottom of Stack

SOAP	Simple Object Access Protocol
REST	Representational State Transfer
XML	Extensible Markup Language
JSON	JavaScript Object Notation
BD	Banco de Dados
NoSQL	Not only SQL
ARP	Address Resolution Protocol
FIFO	First In, First Out
HTB	Hierarchy Token Bucket
RED	Random Early Detection
SFQ	Stochastic Fairness Queueing
FIFO	First-In-First-Out
TCP	Transmission Control Protocol
STP	Spanning Tree Protocol

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Neste capítulo introdutório é apresentado o contexto geral deste trabalho. Com propósito de proporcionar clareza e entendimento do mesmo, são abordados os objetivos a serem alcançados, os desafios que justificam a escolha do tema e a estrutura elaborada para a sua apresentação.

1.1 Visão Geral

A área de Tecnologia da Informação e Comunicações (TIC) se expande de modo contínuo e com notável vigor. A cada dia diversas pessoas são incluídas "digitalmente". Todo esse interesse pelo mundo digital, de alguma forma, está relacionado a utilização da rede mundial das redes de computadores, a *Internet*. São inúmeras formas de acesso a rede mundial: PCs, *laptops*, *netbooks*, *tablets*, *smartphones*, telefones IP, TVs, sensores, entre diversos outros dispositivos. Desde 2011, a quantidade de usuários da *Internet* superou 2.2 bilhões de usuários [ITU-D, 2011].

Diversas são as definições para a *Internet*. Segundo Kurose and W. [2010], existem duas formas para defini-la: (1) rede de computadores que interconectam milhares de dispositivos computacionais ao redor do mundo e/ou (2) infraestrutura que provê serviços a aplicações distribuídas (correio eletrônico, navegação na *Web*, voz sobre IP (VoIP), Internet via rádio, vídeo em tempo real, compartilhamento de arquivos *peer-to-peer* (P2P), mensagem instantânea, televisão pela *Internet* e muito mais). Já para Tanenbaum [2003], a *Internet* não pode ser definida em hipótese alguma como uma única rede, mas sim como um vasto conjunto de redes diferentes que utilizam certos protocolos comuns e fornecem determinados serviços comuns. Kurose and W. [2010] afirma ainda, que se trata de um sistema pouco usual no sentido de não ter sido planejado nem ser controlado por ninguém.

A história da *Internet* teve seu início há aproximadamente 50 anos. Trabalhos desenvolvidos no *Massachusetts Institute of Technology* (MIT), *Rand Institute* e no *National Physical Laboratoy* subsidiaram os trabalhos da *Advanced Research Project Agency* (ARPA - Agência de Projetos de Pesquisa Avançada) que culminaram na criação da primeira rede de computadores por comutação de pacotes, a *ARPAnet*. O avanço tecnológico era perceptível. Diversas outras redes de pacotes surgiram, sendo notável a relevância do papel de integrador do protocolo TCP/IP para a consolidação e interconexão da rede mundial. Na década de 90, ocorreu a explosão da *Internet* impulsionada pelo surgimento do serviço *World Wide Web* (WWW), além de *e-mail*, mensagem instantânea e troca de arquivos P2P [Kurose and W., 2010].

Em dias atuais, a *Internet* continua evoluindo. Impulsionada por anseios das mais variadas naturezas, a *Internet* hospeda aplicações de características e classificações distintas, que vão desde aplicações simples de entretenimento a importantes transações bancárias, ou ainda desde um simples *chat* a uma reunião corporativa de alta relevância através de vídeo-conferência. Em resumo, é uma rede global de aplicações heterogêneas. Para alcançar o nível de maturidade percebido nos dias de hoje, a *Internet* se reforçou de diversas técnicas, como, por exemplo, *Network Address Translation* (NAT), *Classless Inter-Domain Routing* (CIDR), *Internet Protocol Security* (IPSec) etc, que a possibilitou evoluir em termos de escalabilidade, segurança, disponibilidade e desempenho.

Entretanto, de acordo com McKeown et al. [2008] a complexidade da *Internet* (enorme quantidade de equipamentos, protocolos de roteamento) somado a necessidade de alta disponibilidade das aplicações não permite a execução de experimentos sobre a sua estrutura. Portanto, não há como testar novas ideias em redes com configurações realísticas. Consequentemente, a maioria das inovações na área de redes não chegam nem a serem testadas.

Nesse contexto, alguns trabalhos como [Paul et al., 2011; Pan et al., 2011] reforçam a visão de McKeown et al. [2008], e afirmam que a infraestrutura da *Internet* está engessada (tradução do termo "*ossified*", bastante lido nas literaturas em inglês), sobretudo pelo fato da *Internet* ter alterado sua concepção inicial de área de pesquisa e se tornar uma *commodity* comercial. Com a estrutura atual, está extremamente árduo suportar crescentes demandas de segurança, desempenho, confiabilidade, distribuição de conteúdo social, mobilidade e outros. Em outras palavras, para o avanço da *Internet* há a necessidade de desenvolvimento de novos protocolos e arquiteturas de infraestrutura de rede.

Nesse aspecto, para que as inovações tecnológicas possam ter suas características e benefícios atestados é necessário executá-las em ambientes experimentais, chamados *testbeds*, similares aos ambientes de produção [Pan et al., 2011]. Dentre as iniciativas de experimentação, destacam-se o projeto norte-americano *Global Environment for Network Innovations* (GENI) [Geni, 2013], o europeu *Future Internet Research & Experimentation* (FIRE) [Fire, 2013] e japonês Akari [Akari, 2013]. Algumas dessas iniciativas de experimentação, como, por exemplo, o GENI, têm características de funcionar como redes programáveis, portanto se faz necessário interagir com *switches* e roteadores programáveis

capazes de processar pacote simultaneamente para múltiplas redes experimentais de forma isolada. Nesse sentido, é apresentado o padrão *OpenFlow* através do qual torna-se possível realizar experimentos sobre a infraestrutura de produção de modo isolado, sem que haja qualquer interferência entre os tráfegos desses ambientes [McKeown et al., 2008].

Na arquitetura *OpenFlow*, os elementos de rede têm seus planos de dados (responsáveis pelo encaminhamento físico dos pacotes) desacoplados dos planos de controle (responsáveis pelo controle de ações sobre os pacotes que transpõem o plano de dados). Sendo assim, o *OpenFlow* permite que o plano de controle dos equipamentos de rede sejam reprogramados remotamente por uma entidade externa, denominada controlador *OpenFlow*, que é responsável por determinar as ações que devem ser executadas pelos *switches* ou roteadores quando receberem pacotes em suas interfaces de rede.

Para isto, o controlador *OpenFlow* fornece uma *Application Programming Interface* (API) para que sejam desenvolvidas aplicações de diversas naturezas (encaminhamento, *firewall*, controle, monitoramento) que interajam com os mecanismos de encaminhamento dos comutadores *OpenFlow*. Atualmente, diversos controladores estão disponíveis, como o *Nox* [Gude et al., 2008], *Floodlight* [Floodlight, 2013], *Beacon* [Beacon, 2013], *Pox* [POX, 2013] e *Maestro* [Cai et al., 2010].

Como uma plataforma experimental, redes com *OpenFlow* habilitado podem suportar inúmeros cenários de novas arquiteturas e aplicações de redes. Devido suas características e flexibilidade, além do interesse relevante na área de pesquisa, a aceitação do *OpenFlow* no campo da indústria é notável. Diversos fabricantes de *switches* já dispõem de modelos com *OpenFlow* habilitado. Aplicações de controle e de gerência de redes OpenFlow estão emergindo, contudo ainda não há um consenso ou padrão para uma arquitetura.

Nesse contexto, é apresentado neste trabalho o Op3nControl, uma proposta de *framework* para gerência e controle de redes *OpenFlow*. É capaz de obter dados de elementos de comutação e seus enlaces e capturar suas estatísticas da rede, para, posteriormente, disponibilizá-las de forma estruturada. Além disso, fornece um ambiente de desenvolvimento para algoritmos capazes de calcular o caminho entre dois extremos da rede e, sobre este caminho, provisionar circuitos virtuais (CVs) de modo automatizado através do protocolo *OpenFlow*.

Diante do fato de que muitas das novas aplicações fazem ou farão uso de conteúdo multimídia - vídeo sob demanda (*Video on Demand*, VoD), voz sobre IP, serviços multimídia de alta resolução - é imprescindível a utilização de técnicas que possibilitem o tratamento e controle de qualidade de serviço (*Quality of Service*, QoS), com objetivo de adequar o modo de utilização da infraestrutura às características e necessidades de cada aplicação.

Incentivado pelo cenário em que aplicações multimídia ganham espaço e criam um futuro promissor no mundo da *Internet*, foram desenvolvidas APIs no Op3nControl que possibilitam tratamento de QoS sobre os fluxos de dados. Em outras palavras, foram desenvolvidas aplicações de tratamento de disciplinas de filas e criação de classes de fluxos,

além de estender o protocolo *OpenFlow* a fim de suportar as novas funcionalidades de QoS. Portanto, os circuitos virtuais criados pelo Op3nControl podem fazer uso de técnicas de classificação de dados e configuração de filas nos dispositivos de rede com *OpenFlow* habilitado, possibilitando desta forma garantir níveis de qualidade de serviço desejáveis de acordo com as necessidades de cada aplicação.

Além disso, o Op3nControl possui a capacidade de gerenciar equipamentos sem *OpenFlow* nativo (referenciados como legados). Através do desenvolvimento de controles e mensagens *OpenFlow* específicas, o Op3nControl se integra a solução chamada LegacyFlow [Farias et al., 2012]. Desta forma, é possível ter um gerenciamento e controle de equipamentos heterogêneos (com *OpenFlow* nativo e legados) de forma padronizada através do protocolo *OpenFlow*.

1.2 Motivação e Desafios

O *OpenFlow* [McKeown et al., 2008] é um padrão que permite a reprogramação dos elementos de rede (*switches*, roteadores, *access-points*) que possuem *OpenFlow* habilitado. Através da divisão entre plano de controle e plano de dados nestes equipamentos, é possível que este primeiro seja controlado via entidade externa denominado controlador *OpenFlow*. Esse controlador fornece APIs para o desenvolvimento de aplicações para a infraestrutura de rede subjacente.

Esse comportamento de permitir que as redes sejam controladas por *software* é a principal característica do paradigma de *Software Defined Networking* (SDN). Esse padrão de redes traz um conceito que altera bastante o cenário atual de redes, em que grandes empresas comercializam seus equipamentos com softwares proprietários, embarcados e impossibilitados de qualquer alteração ou refinamento por parte dos usuários finais. O *OpenFlow* é a primeira implementação do paradigma SDN [ONF, 2012].

SDN/*OpenFlow* compõe um padrão emergente de paradigma de redes que possibilitam que diversas inovações sejam construídas, como, por exemplo, novos protocolos de roteamento, novas arquiteturas de rede, modelos de *firewall*, controle de tráfego. De posse de uma infraestrutura de redes *OpenFlow* é necessário que haja aplicações que a controle e gerencie. Contudo, essas aplicações compõem um cenário ainda em desenvolvimento, que necessita de pesquisas, propostas e definições. Neste contexto, esse trabalho pretende contribuir para o estado da arte dessas aplicações através do desenvolvimento de um *framework* de controle e gerenciamento de redes SDN/*OpenFlow*. A proposta do Op3nControl torna possível o experimentador ou administrador de redes ter acesso a diversas informações da infraestrutura de redes. Além disso, o mesmo pode alocar recursos para sua utilização através de funcionalidade de descoberta de caminho e criação de circuito virtual com QoS. A proposta oferece um ambiente para desenvolvimento de protocolos de encaminhamento integrado a diversas APIs de recuperação de estatísticas dinâmicas da rede, fomentando, assim, o desenvolvimento de iniciativas inovadoras.

Adicionalmente, este trabalho apresenta uma solução para tratamento de QoS

integrado ao *OpenFlow*, a qual constitui uma área de pesquisa relevante, conforme atestam os trabalhos de Kim et al. [2010] e Civanlar et al. [2010]. Outro objetivo desta pesquisa, é permitir integrar soluções que utilizam arquiteturas do paradigma SDN e tradicional de redes (legada), e proporcionar uma camada de controle e gerenciamento que integre estas arquiteturas distintas.

Nesse contexto, este trabalho pretende contribuir, através da apresentação do Op3nControl, com a área de pesquisa e desenvolvimento de SDN/OpenFlow, objetivando que as soluções inovadoras integradas a esta proposta, agreguem benefícios aos cenários de experimentação requeridos pela área de Internet do Futuro, assim como apresentar novas ferramentas para o mercado.

1.3 Objetivos

O objetivo geral deste trabalho é apresentar um *framework* que realize gerenciamento e controle de redes *OpenFlow*. Como desdobramento de tal objetivo, os seguintes objetivos específicos foram definidos:

- **Trabalhos Relacionados:** verificar e avaliar um conjunto de trabalhos relacionados ao tema desta pesquisa;
- **Detalhamento de mensagens implementadas no *OpenFlow*:** avaliar o escopo de mensagens *OpenFlow* e suas possíveis aplicabilidades no que tange o controle dos dispositivos de rede;
- **Monitoramento:** realizar estudo e levantamento das mensagens de *status* fornecidas pelo protocolo *OpenFlow* (mensagens *OpenFlow Stats*);
- **Provisionamento:** realizar estudo e levantamento das mensagens de alteração de fluxos fornecidas pelo protocolo *OpenFlow* (mensagens *OpenFlow FlowMod* e suas *Actions*);
- **Persistência de dados:** avaliar o modelo de banco de dados adequado a proposta Op3nControl. É necessário armazenar nodos (*switches*) e enlaces (*links*) com um conjunto de características relevantes (baseadas no resultado obtido do item de "Monitoramento");
- **APIs de QoS:** desenvolver APIs de comunicação com *datapaths* com *QoSFlow* Ishimori et al. [2010] e criar aplicações para uso dessas APIs;
- **Desenvolvimento de arquitetura para o *framework* proposto:** modelar um *framework* capaz de interagir com os dispositivos de rede *OpenFlow* e realizar as atividades de gerenciamento e monitoramento;

- **Interfaces de comunicação com aplicações clientes:** desenvolvimento de interfaces para consumo de serviços disponibilizados no *framework* através de chamadas de serviços *web*;
- **Análise de Resultados:** de posse de resultados de experimentos, avaliar atendimento das necessidades funcionais do *framework* e aspectos de desempenho;
- **Comparação de Resultados:** de posse de resultados de experimentos, avaliar e comparar com soluções similares;

1.4 Organização do Trabalho

Além deste capítulo introdutório, o trabalho está dividido seguindo o ordenamento descrito abaixo:

- Capítulo 2: São apresentados os conceitos de SDN (*Software-Defined Networking*), protocolo *OpenFlow* e aplicações de controladores *OpenFlow*.
- Capítulo 3: São apresentadas propostas cujos temas se relacionam aos objetivos deste trabalho.
- Capítulo 4: É apresentada a proposta do *framework* Op3nControl, descrita sua arquitetura geral, assim como são detalhados seus módulos e suas características.
- Capítulo 5: Neste capítulo são demonstrados os cenários de experimentação para verificação de requisitos funcionais que fundamentam a proposta, assim como parâmetros de desempenho que validam sua aplicabilidade.
- Capítulo 6: Apresenta as considerações finais sobre a pesquisa, além de propostas de trabalhos futuros.

CAPÍTULO 2

Referencial Teórico

Este capítulo tem como objetivo descrever os conceitos fundamentais relacionados ao tema deste trabalho, que servirão como base para o entendimento da proposta apresentada. Além disso, serão retratados os estudos sobre estados da arte dos conceitos apresentados.

2.1 *Software-Defined Networking*

Software-Defined Networking (SDN) é um paradigma de redes emergente que possibilita o desacoplamento do plano de controle da rede de seu plano de encaminhamento. O plano de controle - antes inacessível - é retirado dos dispositivos de rede e migrado para entidades externas que proporcionam sua programação remota. Nesse sentido, a infraestrutura passa a ser abstraída na visão de aplicações e serviços de rede, estes, inclusive, podem fazer uso da rede como entidade lógica ou virtual. Consequentemente, os dispositivos físicos da infraestrutura de redes funcionarão apenas como encaminhadores de dados [ONF, 2012].

A Figura 1 representa uma visão lógica da arquitetura relacionada ao paradigma SDN. A inteligência da rede está centralizada nos controladores SDN (camada de controle), entidades que têm uma visão global da infraestrutura de redes. Com isso, as aplicações de rede, situadas na camada de aplicação, enxergam a rede como um único comutador lógico e centralizado. Mesmo comutadores de diferentes modelos e fabricantes irão ser controlados e gerenciados através das mesmas interfaces lógicas e dinâmicas providas pelo SDN. Através deste paradigma, administradores e operadores de rede poderão realizar suas configurações de modo simplificado e padronizado, e, sobretudo, poderão desenvolver aplicações de rede de acordo com suas reais necessidades, em detrimento a aguardar

soluções proprietárias e fechadas dos fabricantes dos equipamentos de rede [ONF, 2012].

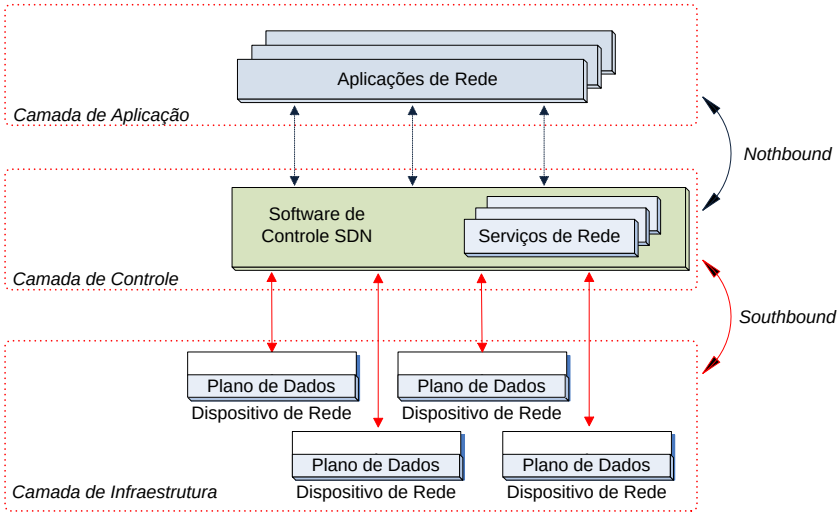


Figura 1: Arquitetura SDN, adaptado de ONF [2012]

No paradigma SDN, do ponto de vista do controlador, a comunicação com os *softwares* da camada de aplicação ocorre pela extremidade norte (*northbound*), enquanto que a comunicação com elementos de comutação, que compõem a camada de infraestrutura, ocorre pela extremidade sul (*southbound*).

A relação entre SDN e *OpenFlow* pode, de modo resumido, ser definida como o *OpenFlow* sendo uma instância do paradigma SDN. ONF [2012] ratifica tal visão, uma vez que define o protocolo *OpenFlow* como sendo o primeiro padrão de comunicação para a SDN via *southbound*.

2.2 *OpenFlow*

A arquitetura de rede *Ethane* [Casado et al., 2007], criada de acordo com grande parte das diretrizes do projeto 4D [Greenberg et al., 2005], serviu de referência para o desenvolvimento do *OpenFlow* [McKeown et al., 2008].

OpenFlow é a primeira interface de padronização de comunicação definido entre as camadas de controle e infraestrutura de redes em uma arquitetura relacionada ao paradigma SDN [ONF, 2012], disponibilizada na Figura 1.

A maior parte dos comutadores *Ethernet* e roteadores modernos contém tabelas de fluxos (*flow-tables*), tipicamente construídas de *Ternary Content Addressable Memory* (TCAM), que são executadas em *line-rate* para implementar diferentes funcionalidades, como, por exemplo, *Firewalls*, NAT, QoS e coleta de estatísticas. A especificação do *OpenFlow Switch* baseia-se na ideia de prover um protocolo aberto para programar as tabelas de fluxos de diferentes *switches* e roteadores. As ações de programação permitidas pelo *OpenFlow* são resultantes de uma pesquisa e mapeamento de um interessante subconjunto de ações comuns a diversos comutadores e roteadores, uma vez que, dependendo

do fabricante, esses equipamentos utilizam tabelas de fluxos diferentes [McKeown et al., 2008].

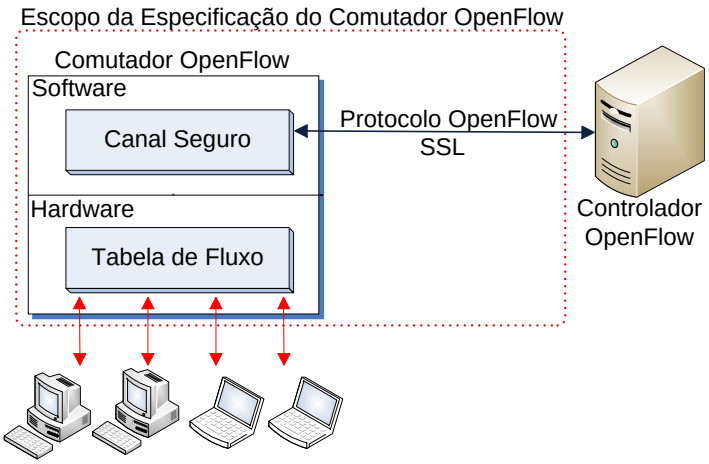


Figura 2: Especificação *OpenFlow Switch* e Controlador *OpenFlow*, adaptado de McKeown et al. [2008]

A definição de *OpenFlow*, em seu sentido mais amplo, engloba duas entidades: o comutador e o controlador *OpenFlow*. Ambos podem ser observados na Figura 2. O comutador *OpenFlow* se constitui dos seguintes componentes [McKeown et al., 2008]:

- Tabela de Fluxos: Armazena as entradas de fluxos. Cada entrada de fluxo contém campos de cabeçalho, contadores e ações. Os campos de cabeçalho da especificação *OpenFlow* podem ser visualizados na Tabela 1, sua função é permitir a verificação do *Match* (combinação) com as informações de cabeçalho dos pacotes da rede. Os contadores podem ser agrupados por tabelas, fluxos, portas e filas. Cada entrada está associada de zero a muitas ações. Caso não encontre uma ação de encaminhamento, o pacote é descartado (ou é enviado ao controlador, se assim definido). De modo resumido, em um fluxo de dados passado pelo comutador, caso haja o *Match* em algum dos campos de cabeçalho, determinada entrada de fluxo é acionada, neste momento, são incrementados os contadores relacionados a essa entrada de fluxo e, em seguida, são executadas as ações mapeadas para a respectiva entrada de dados

- Canal Seguro: Para que a comunicação entre o comutador *OpenFlow* e o controlador ocorra de forma segura, é utilizado o protocolo *Secure Sockets Layer* SSL como meio de acesso entre os mesmos

- Protocolo *OpenFlow*: Especificação aberta que possibilita trocas de informações entre o *switch* (comutador) *OpenFlow* e o controlador. Especifica uma interface padrão que elimina a necessidade de programação diretamente no comutador (ocorre de fato no controlador *OpenFlow*). É composto por informações que serão utilizadas para a determinação/programação de ações que devem ser realizadas sobre determinado fluxo de dados.

Um dos fundamentos do *OpenFlow* é permitir a separação entre plano de dados e plano de controle dos elementos de comutação de rede, ou seja, um comutador com *OpenFlow* habilitado mantem suas características de encaminhamento de dados (plano de dados), este que é comandado não mais por seu *software* embarcado no equipamento (plano de controle), mas sim, por um *software* customizado executando em máquina externa deste equipamento de rede.

Comutadores *OpenFlow* são classificados basicamente em dois tipos ou classes. A primeira classe é composta por equipamentos dedicados, ou seja, apenas "compreendem" *OpenFlow* e não suportam encaminhamento comum de nível 2 e nível 3, é um elemento que apenas encaminha o tráfego entre as portas do comutador de acordo com a tabela de fluxos configurada remotamente via controlador *OpenFlow*. Na primeira geração de comutadores ("Tipo 0") , os cabeçalhos do fluxo formam uma tupla de dez elementos, conforme Tabela 1.

