UNIVERSIDADE FEDERAL FLUMINENSE

RODRIGO ALVES DE OLIVEIRA

Avaliação de ambientes de experimentação baseados em OpenFlow

NITERÓI

2018

UNIVERSIDADE FEDERAL FLUMINENSE

RODRIGO ALVES DE OLIVEIRA

Avaliação de ambientes de experimentação baseados em OpenFlow

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sistemas de Telecomunicações

Orientador:
NATALIA CASTRO FERNANDES

NITERÓI

2018

Ficha catalográfica automática - SDC/BEE

O48a Oliveira, Rodrigo Alves de
Avaliação de ambientes de experimentação baseados em OpenFlow / Rodrigo Alves de Oliveira; Natália Castro Fernandes, orientadora. Niterói, 2018.
134 f.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2018.

1. OpenFlow. 2. Redes Definidas por Software. 3. Plataforma de Teste. 4. Open vSwitch. 5. Produção intelectual. I. Título II. Fernandes,Natália Castro, orientadora. III. Universidade Federal Fluminense. Escola de Engenharia.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

RODRIGO ALVES DE OLIVEIRA

Avaliação de ambientes de experimentação baseados em OpenFlow

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sistemas de Telecomunicações

Aprovada em 28 de fevereiro de 2018.

BANCA EXAMINADORA

Prof[a]. D.Sc. NATALIA CASTRO FERNANDES,
Universidade Federal Fluminense (UFF)
Orientadora

Prof. D.Sc. RODRIGO DE SOUZA COUTO, Universidade do Estado do Rio de Janeiro (UERJ)

Prof. D.Sc. DIOGO MENEZES FERRAZANI MATTOS,
Universidade Federal Fluminense (UFF)

Prof. D.Sc. RICARDO CAMPANHA CARRANO,
Universidade Federal Fluminense (UFF)

Niterói
2018

Dedico este trabalho a todos que acham que nunca é tempo de recomeçar:
"Não importa onde você parou...
em que momento da vida você cansou...
o que importa é que sempre é possível e
necessário "Recomeçar"."

Carlos Drummond de Andrade

Agradecimentos

À minha família, que sempre estiveram presentes nos momentos difíceis da minha vida, apoiando nas minhas decisões e compreendendo a importância do Mestrado para mim, em especial a minha mãe Leda Nasaré Alves de Oliveira, que além de mãe é minha melhor amiga, sempre acreditando na minha capacidade e, principalmente, por confiar em mim.

Agradeço a minha orientadora Natália Castro Fernandes, pela amizade e por, além de orientado, ter realmente participado neste trabalho, demonstrando dedicação, empenho e paciência comigo, os quais foram essências para a conclusão.

Agradeço a todos os membros do Programa de Pós-Graduação e do Laboratório MidiaCom, que sempre foram atenciosos e disponíveis para atender as necessidades e dificuldades ao longo desta trajetória, assim como ao CNPq, CAPES e FAPERJ pelo apoio financeiro e a todos os envolvidos da Rede Nacional de Ensino e Pesquisa (RNP) pelo apoio com os testes realizados na plataforma FIBRE.

Aos meus amigos de DATAPREV pelo apoio e incentivo para que eu pudesse me ausentar do trabalho em horário comercial para as aulas na Universidade Federal Fluminense, especialmente ao meu chefe Ronaldo Luis Ribeiro dos Santos pela compreensão durante todo o tempo de realização do Mestrado.

Finalizando, gostaria de agradecer a todos mais que eu não tenha citado nesta lista de agradecimentos, mas que de uma forma ou de outra contribuíram para a minha dissertação.

Resumo

A escolha de um ambiente de experimentação para Redes Definidas por Software (Software Defined Networking - SDN) adequado ao tipo de experimento é um desafio frequente para pesquisadores devido à gama de possibilidades existentes, peculiaridades de cada ambiente e orçamento restrito para compra de equipamentos. O objetivo deste estudo é analisar os diferentes ambientes de experimentação sob a ótica das soluções existente de switch OpenFlow, considerando questões como desempenho, custos, escalabilidade, credibilidade e facilidade de uso. São avaliados vários parâmetros de desempenho em ambientes com configurações semelhantes de rede, mas implementações do switch OpenFlow, enlaces, controlador e clientes diversas. A proposta do estudo não é afirmar qual a melhor solução, mas sim prover insumos para a escolha do ambiente que melhor se adapte às necessidades da pesquisa, considerando desempenho, controlabilidade e custos. Ambientes emulados, plataformas privadas e públicas são foco destas avaliações.

Palavras-chave: SDN, OpenFlow, Mininet, Pronto, NetFPGA, FIBRE, GENI, OvS, Plataforma de teste.

Abstract

The choice of a Software Defined Networking (SDN) experimentation environment proper to the kind of experiment is a frequent challenge for researchers because of the range of existing possibilities, peculiarities of each environment, and restricted budget for purchasing devices. The objective of this study is to analyze the different experimentation environments from the standpoint of the existing OpenFlow switch solutions, considering issues such as performance, costs, scalability, credibility and ease of use. Several performance parameters are evaluated in environments with similar network configurations, but different OpenFlow switch, links, controller and host implementations. The proposal of the article is not to define the best solution, but to provide inputs for choosing the environment that best suits the research requirements, considering performance, controllability, and costs. Emulated environments, private and public testbeds are the focus of these assessments.

Keywords: SDN, OpenFlow, Mininet, Pronto, NetFPGA, FIBRE, GENI, OvS, Testbed.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API	:	Application Programming Interface;
ARP	:	Address Resolution Protocol;
ASIC	:	Application Specific Integrated Circuits;
BGP	:	Border Gateway Protocol;
CLI	:	Command Line Interface;
CPU	:	Central Processing Unit;
CSMA	:	Carrier Sense Multiple Access;
DHCP	:	Dynamic Host Configuration Protocol;
DNS	:	Domain Name System;
DoS	:	Denial of Service;
DOT	:	Distributed OpenFlow Testbed;
DRAM	:	Dynamic RAM;
FIB	:	Forwarding Information Base;
FIBRE	:	Future Internet Brazilian Environment for Experimentation;
FITS	:	Future Internet Testbed with Security;
GENI	:	Global Environment for Network Innovations;
GRE	:	Generic Routing Encapsulation;
GUI	:	Graphical User Interface;
HS	:	Hardware Switch;
HTTP	:	Hypertext Transfer Protocol;
ICMP	:	Internet Control Message Protocol;
IP	:	Internet Protocol;
MAC	:	Media Access Control;
MSS	:	Maximum Segment Size;
MTU	:	Maximum Transmission Unit;
NetFPGA	:	Networking Field-Programmable Gate Array;
NFV	:	Network Function Virtualization;
NS-3	:	Network Simulator 3;

OCF	:	OFELIA Control Framework;
OFELIA	:	OpenFlow in Europe: Linking Infrastructure and Applications;
OFLOPS	:	OpenFlow Operations Per Second;
OMF	:	Orbit Management Framework;
OMNet++	:	Objective Modular Network Testbed in C++;
ONF	:	Open Networking Foundation;
OSPF	:	Open Shortest Path First;
OvS	:	OpenvSwitch;
RAM	:	Random Access Memory;
RFC	:	Request for Comments;
RNP	:	Rede de Nacional de ensino e Pesquisa;
RSpec	:	Resource Specification;
SANE	:	Secure Architecture for the Networked Enterprise;
SBRC	:	Simpósio Brasileiro de Redes de Computadores;
SDN	:	Software Defined Network;
SDNWLAN	:	Software-Defined Wireless Local Area Network;
SNMP	:	Simple Network Management Protocol;
SRAM	:	Static Random Access Memory;
SS	:	Software Switch;
SSL	:	Secure Socket Layer;
STP	:	Spanning Tree Protocol;
TCAM	:	Ternary Content Addressable Memory;
TCP	:	Transmission Control Protocol;
UDP	:	User Datagram Protocol;
UFF	:	Universidade Federal Fluminense;
UtahDDC	:	Utah Downtown Data Center;
VLAN	:	Virtual Local Area Network;
XML	:	eXtensible Markup Language;
WiMAX	:	Worldwide Interoperability for Microwave Access

Sumário

Capítulo 1

Introdução

As Redes Definidas por Software (Software Defined Network (SDN)) transformam a forma como as redes de computadores são controladas e gerenciadas [8] abrindo uma nova perspectiva para novas aplicações de rede, que passam a poder ser desenvolvidas de forma simples e livre dos limites da arquitetura atual da Internet. De fato, nas últimas décadas foi observada uma revolução no avanço de dispositivos, aplicativos, serviços e soluções de armazenamento, enquanto a forma da rede permaneceu inalterada. Hoje em dia, novas maneiras de projetar e operar redes usando SDN e a virtualização de funções de rede (Network Function Virtualization (NFV)) [9] surgiram, revolucionando a forma de gerenciar as redes.

A idéia-chave do SDN, conforme citado por McKeown em [10], é separar fisicamente os planos de controle e encaminhamento para fornecer uma orquestração e automação mais eficientes da rede; outras literaturas como em [11] focam no fato da inteligência da rede ser logicamente centralizada em um software de controlador que mantém uma visão global da rede. Como esperado, há muitas maneiras de executar esta tarefa, mas uma das mais destacadas é o OpenFlow. O protocolo OpenFlow [3] surge neste ambiente como uma forma de padronização da configuração por meio de um controle unificado de comutadores, roteadores e pontos de acesso sem fio. Usando o OpenFlow, os dispositivos de rede passam a conter com uma interface de programação simples, que permite o acesso ao controle de fluxo, ou à tabela de comutação do hardware.

Dada a sua importância disruptiva, as SDNs e as aplicações de rede desenvolvidas baseadas no protocolo OpenFlow são hoje alvo de grandes investimentos em pesquisas. Estas pesquisas avançam para o estado da arte com soluções inovadoras que trazem diversos benefícios para o setor acadêmico e privado. No entanto, para que estas vantagens sejam de fato reais, os testes das novas soluções precisam ser o mais consistente possível.

1.1 Motivação

A necessidade por cenários de redes vem crescendo em virtude da demanda de testes de novas soluções para redes antes mesmo da sua utilização no mundo real. Entre as metodologias adotadas estão o uso de simuladores e emuladores que tenta representar o mais fielmente possível o sistema real a ser estudado. As informações resultantes deste processo são utilizadas para estimar variáveis de interesse deste sistema.

Alguns fatores como: investimento, relação custo x benefício, complexidade de gerenciamento e tempo necessário para implementação são algumas das preocupações associadas aos ambientes de avaliação das redes atuais e das tecnologias de redes que estão surgindo, no qual questões sobre uso de Central Processing Unit (CPU), memória, tempo computacional e escalabilidade são essenciais para uma boa análise. Por este motivo, o estudo dos ambientes possíveis para a execução de experimentos se torna importante no contexto atual das redes. A avaliação do funcionamento de cada ambiente, seja através de experimentos em um ambiente real, emulado ou simulado pode trazer benefícios para os estudos de novas tecnologias SDN.

O Mininet é uma ferramenta para a simulação de redes definidas por software que permite a emulação de uma grande infraestrutura virtual de rede com a utilização de apenas uma máquina. O usuário pode rapidamente emular, interagir, personalizar e compartilhar um protótipo de rede definida por software para simular uma topologia de rede que utiliza switches OpenFlow. O Mininet até o momento não fornece desempenho e qualidade fiéis a uma rede real, embora o código das aplicações de controle utilizado nele sirva para uma rede real baseada em placas Networking Field-Programmable Gate Array (NetFPGA)s, ou switches comerciais. Isto se deve aos recursos que são tratados pelo kernel da máquina simuladora em tempo real, uma vez que a largura de banda total é limitada por restrições de processamento e memória da mesma. Baseado nestas premissas entende-se que existem certos problemas que devem ser considerados ao criar uma rede no Mininet, principalmente na sua utilização para simulação de grandes redes que pode resultar em prejuízos nos resultados finais [12], pois necessitaria de maior processamento para um bom desempenho.