Tabela 1: Elementos de Cabeçalho *OpenFlow* de Comutador "Tipo 0", adaptado de McKeown et al. [2008]

Porta de Entrada	ID VLAN	Ethernet			IP			TCP	
		End Orig	End Dest	Tipo	End Orig	End Dest	Proto	Orig	Dest

O segundo tipo consiste em um comutador comercial com *OpenFlow* instalado como ferramenta ou funcionalidade adicional, que, além de suportar as funcionalidades do *OpenFlow*, realiza as funções comuns de um comutador. Assim, o tráfego de produção pode ser encaminhado utilizando o encaminhamento normal e permanece isolado do tráfego experimental, que utiliza o mecanismo do *OpenFlow*. Para cada entrada da tabela de fluxos é definida uma ação para ser tomada com os pacotes oriundos de um determinado fluxo.

Cada entrada de fluxo tem uma ação simples associada. As três ações básicas que todos comutadores *OpenFlow* devem suportar são [McKeown et al., 2008]:

- Encaminhar os pacotes do fluxo para uma dada porta ou várias portas. Isso permite que os pacotes sejam roteados através da rede;
- Encapsular os pacotes do fluxo e encaminhá-los para o Controlador utilizando o Canal Seguro de comunicação. Comumente, essa ação é usada no cenário de envio do primeiro pacote de um fluxo para o controlador, este que deverá processar o pacote e decidir sobre a configuração de entrada de fluxos nos comutadores. Além disso, a ação pode ser realizada em um experimento no qual é necessário processamento adicional nos pacotes de um fluxo;
- Descartar o pacote do fluxo. Ação pode ser utilizada para prover segurança como, por exemplo, impedir ataques de negação de serviço ou reduzir *broadcasts* excessivos na rede.

Além das três ações anteriormente citadas, exclusivamente os comutadores com *OpenFlow* habilitado podem realizar ação adicional:

- encaminhar os pacotes do fluxo pelo *pipeline* normal do comutador. Nessa ação, um fluxo identificado como não sendo referente ao *OpenFlow* será processado utilizando o encaminhamento normal.

Como alternativa ao uso da ação anterior, um comutador pode isolar o tráfego de produção do tráfego *OpenFlow* através do uso de *Virtual Local Area Network*(VLANs). Alguns comutadores podem suportar tanto essa alternativa como a anterior.

O tratamento de um pacote entrante em um comutador *OpenFlow* é descrito, em modo de fluxograma, na Figura 3. Quando um pacote é recebido pelo comutador, o mesmo cria uma estrutura de dados (ver Figura 4) e a compara com as entradas de fluxos registradas em sua tabela de fluxo. Caso haja um casamento dos campos (*Match*) de determinada entrada de fluxos com a estrutura de dados do pacote entrante, os contadores e ações associadas a essa entrada de fluxo são executados. Caso o pacote entrante não satisfaça a nenhuma entrada, o pacote será enviado ao controlador (funcionamento padrão) *OpenFlow* para que sejam definidas as ações sobre este.

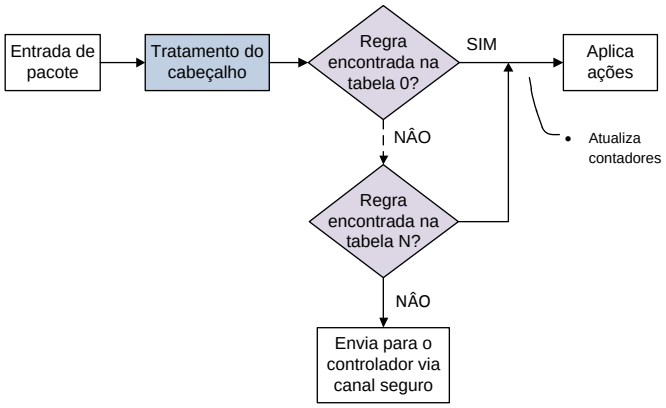


Figura 3: Fluxograma de Tratamento de Pacote Entrante em Comutador *OpenFlow*, adaptado de OpenFlowSpecv1.0 [2013]

A criação da estrutura de dados do pacote entrante, executada pela atividade de "Tratamento de Cabeçalho", é detalhada na Figura 4. Inicialmente, alguns campos são configurados (porta de entrada, endereços MAC, *Ethertype*). Posteriormente, é verificado o tipo de *Ethertype* do pacote entrante e, de acordo com seu valor, diferentes tratamentos serão realizados para a formação da estrutura de dados.

Segundo Farias et al. [2011], o *OpenFlow* permite a execução de experimentos de novas propostas na área de redes de computadores sobre uma infraestrutura de redes já existente. São citadas algumas propostas que podem ser trabalhadas com o *OpenFlow*:

- Desenvolvimento de novos Protocolos de Roteamento: No controlador *OpenFlow* podem ser desenvolvidas aplicações que controlem o roteamento de pacotes em uma

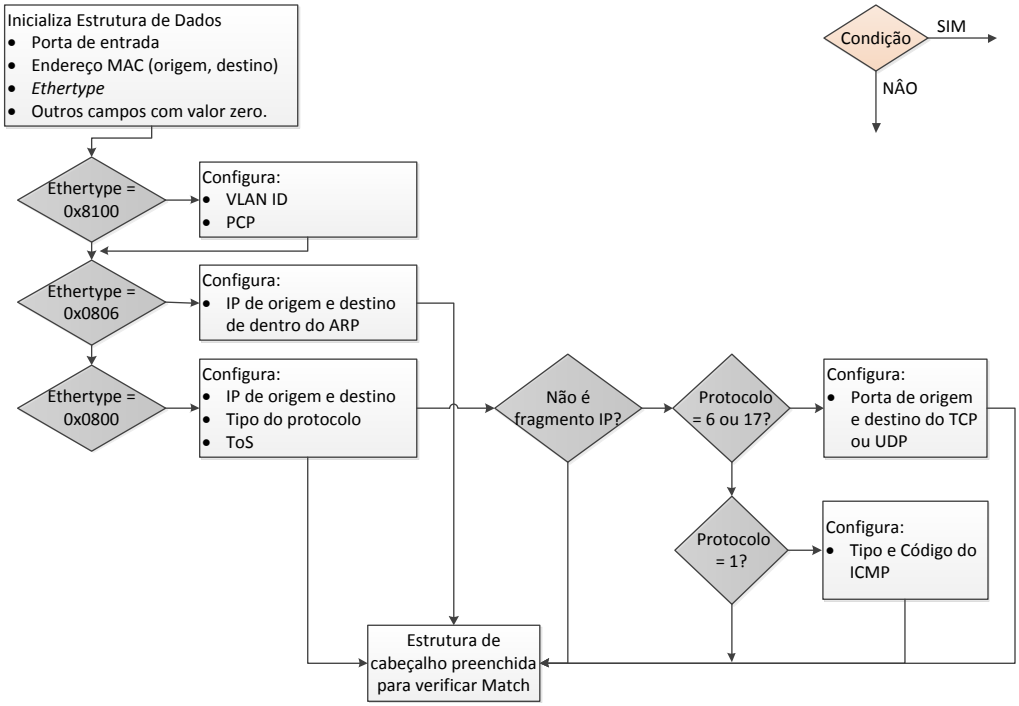


Figura 4: Fluxograma de Criação de Estrutura de Dados para Verificação de *Match*, adaptado de OpenFlowSpecv1.0 [2013]

rede. Para isso, quando um pacote de dados chegar a um comutador, ele é encaminhado para o controlador, que calcula a melhor rota para o pacote seguir na rede, a partir dos mecanismos do protocolo de roteamento desenvolvidos (inovação). Após isso, o controlador envia mensagens *OpenFlow* que adicionam entradas (*flowentries*) na tabela de fluxos de cada comutador pertencente à rota escolhida. Os próximos pacotes desse fluxo que forem encaminhados na rede serão encaminhados em nível de plano de dados através das regras configuradas nas tabelas de fluxos;

- Mobilidade: uma rede *OpenFlow* pode possuir pontos de acesso sem-fio, permitindo que dispositivos móveis utilizem sua infraestrutura para se conectarem a servidores ou à Internet. Assim, no Controlador *OpenFlow* podem ser desenvolvidas softwares que implementem mecanismos de handoff inteligentes, permitindo alteração dinâmica das tabelas de fluxos dos comutadores de acordo com a movimentação do cliente, permitindo a redefinição da rota utilizada;

- Redes Não-IP: um comutador *OpenFlow* pode ser desenvolvido para analisar campos arbitrários de um pacote (campos disponíveis na Tabela 1) permitindo flexibilidade no tratamento e definição do fluxo. Dessa forma, podem ser testadas, por exemplo, novas formas de endereçamento e roteamento. Nos comutadores "tipo 0", os pacotes Não-IP podem ser definidos pelo endereço *Media Access Control* (MAC) de origem ou destino ou a partir de um novo tipo *Ethernet* ou IP;

2.2.1 Mensagens

Tomando como base o protocolo *OpenFlow* v1.0 [OpenFlowSpecv1.0, 2013], no escopo do protocolo *OpenFlow*, são apresentados três grupos de mensagens:

- Controlador para Comutadores: enviadas pelo controlador para os comutadores. Requer ou não respostas dos comutadores. As mensagens deste grupo estão listadas na Tabela 2.

Tabela 2: Mensagens Controlador para Comutadores

Features	Controlador solicita capacidades (características) dos comutadores. Comutador envia resposta
Configuration	Controlador envia ou consulta parâmetros de configuração dos comutadores. Comutador envia resposta apenas para consultas
Modify-State	Controlador gerencia estado dos comutadores
Read-State	Controlador coleta estatísticas dos comutadores
Send-Packet	Mensagens usadas pelo controlador para enviar pacotes por porta específica no comutador
Barrier	Controlador solicita que comutador reportar quando o mesmo finalizar um conjunto de operações *OpenFlow*

- Assíncrona: enviadas pelos comutadores sem solicitação do controlador. As mensagens deste grupo estão listadas na Tabela 3.

Tabela 3: Mensagens Assíncronas

Packet-in	Quando não encontrada nenhum *Match* na tabela de fluxos, uma mensagem *packet-in* é gerada pelo comutador e enviada ao controlador (ou quando há uma ação explícita de envio ao controlador)
Flow-Removed	Indica quando uma entrada de fluxo é eliminada da tabela de fluxos dos comutadores
Port-status	Enviada pelo comutador quando alterado o estado de configuração de uma porta
Error	Enviada pelo comutador quando detectado erro

- Simétricas: enviadas por controlador ou comutadores sem solicitação prévia. Exige respostas. As mensagens deste grupo estão listadas na Tabela 4.

2.2.2 Controladores

O controlador é um elemento da rede *OpenFlow* que possui funcionalidades de programação e configuração, responsável por interagir com a tabela de fluxos presente

Tabela 4: Mensagens Simétricas

Hello	mensagens enviadas (*request/reply*) entre comutador e controlador durante o estabelecimento de conexão
Echo	mensagens enviadas (*request/reply*) entre comutador e controlador indicando conexão ativa. Podem ser utilizadas para indicação de latência e largura de banda
Vendor	Alternativa para utilizar informações adicionais dentro do escopo de mensagens *OpenFlow*

no comutador *OpenFlow*. O controlador pode funcionar de modo estático, quando configura fluxos para a execução dos experimentos, sem alterações, ou pode ser utilizado de modo dinâmico, quando permite que fluxos sejam adicionados e removidos durante a fase de execução dos experimentos. O controlador *OpenFlow* abstrai as peculiaridades e especificidades dos equipamentos de rede para facilitar a programação remota destes.

Enquanto o protocolo *OpenFlow* possibilita a criação de uma abstração para os comutadores de rede, o controlador *OpenFlow* proporciona um nível de abstração integral de toda a rede, contando com a existência de serviços para a rede, muito similar a abstração que sistemas operacionais (SOs) realizam sobre sistemas tradicionais. Por esta razão, muitas vezes, o controlador *OpenFlow* é referenciado como SO de redes; de modo análogo, nesse cenário, o *OpenFlow* seria uma espécie de *"driver"* para dispositivos de rede.

Como implementações relevantes de controlador *OpenFlow*, podem ser citados o *Nox* [Gude et al., 2008], *Pox* [POX, 2013], *Beacon* [Beacon, 2013], *Maestro* [Cai et al., 2010] e *Floodlight* [Floodlight, 2013]. A implementação do *Nox* é baseado em C++; *Pox* em *Python*; e os três últimos baseados em *Java*.

O *Floodlight* trata-se de uma implementação de controlador *OpenFlow* baseado em linguagem *Java*, possui licença *Apache*, é desenvolvido e mantido pela comunidade livre, incluindo engenheiros do *Big Switch Networks* - inclusive o *Floodlight* compõe o *core* da solução de controlador *OpenFlow* comercial desta empresa, o *Big Network Controller*. Possui uma arquitetura modular, possibilitando extensões e otimizações de modo simplificado.

2.2.3 Versões

Nesta sub-seção serão citadas as versões do protocolo *OpenFlow*. Serão descritas as datas de liberação e **algumas** das principais funcionalidades incrementadas em cada uma das *releases*. A versão mais recente é a v1.3.1. A seguir são listadas, ordenadamente, as versões do *OpenFlow* [OpenFlowSpecv1.3.1, 2013]:

Versão 0.2.0. Data de liberação: 28/03/2008

Versão 0.2.1. Data de liberação: 28/03/2008

Versão 0.8.0. Data de liberação: 05/05/2008. Principais alterações:

- Reorganização de tipos de mensagens *OpenFlow*
- Adicionado prioridade nos fluxos
- Adicionado mensagens de erro
- Adicionado mensagens de estatísticas de tabelas e portas

Versão 0.8.1. Data de liberação: 20/05/2008

Versão 0.8.2. Data de liberação: 17/10/2008. Principais alterações: Adicionado mensagens de requisição e resposta de *"echo"*

Versão 0.8.9. Data de liberação: 02/12/2008. Principais alterações:

- Incrementado capacidade de entrada de fluxos para conter máscaras de subrede IP
- Expansão de quantidade de informação da mensagem de estatística de porta (`ODP_PORT_STATS`)
- Adição de capacidade aos comutadores informar alterações nos status de porta e *link* (`OFPPC_PORT_DOWN` e `OFPPS_LINK_DOWN`)
- Otimização das mensagens *Echo Request/Reply Messages* no sentido de melhorar a conectividade entre comutador e controlador
- Criação de extensão para fabricantes
- Criada capacidade de configurar e visualizar implementações do protocolo 802.1D *Spanning Tree* embarcados nos comutadores
- Otimização do comando de alteração de fluxos (`ofp_flow_mod`) para poder alterar *"Actions"*
- Descrições mais flexíveis das tabelas de fluxos
- Refatoração da mensagem `Packet-Out`
- Definição de *hard-timeout* (tempo absoluto para expiração de regra) para entradas de fluxos
- refatoração do processo de *"handshaking"* das mensagens `OFPT_HELLO` para suportar negociação de versões
- Melhoria da mensagem de estatística de *switch* (*switch stat*)
- Alterações das ação de VLAN
- Número máximo de portas alteradas para 65280
- Definição de comportamento quando o comutador perde conexão com controlador
- Incremento de *Macth* para tráfego *Internet Control Message Protocol* (ICMP)

Versão 0.9.0. Data de liberação: 20/07/2009. Principais alterações:

- *Failover*: Incrementado capacidade do *datapath* definir uma lista de controladores e, em caso de falha do controlador principal, o mesmo selecionará o segundo controlador da lista
- *Emergency Flow Cache*: em caso de perda de conexão com o controlador, o comutador irá inserir regras de emergência com objetivo de minimizar a falha.
- *Match* com bit de prioridade em VLANs
- Refinamento da regras de expiração (passa a ser por fluxo)
- Comportamento do *Flow Mod* alterado
- Incorporação do campo duração na expiração do fluxo
- Incremento de notificação para exclusão de fluxos nos comutadores
- Enumeração de porta iniciando por 1 (anteriormente iniciava-se com 0)
- Recomendação da porta 6633/TCP como padrão para *OpenFlow*

Versão 1.0. Data de liberação: 31/12/2009. Principais alterações:

- *Slicing*: capacidade de suportar múltiplas filas por porta de saída
- Configuração de *cookies* para as entradas de fluxo
- Capacidade de realizar *Match* em campos IP nos pacotes ARP
- Capacidade de realizar *Match* em bits IP ToS/DSCP
- Capacidade de realizar consultas de estatísticas de portas de modo individual
- Otimização da unidade para nanosegundos das mensagens de estatísticas e expiração de fluxo
- Alteração de referência SSL para TLS

Versão 1.1. Data de liberação: 28/02/2011. Principais alterações:

- Alteração do processamento (*pipeline*) *OpenFlow* para visão de múltiplas tabelas
- Abstração de grupos de portas
- Incremento de capacidade de trabalhar com VLAN e *Multiprotocol Label Switching* (MPLS)
- Incremento de conceito de portas virtuais
- Melhoria no processo de perda de conexão com controlador *OpenFlow*

Versão 1.2. Data de liberação: 05/12/2011. Principais alterações:

- Incremento de flexibilidade do *macth* através do *OpenFlow Extensible Match* (OXM)
- Extensão do contexto da mensagem `PACKET_IN`

- Simplificação do comportamento da mensagem de requisição `FLOW_MOD`
- Suporte a IPV6
- Exclusão da especificação de parser para os pacotes
- Alteração do mecanismo de role do controlador (no processo de definição do controlador backup em caso de falhas)

Versão 1.3. Data de liberação: 13/04/2012. Principais alterações:

- Refatoração de negociação de características do comutador
- melhoria no suporte de IPV6
- Medidores de fluxos nos comutadores
- Filtros de eventos por conexão
- Novo campo `OXM_OF_MPLSBOS` para realizar *Match* no *bit Bottom of Stack* (BoS) no cabeçalho MPLS
- Adição de campos de duração dos fluxos em estatísticas
- Capacidade de desabilitar contadores nos comutadores na mensagens de instalação de entradas de fluxos

Versão 1.3.1 Data de liberação: 08/2012. Principais alterações:

- Otimização da negociação de versão (estabelecimento de conexão entre comutador e controlador)

2.2.4 Integração com o *FlowVisor*

De acordo com Sherwood et al. [2009], o *OpenFlow* torna possível a existência de camada de abstração de *hardware* da infraestrutura de redes, pois todos os elementos desta estarão aptos a comunicar-se via interface padronizada através do protocolo *OpenFlow*. Sobre essa abstração é possível aplicar camada de virtualização através da ferramenta *FlowVisor*.

A abstração de rede disponibilizada pelo *OpenFlow* deve ser facilmente "fatiável", para que múltiplas redes virtuais (formando topologias virtuais) com diferentes características possam coexistir preservando a individualidade de cada fatia ou *slice*. Cada *slice* deverá ter seu tráfego isolado dos demais, de modo que cada *slice* tenha privilégios de executar suas aplicações de forma autônoma a infraestrutura física de rede compartilhada. A camada de abstração criada pelo *OpenFlow* compatibiliza a forma de tratamento comum a uma variedade de *hardwares* de rede diferentes.

Portanto, em cada *slice* será possível que novos protocolos e formatos de endereçamento sejam executados independentemente sobre a mesma rede física. O responsável por criar e administrar as fatias de virtualização é o componente *FlowVisor*, que

pode ser definido como um controlador especializado. O *FlowVisor* é responsável por interceptar todas as mensagens *OpenFlow* que trafegam na rede, tanto aquelas originárias dos comutadores para os controladores quanto as dos controladores para os comutadores, sem, no entanto, realizar qualquer tratamento ou processamento sobre as mesmas, funcionando de forma semelhantes a um servidor *proxy* transparente. [Sherwood et al., 2009].

Dividir a rede em *slice* significa fatiar todos os recursos de rede. Inicialmente o *slicing*, ou fatiamento, atua em quatro dimensões [Sherwood et al., 2010]:

- Topologia: cada *slice* tem sua própria visão dos nodos de rede (ex: *switches, access-points*, roteadores) e suas respectivas conexões;
- Largura de banda: cada *slice* tem sua própria fração de largura de banda em cada *link* ou enlace. Uma possível falha no isolamento de largura de banda permitiria que um *slice* tomasse parte (ou até integralmente, no pior caso) da vazão de dados de outro *slice*;
- CPU do equipamento: cada *slice* é limitado pela fração a si destinada. *Switches* e roteadores, tipicamente, tem recursos computacionais significativamente limitados. Sem o gerenciamento correto de CPU nos *slices*, os *switches* perderão a capacidade de encaminhar *slow-path packets*, rejeitarão requisições de estatísticas e, sobretudo, tornarão-se incapazes de processar atualizações das tabelas de encaminhamento;
- Tabelas de encaminhamentos: cada *slice* tem uma cota finita de regras de encaminhamento. Equipamentos de rede tipicamente suportam um número finito de regras de encaminhamento (entradas *TCAM*, por exemplo). Falhas de isolamento nas ações de encaminhamento entre os *slices* poderão permitir que um *slice* iniba outro da capacidade de encaminhar pacotes.

2.3 Conclusões do Capítulo

Este capítulo apresentou conceitos básicos de redes *OpenFlow*, suas características, sua arquitetura, descrição da especificação de comutador, principais mensagens e versões do protocolo. Além disso, foram citados os principais controladores e suas integrações com o software *Flowvisor* para se obter "fatias"de recursos da infraestrutura de redes (virtualização). O capítulo também abordou conceitos básicos do paradigma SDN, relatando, inclusive, aspectos da comunicação com extremos norte (*northbound*) e sul (*southbound*). Na próxima seção, é apresentada a proposta do *framework* Op3nControl.

CAPÍTULO 3

Trabalhos Relacionados

O objetivo deste capítulo é apresentar um conjunto de trabalhos e pesquisas associados ao tema desta dissertação, bem como relacioná-los ou compará-los ao trabalho proposto.

3.1 OMNI

Em Mattos et al. [2011], o trabalho intitulado *"OMNI: OpenFlow MaNagement Infrastructure"* descreve uma ferramenta para controle e gerência de redes *OpenFlow*. De forma resumida, o OMNI é capaz de realizar monitoramento da rede (coleta das estatísticas dos fluxos e dados para obtenção da topologia física) via *OpenFlow* e configuração dinâmica de fluxos. Através de um sistema de agentes autônomos (baseados na plataforma *Ginkgo*), quando detectadas falhas na rede, é capaz de configurar dinamicamente os fluxos de dados, afetados por esta falha, por caminhos alternativos por meio de sua interface de manutenção de fluxos na rede. Quando há necessidade de realizar cálculo de caminho é utilizado o algoritmo de *Dijkstra*, de modo a ser estabelecido um caminho completo com o menor número de saltos. A solução é baseada em funções desenvolvidas no controlador NOX.

Por sua vez, o *framework* Op3nControl também tem o objetivo de controlar e gerenciar a rede *OpenFlow* e disponibiliza a possibilidade de criar circuitos dinâmicos. Contudo, o Op3nControl apresenta alguns diferenciais.

O primeiro deles é uma arquitetura que prevê a construção de um módulo de cálculo de caminho flexível que possibilitará o desenvolvimento de diversos algoritmos de cálculo de caminho, sendo que na arquitetura do Op3nControl são disponibilizadas

diversas informações estatísticas da infraestrutura através de simples APIs que poderão ser utilizadas no desenvolvimento de lógica de encaminhamento. Esses dados estarão em um banco de dados orientados a grafos, o que permite uma navegabilidade otimizada sobre os *swicthes* (nodos) e seus *links* (arestas).

Outro diferencial é que o Op3nControl irá proporcionar a possibilidade de criação dos circuitos virtuais com parâmetros de qualidade de serviço. Será possível, por exemplo, realizar o controle de banda (*traffic shaping*) de um *link* e/ou definir, de modo dinâmico, os escalonadores de pacotes das portas dos comutadores compreendidos no caminho entre dois extremos.

Por último, o Op3nControl proporciona o gerenciamento de *switches* legados através de *OpenFlow*. O termo legado refere-se aos equipamentos sem suporte nativo ao protocolo *OpenFlow*.

3.2 QoSFlow

Em Ishimori et al. [2010], o trabalho intitulado "*QoSFlow*: Gerenciamento Automático da Qualidade de Serviço em Infraestruturas de Experimentação Baseadas em *Framework OpenFlow*"define uma forma de realizar o gerenciamento da qualidade de serviço (QoS) de modo dinâmico através do protocolo *OpenFlow*. Essa solução permite que sejam desenvolvidas aplicações em nível de controlador que possam vir a realizar a programação dinâmica de aspectos de QoS nas portas dos dispositivos de rede habilitados com *OpenFlow*.

O *framework* Op3nControl prevê a integração da solução *QoSFlow* em seu escopo. O resultado será um incremento de capacidades para ambos trabalhos, já que, juntos, poderão realizar trabalhos como alocação dinâmica de circuitos com parâmetros de qualidade de serviço através do protocolo *OpenFlow*. Além disso, através dessa integração, o usuário final poderá fazer uso real da aplicação *QoSFlow* de forma bastante intuitiva.

3.3 LegacyFlow

Diante de um cenário de interesse na migração do parque tecnológico de uma empresa ou instituição que utiliza o modelo tradicional (ou legado) para redes *OpenFlow*, é possível, e muito provável, que existam equipamentos utilizados em ambiente de produção desta entidade que não possui suporte ao protocolo *OpenFlow*. As razões podem ser a falta de disponibilidade de modelo de equipamento similar com suporte ao protocolo *OpenFlow* ou ainda um custo elevado que possibilite sua substituição. A solução apontada por Farias et al. [2012] foi desenvolver uma solução chamada *LegacyFlow*.

Esse cenário supracitado é descrito no trabalho intitulado *"A Proposal Manage-*

ment of The Legacy Network Environment using OpenFlow Control Plane". O *LegacyFlow* possui duas camadas: (1) *datapath Openflow* e (2)controlador de *switch*. A camada (1) compreende o plano de controle do *OpenFlow* e comunica-se diretamente com a camada (2), esta por sua vez é responsável por ser uma "ponte"de comunicação com diversos modelos de *switches* legados, em outras palavras, nesta camada existirão diversos módulos que se comunicam com os *switches* através de interfaces de comunicação específicas de cada fabricante, sendo que as mais comuns são *WebService*, *Telnet/CLI* e *SNMP*). O objetivo final dessa integração é possibilitar a criação de circuitos L2 nos dispositivos legados via protocolo *OpenFlow*.

O modelo Op3nControl prevê a integração da solução *LegacyFlow* em seu escopo. O resultado será um incremento de capacidades para ambos trabalhos, já que, juntos, poderão realizar trabalhos, como, a alocação dinâmica de circuitos através do *OpenFlow* sobre infraestrutura heterogênea (equipamentos *OpenFlow* e equipamentos com *LegacyFlow*). O *LegacyFlow* será contemplado com camada de controle de gerenciamento, antes inexistentes em Farias et al. [2012]. Outra possibilidade (trabalho futuro) é a criação de mecanismos para integração de federações autônomas, controladas por *OpenFlow*, através da sinergia entre *LegacyFlow* e Op3nControl.

3.4 Nox

Em Gude et al. [2008], o trabalho intitulado "*NOX: Towards an Operating System for Networks*"é descrito o controlador *NOX*. São informadas suas principais características e funcionalidades. É, de fato, a primeira implementação de controladores do paradigma SDN baseada em *OpenFlow*. O *NOX* disponibiliza *APIs* para desenvolvimento de aplicações de rede baseado em dois conceitos: componente e evento. O primeiro corresponde a um agregado de funcionalidades carregadas com o *NOX* e que podem ser escritas/estendidas pelo usuário. Podem ser utilizadas as linguagem de programação C++ ou *Python*, ou utilizados de forma híbrida. Um evento é uma ação executada sobre um determinado fluxo, que é encaminhado ao controlador, esses eventos são disparados por *handlers* ao serem sensibilizados por mensagens *OpenFlow*, ou seja, um módulo pode conter uma série de eventos que determinam as ações sobre os fluxos encaminhados ao controlador.

A arquitetura do *NOX* e suas funcionalidades foram, de certa forma, inspiradoras para o desenvolvimento do modelo Op3nControl, sobretudo, pelo seu pioneirismo em relação ao desenvolvimento sob prisma da arquitetura SDN. Contudo, devido a própria natureza da SDN, que propicia e fomenta o desenvolvimento de novas aplicações, o modelo Op3nControl traz contribuições inovadoras quando observado sob olhar de controladores *OpenFlow*, uma vez que traz aplicações, por exemplo, módulo de estatística que retorna resultados estruturados em *Json*, módulo que configura CVs com QoS de forma dinâmica ou ainda a possibilidade de gerenciar, via *OpenFlow*, uma infraestrutura composta de equipamentos legados.

3.5 CircuitPusher

Em Floodlight [2013], é apresentada a solução *CircuitPusher* que é uma aplicação independente que acompanha o pacote *Floodlight*. *CircuitPusher* é desenvolvido em linguagem de programação *Python*, utiliza APIs *Rest* do controlador *Floodlight* para criar circuitos virtuais bidirecionais, através da criação de entradas de fluxo em formato *OpenFlow*, permanentes, em todos os *switches* contidos na rota entre dois equipamentos (*hosts*) que tem endereços IP configurados [Kanui and Parraga, 2013].

Na Seção avaliação dos resultados, Seção 5, o *CircuitPusher* será comparado a proposta deste trabalho a fim de se avaliar o desempenho do Op3nControl.

3.6 Conclusões do Capítulo

Este capítulo apresentou alguns trabalhos relacionados ao tema do *framework* Op3nControl, os conceitos obtidos com os mesmos são de fundamental importância para a compreensão da proposta deste trabalho. Foram descritas as principais características e contribuições de cada proposta apresentada e como se relacionaram ao contexto do Op3nControl.

CAPÍTULO 4

Op3nControl

Este capítulo apresenta o Op3nControl, uma proposta de *framework* para gerenciamento e controle de redes baseadas em *OpenFlow*. Para tal objetivo a solução proposta realiza o monitoramento de estatísticas de portas e filas dos equipamentos de rede via protocolo *OpenFlow*, além do controle de topologia da rede subjacente. Adicionalmente, é capaz de controlar os recursos da rede através da alocação dos mesmos em circuitos virtuais, que podem incluir, opcionalmente, controle de QoS.

A Seção 4.1 discute aspectos gerais do Op3nControl e define seu cenário de aplicabilidade. A Seção 4.2 aborda assuntos relacionados à arquitetura da solução e suas principais características, também são detalhadas as estruturas dos seus módulos e conteúdo das mensagens trocadas pelo Op3nControl. Na Seção 4.3 são descritas as funcionalidades providas pelo *framework* proposto e são mostradas as sequências de ações e eventos de duas das principais funcionalidades que tem interação com o usuário. Por fim, a Seção 4.5 conclui o capítulo.

4.1 Visão Geral do Escopo da Proposta

O Op3nControl é uma proposta de *framework* para gerenciamento e controle de redes *OpenFlow* que contempla a captura de dados provenientes dos comutadores e *links* que compõem a infraestrutura de redes *OpenFlow*; também é capaz de obter dados estatísticos de utilização e comportamento desses *links*. Além disso, a proposta disponibiliza um módulo para cálculo de caminho que tem a função principal de oferecer um ambiente de desenvolvimento de algoritmos de encaminhamento. Nesse módulo estão disponíveis várias APIs de consultas de informação da rede, ou seja, o usuário poderá construir um algoritmo que calcule caminhos sobre uma infraestrutura de redes tendo

informações de estatísticas da rede(ver detalhes no módulo *Cálculo de Caminho*, Seção 4.2.1) atualizadas em tempo real.

Outra funcionalidade provida pelo modelo é a capacidade de provisionar circuitos virtuais de modo dinâmico. Para a criação dos CVs são configuradas ações *OpenFlow* nas tabelas de fluxos dos elementos de comutação de rede. Além disso, os CVs tem a opção de utilizar tratamento de QoS, podendo ser realizado controle de tráfego (*traffic shapping*) ou ainda selecionado o tipo de escalonador de pacotes em filas das portas dos comutadores.

Adicionalmente, o Op3nControl utiliza o protocolo *OpenFlow* para a comunicação com elementos de comutação da rede. Portanto, permite conexão com equipamentos com *OpenFlow* habilitado. Como inovação, a proposta apresenta a capacidade de integrar equipamentos comuns, sem *OpenFlow* nativamente habilitado (comumente referenciados como equipamentos legados), ao contexto *OpenFlow*, consequentemente passível de serem gerenciados pelo Op3nControl. Em outras palavras, a proposta reune capacidade de homogeneizar a interface de administração (via OpenFlow) de uma infraestrutura de redes heterogênea (comutadores com *OpenFlow* e legados). Detalhes dessa integração com equipamentos legados serão descritos na Sub-Seção 4.2.1.

Desta forma, esta proposta de *framework* possibilita que um administrador de redes tenha a capacidade de gerenciar uma infraestrutura de redes, monitorando sua disponibilidade, verificando *status* de comutadores e *links*. O mesmo poderá controlar a alocação dos recursos em CVs. Quando detectadas falhas em componentes dessa infraestrutura e estas ocasionarem a interrupção de um CV, o administrador poderá reconstituir este caminho executando novamente uma busca de caminho e criando um novo CV sobre a nova visão da infraestrutura.

O Op3nControl é descrito como *framework* para gerência e controle de redes *OpenFlow*. Em Zhao et al. [2012], o controle da rede é encarregado de computar o roteamento para a rede de dados, enquanto que a gerência é responsável por averiguar o estado desta rede, assim como, distribuição de políticas e configurações. Já em Malik [1990], o controle de rede é relacionado a atividades operacionais, como diagnósticos, configurações, monitoramento e verificação de performance de redes; o gerenciamento de rede compreende diversas funções realizadas para gerenciar os recursos de redes. No contexto deste trabalho, controle corresponde as funcionalidades que geram ação na rede, como, alocação de recurso em CV, criação de classe de fluxo, configuração do escalonador de pacotes. As ações de gerenciamento do Op3nControl são responsáveis por realizar o monitoramento, armazenar o estado da rede (condições dos comutadores, *links*), além de conter políticas para uso dos controles disponibilizados por sua arquitetura.

Por fim, o Op3nControl tem sua estrutura baseada no paradigma *SDN/OpenFlow*. Por esta influência, ou melhor, diretriz, o nome do *framework* proposto apresenta um trocadilho na palavra "*Open*", alterando o "e"por "3", este que referencia as múltiplas camadas da SDN, ver Figura 1: camada de controle, camada de aplicação (*northbound*) e camada de infraestrutura (*southbound*).

4.2 Especificação da Arquitetura do Op3nControl

A arquitetura do modelo proposto é baseada na arquitetura *SDN/OpenFlow*. A estrutura planejada para o Op3nControl se fundamenta na construção de módulos. A justificativa desta escolha é obter uma solução flexível, extensível, além de proporcionar melhor organização, maior independência e isolamento entre as funcionalidades providas. Sobre o prisma da SDN, esses módulos foram desenvolvidos sobre as camadas de controle e aplicação (*northbound*). O Op3nControl prevê customizações e/ou extensões de algumas aplicações (detalhadas nos módulos) da camada de controle, uma vez que em sua estrutura está presente o *core* de um controlador *OpenFlow*. Já na camada de aplicação foi planejado o desenvolvimento de novas aplicações de rede. A arquitetura do *framework* proposto está disponível na Figura 5.

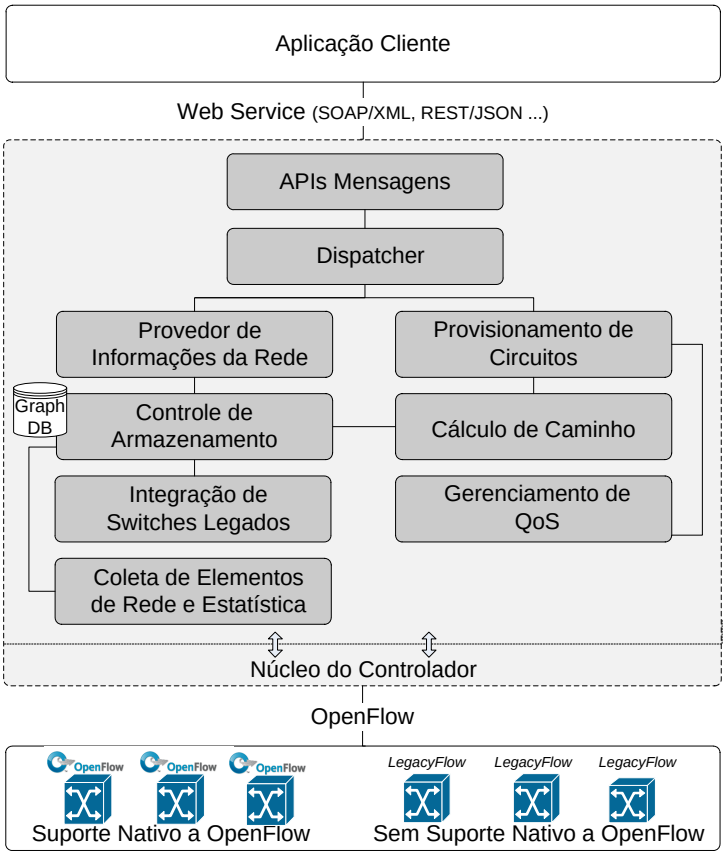


Figura 5: Especificação da Arquitetura do Op3nControl

Analisando sua arquitetura, na Figura 5 verifica-se que o Op3nControl (caixa com fundo em tom de cinza) possui duas extremidades, ideia similar a SDN: a extremidade norte (*northbound*) - que se comunica com aplicações clientes (que podem ser do tipo *web* ou *desktop*) - e a extremidade sul (*southbound*) - que se comunica com elementos de comutação da rede.

Devido esta Seção estar descrevendo a especificação do *framework* Op3nControl, não são determinadas as ferramentas para sua implementação. Contudo, em alguns casos,

são restringidos padrões ou conjunto de tecnologias que devam ser adotados. Na Seção 4.4 será discutida uma implementação do Op3nControl.

Nesse sentido, com objetivo de padronização e questões de assertividade ao padrão *OpenFlow*, deverão ser utilizadas bibliotecas disponíveis que implementem este protocolo de maneira confiável, as mesmas poderão ser conseguidas isoladamente ou embutidas nas implementações de controladores *OpenFlow*. Além disso, a proposta integra a necessidade de utilização do núcleo de um controlador *OpenFlow*, uma vez que as funcionalidades básicas do *OpenFlow* estão em um nível de maturidade bastante consistentes nestes. Como exemplos de implementações de controladores *OpenFlow* citam-se: *Nox* [Gude et al., 2008], *Floodlight* [Floodlight, 2013], *Beacon* [Beacon, 2013], *Pox* [POX, 2013], *Maestro* [Cai et al., 2010], entre outros. Portanto, os módulos do Op3nControl poderão ser tanto extensões e/ou refatorações de aplicações já existentes em um controlador *OpenFlow* como novas implementações de *software* sobre a estrutura destes.

A estrutura do Op3nControl foi construída visando-se obter uma solução flexível e de fácil adaptação para novas funcionalidades. Para tal fim, optou-se pela separação das funcionalidades de natureza similar em agrupamentos ou módulos. A estrutura modular pode ser facilmente observada na Figura 5, na qual podem ser observados os seguintes agrupamentos: (1) APIs Mensagem, (2) *Dispatcher*, (3) Provedor de Informações da Rede, (4) Controle de Armazenamento, (5) Coleta de Elementos de Rede e Estatística, (6) Cálculo de Caminho, (7) Provisionamento de Circuitos, (8) Integração de *Switches* Legados e (9) Gerenciamento de *QoS*. Os módulos (1) a (7) são essenciais para a criação de CVs sobre redes *OpenFlow*; já os módulos (8) e (9) são opcionais para a configuração dos CVs, sendo utilizados para possibilitar funcionalidades extras e específicas.

As próximas Sub-Seções descreverão, respectivamente, detalhamentos dos módulos e aspectos de comunicação no Op3nControl.

4.2.1 Detalhamento dos Módulos

Módulo APIs Mensagem. Responsável por receber solicitações de aplicações clientes através mensagens em formato de serviços *web* (*web services*). As solicitações recebidas por este módulo podem ser de duas naturezas: (1) solicitações de saída, ou seja, aplicações clientes requisitando dados provenientes do Op3nControl; ou (2) solicitações de entrada, ou seja, aplicações clientes enviando dados ao Op3nControl com objetivo de realizar ações internas.

Este módulo deve ser disponibilizado para as aplicações clientes através de um servidor *web*. Em sua implementação deverão ser definidos os padrões tecnológicos - *Simple Object Access Protocol* (SOAP) ou *Representational State Transfer* (REST) - e o formato da estrutura da mensagem - *Extensible Markup Language* (XML), *JavaScript Object Notation* (JSON). Além de realizar o controle de envio e recebimento de mensagens, outra importante funcionalidade deste módulo é realizar o *parser* de mensagens de acordo com o padrão definido para estrutura da mensagem

e a estrutura de dados da linguagem de programação adotada e vice-versa.

Módulo *Dispatcher*. Possui a função de discernir o tipo de solicitação realizada pela aplicação cliente. Após verificada a natureza da solicitação, é responsável por disparar a execução dos módulos que contemplam as respectivas funcionalidades esperadas: módulo `Provedor de Informações da Rede` - quando receber (1) solicitações de saída requisitando envio de dados - ou módulo `Provisionamento de Circuitos` - quando receber (2) solicitações de entrada requisitando ações serem executadas. Adicionalmente, é responsável por controlar a execução do módulo instanciado. Caso a operação solicitada seja executada com sucesso em um dos módulos supracitados, o módulo *Dispatcher* encaminha a mensagem de retorno para o módulo `APIs Mensagem` para ser entregue a aplicação cliente, caso contrário, uma mensagem de exceção é encaminhada a este mesmo módulo.

Módulo Provedor de Informações da Rede. Responsável por definir e organizar a estrutura das mensagens de solicitação de saída do Op3nControl. No processo de exportação das mensagens, o Op3nControl deverá consultar dados persistidos em um Banco de Dados (BD) e utilizá-los como fonte de dados para a composição destas mensagens. Para isso, este módulo requisita o módulo `Controle de Armazenamento`. A especificação do *framework* Op3nControl determina um grupo de mensagens para serem implementadas, as mesmas estão disponíveis na Sub-Seção 4.2.2.

Módulo Controle de Armazenamento. Responsável por controlar o acesso a base de dados Op3nControl. Este módulo recebe requisições para armazenamento de dados dos módulos `Coleta de Elementos de Rede e Estatística` e `Integração de Switches Legados` com o objetivo de armazenar informações a respeito dos elementos de comutação e dos *links* e suas estatísticas. O Op3nControl define que este deverá ser otimizado para estruturas de dados voltada para redes de computadores conectados. Portanto, deve ser utilizado Banco de Dados para grafos, uma vez que redes de computadores podem ser representadas, facilmente, em forma desta estrutura de dados. Neste mapeamento, os elementos de comutação serão os nodos e os *links* os enlaces de um grafo.

O banco de dados em grafo está relacionado ao padrão *Not only SQL* (NoSQL), um conceito moderno de responder a questões de armazenamento específicas, em outras palavras, é o resultado de pesquisas em relação a busca de soluções alternativas ao modelo de banco de dados relacional [Vicknair et al., 2010].

A recomendação de utilização de banco de dados orientados a grafos é baseada em pesquisas de usabilidade e desempenho em manipulações de dados dessa natureza (nós conectados). Os resultados expostos em Angles and Gutierrez [2008] e Vicknair et al. [2010] comprovam a eficácia do BD NoSQL, em sua versão para estrutura de dados para grafos, quando comparado a um BD relacional.

Uma das principais funcionalidades deste módulo é criar uma abstração em relação a implementação de banco de dados, ou seja, deve disponibilizar um conjunto de métodos, funções ou procedimentos bem definidos e suficientes para que solicitações

de gravação e leitura dos dados por parte de outros módulos sejam bem sucedidas. Fornece uma "fachada"de ações para os demais módulos. Internamente, essa abstração é conseguida através da execução da transformação da estrutura de dados em questão, ou seja, transformar dados do modelo de objetos para modelo de persistência de grafos e vice-versa. O armazenamento em estrutura de grafos possibilitará uma fácil navegabilidade entre os nodos e seus enlaces. As informações persistidas podem ser disponibilizadas ao usuário ou ainda ser fonte de informações para tomadas de decisões (por exemplo, algoritmos de roteamento) de outros módulos.

Módulo Coleta de Elementos de Rede e Estatística. Uma das principais atribuições deste módulo é interagir com o estruturas de dados (classes que armazenam nodos e enlaces) que armazenam a topologia de rede no controlador *OpenFlow* e recuperar dados referentes a elementos de comutação e seus respectivos *links*. Essas informações são encaminhadas para o módulo `Controle de Armazenamento` a fim de serem reorganizadas e persistidas em banco de dados, de acordo com a especificação do Op3nControl.

O domínio dos dados relacionados aos elementos de comutação (nós) será composto por: identificação (*datapathid*), quantidade de portas disponíveis e a natureza do equipamento (distinguir entre um elemento de comutação com *OpenFlow* habilitado de forma nativa, ou um elemento de comutação com *OpenFlow* habilitado através do módulo `Integrador de` *`Switches`* `Legados`.

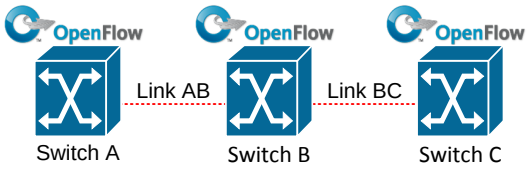


Figura 6: Elementos de Comutação em Linha

Após a etapa em que os nodos e enlaces já são conhecidos, o módulo executa requisições de estatísticas para cada uma das extremidade dos *links* encontrados, ou seja, para cada *link* serão realizadas duas consultas para os elementos de comutação conectados por si, em especial, para as portas em uso destes elementos de comutação na formação do *link* em questão. A Figura 6 apresenta um cenário em que seriam retornados três elementos de comutação (*switches*) `A`,`B` e `C` e dois *links* `AB` (porta 1 do *Switch* `A` com porta 2 do *Switch* `B`) e `BC` (porta 3 do *Switch* `B` e porta 1 do *Switch* `C`). Portanto, deverão ser realizadas solicitações de dados de estatísticas para cada tupla composta por *Switch* e porta, ou seja, quatro requisições.

As informações de estatísticas solicitadas deverão ser baseadas em requisições *OpenFlow* do tipo *Read State Messages*, implementadas através de `OFPT_STATS_REQUEST` [OpenFlowSpecv1.0, 2013]. Essas mensagens pertencem a classe *Controller-to-Switch Messages*. Dentre os sub-tipos possíveis deste tipo de mensagem, serão utilizadas a requisição `OFPST_PORT` e a requisição `OFPST_QUEUE`. O conteúdo da resposta da requisição do sub-tipo `OFPST_PORT` deverá conter [OpenFlowSpecv1.0, 2013]:

- número de pacotes recebidos e transmitidos;
- número de *bytes* recebidos e transmitidos;
- número de pacotes descartados pelo receptor e transmissor ;
- número de erros no receptor e transmissor;
- número de erros de alinhamento de quadro;
- número de erros CRC;
- número de colisões detectadas.

Os dados contidos na listagem acima deverão ser organizados e enviados ao módulo `Controle de Armazenamento` para sua correta persistência, desta forma, poderão ser recuperados e utilizados quando necessário.

Módulo Cálculo de Caminho. Módulo responsável por determinar o melhor caminho para conectar dois equipamentos de comutação, em outras palavras, fornecerá um caminho sobre infraestrutura de redes para a conexão de terminais. A inteligência deste módulo no processo de cálculo de caminho é dada pela utilização de algoritmos para este fim. Esse módulo deve ser flexível em relação ao desenvolvimento desses algoritmos. Ou seja, o módulo não restringe quais algoritmos devam ser implementados, e, sim, oferece um ambiente para o desenvolvimento dos mesmos. Fica claro que este módulo apresenta duas faces: (1) oferecer ambiente de desenvolvimento de algoritmos de cálculo de caminhos e (2) coordenar a execução desses algoritmos. Poderão ser desenvolvidos desde algoritmos de grafos simples, até algoritmos mais robustos que possam vir a utilizar dados dinâmicos, coletados pelo módulo `Coleta de Elementos de Rede e Estatística` e disponibilizados no banco de dados. Ou seja, o Op3nControl fornece uma API para desenvolvimento de novos algoritmos de encaminhamento.