Neste contexto de experimentação de ambientes baseados no protocolo OpenFlow, foco desta dissertação, os resultados apresentados servem de insumos para pesquisadores que possuem como estudo as redes SDN. O aumento de pesquisa no meio acadêmico destas redes ocorre devido as diversas possibilidades de aplicação que esse paradigma traz para

as redes de computadores se apresentando como uma forma potencial de implantação da Internet do Futuro e a possibilidade de implantação de aplicações de rede que realizam lógicas de monitoração e acompanhamento de tráfego mais sofisticado e mais completo que os atuais.

O protocolo OpenFlow é apenas uma das aplicações abordadas pelas Redes Definidas por Software, entretanto muito utilizada no meio acadêmico o que estimulou a realização desta dissertação, pois nesta conjuntura o protocolo OpenFlow aborda a ideia de que não é necessário que a comutação de pacotes seja definida pelo principio de roteamento de redes Internet Protocol (IP), tornando os elementos da rede, roteadores e switches, não mais como os controladores da parte lógica do processo de comutação de pacotes, este processo pode ser controlado por aplicações em software desenvolvidos independentemente do hardware.

Todo este arcabouço que envolve o protocolo OpenFlow e as Redes Definidas por Software abrindo a possibilidade de se desenvolver outras aplicações que controlam os elementos de comutação de uma rede física de maneiras totalmente distintas das que é possíveis com a arquitetura atual das redes de computadores, estimulou esta dissertação, motivando a busca por prover insumos aos pesquisadores sobre ambientes de experimentação OpenFlow de forma a facilitar a escolha de um ambiente apropriado para o desenvolvimento de novas aplicações.

De fato, a importância de testes conclusivos sobre novas aplicações de controle e gerência e sobre propostas de evolução do plano de dados não pode ser negligenciada [13]. Apesar da importância de resultados consistentes, é comum se deparar na literatura com análises limitadas ou resultados duvidosos devido ao mal uso de ferramentas de análise, simulação e emulação. Os resultados podem ser mascarados quando a escolha do ambiente não é bem executada e, da mesma forma, bons resultados podem ser descartados quando o ambiente escolhido não é adequado.

1.2 Objetivos

Este trabalho visa analisar e verificar os pontos fortes e fracos de ambientes de experimentação OpenFlow, provendo insumos para que pesquisadores possam escolher adequadamente a ferramenta que será usada de acordo com as suas necessidades. São consideradas questões como a facilidade de implementação, a precisão dos resultados, a repetibilidade dos resultados e os custos de implementação. Os testes realizados visam

verificar o desempenho dos ambientes de experimentação, com foco na infraestrutura adotada e, através dos resultados obtidos, ponderar as características destes ambientes para contribuir na escolha de ambientes a serem usados por pesquisadores.

É utilizada uma metodologia que consiste na comparação de ferramentas de experimentação diferenciadas: ferramenta de emulação, plataforma de teste pública e o desenvolvimento de plataforma de teste privada. Foram desenvolvidos pequenos cenários que podem ser reproduzidos em todas as ferramentas de experimentação possibilitando avaliação do desempenho da ferramenta como um todo. O objetivo é prover insumos para a escolha de quando usar uma ou outra ferramenta de experimentação.

1.3 Organização do trabalho

Este trabalho está organizado ao longo de oito capítulos. No Capítulo 2 são apresentados os conceitos básicos das Redes Definidas por Software e o protocolo OpenFlow, que são os assuntos que permeiam esta dissertação; logo em seguida, o Capítulo 3 descreve os tipos de ambientes de experimentação, destacando suas características particulares. O Capítulo 4 apresenta o ambiente de implementação com sua arquitetura detalhada, especificando os hardwares, softwares e aplicações utilizadas. Os resultados dos experimentos são expostos no Capítulo 5 com uma análise apurada sobre os resultados obtidos no Capítulo 6. Esse capítulo também traz uma estimativa de custos destes ambientes e a indicação de qual ambiente utilizar dependendo do que o pesquisador almeja. O Capítulo 7 apresenta os trabalhos relacionados ao tema desta dissertação, que permearam as escolhas deste trabalho. Por fim, no Capítulo 8 são apresentadas as conclusões da dissertação e uma breve descrição de trabalhos futuros que podem ser realizados baseados no que esta dissertação apresentou.

Capítulo 2

Redes Defnidas por Software e o protocolo OpenFlow

A proposta nessa dissertação é baseada na aplicação do conceito de Redes Definidas Por Software (SDN), mais especificamente do protocolo OpenFlow. Assim primeiramente, será apresentada a arquitetura, os elementos e as aplicações das Redes Definidas por Software. Na sequência, são apresentadas as características do padrão OpenFlow, bem como os elementos necessários para sua configuração e aplicação em uma Rede Definida por Software.

2.1 Redes Definidas por Software

O termo Internet do Futuro trata das atividades de pesquisa sobre novas arquiteturas para a Internet, objetivando resolver problemas causados pela sua arquitetura monolítica fechada, que ocasiona problemas relacionados ao endereçamento, mobilidade, escalabilidade, gerenciamento e qualidade de serviço. As abordagens da Internet do Futuro visam resolver os problemas atuais e atender às novas demandas de requisitos estabelecidos pela Internet, em sua grande maioria, caracterizando um plano de controle que permite mover grande parte da lógica de tomada de decisões dos dispositivos de redes para controladores externos, que podem ser implementados com o uso de tecnologia de servidores comerciais, um recurso abundante, de fácil programação, escalável e barato [14]. Esse é o conceito chave das redes definidas por software. O principal motivador para substituir a arquitetura de rede convencional para uma Rede Definida por Software é o seu desenho alinhado com as principais tendências de mercado:

- Nuvem híbrida: Sistemas distribuídos geograficamente através de nuvens públicas e

privadas demandam um gerenciamento de tráfego extremamente flexível e acesso à largura de banda sob demanda.

- Gestão em Infraestrutra de TI: A tendência de BYOD (Bring Your Own Device) requer redes que sejam flexíveis e seguras.
- *Big data*: A era do zetabyte significa mais largura de banda, conectividade e acesso aos sistemas do *data center*.

Ainda é possível atribuir como vantagem da arquitetura SDN a dificuldade das atuais redes convencionais em crescer na dinâmica que a TI precisa, a complexidade na implementação de políticas e readapatação do desenho e a dependência de fabricantes.

As redes definidas por software apresentam esse novo paradigma para o desenvolvimento das redes de computadores. As SDN abrem novas perspectivas em ambientes de controle lógico da rede de forma usualmente centralizada, permitindo que novas aplicações de rede possam ser desenvolvidas de forma simples e livre dos limites da arquitetura atual. As principais precursoras do conceito das redes SDN são as propostas Secure Architecture for the Networked Enterprise (SANE) [15] e Ethane [16].

O projeto SANE é uma uma arquitetura de proteção para redes corporativas, concebida em 2006, que define uma única camada de proteção, localizada entre as camadas Ethernet e de Rede, que governa toda conectividade dentro da empresa. Todas as decisões de roteamento e controle de acesso são feitas por um servidor logicamente centralizado que concede acesso usando um ponto de vista global, permitindo implementar políticas de uma forma independente de topologia.

Ethane (sucessor do SANE) é um tipo de arquitetura da qual derivaram as redes SDN, que estabelece uma solução a nível de fluxo para o controle de acesso em redes empresariais. Nas redes Ethane foi proposto um mecanismo de controle de acesso distribuído para a rede, porém com o controle centralizado em um nó, responsável pela supervisão. Nessa arquitetura, a cada novo fluxo, o nó central era consultado para a verificação das políticas de segurança de alto nível. Em caso de o novo fluxo ser permitido, o no controlador central instalava o novo fluxo nos comutadores da rede através de uma interface de programação. Em caso negativo, caso as políticas de acesso bloqueassem a entrada do novo fluxo, os pacotes desse fluxo eram descartados. O modelo de programação de rede seguido pelo Ethane foi entao estendido e formalizado, dando origem ao OpenFlow.

A Open Networking Foundation (ONF) [17] é uma organização sem fins lucrativos, financiada por empresas como a Deutsche Telekom, Facebook, Google, Microsoft, Verizon

e Yahoo visando promover a rede através de SDN e padronizar o protocolo OpenFlow e as tecnologias relacionadas.

2.1.1 Arquitetura

A SDN propõe um arquitetura dinâmica, gerenciável, adaptável e com um custo-benefício adequado, tornando-se a plataforma ideal para a alta largura de banda e a natureza dinâmica das aplicações de hoje.

De acordo com o *Open Networking Foundation*, na arquitetura SDN, os planos de controle e de dados são dissociados. Em um ponto central ficam a inteligência e a monitoração do estado da rede, ao mesmo tempo que a infraestrutura básica torna-se abstrata para as aplicações. Como resultado, o departamento de TI pode construir redes altamente escaláveis e flexíveis que facilmente se adaptam às mudanças de necessidades de negócios, por meio de recursos programáveis que irão automatizar e controlar a rede. Já o plano de dados, comumente chamado de plano de encaminhamento, controla a comutação e o repasse dos pacotes de rede. Esta separação entre o plano de dados e o plano de controle facilita ao plano de controle assumir o papel de orquestrador das relações da rede com outras aplicações e serviços [18]. A Figura 2.1, mostra o intuito das redes definidas por software, separado em algumas camadas que mostram os detalhes dos elementos da mesma.

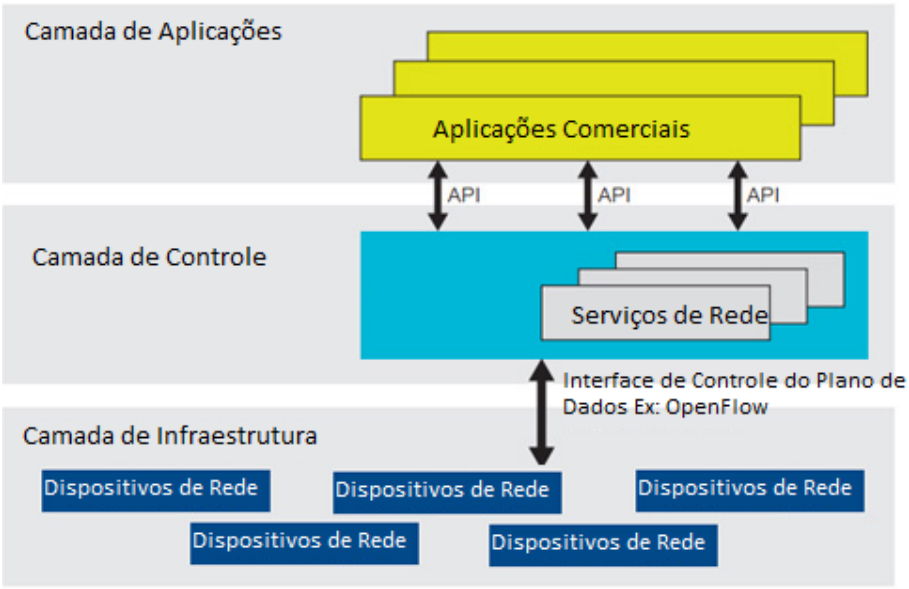


Figura 2.1: Separação dos elementos da arquitetura de redes definidas por software. Adaptado de [1]

Nas redes legadas, o plano de controle, responsável por todas as tomadas de decisões,

é executado no próprio equipamento que também atua no encaminhamento dos pacotes na rede, por meio dos protocolos inseridos em seu *firmware*, não sendo possível trocar qualquer tomada de decisão que não tenha sido prevista nestes protocolos. Este modelo de equipamento tem desvantagens grandes, pois apresenta um complexo sistema de software com milhares de linhas de código fonte e múltiplas funções complexas integradas em sua infraestrutura física. Com esta ligação estabelecida entre o plano de controle, representado pela entidade de software, e o plano de dados, cuja entidade representa o elemento de camada física (hardware), o mercado de rede é forçado a adquirir uma determinada solução de apenas um fabricante, não sendo possível comprar o hardware de um e o software de outro e, com isso, ele deixa de ser competitivo, devido à forte pressão em massa dos grandes fabricantes sobre os pequenos. A Figura 2.2 mostra a evolução das redes SDN com relação às redes legadas.

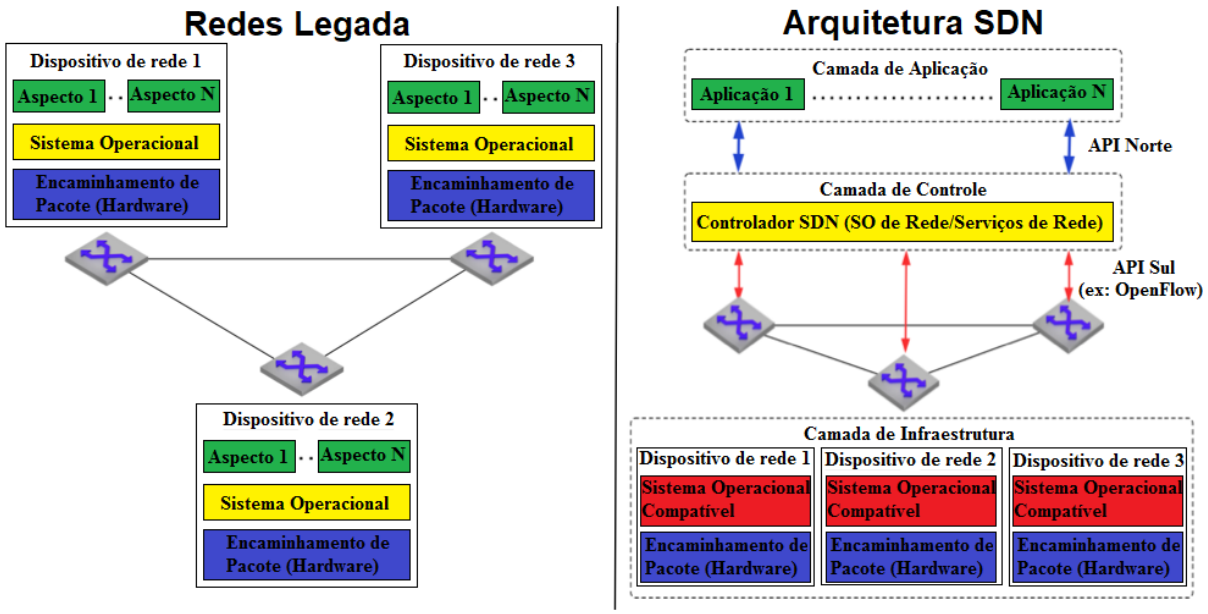


Figura 2.2: Comparação entre as Redes Legadas e Redes Definidas por Software. Adaptada de [2]

Para que seja possível modificar esta restrição, faz-se necessária uma forma de permitir que o equipamento encaminhe os pacotes a partir de protocolos abrigados externamente, de forma que se tenha a separação do plano de controle daquele pré-configurado e não se limite aos protocolos desenvolvidos pelo fabricante.

Um dos elementos que podem ser utilizados na arquitetura SDN é o protocolo OpenFlow, proposto por McKeown em 2008 [3] e padronizado pela ONF. O OpenFlow é a primeira interface padronizada projetada especificamente para SDN, ao estabelecer uma interface de comunicação entre o plano de encaminhamento, que se encontra nos equipa-

mentos de rede, sejam eles, switches ou roteadores, e o plano de controle, que no caso das Redes Definidas por Software, fica externo ao dispositivo de rede, alocado numa máquina física ou virtual. Essa arquitetura proporciona, devido a sua grande programabilidade, alto desempenho e controle de tráfego granular através de dispositivos de rede de diferentes fornecedores.

O protocolo também proporciona a melhoria de automação e gerenciamento, usando Application Programming Interface (API)s comuns para abstrair os detalhes de infraestrutura básica de rede dos sistemas de orquestração e aplicações de aprovisionamento. Com esses recursos é possível melhorar a experiência do usuário final, com aplicações que podem explorar as informações das condições da rede para se adaptar ao comportamento e às necessidades do usuário. A arquitetura SDN e o protocolo Openflow também influenciam diretamente na confiabilidade e segurança da rede, uma vez que a gestão centralizada e automatizada de dispositivos de rede permite a aplicação de políticas uniformes reduzindo possíveis erros de configuração.

2.1.2 Elementos da solução OpenFlow

Basicamente, uma rede definida por software utilizando o protocolo OpenFlow consiste em elementos de rede habilitados para que o estado das tabelas de encaminhamento possa ser instalado através de um canal seguro, conforme as decisões de um controlador em software. Os elementos de comutação, switches e roteadores, exportam uma interface de programação que permite softwares de rede inspecionar, definir e alterar a tabela de roteamento do comutador. Os componentes da arquitetura destas redes são: a tabela de fluxos, o canal seguro, o protocolo OpenFlow e o controlador, conforme mostrado na Figura 2.3.

- Tabelas de fluxos: A entrada na tabela de fluxos do hardware de uma Rede Definida por Software consiste em regras, ações e contadores. A regra a ser aplicada a um determinado fluxo entrante na rede é formada com referência na definição de um ou mais campos do cabeçalho do pacote. Associa-se a ela um conjunto de ações que definem o modo com que os pacotes devem ser processados e para onde eles devem ser encaminhados. Os contadores são usados com a finalidade de manter estatísticas de utilização e também servem para remover fluxos inativos. As entradas nas tabelas de fluxos podem ser interpretadas como decisões em hardware do plano de controle (software), sendo, portanto, a mínima unidade de informação presente no plano de dados da rede.

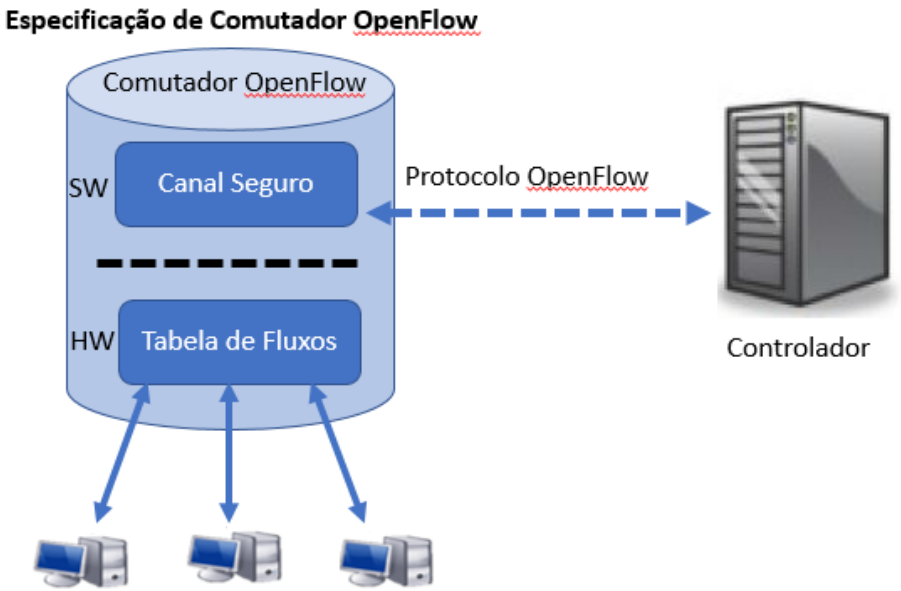


Figura 2.3: Elementos de uma Rede Definida por Software. Adaptado de [3]

- Canal seguro: é necessário um canal para trocar de forma segura informações entre o switch e o controlador, sem que sofra ataque de elementos mal-intencionados. A interface de acesso recomendada é o protocolo Secure Socket Layer (SSL). Interfaces alternativas (passivas ou ativas) incluindo-se o Transmission Control Protocol (TCP) são essenciais em ambientes virtuais e experimentais pela facilidade de utilização, pois não necessitam de chaves criptográficas [14], mas não são recomendadas na prática.

- Protocolo OpenFlow: O protocolo OpenFlow é um protocolo aberto utilizado para a comunicação, fazendo uso de uma interface de acesso, para a troca de mensagens entre os equipamentos de rede e os controladores.

- Controlador: É o software responsável por tomar decisões e adicionar e/ou remover as entradas na tabela de fluxos, de acordo com o objetivo desejado. Exerce a função de uma camada de abstração da infraestrutura física, permitindo de forma mais fácil a criação de aplicações e serviços que gerenciem as entradas de fluxos na rede. A programação do controlador permite a evolução das tecnologias nos planos de dados e as inovações na lógica das aplicações de controle. Foram desenvolvidos diversos controladores para o paradigma SDN, os quais apresentam ambientes de tempo real de execução e uma interface para programação da rede. A Tabela 2.1 apresenta alguns modelos de controladores e suas características básicas.

Tabela 2.1: Principais Controladores SDN.

Nome	**Linguagem**	**Código Aberto**	**Desenvolvedor**	**Caracterísitca**
NOX [19]	C++	Sim	Nicira	É o primeiro controlador SDN, desenvolvido pela empresa Nicira e disponibilizado para a comunidade.
POX [20]	Python	Sim	Nicira	Tem uma interface de programação de aplicações de alto nível, incluindo um gráfico de topologia e suporte para virtualização.
Maestro [21]	Java	Sim	Rice University	É um controlador OpenFlow para orquestrar aplicações de controle de rede.
Beacon [22]	Java	Sim	Stanford	É um controlador que suporta operação baseada em eventos.
Floodlight [23]	Java	Sim	BigSwitch	É um controlador desenvolvido pela empresa Big Switch e disponibilizado para a comunidade como software livre.
Ryu [24]	Python	Sim	Grupo NTT, OSRG	Um controlador que visa proporcionar controle logicamente centralizada e APIs para criar novos aplicativos de gerenciamento e controle da rede.
OpenDaylight [25]	Java	Sim	Linux Foundation	Implementa o protocolo Openflow. Através de uma API HTTP ele permite que outras aplicações possam interagir com os equipamentos/switches controlados por ele.
ONOS [26]	Java	Sim	Open Network Lab	É um projeto de um controlador de código aberto, para o segmento de provedores de Internet.
SNAC [27]	C++	Não	Nicira	É atualmente o controlador da Nicira, que só disponibiliza o código binário, ele usa um gerenciador de política baseado na web para gerenciar a rede.
Trema [28]	C/Ruby	Sim	NEC	É um framework para o desenvolvimento de controladores OpenFlow.

2.1.3 Aplicações

Considerando os controladores do paradigma SDN como sistemas operacionais de rede, o software desenvolvido para criar novas funcionalidades pode ser visto como uma aplicação que é executada sobre uma rede física. As aplicações são concebidas sob uma plataforma extensível, que pode fornecer a base de diferentes visualizações relacionadas à rede. A interface de usuário é capaz de exibir a topologia da rede em alguns casos, assim como controles personalizados e informações relacionadas ao fluxo, que podem ser consultadas e recebidas pelo próprio controlador.

Com SDN, as aplicações passam a ser ativas nas redes, ao contrário das redes convencionais, pois a rede está ciente da aplicação que está sendo executada. Nas redes convencionais as aplicações devem indiretamente descrever seus requisitos para um bom funcionamento, o que geralmente envolve várias etapas e processamentos para negociar recursos suficientes e uma política de controle para suportar o aplicativo. Usando SDN, as aplicações podem monitorar o estado da rede e se adaptar em conformidade.

A flexibilidade que esse paradigma oferece para estruturar sistemas de rede é útil em praticamente todas as áreas de aplicação das redes de computadores. Sua estrutura lógica centralizada permite o desenvolvimento de novas funcionalidades de maneira simples, abrangendo assim uma ampla diversidade de ambientes de rede. Considerando que as redes SDN definem essa nova forma de estrutura, pode-se avaliar que ela seja aplicável em diversos tipos de ambiente no cenário de redes de computadores, beneficiando uma melhor organização das funcionalidades oferecidas em torno de uma visão lógica completa da rede.

2.2 Protocolo OpenFlow

As redes de computadores já são parte da realidade de bilhões de pessoas, e são compostas por inúmeros equipamentos. Atualmente, elas utilizam inúmeros protocolos, como exposto na Figura 2.4, apenas para definir as regras de roteamento. Outro problema enfrentado são os sistemas fechados que fazem uso de Command Line Interface (CLI)s, Simple Network Management Protocol (SNMP), que acabam sujeitos a soluções específicas, exigindo equipamentos com hardware dedicado, geralmente proprietários, para suportar essa gama de protocolos com alto desempenho e segurança para processamento dos pacotes. Isso implica em implementações que exigem um profundo conhecimento para o uso de tais protocolos sem causar inconsistências ou problemas de desempenho.