Além disso, este módulo deverá ser implementado de modo que um utilizador desta instância do Op3nControl consiga, facilmente, escolher entre as opções dos algoritmos de busca de caminho implementados. O padrão de saída da execução de um algoritmo de descoberta de caminho, deve ser estruturado e definido de acordo com a linguagem de programação definida na implementação do *framework*, como *Array, Struct, List, Collection.*

Exemplificando, no cenário ilustrado através da Figura 7, caso fosse solicitado encontrar o melhor caminho (baseado em número de saltos, por exemplo) entre os *hosts* `X` e `Y` o retorno seria o conjunto `{A,B,C}`, nesta ordem. Cada elemento do conjunto apresentado é um comutador que faz parte do caminho encontrado entre os extremos.

Módulo Provisionamento de Circuito . Módulo que recebe solicitação de usuário para a criação de um circuito virtual (CV) sobre a infraestrutura de redes composta de comutadores *OpenFlow*. A criação dos CVs pode ocorrer a partir de dois modos: usuário informando o caminho desejado ou Op3nControl realizando o cálculo de

caminho de forma automatizada. A primeira maneira é trivial, em que o usuário é responsável por determinar quais comutadores e portas pertencerão ao CV. Já na segunda maneira, este módulo requisita o módulo `Cálculo de Caminho`, passando os nodos (e portas) extremos como parâmetros de busca e, recebe, como resultado, um conjunto de nodos, com suas respectivas portas, ordenados integrando os mesmos. De posse deste conjunto de comutadores (em ambos os casos), este módulo criará mensagens *OpenFlow* para cada um desses nodos, e as enviará para os respectivos elementos de comutação da infraestrutura de redes. O resultado será a criação de um CV sobre esses equipamentos de rede que possibilitará a comunicação de dois elementos finais da rede (*hosts*). Essa explicação pode ser traduzida graficamente através da Figura 7 que mostra a conexão do *host* `X` com o *host* `Y`.

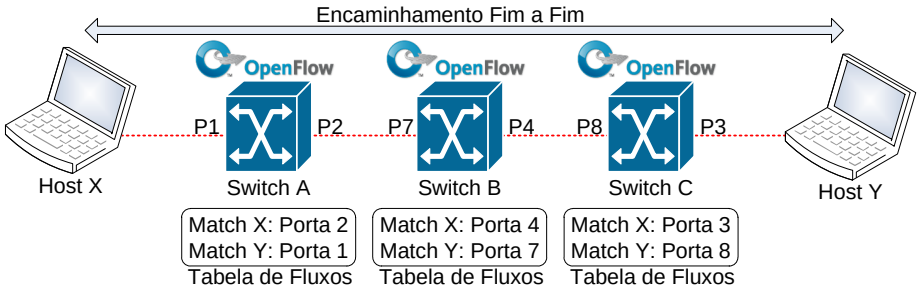


Figura 7: Circuito Virtual Configurado via *OpenFlow*

As mensagens *OpenFlow* utilizas por este módulo são do tipo *Modify State Messages*, implementadas através da mensagem `OFPT_FLOW_MOD`; pertencem a classe *Controller-to-Switch Messages* [OpenFlowSpecv1.0, 2013]. Essas mensagens `OFPT_FLOW_MOD` criarão `FLOW_ENTRIES` (entrada de fluxo) nas tabelas de fluxo dos elementos de comutação que, por sua vez, serão responsáveis por instruir esses equipamentos em relação ao correto encaminhamento de pacotes.

Em cada entrada de fluxo *OpenFlow* estão disponíveis: campos de cabeçalho, contadores e ações. Os campos de cabeçalho são as informações *OpenFlow* nas quais ocorre a verificação se o pacote que está passando pelo elemento de comutação atende a determinada regra da entrada de fluxo, ou seja, se atende ao *Match*, na nomenclatura *OpenFlow*. Caso ocorra o *Match*, os contadores daquela entrada de fluxo são atualizados e o pacote é submetido às ações determinadas pela aplicação ou operador de rede.

Para a criação dos CVs através do *OpenFlow*, o Op3nControl realiza a aplicação de dois grupos de regras de encaminhamento: *Match* para pacotes por endereçamento *IP* e *Match* por endereçamento *MAC*. A prioridade maior deve ser atribuída para a primeira regra, uma vez que o *Match* por endereço *IP* realizará o encaminhamento de pacotes com *Ethertype* dos pacotes IPV4; enquanto que a segunda é fundamental para o controle dos protocolo *Address Resolution Protocol* (ARP). O controle de `ARP` evita que sejam propagadas solicitações deste tipo para toda rede *broadcast*, pois apenas serão transmitidas as requisições `ARPs` nos comutadores onde as regras

de encaminhamento estiverem sido configuradas. Uma outra opção para o controle de `ARP` poderia ter sido realizado através da implementação de aplicação no controlador *OpenFlow*, mas, no escopo deste trabalho, o controle de `ARP` continuará a ser utilizado conforme sua estrutura padrão nos equipamentos de rede, contudo delimitada ao encaminhamento via *OpenFlow*.

A Figura 7 ilustra a comunicação dos *hosts* `X` e `Y` (para efeito de simplificação e clareza foi suprimido o controle de `ARP`). Os *switches* `A`, `B` e `C` realizam o tratamento dos fluxos de dados correntes verificando suas respectivas tabelas de fluxos e, caso satisfeito determinado *Match*, os pacotes serão encaminhados segundo *Action* definida na entrada de fluxo, no caso a *Output*, que significa redirecionamento do fluxo de pacotes para determinada porta.

O Op3nControl prevê que sejam implementadas, além de mensagens da criação dos circuitos virtuais, mensagens para exclusão de regras, disponibilizando os recursos em sua capacidade integral para novos experimentos.

Módulo de Integração de *Switches* Legados . Módulo que adiciona capacidade extra ao Op3nControl. Possibilita a integração com a solução de *datapath OpenFlow* Virtual, chamado *LegacyFlow*, apresentado por Farias et al. [2012]. Através desta integração, será possível adicionar ao escopo de equipamentos suportados pelo Op3nControl aqueles ditos legados, isto é, sem suporte nativo ao *OpenFlow*. Dois aspectos relevantes são: (1) quando houver conexão de um *datapath OpenFlow* ao Op3nControl, este deverá ser capaz de discernir o tipo desse comutador (legado ou *OpenFlow* nativo); e (2) aplicar as regras *OpenFlow* de modo dinâmico de acordo com as interfaces de comunicação definidas pelo *LegacyFlow* [Farias et al., 2012]. Para o atendimento do aspecto (1) deve ser alterado ou estendido a aplicação de controle da topologia do controlador *OpenFlow* com objetivo de comportar a informação do tipo do comutador.

De posse dessa integração, alguns novos cenários de uso podem ser atendidos pelo Op3nControl. A Figura 8 ilustra a possibilidade de criar um circuito virtual contendo equipamentos legados integralmente controlado via *OpenFlow*.

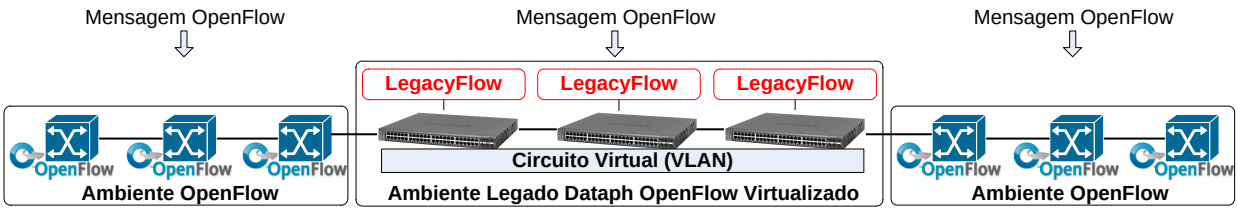


Figura 8: Circuito Virtual Contendo Equipamentos Legados Configurado via *OpenFlow*

Este módulo cria uma camada de gerenciamento e controle único sobre uma rede de comutadores de diferentes fabricantes e natureza (com *OpenFlow* nativo e legados (sem *OpenFlow* nativo). Em outras palavras, permite o acréscimo de um nível de abstração que torna toda a infraestrutura de rede passível de gerenciamento e controle padronizado via *OpenFlow*.

Módulo Gerenciamento de QoS . Módulo que adiciona capacidade extra ao Op3nControl (não é obrigatório para circuitos virtuais simples) de configurar parâmetros de qualidade de serviço em CVs. Possibilita a integração com a solução de *datapath QoSFlow* Virtual apresentado por Ishimori et al. [2010]. Através dessa integração será possível utilizar interfaces de criação de classes de fluxos, mensagens de configuração das disciplinas de fila através de aplicações de controle de redes *OpenFlow*, além de controle de tráfego.

Para possibilitar o desenvolvimento desta integração devem ser implementadas as *APIs QoSFlow* no Op3nControl, descritas em Ishimori et al. [2010]. O desenvolvimento de APIs deve levar em consideração dois aspectos: (1) extensão do protocolo *OpenFlow* para suportar as mensagens de configuração esperadas pelo *QoSFlow*; (2) desenvolvimento de interfaces de controle das filas dos comutadores. Este módulo permitirá que o usuário escolha as seguintes disciplinas de fila: *First In, First Out* (FIFO), *Hierarchy Token Bucket* (HTB), *Random Early Detection* (RED), *Stochastic Fairness Queueing* (SFQ). A partir destas implementações, o Op3nControl será provido de capacidade de configuração desses escalonadores de pacotes que controlam o envio de pacotes nos elementos de comutação.

Na implementação da solução *QoSFlow*, o *datapath QoSFlow*, quando iniciado, configura a disciplina de fila HTB e cria uma fila padrão FIFO [Ishimori et al., 2010]. O *datapath* fica aguardando a configuração de outras classes e/ou reconfiguração desta fila padrão. Nesse momento, o Op3nControl entra em ação enviando as mensagens de QoS implementadas neste módulo.

Portanto, será possível, por exemplo, criar um circuito virtual com largura de banda controlada, isto é, com definição da largura de banda (*rate limiting*) para os fluxos. Assim, poderão ser relacionados a capacidade total dos recursos e utilização da banda nos circuitos. Consequentemente, poderá ser quantificada a capacidade ociosa dos *links*, por exemplo. Esse conceito de utilização de recursos de rede de forma isolada e controlada pode ter seu conceito expandido em termos de fatias de virtualização com garantias em nível de *hardware*, essa funcionalidade é similar a *rate limiting* com VLAN, contudo aplicada em cenário SDN.

4.2.2 Aspectos de Comunicação

A comunicação entre os módulos presentes na especificação da arquitetura do Op3nControl ocorrerá através da invocação de estruturas de código (instância de classes, métodos, funções, procedimentos) em um escopo local de acordo com a linguagem de programação a ser adotada para a implementação (instanciação) do Op3nControl.

Por outro lado, a comunicação do Op3nControl com os elementos de comutação, através do extremo sul (*southbound*), deve ocorrer através de mensagens *OpenFlow*. Já a comunicação do *framework* através do extremo norte (*northbound*), com as aplicações clientes, deve ocorrer por meio de mensagens em formato de serviços *web*, independentes de tecnologia e de fácil manipulação. O *framework* recomenda a implementação de mensa-

gens de serviços *web* (*webservices*) através de SOAP ou REST). A estrutura da mensagem deve ser XML ou JSON. As interfaces de comunicação disponibilizadas pelo Op3nControl devem ser bidirecionais, ou seja, deverá haver tanto interfaces de saída quanto interfaces de entrada de dados no *framework*. As mensagens referentes a *northbound* que são disponibilizadas:

- **Tipo: Saída. Mensagem 1 - Elementos de Comutação.** Ao receber esta solicitação, o Op3nControl deverá fornecer os elementos de comutação que estiverem a si conectados assim como suas respectivas informações (disponíveis na Sub-Seção 4.2.1 - módulo `Coleta de Elementos de Rede e Estatística`);
- **Tipo: Saída. Mensagem 2 - *Links* entre Elementos de Comutação.** Ao receber esta solicitação, o Op3nControl deverá fornecer os enlaces (*links*) entre os elementos de comutação que estiverem a si conectados assim como suas respectivas informações (disponíveis na Sub-Seção 4.2.1 - módulo `Coleta de Elementos de Rede e Estatística`);
- **Tipo: Entrada. Mensagem 1 - Provisionar Circuito Digital.** Ao receber esta requisição, o Op3nControl deverá executar ações necessárias para a criação de circuitos digitais;
- **Tipo: Entrada. Mensagem 2 - Configurar Classe em Fila.** Ao receber esta requisição, o Op3nControl deverá executar ações de criação de mensagens *OpenFlow* através da API *QoSFlow* relacionada a criação de classe (detalhes na Sub-Seção 4.2.1 - módulo `Gerenciamento de QoS`)) e enviá-las aos elementos de comutação a fim de serem configuradas filas em suas portas de comunicação;
- **Tipo: Entrada. Mensagem 3 - Configurar Disciplina de Fila.** Ao receber esta requisição, o Op3nControl deverá executar ações de criação de mensagens *OpenFlow* através da API *QoSFlow* relacionada a configuração de disciplina de fila (detalhes na Sub-Seção 4.2.1 - módulo `Gerenciamento de QoS`)) e enviá-las aos elementos de comutação a fim de serem configuradas filas em suas portas de comunicação;
- **Tipo: Entrada. Mensagem 4 - Provisionar Circuito Digital com Parâmetros de QoS.** Ao receber esta requisição, o Op3nControl deverá executar ações necessárias para a criação de circuitos digital com determinadas características de QoS;
- **Tipo: Entrada. Mensagem 5 - Provisionar Circuito Digital em Elemento de Comutação Legado.** Ao receber esta requisição, o Op3nControl deverá executar ações necessárias para a envio de mensagem *OpenFlow* ao *datapath* virtual *LegacyFlow*. Espera-se que essa mensagem seja descontinuada após ajustes na interface de integração entre o *LegacyFlow* [Farias et al., 2012] e o Op3nControl;

- **Tipo: Entrada. Mensagem 6 - Remover Entradas de Fluxos.** Ao receber esta requisição, o Op3nControl deverá criar e enviar mensagens de exclusão de entradas de fluxos armazenados nos elementos de comutação previamente configurados para o estabelecimento de circuitos;
- **Tipo: Entrada. Mensagem 7 - Remover Classes de QoS.** Ao receber esta requisição, o Op3nControl deverá criar e enviar mensagens de exclusão de classes que armazenam os escalonadores de fila nos elementos de comutação previamente configurados para o estabelecimento de circuitos virtuais com parâmetros de QoS.

4.3 Funcionalidades do Op3nControl

Esta Seção tem como objetivo listar e descrever sucintamente as funcionalidades essenciais que devem ser providas pelas implementações do modelo de *framework* Op3nControl. Neste conjunto, existem as funcionalidades que têm interação com o usuário, iniciadas através de comandos de usuário, e outras que não têm acesso a manipulações dos usuário. Estas últimas, pertencem ao grupo de funcionalidades que criam a estrutura funcional da solução, ou seja, agem em "*background*"com objetivo de criar o alicerce funcional do modelo que possibilita que as funcionalidades com interação com o usuário sejam executadas com sucesso. A seguir são listadas as funcionalidades elencadas no modelo:

Obter dados dos elementos de rede. Devem ser obtidos dos dados dos elementos da infraestrutura de redes. Esses elementos podem ser do tipo:

- Elementos de comutação como *swicthes*, roteadores ou outros equipamentos de rede com *OpenFlow* nativamente habilitado;
- *Links* entre os equipamentos de comutação com *OpenFlow* nativamente habilitado;
- Elementos de comutação como *swicthes*, roteadores ou outros equipamentos de rede com *OpenFlow* habilitado através do *LegacyFlow*.

Coletar estatísticas dos elementos de rede. Devem ser coletadas estatísticas de portas de comutadores que estejam conectados com outro comutador remoto. As estatísticas coletadas serão referentes a portas, filas e descrição de fabricante dos comutadores.

Transformação e persistência de dados em modelo de grafos. Os dados de elementos de rede e dados estatísticos deverão ser persistidos em estrutura de dados de grafos. Podem para servir de fonte de informação para consultas gerais e fontes de tomadas de decisão. Os seguintes mapeamentos fazem-se necessários:

- Dispositivos como *swicthes*, roteadores ou outros equipamentos de rede com *OpenFlow* habilitado serão os nós;

- *Links* entre os equipamentos de rede serão os enlaces.

Disponibilização de informações da infraestrutura de redes. O Op3nControl é capaz de exportar mensagens para aplicação cliente sobre os dados coletados anteriormente.

Calcular caminho. Funcionalidade provida pelo modelo que possibilita que seja determinado um caminho entre dois extremos da rede de comutadores de acordo com o algoritmo definido pelo usuário (deve ser previamente implementado no *framework*).

Gerar circuitos dinâmicos entre dois extremos da rede . Funcionalidade provida pelo Op3nControl que habilita o encaminhamento de pacotes sobre um CV entre dois extremos da rede de comutadores.

Gerar circuitos dinâmicos entre dois extremos da rede com QoS . Funcionalidade provida pelo Op3nControl que habilita o encaminhamento de pacotes sobre um CV entre dois extremos da rede de comutadores com tratamento de QoS, que incluem:

- Controle de Tráfego (*Traffic Shapping*);
- Configuração de escalonadores de pacotes dos comutadores de rede.

Integrar redes legadas ao *OpenFlow* . O Op3nControl permite que sejam gerenciados equipamentos legados via protocolo *OpenFlow*. Cria uma camada de abstração que permite o gerenciamento e controle da infraestrutura em sua totalidade.

Entre as funcionalidades do Op3nControl que terão interação com o usuário, merecem destaque (pois fornecem um serviço completo ao usuário): "Gerar circuitos dinâmicos entre dois extremos da rede"e "Disponibilização de informações da infraestrutura de redes".

De modo simplificado, tendo a premissa que os dados de elementos de comutação *OpenFlow* estão persistidos no banco de dados, o diagrama de sequência ilustrado na Figura 9 mostra a integração dos módulos para atender a funcionalidade "Gerar circuitos dinâmicos entre dois extremos da rede". Nesse caso serão repassados dois parâmetros (origem e destino) pela aplicação cliente, encapsulados em um objeto pelo módulo de "API Mensagem", e este, é encaminhado para o módulo "Provisionamento de Circuito"pelo módulo "Dispatcher". O módulo "Provisionamento de Circuito"requisita o caminho ao módulo "Cálculo de Caminho"que por sua vez necessita de informações persistidas em banco de dados para realizar tal ação.

Já na Figura 10 é exibido o diagrama de sequência da funcionalidade "Disponibilização de informações da infraestrutura de redes". O módulo "API Mensagem"recebe a requisição e encaminha os dados ao módulo "Dispatcher", este que faz o encaminhamento para o módulo "Provedor de Informações da Rede"que busca dados persistidos em banco de dados, os organiza e envia o resultado desejado.

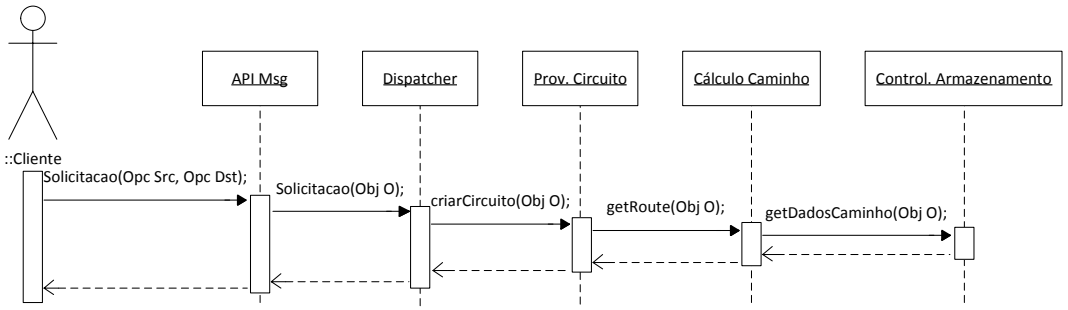


Figura 9: Diagrama de Sequencia da Funcionalidade "Gerar circuitos dinâmicos entre dois extremos da rede´´

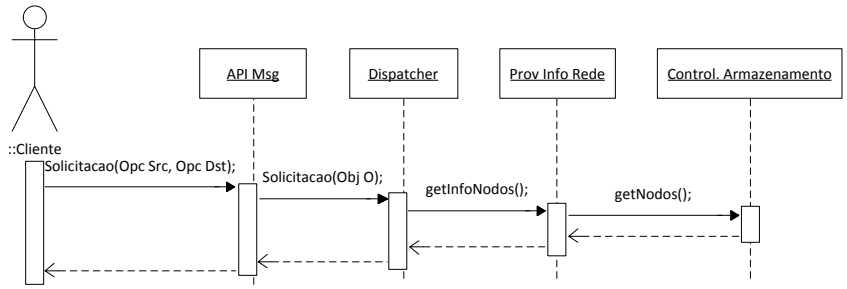


Figura 10: Diagrama de Sequencia da Funcionalidade "Disponibilização de informações da infraestrutura de redes´´

4.4 Implementação da Proposta Op3nControl

Nesta seção será descrita a implementação realizada a partir da especificação de *framework* Op3nControl. Para realizar a implementação da aplicação Op3nControl fez-se necessário definir as tecnologias e padrões conforme previsto na especificação de arquitetura (Seção 4.2).

Como linguagem de programação, foi utilizado Java (JDK7). Como controlador *OpenFlow*, foi adotado o *Floodlight* v.085 [Floodlight, 2013], a biblioteca *OpenFlow* utilizada é mesma encapsulada no próprio *Floodlight*, similar a biblioteca *OpenFlowJ* [OpenFlowJ, 2013].

A solução técnica escolhida para banco de dados utilizado foi o Neo4J que é classificado como um banco de dados do padrão `NoSQL` e trabalha com estrutura de dados de grafos. A escolha desta solução foi baseada em pesquisas bibliográficas, ratificando a percepção de flexibilidade e robustez da mesma, como atesta Vicknair et al. [2010]. O modelo de funcionamento dentro do *framework* Op3nControl é *"embedded"*, ou seja, será acoplado internamente a aplicação Op3nControl. A outra opção possível era criar um "Neo4J *Server*"e trabalhar com requisições remotas (atualmente suporta REST) ao mesmo através de chamadas da aplicação. A opção pelo modelo "*embedded*"ocorreu em

função da busca por maior desempenho.

O módulo de `Cálculo de Caminho` foi implementado seguindo as características definidas pela especificação da arquitetura do Op3nControl, ou seja, é flexível em relação ao desenvolvimento de novos algoritmos de encaminhamento. No escopo da solução atual foram implementados os algoritmos de encaminhamento de menor caminho e *Dijkstra*. Os mesmos se utilizam de APIs do Neo4J para a descoberta dos caminhos. É importante ressaltar que, seguindo diretrizes do modelo, a implementação Op3nControl disponibiliza diversas APIs que fornecem estatísticas da rede atualizadas a cada vinte segundos. As informações retornadas por estas APIs estão detalhadas na descrição do módulo `Coleta de Elementos de Rede e Estatística` do Op3nControl. Com isso, a plataforma está pronta para o desenvolvimento de novos algoritmos mais robustos e inteligentes e que utilizem as características da rede para definir caminhos na infraestrutura de rede.

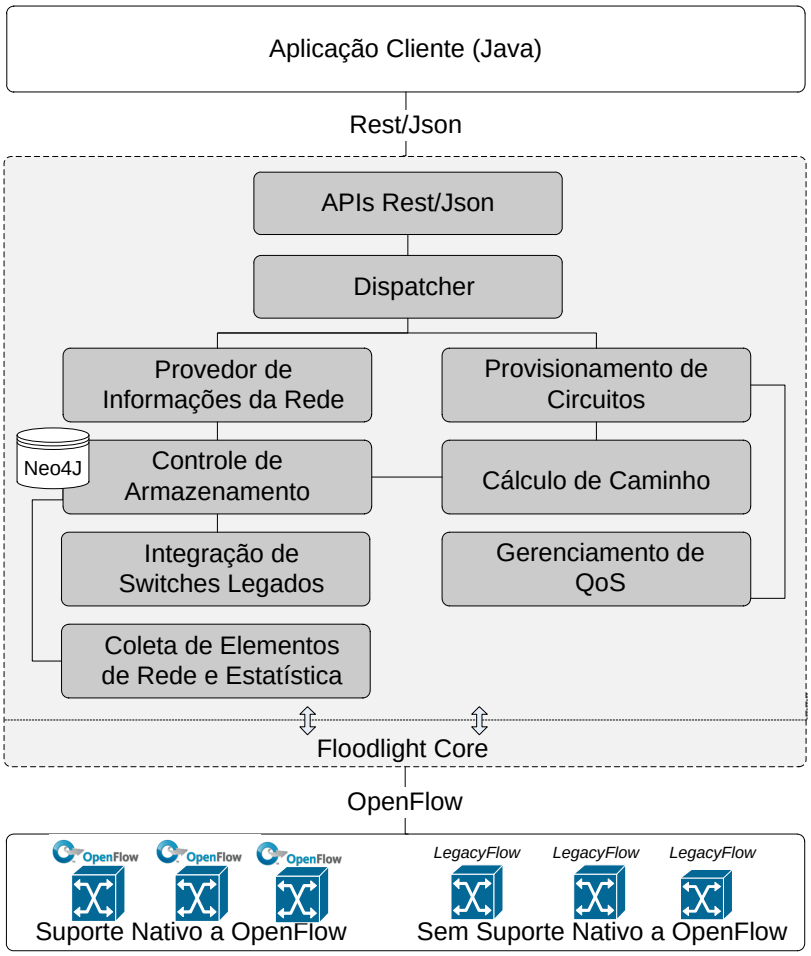


Figura 11: Arquitetura da Implementação do Op3nControl

A Figura 11 ilustra a arquitetura da implementação do Op3nControl realizada neste trabalho. Em relação aos aspectos de comunicação, os processos internos do *framework* utilizarão chamadas da própria linguagem de programação, neste caso, instanciação de classes e chamadas de métodos; as mensagens da extremidade norte terão formato JSON e utilizarão tecnologia REST; já para o atendimento das mensagens da

extremidade sul foram desenvolvidos métodos que geram, enviam e recebem mensagens *OpenFlow*, por exemplo: `OFPT_STATS_REQUEST` com tipo `OFPST_PORT`, `OFPST_QUEUE` e `OFPST_DESC` para a solicitação de dados estatísticos e informações gerais; a coleta desse dados é realizada com o tratamento da mensagem `OFPT_STATS_REPLY`; outra mensagem muito utilizada é `OFPT_FLOW_MOD` com estrutura de *Match* e *Actions* diversas.

O módulo de Gerenciamento de Qualidade de Serviço foi disponibilizado através da implementação de APIs para comunicação com o *datapath QoSFlow* [Ishimori et al., 2010]. Para isso realizou-se uma extensão do protocolo *OpenFlow*, criando-se a mensagem *OpenFlow* `QUEUE_DISCIPLINE`, além de diversas classes que permitem a criação de mensagens de configuração do *datapath* de QoS. Duas importantes funcionalidades garantidas na integração do Op3nControl com o *QoSFlow* são o controle de banda, ou *rate limiting*, e a possibilidade de configuração das disciplinas de filas da portas dos elementos comutadores de rede.

A integração do Op3nControl com *swicthes* legados através do *datapath* virtual *LegacyFlow* ocorreu com sucesso. O Op3nControl implementa uma verificação do tipo de equipamento conectado através do disparo de mensagens `OFPT_STATS_REQUEST` do tipo `OFPST_DESC`. Ao receber a resposta, a implementação do Op3nControl busca pelo campo do tipo "char"`hw_desc` que informará se o comutador é um equipamento legado ou com *OpenFlow* nativo. A partir desta diferenciação é possível gerenciar esse *switch* de forma a atender necessidades específicas de cada tipo de *datapath*. Por exemplo, a criação de mensagem do tipo `OFPT_FLOW_MOD` para o *LegacyFlow* terá estrutura pouco diferente da usual quando o *framework* solicita a criação de mensagem do tipo `OFPT_FLOW_MOD` para um comutador *OpenFlow* tradicional.

Esta aplicação será utilizada em cenários de experimentos a fim de prover informações acerca de validação funcional e testes de desempenho.

4.5 Conclusões do Capítulo

O Op3nControl, um modelo de *framework* para criação de circuitos digitais baseados em *OpenFlow*, foi descrito neste capítulo. A especificação de sua arquitetura e seus módulos foram detalhados, entre eles os módulos "Integrador de Equipamentos Legados"e "Gerenciamento de QoS"que possibilitam, respectivamente, configurar parâmetros de qualidade de serviço nos equipamentos de rede de forma dinâmica e integrar *switches* legados (sem suporte nativo a *OpenFlow*) a um gerenciamento *OpenFlow*.

Além disso, foram expostos os aspectos gerais de comunicação entre os módulos do Op3nControl e, destes com entidades externas ao seu contexto, através de suas pontes norte e sul. Por fim, foi exposta uma implementação do modelo que será a base para os resultados de validação e desempenho deste trabalho. No próximo capítulo, é realizada a avaliação da proposta Op3nControl.

CAPÍTULO 5

Avaliação de Resultados

Este capítulo tem o objetivo de validar a proposta do *framework* Op3nControl. Para atingir tal objetivo foi realizada a implementação desta proposta, descrita no capítulo anterior. Para isso são realizados diversos testes sobre a aplicação Op3nControl, executando seus principais requisitos funcionais com objetivo de aferir métricas que atestem a eficácia e eficiência da proposta.

A metodologia a ser utilizada para a avaliação de resultados será descrita na Seção 5.1. Além disso, são listados e detalhados os cenários de experimentação na Seção 5.2, incluindo informação da especificação técnica dos recursos utilizados; em seguida, são apresentados e discutidos os resultados aferidos nas execuções dos cenários de experimentação na Seção 5.3; e, por fim, a Seção 5.4 conclui o capítulo.

5.1 Metodologia dos Experimentos

A fim de validar a proposta de *framework* elaborado no contexto deste trabalho foi escolhida a opção de construir um ambiente experimental (*testbed*). Para a disponibilização da rede *OpenFlow* foram utilizados equipamentos `TP-Link 1043ND` com *OpenFlow* habilitado. Estes comutadores foram configurados para executar tanto o *datapath OpenFlow* original quanto o *datapath OpenFlow* com suporte ao *QoSFlow*. Sendo assim, para controlar e gerenciar essa rede experimental, utilizou-se a implementação do Op3nControl.

De posse do cenário supracitado, os testes foram conduzidos sobre um *testbed* com nove comutadores (do tipo `TP-Link 1043ND`), tendo a grande vantagem de oferecer resultados similares a um ambiente *OpenFlow* de grande escala. A única exceção é relacionada ao poder de processamento reduzido no processo de tratamento dos pacotes via

OpenFlow desses equipamentos `TP-Link 1043ND`, em comparação com equipamentos de alta desempenho e dedicados para este fim. De forma inversamente proporcional, a taxa de transmissão entre dois *hosts* extremos reduz a medida que são acrescentados comutadores em série. Contudo, este detalhe não compromete, em fator algum, os resultados obtidos, uma vez que não é foco deste trabalho testar a desempenho de processamento *OpenFlow* dos equipamentos de comutação.

Além disso, a fim de serem analisados aspectos de escalabilidade da solução foi adicionado um cenário de realização de testes com um *software* capaz de simular uma rede *OpenFlow* virtual, o *Mininet* [Lantz et al., 2010]. Para melhor organização dos experimentos, foram criados Cenários e Sub-Cenários de experimentação.

5.2 Cenário dos Experimentos

A fim de melhor organizar os experimentos foram criados Cenários e Sub-Cenários para a execução dos experimentos. Em cada um deles foi elaborado um ou mais "itens de medição", que são unidades de avaliação dos cenários. Esses "itens de medição"relacionam-se com os requisitos funcionais do Op3nControl.

Com objetivo de avaliar a implementação das funcionalidades relevantes da arquitetura do Op3nControl, foram elencados os seguintes cenários de testes e seus respectivos itens de medição:

- **Cenário 1.** Topologia em Linha com Comutadores `TP-Link 1043ND`
  - **Sub-Cenário 1a** Circuitos Virtuais Simples
    1. Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches*
    2. Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces
    3. Intervalo de Tempo para Cálculo de Caminho
    4. Intervalo de Tempo para Envio de Mensagens *OpenFlow*
    5. Intervalo de Tempo para Criação de Circuito Virtual (CV)
  - **Sub-Cenário 1b.** Circuitos Virtuais com QoS
    1. Intervalo de Tempo para Envio de Mensagens *OpenFlow* com Definição de QoS
    2. Intervalo de Tempo para Criação de Circuito *com* Definição de QoS
  - **Sub-Cenário 1c.** Avaliação de Taxas de Transmissão nos CVs
    1. Comparação da Taxa de *Bits* Recebidos nos CVs
- **Cenário 2.** Circuitos Virtuais Simples em Topologia com *Links* Redundantes com comutadores `TP-Link 1043ND`
  - Comparação entre Intervalo de Tempo para Cálculo de Caminho de Diferentes Topologias de Rede

- **Cenário 3.** Topologia em Linha com comutadores virtualizados através do *Mininet*
  - Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches* (*Mininet*)
  - Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces (*Mininet*)
  - Intervalo de Tempo para Cálculo de Caminho (*Mininet*)
  - Intervalo de Tempo para Envio de Mensagens *OpenFlow* (*Mininet*)
  - Intervalo de Tempo para Criação de Circuito (*Mininet*)
- **Cenário 4.** Avaliação de Operações do Módulo Integrador de *Switches* Legados
  - Intervalo de Tempo de Operações de Criação de VLANs
- **Cenário 5.** Avaliação Comparativa entre Implementação do Op3nControl e *CircuitPusher*
  - Comparação de desempenho para Criação de Circuitos

Grande quantidade dos experimentos ocorrem sobre topologia de rede em linha, visando simplificação e objetividade em aferir, de acordo com o incremento de número de *switches* envolvidos, a desempenho ou capacidade de itens de medição elaborados. Utilizam essa topologia os Cenários 1, 3, 4 e 5. O Cenário 1 busca trazer informações gerais acerca do processo de criação de circuito virtual, com e sem configuração de QoS; o Cenário 3 objetiva verificar a escalabilidade da implementação Op3nControl; já no Cenário 4, busca-se mensurar a desempenho de operações envolvendo a integração com o *datapath virtual LegacyFlow* [Farias et al., 2012]; por fim, o Cenário 5 traz uma análise comparativa da proposta deste trabalho com uma implementação de funcionalidade semelhante, o *CircuitPusher* [Kanui and Parraga, 2013].

A fim de complementar o escopo dos experimentos será mostrado no Cenário 2 uma comparação de desempenho do Op3nControl em uma topologia de rede com *links* redundantes entre os *switches*.

A forma de como calcular os resultados da maioria dos Cenários seguiu uma única estratégia. Com exceção do `Sub-Cenário 1c` - Avaliação de Taxas de Transmissão nos Circuitos Virtuais - os demais Cenários e Sub-Cenários utilizaram os seguintes procedimentos:

- Configurados laços de repetição que executaram cem (100) ciclos de repetição de cada Cenário ou Sub-Cenário;
- Para cada item de aferição foram gerados registros em arquivos de texto disponibilizando os tempos de execução;
- Com esses resultados, calculou-se a média aritmética e o desvio padrão das amostragens;

- Utilizou-se nível de confiança de 95% para determinação dos respectivos intervalos de confiança;
- Plotagem dos resultados em gráfico.

Devido a natureza do experimento ser diferente, em que o objetivo é mostrar as taxas de *bits* recebidos em uma comunicação entre dois terminais, os resultados do Sub-Cenário 1c - Avaliação de Taxas de Transmissão nos Circuitos Virtuais - foram extraídos fazendo-se uma relação direta entre número de *bits* recebidos *versus* tempo. Na Sub-Seção 5.3.1.3 mais detalhes são fornecidos. A seguir são apresentadas os especificações técnicas dos equipamentos utilizados nos cenários dos experimentos.

5.2.1 Especificações Técnicas dos Equipamentos Utilizados

Para a realização dos experimentos, os seguintes equipamentos e *softwares* foram envolvidos:

Notebook HP Pavilion dv5-2040br. *Intel Core* I3-350M, 4GB *RAM*, 500GB HDD, 64 *bits*. Executa *Linux Ubuntu* 12.04.1 LTS, *Kernel 3.2.0-32-generic-pae*. Suporta os seguintes itens presentes nos cenários de experimentação:

- Implementação Op3nControl;
- Cliente que realiza invocação de serviços (*Rest/Json*) do Op3nControl;
- Máquina Virtual *VirtualBox* v4.1.12 (Imagem *Linux* com *Mininet*) .

Netbook Asus Eee PC 900. *Intel Celeron M353 900 MHz (Cache 512k L2),Chipset Intel 910/915, 1GB DDR2*, Memória *Flash* de 4GB, 32 bits. Executa *Ubuntu* 11.04, *Kernel* 2.6.38-15-*generic*. Suporta o seguinte item presente nos cenários de experimentação:

- Clientes (*hosts*) conectados nas extremidades dos CVs (executam aplicações que verificam conectividade, como, `Ping` e `Iperf`).

Ultimate Wireless N Gigabit Router TL-WR1043ND. 1 Interface WAN e 4 Interfaces LAN 10/100/1000Mbps *Ports*, 1 USB 2.0 *Port*. Executa *OpenWRT* 2.6.32.27 (*Backfire* 10.03.1) com *OpenFlow*. Suporta o seguinte item presente nos cenários de experimentação:

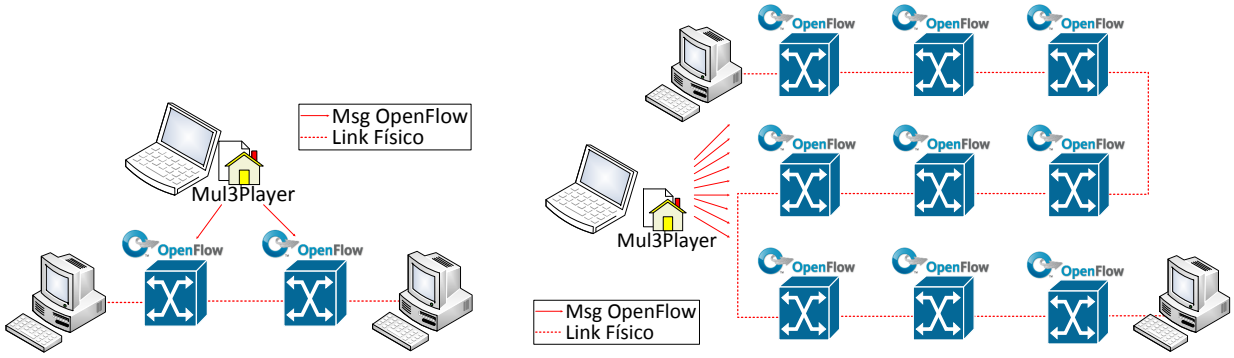
- Comutadores (*switches*) *OpenFlow*.

5.3 Avaliação dos Experimentos

Nesta Seção são descritas as características, condições e resultados de testes de cada Cenário de experimentação realizado neste trabalho.

5.3.1 Cenário 1. Topologia em Linha com Comutadores `TP-Link 1043ND`

Neste cenário de experimentação foram utilizados dois, três, quatro, cinco, sete e nove comutadores do tipo `TP-Link 1043ND` com *OpenFlow* habilitado. A disposição e conexão entre esses equipamentos são baseados em topologia em linha. Através da Figura 12, é possível visualizar duas variações de topologias desse cenário, elas correspondem aos dois extremos do intervalo de comutadores informado. Em 12(a) está ilustrado a variação de Cenário com dois *switches* e, em 12(b), está ilustrado uma outra variação deste Cenário com nove *switches*; as demais variações da topologia em linha desse Cenário são similares, bastando-se alterar o número de *switches* envolvidos.



(a) Topologia em Linha com 3 *Switches* (b) Topologia em Linha com 9 *Switches*

Figura 12: Topologia de Rede Cenário 1

Os Sub-Cenários descritos a seguir utilizam um sub-conjunto, ou todas as seis variações de topologia: dois, três, quatro, cinco, sete e nove *switches*. Os Sub-Cenários 1a (Sub-Seção 5.3.1.1) e 1b (Sub-Seção 5.3.1.2) utilizam todas as seis variações para a execução de seus experimentos. Já o Sub-Cenário 1c (Sub-Seção 5.3.1.3) utiliza apenas a variação de topologia em linha com nove *switches*, conforme ilustra a Figura 12(b).

5.3.1.1 Sub-Cenário 1a. Circuitos Virtuais Simples

O objetivo deste Sub-Cenário de testes é atestar que as funcionalidades de coleta de dados de elementos de comutação, *links*, coleta de estatísticas, cálculo de caminho e criação de circuito virtual estão de acordo com as funcionalidades descritas na especificação do Op3nControl e estão retornando os resultados desejados. Para isso, os itens de medição elencados foram:

- Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches*;
- Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces;
- Intervalo de Tempo para Cálculo de Caminho;

- Intervalo de Tempo para Envio de Mensagens *OpenFlow*;
- Intervalo de Tempo para Criação de Circuito Virtual (CV).

Os dois primeiros itens da listagem "Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches*"e "Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces", quando somados, representam o intervalo de tempo para o Op3nControl descobrir a topologia da rede *OpenFlow* e armazenar seus respectivos dados no repositório de dados local. Esses resultados são importantes para saber por quanto tempo o banco de dados estará inativo (sofrendo atualização) e para determinar qual estratégia utilizar para aumentar a disponibilidade dos serviços *web* que utilizam essas informações de topologia.

Neste sub-cenário foram executados cem (100) ciclos de repetições para cada variação de topologia em linha, ou seja, com dois, três, quatro, cinco, sete e nove *switches*.

O primeiro item de medição é o "Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches*". O atendimento desta funcionalidade qualifica a implementação do Op3nControl quanto a necessidade de descobrir os comutadores da rede e armazenar os dados relevantes referentes aos mesmos. O tempo de "pesquisa"é o tempo de coleta de dados dos *switches* junto ao controlador *OpenFlow Floodlight*; já o tempo de "persistência"é o somatório das ações de transformação de dados de formato objeto para a estrutura de dados do banco de dados do Neo4J, somado ao tempo de chamada de gravação por parte da API deste BD. Este item de medição está vinculado aos módulos `Coleta de Elementos de Rede e Estatística` e `Controle de Armazenamento`.

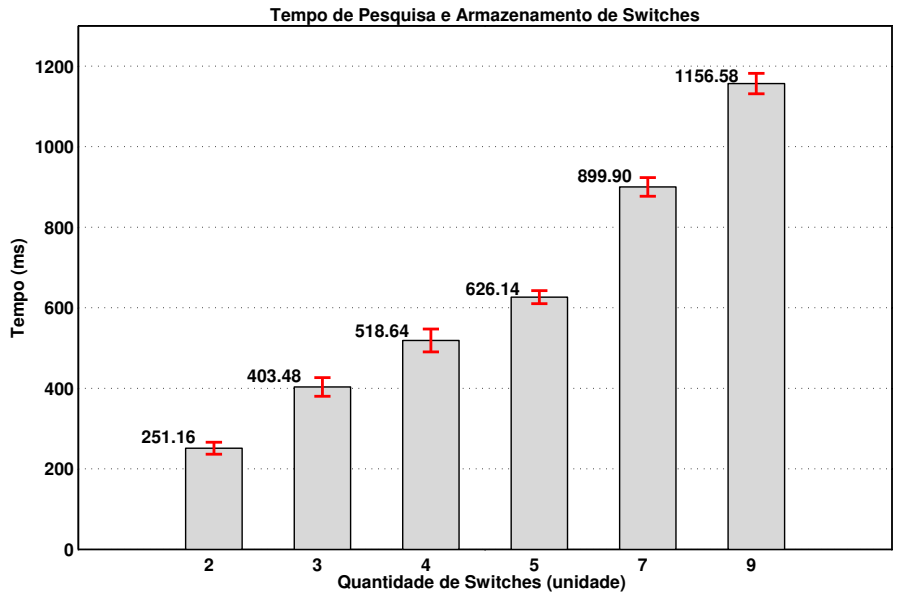


Figura 13: Intervalo de Tempo para Pesquisa e Persistência de dados de Switches

Na Figura 13 estão disponíveis os resultados desse primeiro item de medição. A mesma mostra o tempo, em milissegundos, de operação para cada variação da topologia de testes (dois, três, quatro, cinco, sete e nove *switches*). Para dois *switches* o tempo de

processamento foi de 251.16 milissegundos, enquanto que, no outro extremo, para nove *switches*, o tempo de processamento foi de 1.16 segundos.

O próximo item de medição é o "Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces", o qual tem como objetivo descobrir os tempos de execução dos processos de busca dos *links* da rede e armazenamento dos dados referentes aos mesmos. Seu funcionamento é similar ao item anterior (relacionado a *switches*). O tempo de "pesquisa"é o tempo de coleta de dados de *links* junto ao controlador *OpenFlow Floodlight*; já o tempo de "persistência"é o somatório do ações de transformação de dados de formato objeto para a estrutura de dados do banco de dados do *Neo4J* somado ao tempo de chamada de gravação por parte da *API* deste BD. Este item de medição está vinculado aos módulos `Coleta de Elementos de Rede e Estatística` e `Controle de Armazenamento`.

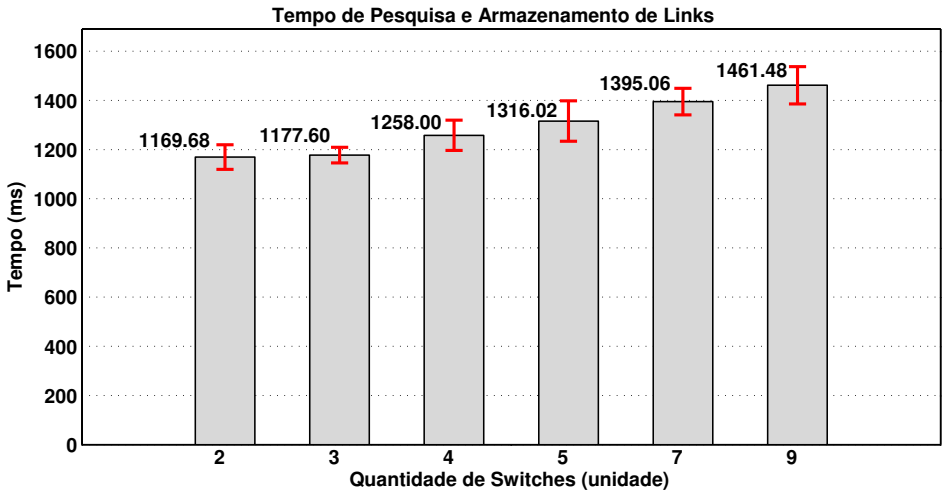


Figura 14: Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces

Os resultados desse segundo item de medição, podem ser visualizados na Figura 14. Cada barra representa uma variação da topologia dos experimentos (dois, três, quatro, cinco, sete e nove *switches*). Os resultados mostram os tempos medidos para as ações de pesquisa e persistência dos enlaces, na unidade de milissegundos. Para dois *switches* o tempo de processamento foi de 1.17 segundos, enquanto que, no outro extremo, para nove *switches*, o tempo de processamento foi de 1.47 segundos.

Se analisado do ponto de vista do maior número de comutadores (nove), a soma dos resultados disponibilizados na Figuras 13 e 14 resultam em 2.62 segundos. Este tempo corresponde ao tempo para atualização dos dados da topologia da rede *OpenFlow* no Op3nControl, sendo que, uma parcela é para a descoberta da topologia e a outra para a atualização do banco de dados (ficará inativo para consultas). Como a cada 20 segundos (definido na implementação) o Op3nControl atualiza seus dados de topologia, os 2.62 segundos representam 13.1% de período de indisponibilidade (por atualização de dados). Para esses casos, quando uma aplicação cliente requisita o serviço durante um período de atualização do banco de dados, foi implementada uma rotina de tratamento que "segura"a requisição até que o banco de dados esteja atualizado, mantendo a disponibilidade do

serviço.