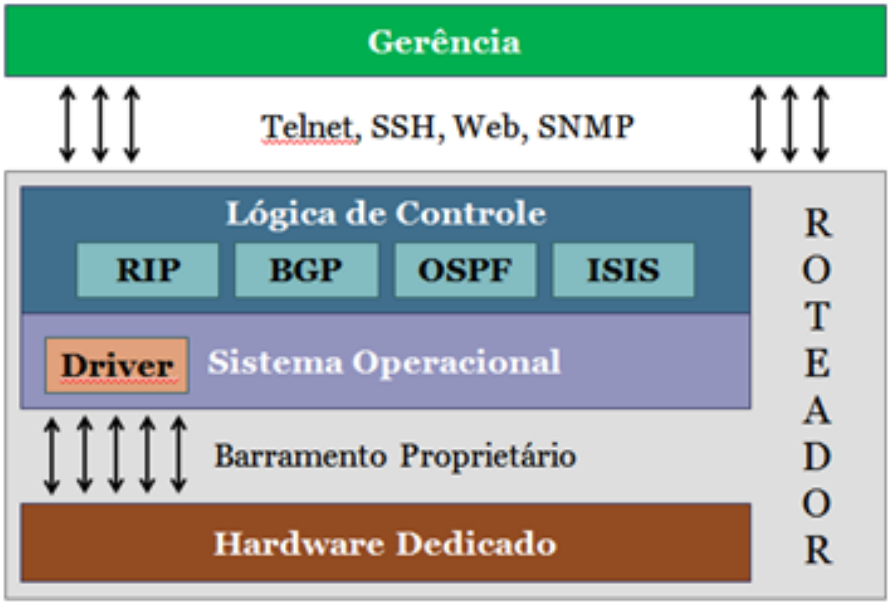


Figura 2.4: Modelo da rede atual. Adaptado de [4]

Para facilitar a implantação destas redes, já que existe em um roteador, por exemplo, mais de 5700 Request for Comments (RFC)s, surgiu uma nova proposta para um protocolo de redes, que seja mais dinâmico e operacional. Este protocolo recebeu o nome de OpenFlow.

O Openflow é um padrão em desenvolvimento para a administração de redes locais e de longa distância com foco em equipamentos comerciais como switches, roteadores, access points e outros; com a idéia de trazer o plano de gerenciamento de todos dispositivos da rede para um único software que será responsável por criar, por exemplo, redes virtuais, roteamento e qualidade de serviço. O OpenFlow foi proposto pela Universidade de Stanford e seu objetivo inicial é atender à demanda de validação de novas propostas de arquitetura e protocolos de rede sobre equipamentos comerciais. Dessa forma, é possível implementar uma tecnologia capaz de promover a inovação no núcleo da rede, por meio da execução de redes de teste em paralelo com as redes de produção.

O protocolo OpenFlow foi criado como uma solução para a construção de uma rede personalizada para experiências acadêmicas. Ele tem por objetivo ser flexível e possibilitar implementações de alto desempenho e baixo custo. Outros benefícios gerados pelo protocolo giram em torno da programação da rede feita pelos operadores, fornecedores de software independentes e por usuários, e não apenas por fabricantes de equipamentos, utilizando ambientes de programação comuns. Isso oferece a todas as partes, novas oportunidades para gerar receita e diferenciação, aumentar a confiabilidade da rede e da segurança, assim como realizar uma gestão centralizada e automatizada dos dispositivos

da rede, aplicando uma política uniforme e sujeita a menos erros de configurações.

2.2.1 Características

Uma das vantagens em se utilizar uma arquitetura OpenFlow é a flexibilidade que ela oferece para se programar de forma independente o tratamento de cada fluxo da rede e como ele deve ou não ser encaminhado nela. De uma maneira mais prática, o OpenFlow determina como um fluxo pode ser definido, quais serão as ações que podem ser realizadas para cada pacote pertencente a este fluxo e qual é o protocolo de comunicação entre o controlador e os comutadores utilizados para realizar as definições de fluxo e ações.

Dessa forma, uma entrada na tabela de fluxos de um comutador OpenFlow é formada pela união da definição do fluxo e um conjunto de ações a ele determinado.

Um fluxo é especificado pelos cabeçalhos das camadas de enlace, de rede e de transporte de cada pacote segundo o modelo TCP/IP, formando um conjunto de doze elementos, conforme modelo OpenFlow versão 1.0 e apresentado na Figura 2.5.

Uma tabela de fluxo consiste em um banco de dados de entradas de fluxos com alguns componentes principais, onde algumas ações podem ser destacadas: encaminhar o pacote para a uma porta específica do switch, descartá-lo, encapsulá-lo e criptografar esse pacote, bem como limitar a banda. Com a definição de fluxo, é possível observar que as regras de encaminhamento de um pacote não se restringem ao endereço IP ou endereço MAC dos pacotes.

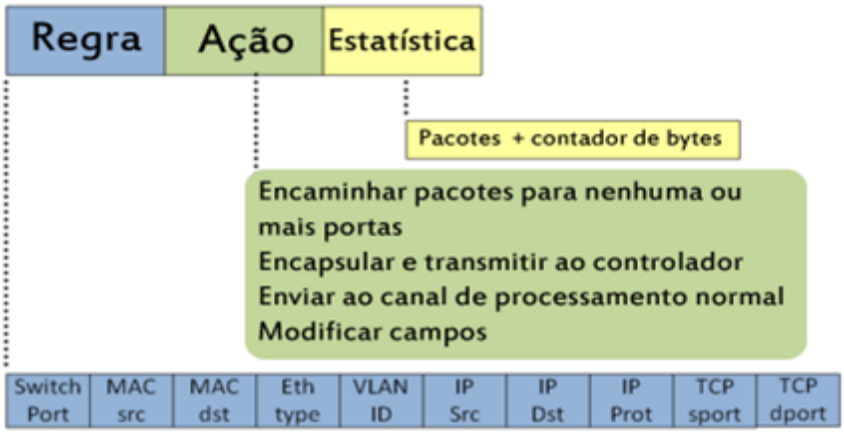


Figura 2.5: Formato da tabela de fluxos do OpenFlow versão 1.0, adaptado de [5]

Cada entrada da tabela de fluxo apresenta um conjunto de ações correspondentes que permitem encaminhar os pacotes, modificar os campos, ou descartar o pacote. Quando um switch recebe um pacote para o qual não possui uma entrada em sua tabela de

fluxos, ele envia esse pacote ou apenas o seu cabeçalho para o controlador. Este então toma as decisões de como lidar com esse pacote podendo descartá-lo ou adicionar uma entrada de fluxo para encaminhar pacotes semelhantes no futuro. Esta ação do protocolo OpenFlow ocorre na versão 1.0, para as demais versões por meio da *table-miss* pode ser configurado as ações de encaminhamento que devem ser executadas referente aos pacotes que não encontraram correspondência em outras entradas de fluxos. A princípio, tabelas que não possuem uma entrada *table-miss* para processar os pacotes sem correspondência, descartam estes pacotes.

2.2.2 Mensagens do Protocolo OpenFlow

O protocolo OpenFlow possui três tipos de mensagens:

- Controlador-Switch: Geradas pelo controlador para gerenciar e inspecionar o estado de um switch. As mensagens comuns são: "Features", "Configuration", "Modify-state", "Read-state", "Send-Packet"e "Barrier".
- Assíncronas: Geradas pelo switch para atualizar o controlador sobre eventos da rede, chegada de novos pacotes, mudanças no estado do switch ou algum respectivo erro e são enviadas sem a solicitação do controlador. Dentre as mensagens podem ser destacadas: "Packet-in", "Flow-removed", "Port-status"e "Error"
- Simétricas: Podem ser geradas tanto pelo controlador quanto pelo switch. São enviadas sem solicitação. Destacam-se as mensagens: "Hello", "Echo"e "Experimenter/Vendor"

No estabelecimento de uma comunicação OpenFlow, cada um dos lados (controlador e switch) devem enviar imediatamente uma mensagem Hello (`OFPT_HELLO`), contendo a mais alta versão do protocolo OpenFlow suportada pelo dispositivo. Ao receber a mensagem, o dispositivo deve escolher a menor versão do OpenFlow entre a que foi enviada e recebida. Se as versões forem compatíveis, a comunicação tem continuidade. Caso não sejam, uma mensagem de erro é gerada (`OFPT_ERROR`) e a conexão é encerrada.

É interessante informar também que, caso haja alguma perda de conexão entre o switch e o controlador, o switch tentará se conectar ao controlador backup, caso exista. Caso essa conexão também falhe, o switch entrará em modo de emergência, modo esse que utilizará apenas as entradas na tabela de fluxos marcadas com um bit de emergência e todas as outras entradas serão deletadas.

2.2.3 Tabela de fluxos OpenFlow

A definição de fluxo é representada pelo conjunto de ações que formam uma entrada da tabela de fluxos. Os switches quando em operação, analisam cada pacote que chega e comparam os cabeçalhos dos pacotes com as entradas das tabela de fluxos. Caso seja encontrado uma correspondência, considerasse que o pacote pertence àquele fluxo, então são aplicadas as ações existentes ou se por acaso, nenhuma regra for encontrada, o pacote é encaminhado para o controlador.

Existem algumas ações básicas associadas a cada entrada na tabela de fluxos de um comutador programável:

- Encaminhar os pacotes deste fluxo para uma determinada porta (ou portas);
- Encapsular e transmitir pacotes deste fluxo para um controlador;
- Descartar pacotes deste fluxo e;
- Alterar o conteúdo dos campos.

Estas ações são realizadas através de trocas de mensagens de controle entre o controlador e o switch, de forma assíncronas e simétricas.

A tabela de fluxos contém doze campos que serão utilizados para a classificação de um fluxo, uma ação para cada fluxo e contadores. Uma entrada na tabela de fluxos é composta por campo dos cabeçalhos, ações e contadores. Os campos de cabeçalho são os campos que serão comparados com o fluxo entrante no switch OpenFlow. Quanto aos contadores existem quatro tipos diferentes: Os contadores por tabela, por fluxo, por porta e por fila. Os mesmos serão incrementados sempre que fluxos correspondentes às entradas na tabela entrem no switch e serão utilizados em mensagens de estatísticas, também definidas pelo OpenFlow. Quanto às ações que deverão ser tomadas com os pacotes, existem ações obrigatórias que todos os switches OpenFlow devem implementar e outras opcionais. Ao se conectar ao controlador, o switch deve informar quais ações opcionais o mesmo implementa. Cada entrada na tabela de fluxos é associada a uma ou mais ações. Se para uma determinada entrada na tabela não houver uma ação especificada, os pacotes desse fluxo serão descartados.

Saber a quantidade máxima de fluxos suportados pelo switch é necessário, uma vez que, caso a tabela de fluxos não possua uma capacidade suficiente para comportar os diferentes fluxos do plano de dados, resultará em uma grande quantidade de troca de

mensagens entre controlador e switch, e como consequência existirá o delay da instalação das novas regras na tabela de fluxos adicionado ao tráfego de dados. Além disso, novas políticas para remoção de fluxos da tabela devem ser utilizadas, como por exemplo, diminuir o tempo que um fluxo fica ativo na tabela de fluxos para garantir que novos fluxos possam ser instalados.

Atualmente, a maioria dos switches OpenFlow utilizam memórias TCAM para implementação da tabela de fluxos, por serem mais eficientes para as rotinas de match, uma vez que diferentemente de uma memória convencional, na TCAM cada célula da memória contém uma lógica que pode efetuar comparações bit-a-bit, onde todas as células trabalham em paralelo para efetuar um casamento de padrões. Porém, este tipo de memória apresenta um alto custo ao projeto e uma capacidade de armazenamento pequena, geralmente entre 4K até 32K regras [11].