Outra percepção sobre os resultados das Figuras 13 e 14 é a diferença de elevação de tempo ao incrementar o número de comutadores. Os resultados da Figura 14 demonstram que a variação de tempo para atualização dos enlaces é menor quando comparado aos resultados de atualização de comutadores (Figura 13). O motivo é que para a descoberta dos nodos há um tempo de mensagens de conexão (`OFPT_HELLO`) - *handshaking* entre o comutador e o controlador *OpenFlow* - enquanto que, para a atualização de dados dos *links*, existe a premissa que todos os nós já estejam conectados, consequentemente, menos suscetível a variações de tempo.

Com objetivo de medir o desempenho da funcionalidade de calcular o melhor caminho entre dois extremos da rede, na implementação do Op3nControl é utilizado o item de medição "Intervalo de Tempo para Cálculo de Caminho". O cálculo de caminho produz, como retorno, um conjunto de nós ordenados entre dois *hosts* na rede. Para atingir este objetivo, foi adotado o algoritmo do menor caminho (menor número de saltos), implementado na API *Neo4J*. A outra opção disponível seria o algoritmo de *Dijkstra*, contudo, atribuindo-se pesos padrões, o resultado seria o próprio menor caminho. Esse item de medição corresponde ao tempo de processamento da API *Neo4J* para execução de algoritmo de busca sobre estrutura de dados criada com o Op3nControl. Este item de medição está vinculado ao módulo `Cálculo de Caminho`. Esta aferição é importante para saber o desempenho do algoritmo desenvolvido.

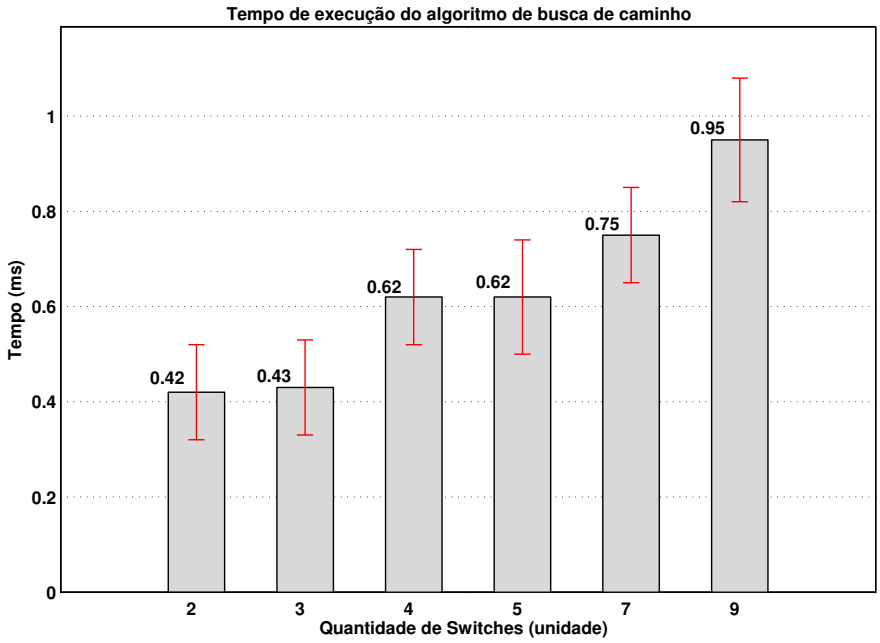


Figura 15: Intervalo de Tempo para Cálculo de Caminho

Na Figura 15 estão disponíveis os resultados desse terceiro item de medição. A mesma mostra o tempo, em milissegundos, de operação para cada variação da topologia (dois, três, quatro, cinco, sete e nove *switches*). Para dois *switches* o tempo de processamento foi de 0.42 milissegundos, enquanto que, no outro extremo, para nove *switches*,

o tempo de processamento foi de 0.95 milissegundos. Esses resultados servirão de base de comparação para os novos algoritmos de cálculo de caminho desenvolvidos sobre o Op3nControl. Além disso, esse resultado é utilizado na comparação com respostas obtidas no Cenário 2, em que o Op3nControl é submetido a gerenciar uma topologia de rede mais complexa.

"Intervalo de Tempo para Envio de Mensagens *OpenFlow*"é o item de medição que qualifica a implementação do Op3nControl quanto as necessidades de (1) criação de mensagens *OpenFlow*, conforme padrão definido em OpenFlowSpecv1.0 [2013], e (2) envio das mesmas para a rede de comutadores *OpenFlow*. As mensagens *OpenFlow* são criadas de acordo com o resultado do módulo de "Cálculo de Caminho", em que, para cada elemento de comutação, são criados dois pares de mensagens *OpenFlow* a fim de encaminhar pacotes na rede. O primeiro par possui *Match* nos cabeçalhos de endereço IP (fluxo de dados) e o segundo par realiza *Match* com endereço *MAC* (protocolo *ARP*). A configuração das regras em pares é relacionada a necessidade de transmissão bidirecional nos CVs. Esse item de medição é essencial para medir o tempo incrementado quando se utiliza os módulos de QoS, Sub-Cenário 1b, Sub-Seção 5.3.1.2.

O envio das mensagens *OpenFlow* é realizado utilizando *APIs* do *core* do controlador *OpenFlow Floodlight*. As mensagens *OpenFlow* utilizadas são do tipo `OFPT_FLOW_MOD`, contendo ações do tipo `Output`. Ou seja, nas tabelas de fluxos dos elementos de comutação que receberem as mensagens, são criadas entradas de fluxo que verificam se há *Match* nos cabeçalhos *OpenFlow* com os pacotes de rede, caso positivo, esses fluxos serão submetidos a ação `Output`, ou seja, serão encaminhados para uma porta específica. Este item de medição corresponde a um sub-conjunto de funcionalidades do módulo "Provisionamento de Circuito".

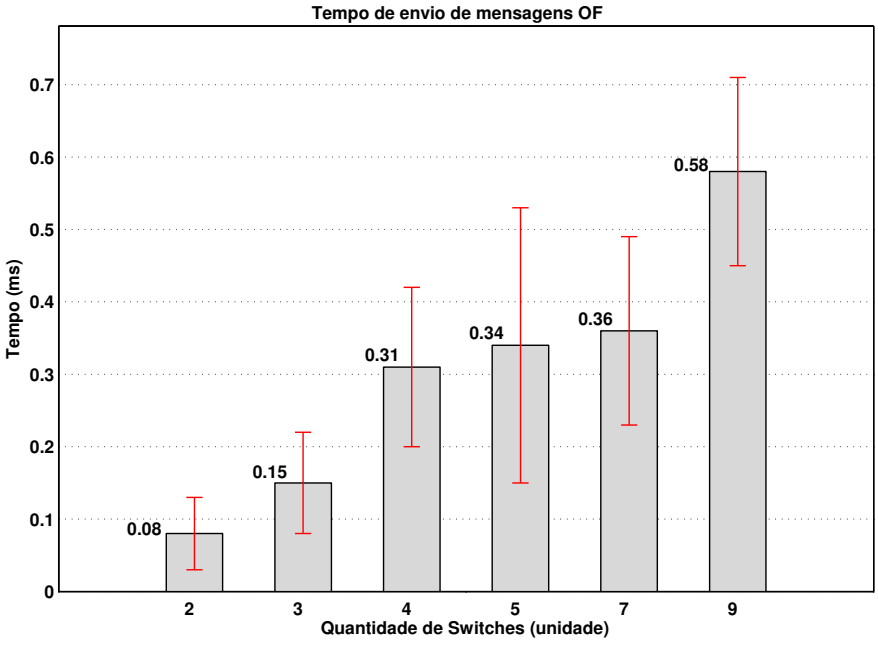


Figura 16: Intervalo de Tempo para Envio de Mensagens *OpenFlow*

Na Figura 16 estão disponíveis os resultados do item de medição "Intervalo de Tempo para Envio de Mensagens *OpenFlow*". É importante ressaltar que o tempo medido de envio de mensagens *OpenFlow* corresponde ao momento que o Op3nControl dispara as mesmas para os comutadores, não é mensurado o tempo que um *datapath* necessita para tornar as entradas de fluxos funcionais (controlar efetivamente o tráfego do fluxo de dados na rede). Os resultados são expostos através de intervalos de tempo, em milissegundos, de operação para cada variação da topologia (dois, três, quatro, cinco, sete e nove *switches*). Para dois *switches* o tempo de processamento foi de 0.08 milissegundos, enquanto que, no outro extremo, para nove *switches*, o tempo de processamento foi de 0.58 milissegundos. Esses resultados serão comparados aos resultados do Sub-Cenário 1b, Sub-Seção 5.3.1.2, estes que adicionam a criação e envio de mensagens de QoS, além do *OpenFlow* tradicional.

Por fim, o quinto item de medição é o "Intervalo de Tempo para Criação de Circuito Virtual (CV)". Esta medição comprova o atendimento da funcionalidade de criação de CVs através do protocolo *OpenFlow* e, sobretudo, é capaz de fornecer dados de desempenho. Este experimento representa o somatório dos últimos dois intervalos de tempo anteriormente descritos (cálculo de caminho, criação e envio de mensagens *OpenFlow*), mais os tempos de processamento dos módulos de "API Rest/Json"e "Dispatcher", que correspondem ao atendimento da requisição *REST* e realização dos devidos *parsers* do formato *JSON* para objeto. Há de se considerar o tempo de comunicação interna do Op3nControl entre as classes dos módulos envolvidos.

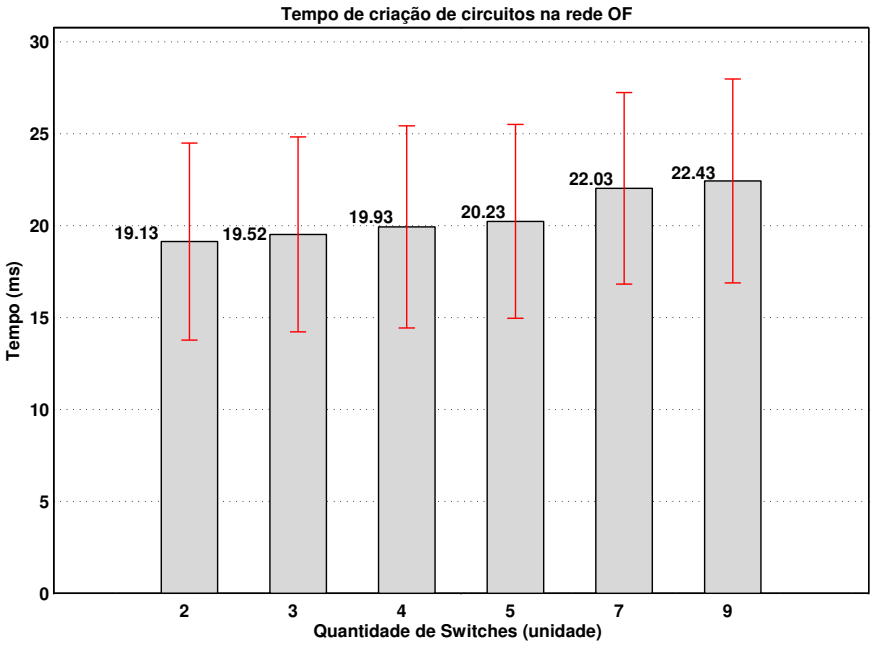


Figura 17: Intervalo de Tempo para Criação de Circuito

Na Figura 17 estão disponíveis os resultados desse quinto item de medição. A mesma mostra o tempo, em milissegundos, de operação para cada variação da topologia (dois, três, quatro, cinco, sete e nove *switches*). Para dois *switches* o tempo de processamento foi de 19.13 milissegundos, enquanto que no outro extremo, para nove *switches*, o tempo de processamento foi de 22.43 milissegundos.

O valores elevados dos intervalos de segurança dos resultados dos itens de medição "Intervalo de Tempo para Cálculo de Caminho"e "Intervalo de Tempo para Envio de Mensagens *OpenFlow*"estão relacionados a heterogeneidade das amostras, sobretudo devido aos resultados representarem unidades bastante reduzidas (poucos milissegundos).

Com objetivo de determinar o nível de desempenho do Op3Control sobre os itens de medição deste Cenário, a métrica de tempo de criação de circuito virtual é comparada na execução da aplicação *CircuitPusher* no Cenário 5, Sub-Seção 5.3.5. Os itens de medição explorados englobam diversas funcionalidades do *framework*. A seguir, no Sub-Cenário 1b, são verificados os tempos para criação de circuito virtual com QoS.

5.3.1.2 Sub-Cenário 1b. Circuitos Virtuais com QoS

O objetivo deste Sub-Cenário de testes é atestar que as funcionalidades de coleta de dados de elementos de comutação, *links*, coleta de estatísticas, cálculo de caminho e criação de circuito virtual com qualidade de serviço (QoS) estão de acordo com os requisitos e estão retornando os resultados desejados.

Este Sub-Cenário é similar ao Sub-Senário 1a, Sub-Seção 5.3.1.1, no entanto, ao invés do cliente requisitar a criação de CV comum, é solicitado a criação do mesmo com parâmetros de QoS. A Tabela 5 mostra a relação dos itens de medição entre os sub-cenários 1a e 1b. Os três primeiros itens da tabela, por representarem funcionalidades idênticas em ambos os módulos, não serão aferidos novamente, uma vez que os resultados seriam idênticos, nas mesmas condições de testes. Portanto, apenas os itens marcados com (*) terão valores diferentes entre os cenários 1a e 1b e terão seus dados aferidos.

Tabela 5: Relacionamento Sub-Cenários 1a e 1b

Item de Medição	Sub-Cenários	
	1a	1b
Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches*	Idêntico	
Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces	Idêntico	
Intervalo de Tempo para Cálculo de Caminho	Idêntico	
Intervalo de Tempo para Envio de Mensagens *OpenFlow**	1a	1b
Intervalo de Tempo para Criação de Circuito Virtual (CV)*	1a	1b

Neste Sub-Cenário, similarmente ao Sub-Cenário 1a, foram executados cem (100) ciclos de repetições para cada variação de topologia em linha, ou seja, com dois, três, quatro, cinco, sete e nove *switches*.

O item de medição de "Intervalo de Tempo para Envio de Mensagens *OpenFlow*", para esse Sub-Cenário, comporta dois grupos de mensagens *OpenFlow*: (1) as mensagens *OpenFlow* tradicionais e (2) mensagens de configuração de parâmetros de QoS. Este segundo grupo utiliza a API de integração com o *datapath QoSFlow*, implementada no Op3nControl. O atendimento desta funcionalidade qualifica a implementação do

Op3nControl quanto a necessidade de criação de mensagens *OpenFlow* (padrão), criação de mensagens *OpenFlow* para QoS, e envio das mesmas para a rede de comutadores *OpenFlow*. As mensagens *OpenFlow* são criadas para os comutadores elencados no resultado do módulo de "Cálculo de Caminho". Em relação as mensagens *OpenFlow* padrão, para cada elemento de comutação, são criados dois pares de mensagens *OpenFlow* a fim de encaminhar pacotes com *Match* nos cabeçalhos dos protocolos `IP` e `ARP` em ambos os sentidos (fluxo bidirecional).

Em relação as mensagens de QoS, para cada elemento, são enviadas mensagens que possibilitam fazer controle de banda (*rate limiting*) e configurar a disciplina de fila na porta do comutador. Neste experimento foram criadas classes com escalonadores de pacote do tipo *First-In-First-Out* (FIFO). O envio das mensagens *OpenFlow* é realizado utilizando APIs do *core* do Controlador *Floodlight*. As mensagens *OpenFlow* utilizadas são do tipo `OFPT_FLOW_MOD`, contendo ações do tipo *ENQUEUE*. Ou seja, nas tabelas de fluxos dos elementos de comutação que receberem as mensagens, serão criadas entradas de fluxo que verificam se há *Match* nos cabeçalhos *OpenFlow* com os pacotes de rede, caso positivo, esses fluxos serão submetidos a ação *ENQUEUE*, através da qual serão encaminhados para uma classe com determinada disciplina de fila configurada na porta do comutador. Este item de medição corresponde a um sub-conjunto de funcionalidades do módulo "Provisionamento de Circuito"mais o módulo "Gerenciamento de QoS".

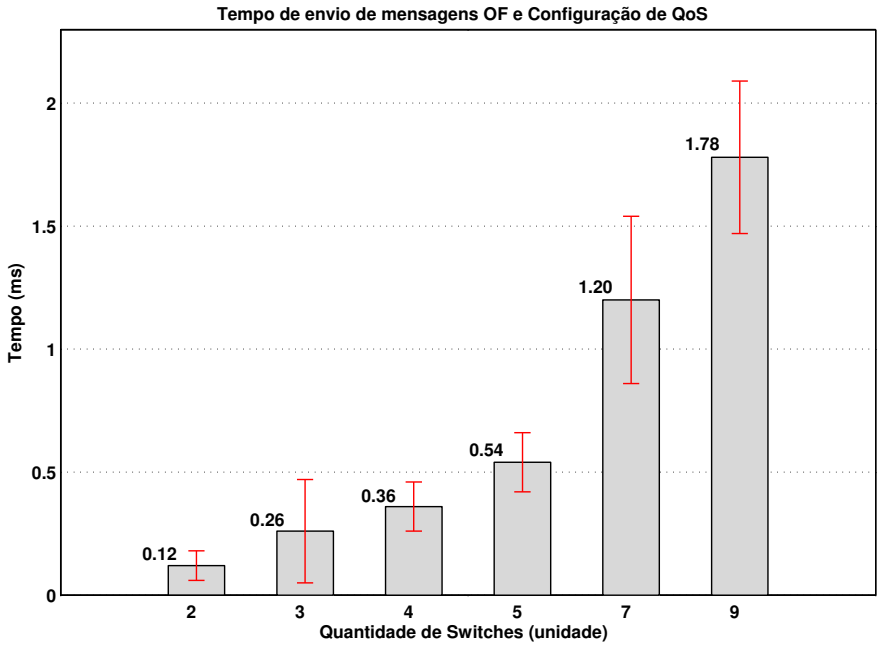


Figura 18: Intervalo de Tempo para Envio de Mensagens *OpenFlow* com Definição de QoS

Na Figura 18 podem ser visualizados os resultados do item medição Intervalo de Tempo para Envio de Mensagens *OpenFlow* deste Cenário. São mostrados os tempos, em milissegundos, de operação para cada variação da topologia (dois, três, quatro, cinco, sete e nove *switches*). No extremo com menor quantidade de *switches* o tempo de operação foi de 0.12 milissegundos. Já para nove *switches*, o tempo de operação foi de

1.78 milissegundos.

O próximo item de medição, "Intervalo de Tempo para Criação de Circuito", é composto das funcionalidades: cálculo de caminho, criação e envio de mensagens *OpenFlow*, criação e envio de mensagens *QoSFlow*, somados os tempos de execução dos módulos de "*API Rest/Json*"e "*Dispatcher*". Estes dois últimos módulos correspondem ao atendimento da requisição *REST* e realização dos devidos *parsers* do formato JSON para objeto. De modo similar ao Sub-Cenário 1a, é necessário somar ao resultado o tempo de comunicação interna do Op3nControl , isto é, o tempo que comunicação entre as classes dos módulos envolvidos. O atendimento desta funcionalidade qualifica a implementação do Op3nControl quanto a necessidade de criação de circuitos virtuais com QoS através do protocolo *OpenFlow*. Este item de medição corresponde as funcionalidades inerentes aos módulos `Provisionamento de Circuito` e `Gerenciamento de QoS`.

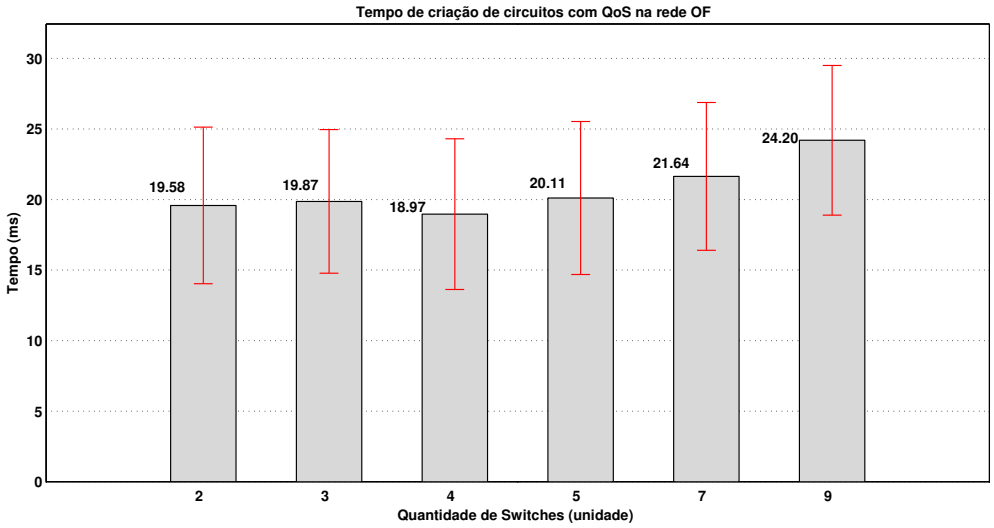


Figura 19: Intervalo de Tempo para Criação de Circuito com Definição de QoS

Os resultados do item medição "Intervalo de Tempo para Criação de Circuito"estão descritos na Figura 19. Os resultados mostram o intervalo de tempo, em milissegundos, de operação para cada variação da topologia (dois, três, quatro, cinco, sete e nove *switches*). Para dois *switches* o tempo de processamento foi de 19.58 milissegundos, enquanto que, no outro extremo, para nove *switches*, o tempo de processamento foi de 24.20 milissegundos. O resultado da medição de quatro *switches* chamou a atenção, pois apresentou valor (18.97) fora do crescimento esperado, pois sofreu decréscimo em relação ao aferição para três *switches*. Ao analisar analiticamente os resultados, foi percebido que este valor foi consequência das amostragens 30, 31, 33 e 34 (do total de 100), cujos intervalos de tempo foram extremamente baixos. Essa heterogeneidade das amostras reflete nos intervalos de confiança de maior amplitude. Portanto, se analisado o intervalo de confiança, este valor de 18.97 é considerado normal.

Os resultados expostos neste Sub-Cenário também apresentaram valores elevados para os intervalos de segurança dos dois itens de medição. A explicação se dá, da mesma forma do Sub-Cenário 1a, pela heterogeneidade das amostras, sobretudo devido

aos resultados representarem unidades bastante reduzidas (poucos milissegundos).

Os resultados dos itens de medição "Intervalo de Tempo para Envio de Mensagens *OpenFlow*"e "Intervalo de Tempo para Criação de Circuito"confirmam a expectativa de que os tempos nesse Sub-Cenário seriam superiores aos tempos apresentados no Sub-Cenário 1a, uma vez que foi necessário a criação e envio de mensagens *OpenFlow* adicionais, pois foi necessário realizar a configuração de parâmetros de QoS.

5.3.1.3 Sub-Cenário 1c. Avaliação de Taxas de Transmissão nos CVs

O objetivo deste Sub-Cenário de testes é atestar que o processo de criação de circuito virtual ocorreu com sucesso e o mesmo foi disponibilizado na infraestrutura dos comutadores de rede. Outro objetivo é atestar que a integração com o módulo `Gerenciamento de QoS` está de acordo com a especificação do Op3nControl, pois apenas em caso de sucesso, as técnicas de controle de tráfego poderiam ser utilizadas.

Nesse sentido, foram criados dois CVs, um com regras de QoS e o outro não. Sobre cada um desses CVs criados com a implementação da proposta Op3nControl, foi realizada a transmissão de dados do tipo *Transmission Control Protocol* (TCP), através da aplicação `Iperf`, envolvendo terminais conectados aos elementos comutadores. O `Iperf` foi executado duas vezes, a primeira execução ocorreu sobre o CV comum, e na segunda execução, sobre o CV com QoS criado com o escalonador do tipo *HTB*, que possibilita a realização de controle de banda (*rate limiting*). No experimento com QoS, foi configurado para que os comutadores limitassem a largura de banda para 2 Mbits/Seg.

Esse Sub-Cenário utilizou apenas uma variação da topologia em linha com nove *switches*, exposto na Figura 12(b). Para atender aos objetivos deste Sub-Cenário é utilizado o item de medição "Comparação da Taxa de *Bits* Recebidos nos CVs". Neste item de medição, ao executar a aplicação `Iperf`, um terminal foi configurado como cliente e o outro como servidor. Os experimentos tiveram duração de 120 segundos, as taxas foram medidas na unidade `Mbits/seg` e as visualizações (saídas) das mesmas foram configuradas para serem impressas no intervalo de 1 segundo, sendo assim, optou-se por representar graficamente esta relação direta entre a taxa de *bits* recebidos `x` intervalo de tempo.

Na Figura 20 são apresentados os resultados do item de medição deste Sub-Cenário. A linha vermelha (pontilhada) representa a taxa de *bits* recebidos (aferidos no lado servidor) no CV comum durante o intervalo de 120 segundos, ou seja, é a taxa máxima suportada na topologia de nove *switches* em linha, algo em torno de 9,3 *Mbits*/Seg. A taxa de transmissão é considerada máxima devido a característica do TCP de "crescer", caso haja banda suficiente, até encontrar seus limites (*treshold*). A linha em azul (contínua) representa a taxa de bits recebidos (aferidos no lado servidor) no CV com QoS realizando *rate limiting*. Durante o mesmo intervalo de 120 segundos, o tratamento do fluxo é facilmente perceptível, sendo que a taxa de *bits* recebidos respeita o limite de carga de 2 *Mbits*/Seg, configurado no momento de criação de regras de QoS. Portanto, desse modo, fica claro que a solução de QoS proposta pelo Op3nControl tem o funcionamento conforme

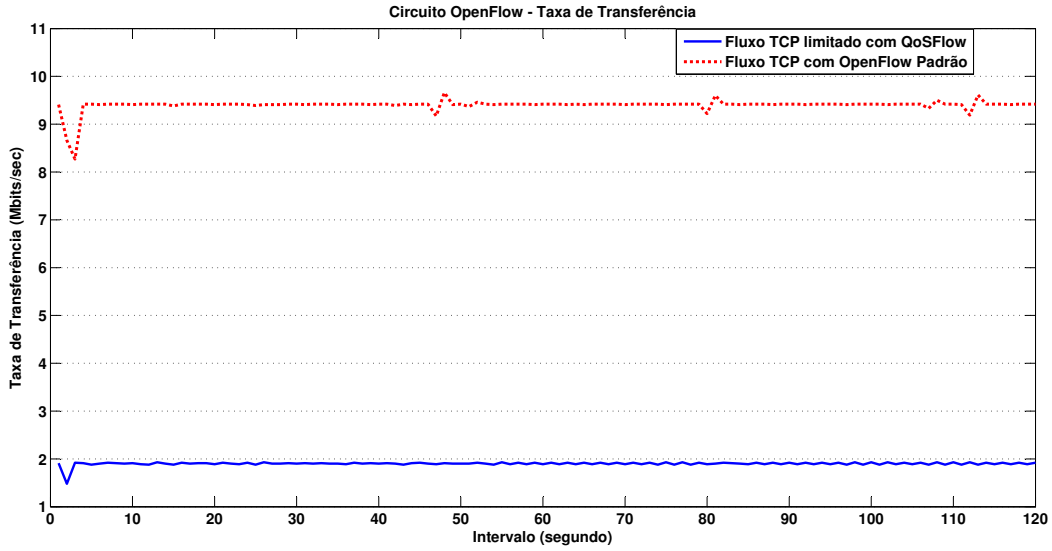


Figura 20: Controle da largura de banda (taxa de transmissão) no *TP-Link 1043ND*

o esperado em relação ao controle de tráfego.

5.3.2 Cenário 2. Circuitos Virtuais Simples com Topologia com *Links* Redundantes com comutadores TP-Link 1043ND

O Cenário 1, explicado na Seção 5.3.1, mostrou uma série de itens de medição em seus Sub-Cenários, sendo que todos foram conduzidos sobre uma topologia de redes em linha. Neste cenário, foram realizados testes com uma topologia distinta, que possui mais de um caminho para alguns nós da rede. O objetivo deste experimento é verificar o funcionamento da implementação do Op3nControl em condições de rede com este tipo de topologia. O objeto de verificação, ou seja, a fundamentação deste Cenário, é a criação de CVs sobre essa infraestrutura diferenciada. Nesse sentido, foi construída a topologia de rede mostrada na Figura 21, composta de comutadores do tipo *TP-Link 1043ND* com *OpenFlow* habilitado, que é a base para a realização dos experimentos, neste Cenário. Ressalta-se que, cada execução de experimento para este cenário, possuiu o número de *swicthes* alterado, similarmente ao Sub-Cenário 1a, Sub-Seção 5.3.1.1, com a seguinte regra: apenas os *swicthes* que possuíam conexões com outros comutadores que formavam um possível caminho entre os extremos eram iniciados.

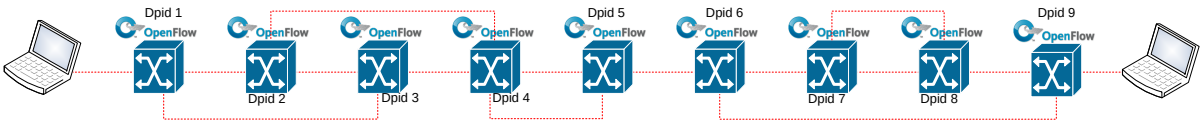


Figura 21: Estrutura do Experimento do Cenário 2

Por ter a fundamentação idêntica ao Sub-Cenário 1a, Sub-Seção 5.3.1.1, os mesmos itens de avaliação devem ser utilizados. Contudo, após um estudo sobre a arquitetura

do Op3nControl, percebeu-se que, devido sua estrutura modular, o impacto nos resultados seria restrito a apenas alguns itens de medição. Sendo assim, foi mapeado que apenas o tempo relacionado ao item de medição "Intervalo de Tempo para Cálculo de Caminho"teria alterações em seus valores aferidos entre os itens de medição do Sub-Cenário 1a. Este fato foi ratificado através da execução prática dos experimentos.

Com exceção do item de medição "Intervalo de Tempo para Cálculo de Caminho", os demais permaneceram idênticos, nas mesmas condições de testes, aos resultados do experimento de cenário 1a (Sub-Seção 5.3.1.1). A Tabela 6 mostra a relação dos itens de medição entre os Sub-Cenários 1a e o Cenário atual. No item de medição "Intervalo de Tempo para Criação de Circuito Virtual (CV)", marcado com (**), foi apresentada uma alteração pontual, contudo, a mesma representa fielmente a diferença do tempo de cálculo de caminho entre os Cenários (1a e atual), por esse motivo, ao encontrar o novo tempo de cálculo de caminho, esse item de medição torna-se automaticamente conhecido e, por isso, não terá seu resultado aferido. Portanto, apenas o item marcado com (*) terá valores diferentes entre os Cenários 1a e o Cenário atual, e terá seus dados .

Tabela 6: Relacionamento entre Sub-Cenário 1a e Cenário 2

Item de Medição	(Sub)Cenários	
	1a	2
Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches*	Idêntico	
Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces	Idêntico	
Intervalo de Tempo para Cálculo de Caminho*	1a	2
Intervalo de Tempo para Envio de Mensagens *OpenFlow*	Idêntico	
Intervalo de Tempo para Criação de Circuito Virtual (CV)**	Idêntico	

O item de medição "Intervalo de Tempo para Cálculo de Caminho Virtual (CV)", como exposto na Sub-Seção 5.3.1.1, visa qualificar a implementação do Op3nControl quanto a necessidade de calcular o melhor caminho entre dois extremos da rede (fornecidos pelo usuário) na solicitação de criação de um CV. Esse item de medição é o mesmo utilizado no Cenário 1. Também foi adotado o algoritmo do menor caminho (menor número de saltos), implementado na *API Neo4J*, para realizar os testes. A diferença nos valores dos resultados são referentes as condições mais complexas da rede de comutadores. Este item de medição corresponde ao tempo de processamento da *API Neo4J* para a estrutura de dados criada com o Op3nControl. Este item de medição está vinculado ao módulo `Cálculo de Caminho`.

Neste cenário, para atendimento do item de medição relacionado ao cálculo de caminho, foram executados cem (100) ciclos de repetições sobre a topologia representada na Figura 21, para buscas variando entre dois a cinco *swicthes* (quatro saltos na rede). As execuções dos testes e seus respectivos resultados encontrados, estão dispostos na Tabela 7, que contém informações sobre as buscas realizadas, por exemplo, foi solicitado encontrar o caminho entre a origem de `dpid` (*datapath id*) 1 e o destino *dpid* 3, logo o algoritmo

Tabela 7: Organização dos Experimentos do Cenário 2

Quantidade de Swicthes	Origem	Destino	Resultado
2	1	3	1,3
3	2	5	2,4,5
4	4	7	4,5,6,7
5	2	9	2,4,5,6,9

retorna uma lista com dois *switches* apenas {1,3} (primeiro registro da Tabela).

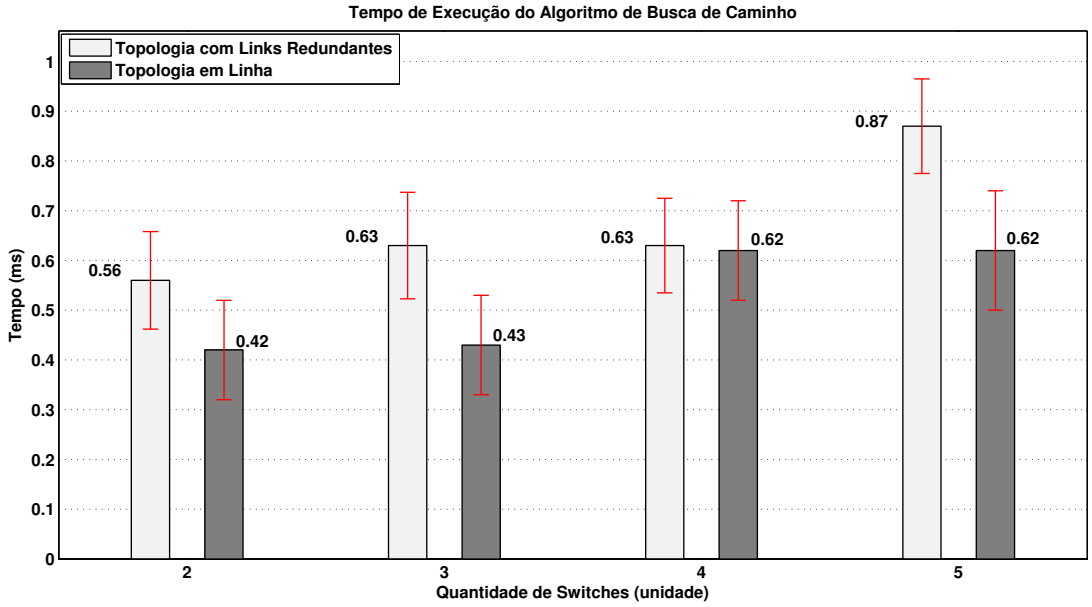


Figura 22: Comparação de Intervalo de Tempo para Cálculo de Caminho entre Diferentes Topologias

Na Figura 22, estão disponíveis os resultados do item de medição deste cenário. De fato, este item de medição é a mensuração do intervalo de tempo necessário para atingir os resultados mostrados na Tabela 7. A fim de facilitar a comparação do comportamento da implementação do Op3nControl sobre as diferentes topologias de rede, é disponibilizado um gráfico contendo informação de intervalo tempo - em milissegundos - de processamento do item de medição sobre ambas topologias.

Nesta Figura 22, há vários pares de barras, cada par representa a medição para dois, três, quatro e cinco *switches*. Dentro do par de barras, uma barra mostra os resultados do cenário atual e a outra mostra resultados do Sub-Cenário 1a, Sub-Seção 5.3.1.1. Nos resultados deste Cenário, para dois *switches*, o tempo de processamento foi de 0.56 milissegundos, enquanto que, no outro extremo, para cinco *switches*, o tempo de processamento foi de 0.87 milissegundos. O valor elevado dos intervalos de segurança está relacionado a heterogeneidade das amostras, sobretudo, devido aos resultados representarem unidades bastante reduzidas. Pode-se perceber que o tempo para calcular a rota em uma rede que contém *links* redundantes entre os nodos *switches* é superior se comparado

a rede com topologia em linha, uma vez que apresenta maior complexidade de cálculo e maior necessidade de verificações de caminho.

5.3.3 Cenário 3. Topologia em Linha com comutadores virtualizados através do *Mininet*

O objetivo deste cenário é atestar que a proposta Op3nControl é escalável para trabalhar com uma maior quantidade de comutadores *OpenFlow*. Para atender a este objetivo, foi utilizado o simulador *Mininet* [Lantz et al., 2010], capaz de criar uma rede virtual *OpenFlow* no próprio computador. A arquitetura dos testes deste cenário está exposto na Figura 23.

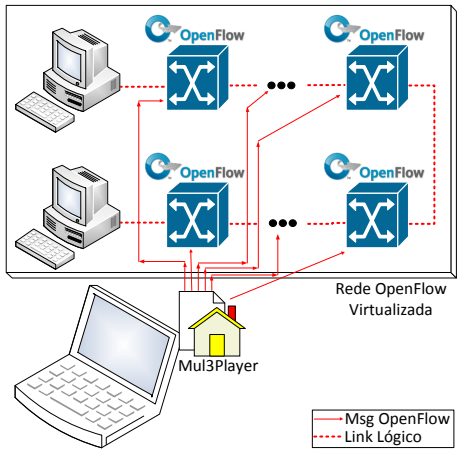


Figura 23: Arquitetura de Experimentação do Cenário 3

Neste Cenário, para a execução dos experimentos, as variações de topologias foram formadas por trinta, quarenta e cinquenta *switches* virtuais, criados com o *Mininet*. Nas três variações de topologia, todos equipamentos foram conectados utilizando topologia em linha. Para cada uma dessas três variações foram executados cem (100) ciclos de repetições.

A fim de comprovar escalabilidade da proposta são verificados e atestadas as funcionalidades especificadas na Sub-Cenário 1a, Sub-Seção5.3.1.1, que são: de coleta de dados de elementos de comutação, *links*, coleta de estatísticas, cálculo de caminho e criação de circuito virtual. Consequentemente, serão utilizados os mesmos itens de medição elencados neste mesmo Sub-Cenário:

- Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches*;
- Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces;
- Intervalo de Tempo para Cálculo de Caminho;
- Intervalo de Tempo para Envio de Mensagens *OpenFlow*;

- Intervalo de Tempo para Criação de Circuito.

É importante ressaltar que este Cenário não tem como objetivo comparar os resultados do Sub-Cenário 1a, Sub-Seção 5.3.1.1, que mostra resultados de um *testbed* real, com resultados aferidos de uma rede virtual, *Mininet*. No *testbed*, cada comutador é uma estrutura de *hardware* isolada, possuindo processamento e gerenciamento de recursos completamente isolado dos demais. Já na virtualização, todos os comutadores executam sobre o mesmo *hardware*, em que recursos de CPU e memória são compartilhados. Portanto, reitera-se que os resultados deste Cenário servem para validar aspecto que escalabilidade do Op3nControl.

A seguir serão mostrados os resultados dos experimentos. As justificativas e explicações dos mesmos são idênticas ao resultados Sub-Cenário 1a, por isso serão omitidas neste Sub-Cenário, mas podem ser observadas no Sub-Cenário 1a, Sub-Seção5.3.1.1.

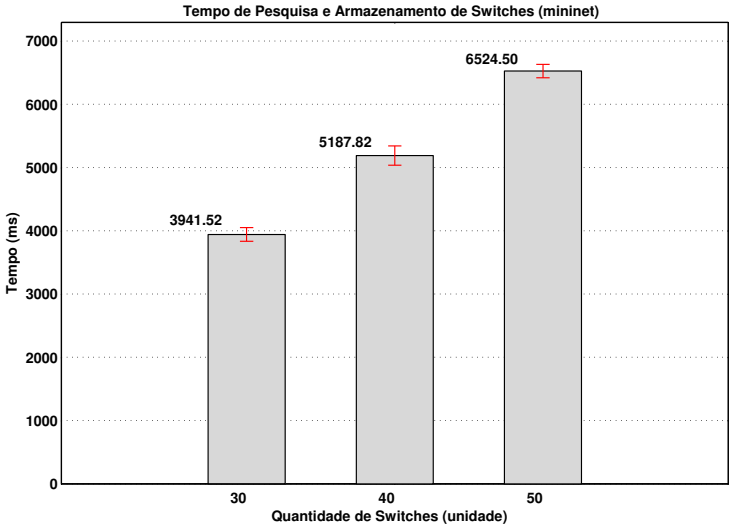


Figura 24: Intervalo de Tempo para Pesquisa e Persistência de dados de Switches (*Mininet*)

Na Figura 24, estão disponíveis os resultados do item de medição "Intervalo de Tempo para Pesquisa e Persistência de dados de *Switches*". A mesma mostra o tempo de operação para cada variação da topologia (trinta, quarenta e cinquenta *switches*). Para trinta *switches* o tempo de processamento foi de 3.94 segundos, para quarenta *switches* o tempo foi 5.18 segundos e, para cinquenta *switches*, o resultado foi de 6.52 segundos.

Já na Figura 25, são revelados os resultados do item de medição "Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces". Os resultados expostos mostram o tempo de operação para cada variação da topologia. Para trinta *switches* o tempo de processamento foi de 2.25 segundos, para quarenta *switches* o tempo foi 2.77 segundos e para cinquenta *switches* o resultado foi de 3.35 segundos.

Os resultados do item de medição "Intervalo de Tempo para Cálculo de Caminho"são mostrados na Figura 26. Estes que disponibilizam o tempo de operação para cada variação da topologia. Para trinta *switches* o tempo de processamento foi de 1.13

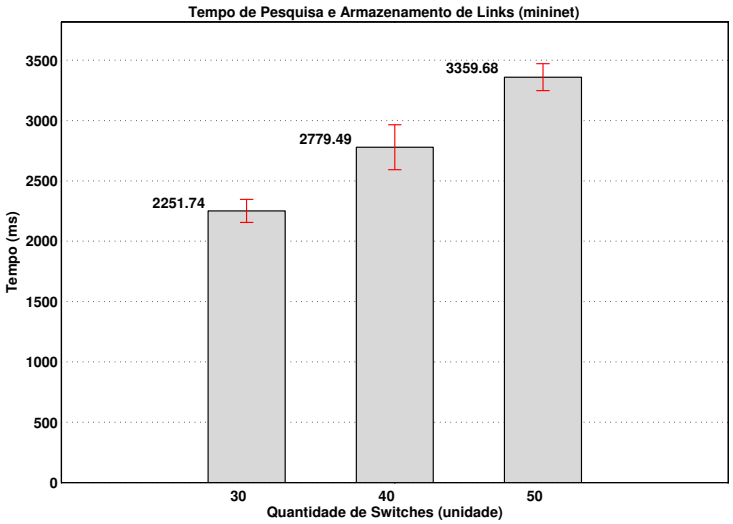


Figura 25: Intervalo de Tempo para Pesquisa e Persistência de dados de Enlaces (*Mininet*)

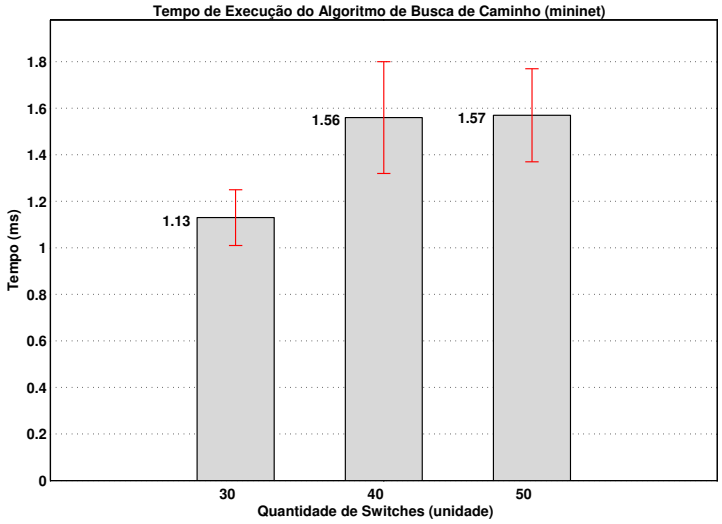


Figura 26: Intervalo de Tempo para Cálculo de Caminho (*Mininet*)

milissegundos, para quarenta *switches* o tempo foi 1.56 milissegundos e para cinquenta *switches* o resultado foi de 1.57 milissegundos.

Na Figura 27, estão disponíveis os resultados do item de medição "Intervalo de Tempo para Envio de Mensagens *OpenFlow*". A mesma mostra o tempo de operação para cada variação da topologia. Para trinta *switches* o tempo de processamento foi de 4.88 milissegundos, para quarenta *switches* o tempo foi 7.70 milissegundos e para cinquenta *switches* o resultado foi de 9.69 milissegundos.

Por fim, na Figura 28 são visualizados os resultados do item de medição "Intervalo de Tempo para Criação de Circuito", na qual podem ser observados os tempos de operação para as três variações da topologia. Para trinta *switches* o tempo de processamento foi de 29.77 milissegundos, para quarenta *switches* o tempo foi 35.15 milissegundos e para cinquenta *switches* o resultado foi de 36.55 milissegundos.

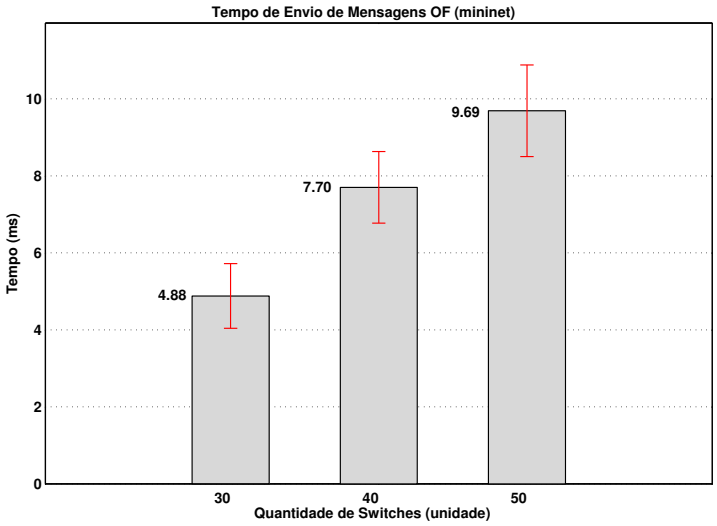


Figura 27: Intervalo de Tempo para Envio de Mensagens *OpenFlow*

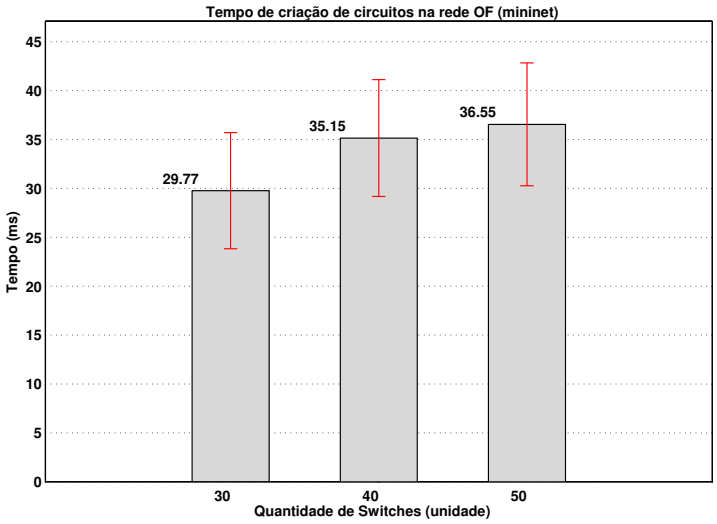


Figura 28: Intervalo de Tempo para Criação de Circuito

5.3.4 Cenário 4. Avaliação de Operações do Módulo Integrador de Switches Legados

O objetivo deste cenário é atestar que a integração com o *datapath OpenFlow* Virtual *LegacyFlow* [Farias et al., 2012] atende aos requisitos elaborados na proposta Op3nControl. Para isso, deve ser verificado o atendimento das funcionamento dessa integração e aferida a sua desempenho. O item de medição definido para satisfazer este Cenário é "Intervalo de Tempo de Operações de Criação de VLANs".