Para contornar esse problema é utilizado em alguns casos um modelo hibrido, utilizando um tipo de memória diferente para dar suporte a pequena capacidade de armazenamento da memória TCAM. Uma alternativa é a utilização da memórias RAM, de acesso aleatório que permite a leitura e a escita, utilizada como memória primária em sistemas eletrônicos digitais. Tendo em vista esse contexto, a análise da memória utilizada pelos switches pode identificar possíveis picos de utilização da memória. Caso o switch necessite de uma quantidade de memória acima da que esta disponível, isso pode ocasionar em aumento no tempo para instalação de fluxos ou, no pior dos casos, no não funcionamento do switch.

A execução de ataque de negação de serviço é um tipo clássico de ataque que consome os recursos do switch. Ele se caracteriza pelo preenchimento completo da tabela de fluxos do switch. Teoricamente, se o atacante preencher a tabela de fluxo com fluxos que não correspondam ao tráfego real da rede, então qualquer novo cliente que queira usar a rede não será capaz de fazer isso. O controlador não conseguirá instalar quaisquer novos fluxos e o cliente terá efetivamente negado o uso da rede. O ataque descrito acontece no plano de dados SDN e com o preenchimento intenso da tabela de fluxos do switch após certo tempo os recursos são esgotados, tornando incapaz a inclusão de novos fluxos na tabela de fluxos, por falta de memória. O switch aceita fluxos enquanto houver memória livre na máquina.

Em switches Openflow os controladores instalam novos fluxos em resposta a mensagens do tipo Packet_in enviadas pelo switch. Quando um novo fluxo começa a passar pelo plano de dados o tráfego de controle e a utilização da CPU podem ter picos de utilização.

Isso faz com que o canal de controle fique instável. Saber o impacto deste pico de utilização de CPU é imprescindível para detectar problemas que podem ocasionar com que o switch fique indisponível, ou o impacto que este comportamento terá no tráfego de dados e na disponibilidade do canal de controle. Caso o controlador requisite as estatísticas de vários fluxos paralelamente, isso poderá afetar o tráfego de dados e a latência de instalação de novas regras.

Em geral, os switches reais possuem uma maior capacidade de encaminhamento de dados por possuírem uma maior quantidade de interfaces, enquanto que os switches virtuais podem apresentar maior capacidade de armazenamento de fluxos, memória e robustez quanto ao seus processadores se estiverem embarcados em máquinas com uma grande quantidade de recursos disponíveis.

2.2.4 Aplicação do protocolo OpenFlow

A primeira versão do protocolo OpenFlow foi a 0.2.0 lançada em maio de 2008 e atualmente está obsoleta. A versão 1.0 lançada em dezembro de 2009 foi a mais amplamente divulgada e implementada. Após a versão 1.0, foram lançadas as versões 1.1, 1.2, 1.3, 1.4 e 1.5, dentre as quais vem se destacando a 1.3, em termos de implementação em produtos e softwares de emulação.

Na especificação do OpenFlow 1.1 [29], os switches OpenFlow contêm diversas tabelas de fluxos e uma tabela de grupo, ao invés de uma única tabela de fluxos como na versão 1.0.0, logo, um fluxo de pacotes ao entrar no switch OpenFlow, pode passar por várias tabelas de fluxos, para que diversas ações diferentes sejam realizadas. Além disso, na versão 1.1.0, três novos campos de cabeçalho foram incluídos: *Metadata* (usada para passar informações entre as tabelas no switch), *MPLS label* e *MPLS traffic class*.

O OpenFlow 1.2 [30] foi lançado em dezembro de 2011 e uma das principais implementações foi o suporte ao protocolo IPv6, incluindo novos campos de cabeçalho nas tabelas de fluxos. Além disso, o mesmo passou a suportar a possibilidade dos switches se conectarem a mais de um controlador ao mesmo tempo. Ao se conectar a vários controladores, aumenta-se a segurança já que o switch pode continuar a operar normalmente mesmo se um controlador ou a conexão ficar indisponível.

Sendo lançada em Junho de 2012, a versão 1.3 [31] do OpenFlow passou a suportar o controle das taxas de pacotes através de medidores de fluxos, introduzindo a tabela de medições *Meter Table*. A tabela de medições por fluxo permite ao OpenFlow implementar

simples operações de qualidade de serviço, como limitação de taxas de transmissão. Além disso, a versão 1.3 permite ao switch criar conexões auxiliares para o controlador, ajudando a melhorar o desempenho de processamento do switch, explorando o paralelismo presente na maioria dos switches

Na versão 1.4 [32] do OpenFlow portas ópticas passaram a ser suportadas. Além disso, os conceitos de *Eviction* e *Vacancy events* foram introduzidos para evitar que as tabelas de fluxos fiquem cheias, já que as mesmas tem capacidades finitas. Nas especificações anteriores quando uma tabela de fluxos enchia, novos fluxos não eram inseridos nas tabelas e uma mensagem de erro era enviada para o controlador. Entretanto, atingir tal situação era problemático e poderia causar interrupções no serviço. O *Eviction* adiciona um mecanismo que permite ao switch apagar automaticamente as entradas nas tabelas de fluxos que tenham menor importância. Já o *Vacancy events* permite ao controlador configurar um limiar, que ao ser atingido, faz o switch enviar mensagens de alerta para o controlador. Isso permite que o controlador possa reagir com antecedência, evitando que as tabelas de fluxos fiquem cheias.

A versão 1.5 [33] foi lançada em 09 Janeiro 2015, esta versão teve poucas inovações quando comparadas com as anteriores, possuindo apenas melhorias nas funcionalidades existentes.

O OpenFlow permite o experimento de novas propostas na área de redes de computadores podendo até mesmo utilizar uma infraestrutura de rede já existente, entre elas [3]:

- Novos protocolos de roteamento: Aplicações diversas podem tratar de fluxos específicos de uma rede OpenFlow. Assim, quando um pacote de um novo fluxo chegar a um switch, ele é encaminhado para o controlador que é responsável por escolher a melhor rota para o pacote seguir na rede a partir de uma política adotada pela aplicação (um novo protocolo de roteamento) correspondente. Após isso, o controlador adiciona as entradas na tabela de fluxos de cada switch pertencente à rota escolhida e os próximos pacotes desse fluxo que forem encaminhados na rede não necessitarão ser enviados para o controlador. Dessa forma, novos protocolos de roteamento podem ser implementados para diversos fluxos específicos na rede de modo que tais protocolos cuidem separadamente de um conjunto de fluxos;

- Mobilidade: Uma rede OpenFlow que possui pontos de acesso sem fio permite que clientes móveis utilizem sua infraestrutura para se conectarem à Internet. Assim, mecanismos de *handoff* podem ser executados no controlador com uma simples

migração dinâmica das tabelas de fluxo dos switches de acordo com a movimentação do cliente, permitindo a redefinição da rota utilizada[1];

- Redes não IP: O protocolo OpenFlow não exige que os pacotes sejam de qualquer formato - desde que a tabela de fluxo possa corresponder no cabeçalho do pacote. O switch OpenFlow analisa arbitrariamente os campos do pacote para definir a qual fluxo ele pertence, de acordo com a tabela de fluxos do switch, permitindo a flexibilidade na definição deste. Isso permitiria experiências usando novos esquemas de nomeação, endereçamento e roteamento. Há várias maneiras pelas quais um switch habilitado para OpenFlow pode suportar tráfego não IP. Por exemplo, os fluxos podem ser identificados usando seu cabeçalho Ethernet (endereços MAC src e dst), um novo valor EtherType ou a nível IP, por um novo número de versão IP. De forma mais geral, é esperado que futuros switches permitam que um controlador crie uma máscara genérica (offset + value + mask), permitindo que os pacotes sejam processados de modo especificado pelo pesquisador.

- Redes com processamento de pacotes: Existem aplicações OpenFlow que realizam processamento de cada pacote de um fluxo; nesse caso, a aplicação implementada no controlador OpenFlow cria uma regra nos switches da rede forçando que cada pacote recebido seja enviado ao controlador para ser analisado e processado pela aplicação. Esses casos correspondem, em geral, a pacotes de controle (Internet Control Message Protocol (ICMP), Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP)) ou protocolos de roteamento (Open Shortest Path First (OSPF), Border Gateway Protocol (BGP)).

Assim, o OpenFlow pode ser visto como uma tecnologia promissora e inovadora que abre uma série de oportunidades de desenvolvimento tecnológico na área de redes de computadores.

[1] O controle da rede sem fio por software usualmente requer o uso de outros softwares associados ao OpenFlow, normalmente a solução é a utilização de API abertas que permitam o desenvolvimento de aplicações e algoritmos de controle sensíveis ao contexto.

Capítulo 3

Tipos de Ambientes de Experimentação

O paradigma SDN tem ganhado força nos últimos anos e atraído cada vez mais a atenção da comunidade acadêmica e da indústria [34]. Com isso, muitos equipamentos de rede passaram a ser compatíveis com o protocolo OpenFlow. Da mesma forma, o interesse em projetos de pesquisa relacionados ao uso do OpenFlow como arcabouço para novas soluções também aumentaram [35].

Depois de projetar uma nova aplicação de rede baseada em OpenFlow, é necessário avaliar o impacto desse novo aplicativo. Neste ponto, o experimentador deve decidir em qual ambiente testar. Entre as possíveis ferramentas de experimentação, utilizam-se simuladores, emuladores, testbeds privados e públicos. Existe a questão, no entanto, sobre quais dessas possibilidades são adequadas para o tipo de experiência a ser realizada, considerando funcionalidades, capacidade de processamento e de armazenamento, tipos de clientes e controladores, custos, escalabilidade, entre diversos outros.

Importante salientar que os resultados obtidos nos experimentos realizados podem ser influenciados pelo ambiente utilizado na experimentação. Por exemplo, o ambiente de experimentação tem um efeito notável sobre o tempo necessário para a construção de uma topologia [36], para configuração de um fluxo, para comutação para a porta backup em caso de falha, etc. Além disso, existem diferenças significativas de desempenho e funcionalidade nas implementações em hardware do OpenFlow [36]. Resultados inteiros podem ser prejudicados caso experimentados em ambientes que não sejam próprios para a avaliação requerida. Caso o ambiente tenha uma limitação que afete diretamente os objetivos dos testes, resultados falsos negativos podem ser gerados e uma boa solução pode ser descartada. Além disso, resultados falsos positivos podem ser encontrados. Ambos os casos devem ser evitados com a escolha prévia de um ambiente de experimentação mais

adequado para os testes que serão realizados. Além de um ambiente de experimentação com equipamentos reais, emulação e simulação podem ser utilizados.

3.1 Ambiente OpenFlow Simulado

Um simulador de rede OpenFlow reproduz o comportamento dos equipamentos através de um modelo, para verificar seu comportamento [37, 38]. Com isso, um sistema real pode ser avaliado numericamente a partir de um modelo de simulação. Este modelo deve ser cuidadosamente construído para representar o ambiente de forma a ser fielmente simulado. Assim, o resultado da experiência pode ser o mais próximo possível do que seria encontrado em um ambiente real. A simulação também pode ser executada em plataformas paralelas e modelo de redes de grande escala [39, 40, 41].

Os testes cuja preparação em ambiente real são caros, demorados ou dispendiosos devido a um número elevado de nós devem considerar a simulação como uma forma de execução viável. Porém uma desvantagem é que o desenvolvimento de modelos de simulação exigiria um esforço significativo, além de demandar uma extensa validação, o que pode tomar muito tempo. Esses custos indiretos, tornam os estudos de simulação menos atraentes em muitos casos. Como resultado, existem poucos projetos de pesquisa que usam extensivamente a simulação de rede detalhada para estudos de desempenho em redes OpenFlow.

Apesar dos muitos simuladores de rede disponíveis, existem apenas alguns que oferecem suporte ao OpenFlow. O Objective Modular Network Testbed in C++ (OMNet++) é um simulador de rede baseado em eventos discretos e modular compatível com OpenFlow, orientado a objetos e desenvolvido para diferentes sistemas operacionais como Linux, Mac OS/X e Windows.

A motivação do desenvolvimento do OMNet++ foi produzir um simulador que pudesse ser usado no meio acadêmico, educacional e em pesquisas. Sua principal área de aplicação é a simulação de redes de comunicação por fio e sem fio. OMNet++ fornece uma arquitetura de módulos programados em C++ e linguagens de programação alternativas como Java e C. Este simulador iniciou com suporte ao OpenFlow 1.0 e posteriormente foi desenvolvido um novo módulo de extensão para OpenFlow 1.3 [42, 43].

O Network Simulator 3 (NS-3) [44] é um exemplo de um simulador compatível com o OpenFlow. Contudo, a modelagem feita é baseada no protocolo OpenFlow versão 0.89, que já está desatualizada. O NS-3 implementa um switch OpenFlow em um módulo

C++, além de fornecer um módulo C++ para o controlador. Isso implica que todos os aplicativos OpenFlow no NS-3 devem ser desenvolvidos para funcionar com este módulo controlador C++. Ou seja, nesse tipo de modelo, não é possível utilizar os controladores tradicionalmente utilizados, como NOX, Ryu, Floodlight e outros. Assim, os algoritmos testados precisam ser posteriormente portados para um controlador real, o que pode causar a inserção de novos erros na aplicação de rede. Além disso, não há maneiras simples de determinar se um controlador modificado para NS-3 funcionará como um não modificado, já que é sabido que os controladores apresentam comportamentos muito distintos entre si [45, 46, 47] .

Uma abordagem de pesquisa recente desenvolveu um novo módulo para switches OpenFlow e controlador em NS-3 que executa o protocolo versão 1.3 [48]. Com este módulo, chamado OFSwitch13, é possível interconectar nós NS-3 para trocar tráfego usando os dispositivos Carrier Sense Multiple Access (CSMA) existentes de NS-3. As restrições ao uso de códigos de controladores reais ainda permanecem ao usar este novo módulo.

Assim, simuladores como o NS-3 são bons para testar algoritmos explorando um grande número de dispositivos ou topologias diferentes com um custo menor do que construir a infraestrutura real. No entanto, essa abordagem não produz um primeiro protótipo da solução ou pode falhar ao considerar o tempo de processamento do controlador em respostas de rede em tempo real. Neste contexto de experimentação de ambientes OpenFlow a prototipação é um item de suma importância, uma vez que tem como objetivo facilitar o entendimento dos requisitos, apresentar conceitos e funcionalidades do software, permitindo propor uma solução adequada que supra as necessidades do usuário. Os protótipos são grandes aliados das metodologias ágeis de desenvolvimento, uma vez que garantem maior alinhamento entre a equipe e o cliente; e os ambientes simulados não tem como objetivo a geração de um protótipo, tornando-os menos atrativos para os obejtivos desta dissertação.

Diferentes estudos já provaram que a implementação do controlador OpenFlow afeta diretamente o desempenho de uma aplicação proposta [46, 49, 47]. Esse tipo de impacto geralmente não se reflete nos resultados da simulação usando modelos como o usado em NS-3. De fato, a possibilidade de testar efetivamente o aplicativo desenvolvido em um controlador real fez da emulação uma técnica muito mais usual nas SDNs do que a simulação.

O OpenNet é outra ferramenta de simulação compatível com OpenFlow que possui um ambiente de desenvolvimento em que os scripts baseados na linguagem de programação

C++ são feitos para modificar, construir ou modelar diferentes componentes de uma rede, como links e nós. É conhecido como um simulador para Software-Defined Wireless Local Area Network (SDNWLAN) que conecta o emulador Mininet ao simulador NS-3 para aproveitar tanto as vantagem do Mininet, no que se refere a compatibilidade com controladores, quanto a capacidade do NS-3 na modelagem sem fio [50].

O S3Fnet [51] é um simulador de rede criado no topo do kernel S3F, capaz de criar modelos de rede de comunicação com dispositivos de rede (por exemplo, host, switch e roteador) e protocolos em camadas (por exemplo, OpenFlow). S3FNet é uma plataforma de teste SDN híbrida baseada em OpenFlow que integra um simulador de rede paralelo com um emulador de rede baseado em OpenVZ[1].

Outra abordagem para simular redes OpenFlow é apresentada em [52], que é chamado de Estinet. O Estinet [53] é um simulador comercial que se originou do NCTUns [54], o qual é um simulador de redes que roda em Linux e que foi desenvolvido no *Department of Computer Science* da *National Chiao Tung University*. O Estinet funciona como simulador e emulador e permite o uso de códigos de controladores reais em simulação através do uso de uma técnica chamada de reentrada do kernel. Neste caso, o tempo de simulação é diferente do tempo real, podendo ser mais rápido ou mais lento, assim permitindo que a máquina simule corretamente o comportamento de uma rede OpenFlow muito grande. Isso significa que nenhuma entrega de pacotes será adiada no tempo de simulação devido à falta de processamento disponível na máquina executando a simulação, mas o tempo para executar toda a simulação será maior do que o tempo para executar o mesmo experimento em uma rede real. Isso pode ter consequências para o experimento nos casos em que o controlador mantém os estados com base em tempos que são considerados ao decidir como controlar a rede.

Em geral, no Estinet, um controlador OpenFlow real pode ser prontamente executado sem modificação para controlar switches OpenFlow simulados. Todas as mensagens trocadas entre um controlador OpenFlow real e um switch OpenFlow simulado, assim como as interações entre as aplicações reais, os hosts e os links em uma rede são agendadas com precisão baseado no relógio de simulação, em vez de serem multiplexados e executados de forma imprevisível pelo sistema operaciona. Por esse motivo, no Estinet os resultados de desempenho de uma rede OpenFlow simulada e gerenciada por controladores OpenFlow reais são corretos, precisos e repetíveis.

[1]Tecnologia de virtualização de nível de sistema operacional, que cria e gerencia vários containers Linux isolados, também chamados de ambientes virtuais (VEs), em um único servidor físico

3.2 Ambiente OpenFlow Emulado

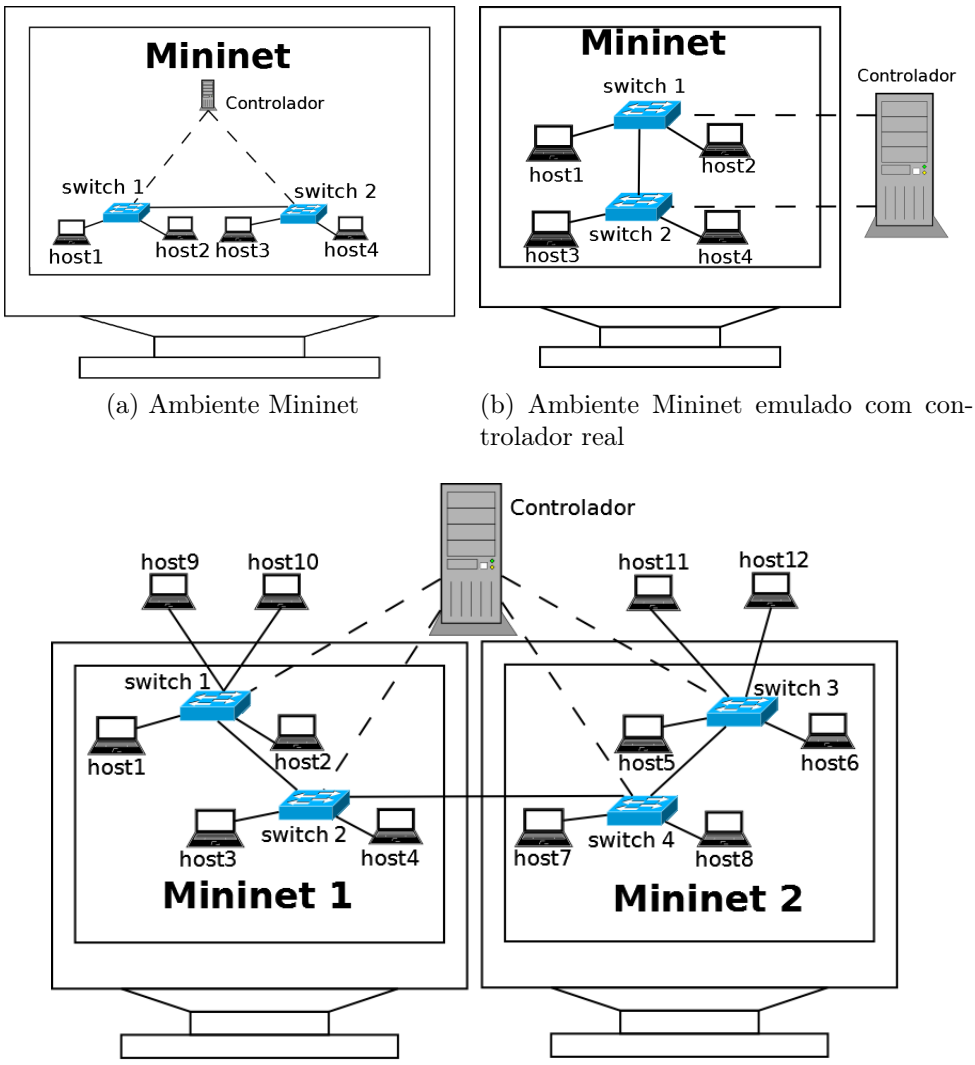
Um emulador de redes SDN é um sistema computacional que imita o funcionamento de switches OpenFlow, controladores e/ou hosts, tendo resposta em tempo real e suporte a modelagem de tráfego, introduzindo atrasos artificiais e perdas de pacotes [55]. Trata-se de uma boa alternativa à simulação, pois facilita a prototipação e permite o teste com o controlador que será usado na prática. O Estinet, como mencionado anteriormente, é um simulador/emulador comercial e, devido a essa condição comercial, geralmente não é usado em trabalhos acadêmicos.

O Mininet é um dos emuladores mais utilizados atualmente para a avaliação de redes SDN, porque pode ser usado para rápida prototipagem. Ele permite a emulação de uma rede OpenFlow com o uso de apenas uma máquina [56], bem como a mistura de máquinas reais com o cenário emulado, conforme mostrado na Figura 3.1c. O Mininet usa primitivas de virtualização do sistema operacional onde o usuário pode criar, interagir, customizar e compartilhar um protótipo de rede OpenFlow com switches, hosts e controladores. Além disso, este emulador executa controladores reais e pode ser integrado com equipamentos externos à emulação.

O Mininet cria switches, hosts e controladores do OpenFlow usando a virtualização baseada em processos com suporte para *namespaces* [57]. Isso cria um ambiente semelhante, porém mais simples que a arquitetura de contêiner do Linux. Os switches OpenFlow são criados usando o software OpenvSwitch (OvS) [58], enquanto os links que conectam switches, hosts e controladores em uma única máquina são criados usando o mecanismo de enlace virtual Ethernet fornecido pelo kernel do Linux [52]. Como hosts e controladores são executados em ambientes virtuais, eles podem executar qualquer aplicativo de computador sem modificações.

No entanto, como qualquer outro software de emulação, o Mininet tem limitações devido às restrições da plataforma de hardware/software onde é executado, o que impede que ele escale para grandes redes [56]. Essa limitação é devido ao uso de um processo shell para emulação e da necessidade de execução de um processo para cada switch OpenFlow virtual e/ou host em espaço de usuário. Assim, o Mininet não fornece desempenho e qualidade fiéis a uma rede real [36]. De fato, essa fidelidade depende da disponibilidade de recursos de processamento e memória para emular todos os switches, hosts e enlaces da rede em tempo real.

Em [59], há a explicação desta falta de fidelidade ao desempenho no Mininet, devido ao



(a) Ambiente Mininet

(b) Ambiente Mininet emulado com controlador real

(c) Ambiente Mininet emulado, integrando redes em emuladores diferentes e com elementos externos e real

Figura 3.1: Exemplos de cenários usando o Mininet.

fato de que este emulador não oferece garantias de que um nó capaz de enviar um pacote será prontamente agendado pelo sistema operacional. Além disso, eles apontam que o Mininet não pode garantir que todos os switches encaminhem pacotes à mesma taxa, pois o tempo de encaminhamento é imprevisível e depende das condições da máquina host [59].