Neste Cenário, há a presença de dois novos elementos: o componente *LegacyFlow* e o comutador *Cisco Catalyst* 3560. O cenário idealizado para os testes está disponível na Figura 29, na qual fica explícito o objetivo da integração do Op3nControl com o *LegacyFlow* no sentido prover circuitos virtuais (L2) através do *OpenFlow*, inclusive quando da existência de equipamentos legados.

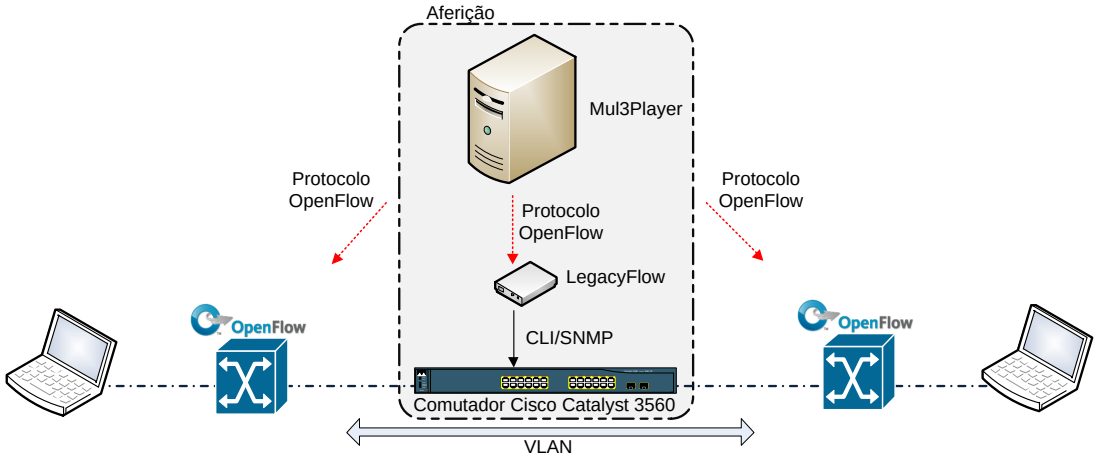


Figura 29: Cenário do Experimento com Datapah LegacyFlow

Contudo, devido a limitações de recursos técnicos de falta de comutadores *OpenFlow* nativo (os equipamentos `TP-Link 1043ND` apresentaram instabilidade no tratamento de VLAN), o experimento teve que ser adaptado para apenas um comutador *OpenFlow* nativo(*NetGear* 7328S-Indigo) mais o comutador com *LegacyFlow*. Ainda assim foi possível realizar a comunicação fim a fim entre os *hosts* sobre o CV criado, sem qualquer problema. Essa limitação em nada comprometeu o resultado do item de medição deste Cenário, que pode ser observado, claramente, na Figura 29 em uma seleção centralizada com fundo destacado. Traduzindo a imagem, o objetivo do item de medição é aferir o intervalo de tempo de comunicação entre a implementação do Op3nControl e o *LegacyFlow* no sentido de criação das VLANs na infraestrutura legada. Desa forma, serão mostrados os tempos de criação e exclusão de CVs. Reforça-se que as medições são apenas para a criação do circuito L2 no *LegacyFlow*, o intervalo de tempo para criação de CV sobre a rede *OpenFlow* não foi aferido neste experimento.

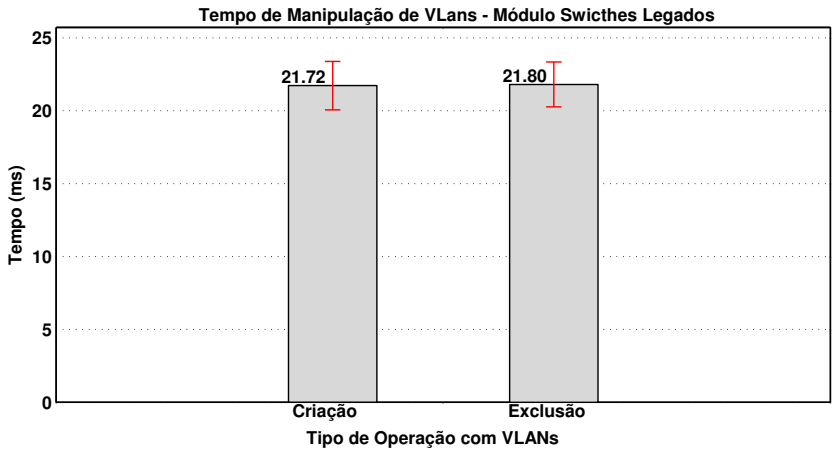


Figura 30: Intervalo de Tempo Para Operações *datapath LegacyFlow*

Na Figura 30, estão disponíveis os resultados do item de medição deste cenário. As barras representam o tempo médio para a operação de criação e exclusão do CV.

Como resultado deste cenário, para criação da CV a média do tempo de processamento foi de 21.72 milissegundos, enquanto que para a exclusão do mesmo, a média do tempo de processamento foi de 21.80 milissegundos. Os tempos são praticamente idênticos, uma vez que as funcionalidades em questão apresentam complexidades similares.

A comunicação com o *datapath LegacyFlow* ocorre através de mensagem *OpenFlow* do tipo `OFPT_FLOW_MOD`, contendo ações do tipo `Output`. Essa mensagem é traduzida para mensagens específicas da implementação do *datapath LegacyFlow*, referente ao modelo de *switch* em questão, no caso deste experimento trata-se de um modelo *Cisco Catalyst* 3560, que aceita mensagens em formato CLI/SNMP. Outro detalhe importante é que os tempos expostos referem-se ao somatório dos tempos que um cliente invoca determinada ação (criação ou exclusão) mais o tempo que Op3nControl consome para criar mensagem *OpenFlow* para o *datapath LegacyFlow*. Não foi objeto de medida deste cenário de uso verificar em quanto tempo os circuitos L2 serão criados após os comandos CLI/SNMP serem enviados pelo *LegacyFlow*.

5.3.5 Cenário 5. Avaliação Comparativa entre Implementação do Op3nControl e *CircuitPusher*

Esse Cenário tem como objetivo comparar a implementação do Op3nControl com soluções que tenham algum nível de similaridade em suas funcionalidades. Nesse sentido, é realizada a comparação entre a implementação do Op3nControl e a aplicação *CircuitPusher* [Kanui and Parraga, 2013], disponibilizada com o controlador *OpenFlow Floodlight* [Floodlight, 2013]. A comparação é realizada sobre o item de medição: "Comparação de desempenho para Criação de Circuitos".

A topologia utilizada para esta comparação foi a mesma do Cenário 1, Seção 5.3.1, ou seja, com dois, três, quatro, cinco, sete e nove comutadores do tipo *TP-Link 1043ND* com *OpenFlow* habilitado. Neste cenário de experimentação foram executados cem (100) ciclos de repetições para cada uma dessas variações de topologia em linha.

Os resultados relacionados a implementação do Op3nControl já foram aferidos na Sub-Seção 1a, especificamente no item de medição "Intervalo de Tempo para Criação de Circuito". Portanto, para efeito de comparação, foi necessário somente a execução da aplicação *CircuitPusher* e aferir seus resultados. Os resultados dessa comparação de desempenho entre as execuções das aplicações supracitadas podem ser conferidos na Figura 31.

Em ambas aplicações conseguiu-se estabelecer a comunicação fim-a-fim através de CV entre os *hosts* nas extremidades da rede. Contudo, o tempo para atendimento da funcionalidade foi consideravelmente discrepante entre as duas aplicações. A Figura 32 mostra os resultados que atendem ao item de medição desse cenário "Comparação de desempenho para Criação de Circuitos", na qual pode ser verificada que, para configurar um CV através de entradas *OpenFlow*, para dois *switches*, enquanto o *CircuitPusher* necessita de mais de 4 segundos o Op3nControl realizou tal ação em 19.13 milissegundos.

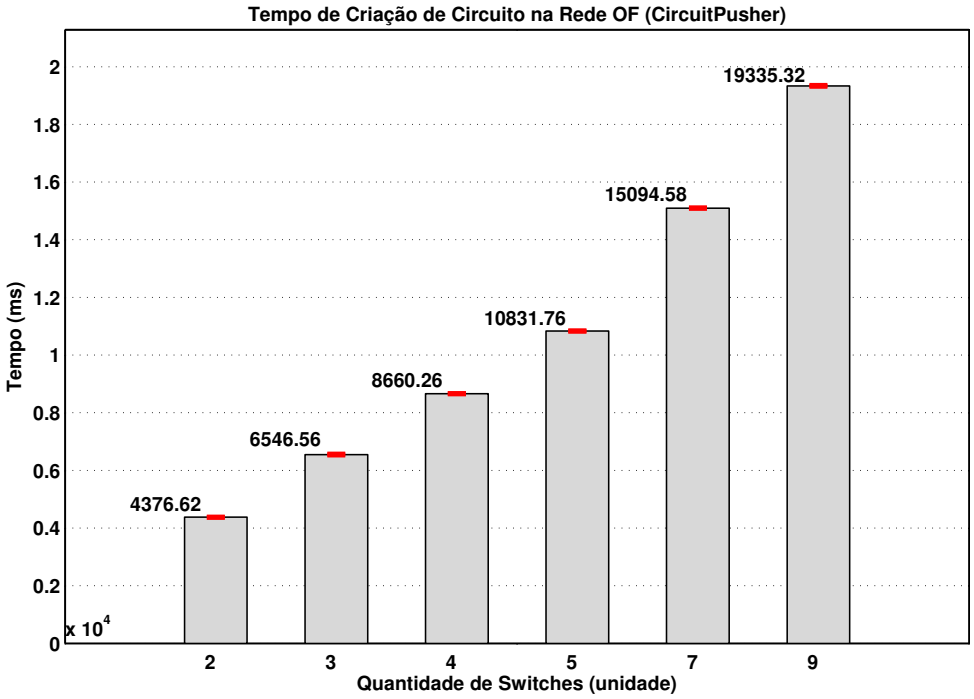


Figura 31: Tempo para Criação de Circuitos da Solução CircuitPusher

A discrepância é verificada nas demais aferições com diferentes quantidades de comutadores. Desta forma, ratifica-se a impressão de boa desempenho do Op3nControl para a funcionalidade de criação de CV.

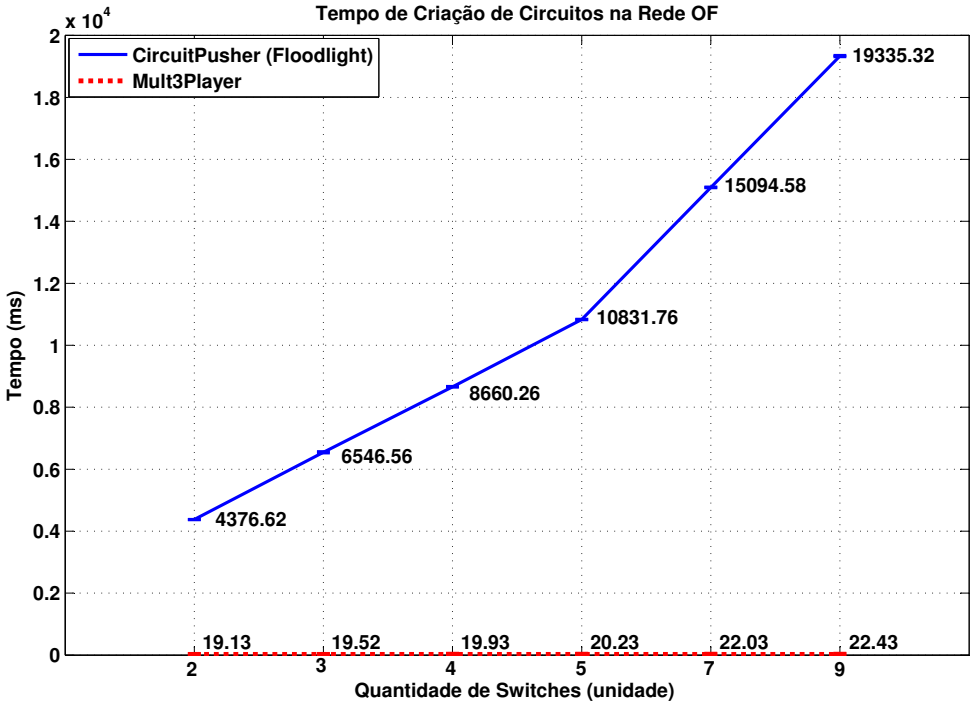


Figura 32: Comparação de desempenho para Criação de Circuitos

5.4 Conclusões do Capítulo

Nesse capítulo foram observados cinco cenários de experimentação envolvendo a implementação do Op3nControl. O Cenário 1 subdividiu-se em três Sub-Cenários. No primeiro, foram medidas as funcionalidades relacionadas a comutadores, *links*, busca de caminho e criação de circuito; no segundo Sub-Cenário, foram aferidas as grandezas relacionadas ao tempo de criação de circuito com QoS; e, o terceiro abordou uma visão de controle de tráfego alcançado com técnicas de QoS do Op3nControl.

No Cenário 2 foi observado como o Op3nControl se comportou ao trabalhar com uma rede com topologia de *links* redundantes entre os comutadores. Já o Cenário 3 abordou aspectos de escalabilidade, trazendo resultados de execução do Op3nControl com o *Mininet*, um software capaz de criar uma rede *OpenFlow* virtual. O Cenário 4 apresentou testes integrando equipamentos sem *OpenFlow* nativo (equipamentos legados) com comutadores com *OpenFlow* nativos, ambos gerenciados através do Op3nControl. Por fim, no Cenário 5 foi realizada uma comparação de desempenho do processo de criação de circuito entre o Op3nControl e aplicação com funcionalidade similar, o *CircuitPusher*.

Nos experimentos envolvendo o *testbed*, houve limitação de número de comutadores disponíveis (nove). Apesar da utilização do *Mininet* ter minimizado tal limitação, testes com *testbed* mais robustos são indicados. Através dos resultados apresentados, atesta-se que o Op3nControl é uma solução que merece atenção e que pode ser bastante útil no cenário de redes SDN/OpenFlow.

CAPÍTULO 6

Conclusões

Esta dissertação apresentou um estudo sobre o padrão SDN/*OpenFlow*, descrevendo sua arquitetura e principais características. De modo simplificado, o *OpenFlow* é um padrão que prevê a separação dos planos de dados e controle nos comutadores, permitindo assim, que o controle desses comutadores possa ocorrer através de aplicações externas e centralizadas nos controladores *OpenFlow*.

Contudo, para ter maior acessibilidade e ser funcional, uma rede *OpenFlow* necessita de um gerenciador de ações, um *framework* de gerência e controle, e que possibilite interação com o usuário, este que pode ser um experimentador ou um administrador de redes. Neste contexto foi elaborada a proposta deste trabalho, o Op3nControl. A arquitetura, funcionalidades e principais características do Op3nControl foram discutidas durante o trabalho. Os módulos de coleta de elementos de rede e estatísticas, cálculo de caminho, provisionamento de circuitos virtuais, controle de armazenamento e, sobretudo, as propostas de gerenciamento de QoS e integração com equipamentos legados fundamentam os objetivos desta dissertação.

Para a execução dos experimentos da dissertação foi criado um ambiente de experimentação real (*testbed*), contendo equipamentos habilitados com *OpenFlow*. A implementação do Op3nControl foi executada sobre esse *testbed* sob a perspectiva de cinco cenários de avaliação. Nestes, foram aferidas diversas métricas: intervalo de tempo para coleta de dados de comutadores e links e seus respectivos dados estatísticos, intervalo de tempo para criação de mensagens OpenFlow e de configuração de QoS, performance para configuração dos comutadores, desempenho para controle de *switches* legados, integração com o *Mininet*.

Os resultados foram satisfatórios, apresentando medidas de desempenho com baixo tempo de processamento e, sobretudo, demostraram o atendimento aos requisi-

tos funcionais esperados do Op3nControl. Em um dos experimentos (comparação com o *CircuitPusher*), comparou-se a performance de uma das funcionalidades do Op3nControl (criação de circuito virtual) com uma solução similar, e comprovou-se a impressão positiva acerca do desempenho da proposta Op3nControl, uma vez que a mesma reduziu em escala considerável (superior a 99%) o tempo para concluir tal funcionalidade.

Por fim, através da apresentação do Op3nControl e suas funcionalidades pretende-se contribuir para o avanço de pesquisas na área de Internet do Futuro. A realização das avaliações de resultados da implementação do *framework* Op3nControl ocorreu através da execução de experimentos em um *testbed* simples, com ajustes pontuais, sua estrutura já se encontra pronta para ser disponibilizada em um ambiente de experimentação de maior escala.

6.1 Trabalhos Futuros

Como trabalhos futuros, deverão ser incorporadas novas funcionalidade ao Op3nControl, descritas a seguir. Além disso, sua disponibilização em ambiente de experimentação baseado em *OpenFlow* de grande escala seria muito interessante para atestar seu bom funcionamento.

De forma prioritária, deve ser elaborado um trabalho que disponibilize um algoritmo de cálculo de caminho inovador, que utilize as APIs de estado da rede oferecidas pelo Op3nControl.

Outra contribuição relevante é incrementar o *framework* com funcionalidade de tolerância a falhas. Uma abordagem interessante seria a utilização do protocolo de *Spanning Tree* (STP) integrado ao *OpenFlow* para reconstruir, de modo imediato, circuitos virtuais interrompidos, quando verificada ocorrência de falhas em dispositivos da rede.

O Op3nControl é capaz de realizar a integração de federações *OpenFlow* de forma automatizada. Como é comum a existência de equipamentos legados em ambientes de redes de produção, o módulo que controla dispositivos legados poderia realizar a administração desses equipamentos, criando um circuito virtual L2, por exemplo, estabelecendo a comunicação entre duas ou mais federações *OpenFlow*. Talvez a solução atual necessite de pequenos ajustes (relacionada a regra de *Match* nos CVs criados, alterando-se de dados L3 para L2) para funcionar de forma adequada neste cenário.

Por fim, será criado uma página web contendo artefatos de documentação e código fonte da proposta deste trabalho, com objetivo de compartilhar o Op3nControl com a comunidade.

Referências Bibliográficas

Akari. Akari architecture design project for new generation network. http://akari-project.nict.go.jp/eng/index2.htm, 2013. [Online; accessado Janeiro-2013].

Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, February 2008. ISSN 0360-0300. doi: 10.1145/1322432.1322433. URL `http://doi.acm.org/10.1145/1322432.1322433`.

Beacon. Beacon. https://openflow.stanford.edu/display/Beacon/Home, 2013. [Online; accessado Janeiro-2013].

Zheng Cai, A. L. Cox, and Eugene T. S. Ng. Maestro: A system for scalable openflow control. tech. rep. tr10-11. Technical report, Department of Computer Science: Rice University, 2010.

Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane: taking control of the enterprise. *SIGCOMM Comput. Commun. Rev.*, 37(4):1–12, August 2007. ISSN 0146-4833. doi: 10.1145/1282427.1282382. URL `http://doi.acm.org/10.1145/1282427.1282382`.

S. Civanlar, M. Parlakisik, A.M. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem. A qos-enabled openflow environment for scalable video streaming. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 351–356. IEEE, 2010.

F.N.N. Farias, J.M.D. Júnior, J.J. Salvatti, S. Silva, A.J.G. Abelém, M.R. Salvador, and M.A. Stanton. Pesquisa experimental para a internet do futuro: Uma proposta utilizando virtualização eo framework openflow. *Minicurso,Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2011.

F.N.N. Farias, J.J. Salvatti, E.C. Cerqueira, and A.J.G. Abelem. A proposal management of the legacy network environment using openflow control plane. In *Network Operations*

and Management Symposium (NOMS), 2012 IEEE, pages 1143 –1150, april 2012. doi: 10.1109/NOMS.2012.6212041.

Fire. Future internet research and experimentation. http://cordis.europa.eu/fp7/ict/fire/, 2013. [Online; accessado Janeiro-2013].

Floodlight. Floodlight - an open sdn controller. http://floodlight.openflowhub.org, 2013. [Online; accessado Dezembro-2012].

Geni. Exploring networks of the future. http://www.geni.net, 2013. [Online; accessado Janeiro-2013].

Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35 (5):41–54, October 2005. ISSN 0146-4833. doi: 10.1145/1096536.1096541. URL `http://doi.acm.org/10.1145/1096536.1096541`.

N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

A. Ishimori, J. Salvatti, Fernando F. Farias, Gaspary L., L Granville, Cerqueira E., and Abelém A. J. G. Qosflow: Gerenciamento automatico da qualidade de servico em infraestruturas de experimentacao baseadas em framework openflow. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos WPEIF*, 2010.

ITU-D. Itu telecommunication development sector. itu information and communication technology. statistics and database. internet users, 2011. [Online; accessado Janeiro-2013].

B. Kanui and J. Parraga. Circuit pusher. http://www.openflowhub.org/display /floodlightcontroller/Circuit+Pusher, 2013. [Online; accessado Fevereiro-2013].

W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.J. Lee, and P. Yalagandula. Automated and scalable qos control for network convergence. *Proc. INM/WREN*, 2010.

James F. Kurose and Ross Keith W. *Redes de Computadores e a Internet: uma abordagem top-down*. Addison Wesley, 5 edition, 2010.

Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0409-2. doi: 10.1145/1868447.1868466. URL `http://doi.acm.org/10.1145/1868447.1868466`.

A.K. Malik. Network management and control systems and strategic issues. *Communications Magazine, IEEE*, 28(3):26–29, 1990. ISSN 0163-6804. doi: 10.1109/35.52888.

D.M.F. Mattos, N.C. Fernandes, V.T. da Costa, L.P. Cardoso, M.E.M. Campista, L.H.M.K. Costa, and O.C.M.B. Duarte. Omni: Openflow management infrastructure. In *Network of the Future (NOF), 2011 International Conference on the*, pages 52 –56, nov. 2011. doi: 10.1109/NOF.2011.6126682.

Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38 (2):69–74, March 2008. ISSN 0146-4833. doi: 10.1145/1355734.1355746. URL `http://doi.acm.org/10.1145/1355734.1355746`.

ONF. Software-defined networking:the new norm for networks. https://www.opennetworking.org/. [online; accessado janeiro-2013]. Technical report, Open Networking Foundation, 2012.

OpenFlowJ. Openflowj. https://openflow.stanford.edu/fisheye/browse/OpenFlowJ, 2013. [Online; accessado Janeiro-2013].

OpenFlowSpecv1.0. Openflow specification 1.0. http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf, 2013. [Online; accessado Janeiro-2013].

OpenFlowSpecv1.3.1. Openflow specification 1.3.1. https://www.opennetworking.org/images/stories /downloads/specification/openflow-spec-v1.3.1.pdf, 2013. [Online; accessado Fevereiro-2013].

Jianli Pan, S. Paul, and R. Jain. A survey of the research on future internet architectures. *Communications Magazine, IEEE*, 49(7):26 –36, july 2011. ISSN 0163-6804. doi: 10.1109/MCOM.2011.5936152.

Subharthi Paul, Jianli Pan, and Raj Jain. Architectures for the future networks and the next generation internet: A survey. *Comput. Commun.*, 34(1):2–42, January 2011. ISSN 0140-3664. doi: 10.1016/j.comcom.2010.08.001. URL `http://dx.doi.org/10.1016/j.comcom.2010.08.001`.

POX. Pox. http://www.noxrepo.org/pox/about-pox/, 2013. [Online; accessado Janeiro-2013].

R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 2009.

Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. Can the production network be the testbed? In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1924943.1924969`.

Andrew S Tanenbaum. *Redes de Computadores*. Elsevier, 4 edition, 2003.

Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, pages 42:1–42:6, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0064-3. doi: 10.1145/1900008.1900067. URL `http://doi.acm.org/10.1145/1900008.1900067`.

Feng Zhao, Dan Zhao, Xiaofeng Hu, Wei Peng, Baosheng Wang, and Zexin Lu. A 3n approach to network control and management. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1237–1242, 2012. doi: 10.1109/IPDPSW.2012.156.