Em [60], é apresentado uma proposta de utilização da plataforma Mininet através de uma configuração distribuída usando uma abordagem simbiótica, que estende um método existente para combinar simulação e emulação em tempo real. É apresentado um estudo de caso que replica com sucesso o comportamento de um protocolo de ataque de negação de serviço Denial of Service (DoS) e mostra que uma única instância do Mininet gera resultados incorretos quando existe vários tráfegos simultâneos e um deles realiza DoS e

com a implementação simbiótica tal efeito não é observado.

Como consequência, o Mininet é uma boa ferramenta para desenvolver um protótipo, mas não é confiável para a realização de testes para medir atrasos, *jitter* ou qualquer outra medida de tempo. Ele também apresenta outras restrições de acordo com a carga na máquina host.

3.3 Plataforma de Testes Privadas OpenFlow

O ambiente de teste mais realista que um pesquisador/desenvolvedor poderia usar seria baseado em equipamentos reais em um ambiente real. Esta opção geralmente é ignorada devido aos altos custos de equipamentos de rede. No entanto, como existem opções para criar ambientes reais usando switches de software ou para usar infra-estruturas públicas para testes, esta é uma opção real ao discutir testes em redes OpenFlow.

Para criar uma plataforma de teste privada, um pesquisador deve ter vários switches, hosts e um servidor para executar o controlador. Os switches OpenFlow podem ser implementados usando switches comerciais bem como computadores pessoais que emulam switches.

Atualmente, diversos fabricantes já disponibilizam switches com suporte ao protocolo OpenFlow. O desempenho de um switch OpenFlow real pode ser influenciado pela característica do hardware utilizado, já que pode-se implementar o switch OpenFlow como Software Switch (SS) [61], ou Hardware Switch (HS) [62]. O SS implementa as tabelas de fluxo, campos, entradas, etc, como uma estrutura de dados através de software, geralmente utilizando sistemas UNIX/Linux e memória comum (Static Random Access Memory (SRAM) - ou Dynamic RAM (DRAM)). Nesse modelo, os dados são processados através da Central Processing Unit (CPU), e por consequência, o desempenho deste é limitado pelo poder de processamento da(s) CPU(s) utilizada(s) [63], [64].

Nos HS, geralmente, é utilizada uma memória especializada chamada de Ternary Content Addressable Memory (TCAM), que age no tamanho das tabelas de fluxo de hardware, taxa de inserção de regra de fluxo, entradas de tabela de fluxo e complexidade de regra de fluxo [65],porém, este tipo de memória apresenta alguns problemas como citado na subseção 2.2.3. Nesses casos, é usual utilizar firmware proprietário para implementar o protocolo OpenFlow e as tabelas de fluxo [66]. Assim, os pacotes são processados através de componentes especializados de hardware, orientados para implementação de tarefas específicas em domínios bem definidos, denominados Application Specific Integrated Cir-

cuits (ASIC). Com isso, a queda de desempenho é mínima.

Uma comparação imediata é que os HSs são considerados mais rápidos do que SSs pois utilizam memórias especializadas para realizar a checagem de regras em paralelo, apesar de possuírem menor capacidade de armazenamento. Com isso, os HSs suportam um número limitado de regras enquanto SS suportam uma quantidade de regras muito maior. Com isso, geralmente, os SSs utilizam altas taxas de processamento para alcançar um menor desempenho que os HSs. No entanto, SSs possuem uma maior flexibilidade para implementar ações mais complexas [67].

Dentro deste cenário, o OvS, um comutador virtual com diversas camadas disponível como um software livre na licença do Apache 2.0, se encaixa como um SS, tanto para switches reais [68] quanto para switches emulados em computadores pessoais. O OvS é um switch virtual multicamada projetado para permitir a automatização maciça da rede através da extensão programática, enquanto ainda suporta interfaces e protocolos de gerenciamento padrão. O OvS tem suporte OpenFlow, é de código aberto e funciona em ambiente virtualizado. Pela sua facilidade de implementação, costuma ser bastante utilizado.

Para implementar a funcionalidade de switch OpenFlow, o OvS utiliza o kernel Unix/-Linux. Na arquitetura OvS, o daemon switch (ovs-vswitchd), situado no espaço de usuário implementado independente do sistema operacional, e o caminho de dados situado no módulo kernel que varia de sistema operacional para sistema operacional, são os componentes mais importantes que afeta o desempenho.

O caminho de dados (*fast path*) no módulo do kernel, recebe os pacotes das Network Interface Card (NIC) e é responsável pelo encaminhamento de pacotes com base nas regras mantidas na tabela de fluxo para encaminhar fluxos correspondentes com as ações correspondentes. O vswitchd gerencia os fluxos no módulo Kernel. Ele contém os componentes ofproto, netdev e dpif, que ajudam a configurar as instâncias do OVS, conectam controladores externos e se comunicam com o sistema subjacente.

Quando o caminho de dados não tem informações de que ação realizar, ele passa o pacote para o ovs-vswitchd, se existe um controlador e o ovs-vswitchd não tem instruções de como manipular este tipo de pacote, ele encaminhará o pacote para o controlador e aguardará por uma resposta. Caso não exista um controlador, o ovs-vswitchd usará sua lógica interna para definir ações para os pacotes recebidos.

Ressalta-se que alguns switches de mercado, como o WRT54GL da *LinkSys*, utilizam o

OvS na sua implementação [34]. Trabalhos como o de Katta et al. [13] propõem modelos híbridos de switch no intuito de combinar as vantagens dos HS e SS em um modelo que consiga lidar com o cache de regras de forma mais eficiente, provendo uma grande quantidade de regras a baixo custo.

Espera-se que o uso de SS em computadores pessoais traga perdas de desempenho quando comparado a switches comerciais. No entanto, como alguns switches comerciais são baseados no mesmo *software switch*, essas perdas não costumam causar impactos significantes nos resultados dos testes. Nesse cenário, é importante notar que diferentes switches de diferentes fornecedores proverão um desempenho diferente, que pode ou não ter impactos na avaliação de uma aplicação de rede. No caso de causar um impacto, o desenvolvedor da aplicação deve considerar em que cenários reais sua aplicação pode ser aplicada.

3.4 Plataforma de Testes Públicas OpenFlow

Com o avanço da área de pesquisa em Internet do Futuro, surgiram diferentes propostas de plataforma de testes públicas (em inglês *testbed* ou *experimental facilities*), que fornecem diferentes infraestruturas que podem ser usadas para testar novas arquiteturas e aplicações sobrepostas à Internet. Dessa forma, funcionam como um laboratório virtual para ensino e pesquisa em rede de computadores. O uso destas plataformas, ao invés de softwares emuladores, ajuda a treinar a próxima geração de pesquisadores em redes. Muitas dessas plataformas são agnósticas quando se considera o uso de SDN, tais como PlanetLab [69] e Orbit [70]. Entretanto, algumas plataformas públicas provêem recursos OpenFlow para experimentação, sendo elas: Future Internet Brazilian Environment for Experimentation (FIBRE) [71], OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) [72], Global Environment for Network Innovations (GENI) [73] e Future Internet Testbed with Security (FITS) [74]. Em geral, estas plataformas incluem tanto hardware switches quanto software switches em suas arquiteturas.

3.4.1 FIBRE

A plataforma de teste FIBRE foi desenvolvida, em 2010, como uma parceria entre institutos de pesquisa brasileiros e europeus, sendo considerada a principal iniciativa brasileira no que se refere a uma plataforma de teste de rede aberto [75, 76].

O FIBRE está aberto para uso sem fins lucrativos e fornece recursos para redes sem

fio e redes OpenFlow, podendo ser usado por estudantes e pesquisadores para realização de testes de novas aplicações e de modelos de arquitetura de rede.

O FIBRE permite a execução de experiências distribuídas geograficamente em larga escala. Sua infraestrutura consiste em uma coleção expansível de componentes localizados em diferentes ilhas, as quais são usualmente universidades. O conjunto de componentes escolhidos para inclusão dentro de um experimento é destinado a permitir a criação de redes virtuais com características distintas, de forma a cobrir toda a gama de experimentos necessária para a comunidade de usuários do FIBRE no Brasil.

Desse modo, a experimentação no FIBRE é realizada por meio de instanciação de máquinas virtuais e reserva de fatias em switches OpenFlow, fornecidas usando o FlowVisor [77], com links de *backbone* fornecendo uma largura de banda de 1 Gb/s compartilhada entre todos os experimentadores.

O FlowVisor [78] é um controlador específico que atua como um proxy entre os dispositivos de rede e controladores OpenFlow, criando uma camada de virtualização sobreposta à rede permitindo que o mesmo hardware do plano de dados possa ser compartilhado entre múltiplas redes virtuais, cada uma com lógicas de encaminhamento distintas através da criação de várias fatias. Sendo assim, diferentes redes podem utilizar os recursos do equipamento de forma independente e aplicando seu próprio controlador. As mensagens OpenFlow enviadas de e para os controladores são interceptadas pelo FlowVisor e encaminhadas de acordo com as redes virtuais estabelecidas, através de um conjunto de fluxos definidos para cada fatia, também chamados de "espaço de fluxo" (ou flowspace).

Entretanto o FlowVisor ainda apresenta algumas limitações, como a definição de mecanismos que permitam provisionar recursos às diferentes redes virtuais, e provisionamento de largura de banda sem interferência entre as fatias. A configuração dos dispositivos ainda é dependente de ferramentas ou comandos externos, como as realizadas estaticamente pelo comando dpctl do framework OpenFlow. Além disso sua arquitetura gera um overhead, pois a mesma entidade responsável pelas informações de gerência da rede virtualizada é a responsável por encaminhar as mensagens de controle geradas, incluindo os packet_in que podem consumir muitos recursos ao serem reencaminhadas entre os FlowVisors até chegarem aos switches/controladores de destino.

O FIBRE inclui diversas ilhas montadas em diferentes universidades e/ou centro de pesquisas espalhados pelo Brasil. O mapa apresentado na figura 3.2 mostra as localizações geográficas das ilhas brasileiras e seus pontos de conexão com o exterior.

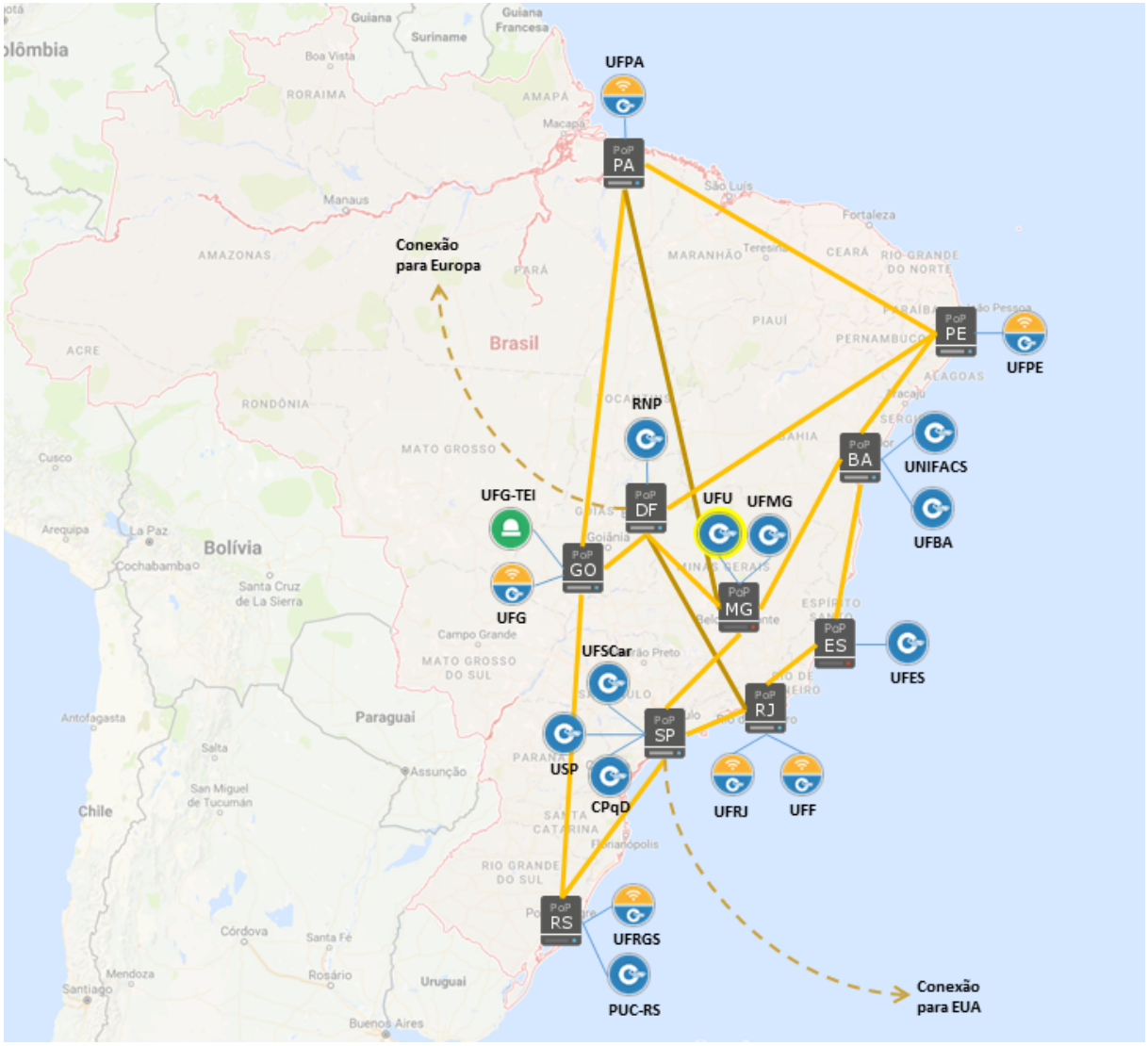


Figura 3.2: Topologia das ilhas do FIBRE no Brasil em novembro 2017. [6]

Detalhes sobre a arquitetura típica de uma ilha FIBRE podem ser vistos na Figura 3.3, sendo composta dos seguintes itens:

- 1 IBM server system x3650 M3 para virtualização.
- 2 Dell Server PowerEdge R210 II para a ferramenta de monitoração.
- 3 NetFPGA Supermicro V9SCM-f para experimentos OpenFlow.
- 1 DATACOM switch DM4100 Firmware 1.0.8 funcionando como switch de topo de *rack*.
- 1 Pronto Switch PICA8 (Open Networking) Firmware 2.1.5 para experimentos OpenFlow.

- 3 ou mais nós *Wireless* Icarus para experimentos associados a redes sem fio.

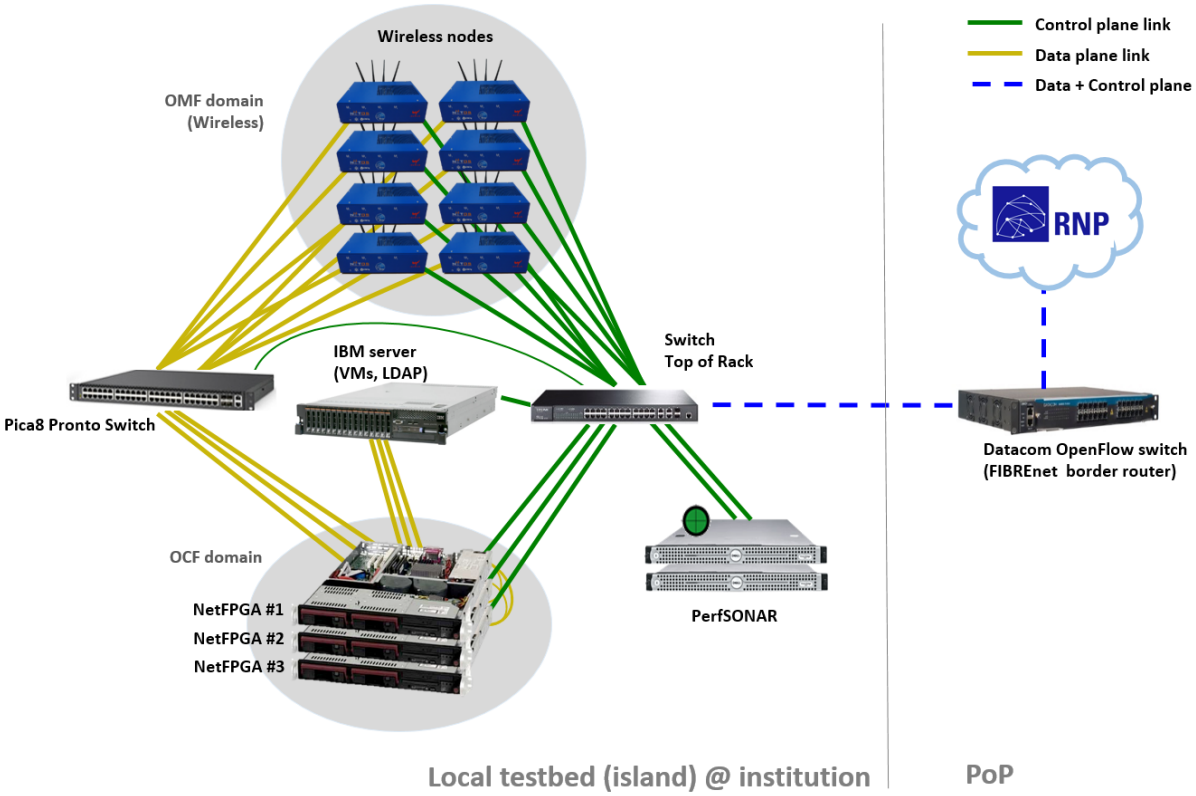


Figura 3.3: Componentes de uma ilha FIBRE. [6]

3.4.2 GENI

O GENI é uma infraestrutura aberta para experimentação em redes, com larga escala, abrangendo todo os EUA, conforme a Figura 3.4. O GENI é um laboratório virtual distribuído com múltiplas plataformas de testes, promovendo, assim, inovações em tecnologias de rede, segurança, serviços e aplicações, atendendo modelos sem fio, óptico e elétrico. Possui sistemas de instrumentação e medição que fornecem sondas para medidas ativas e passivas, armazenamento de dados de medição e ferramentas para visualizar e analisar dados de medição. O GENI permite configurar conexões de camada 2 entre recursos de computação e executar seus próprios protocolos de camada 3 e acima, conectando esses recursos. O laboratório virtual possibilita pesquisas sobre o futuro das redes de grande porte, criando oportunidades de compreensão, inovação e transformação das redes globais e suas interações com a sociedade.

O GENI provê:

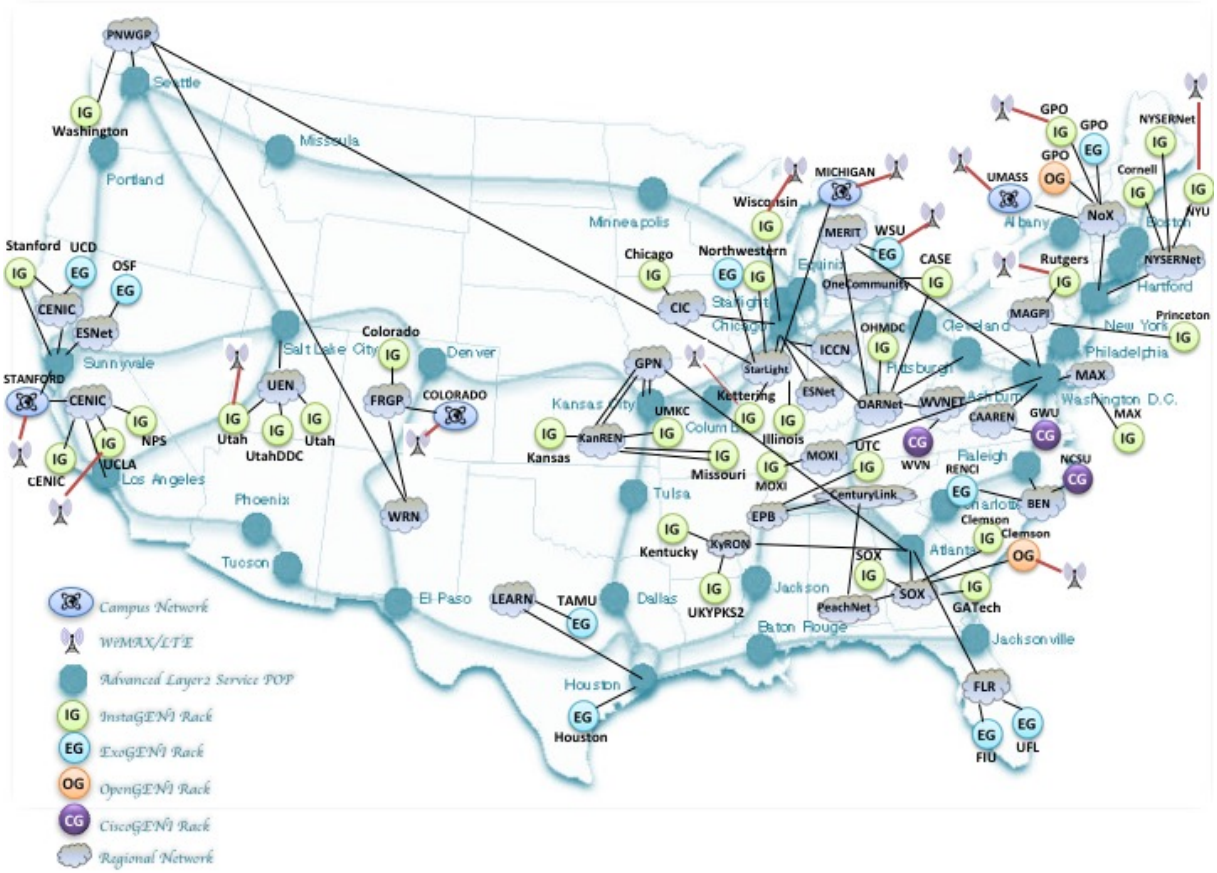


Figura 3.4: Federação GENI. [7]

- Suporte à experimentação em larga escala em uma infraestrutura compartilhada, heterogênea e altamente equipada;
- Alta programabilidade da rede, promovendo inovação nas áreas de redes, segurança, tecnologia, serviços e aplicações;
- Ambientes colaborativos para academia, indústria, governo e público em geral, catalisando descobertas e inovação.
- Acesso exclusivo a certos recursos GENI, incluindo recursos de CPU e recursos de rede, dando ao usuário controle sobre o ambiente da sua experiência e, portanto, a capacidade para repetir experimentos em condições idênticas ou muito similares.
- Programação dos hosts finais e dos switches que compõem o núcleo da rede, permitindo ao usuário experimentar novos protocolos da camada de rede ou novos algoritmos de roteamento IP.

Para iniciar os estudos no ambiente GENI, faz-se necessário ser filiado a uma instituição parceira do projeto ou, caso não exista, preencher um formulário especificando

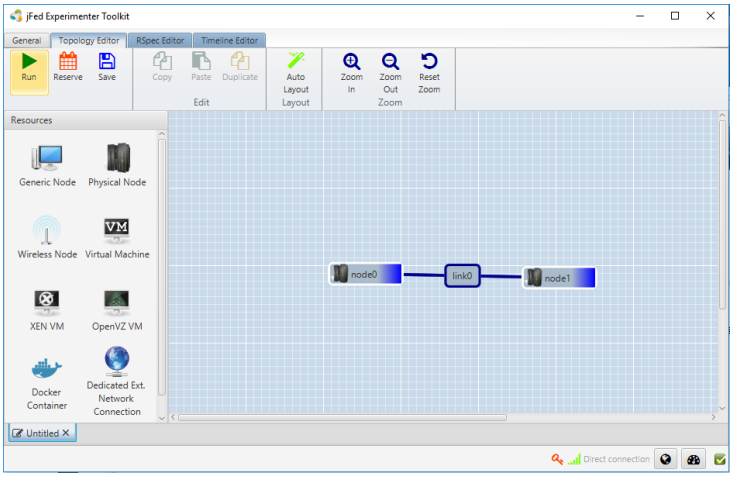
o projeto e a instituição de vínculo. Logo após a criação da conta, alguns itens precisam ser definidos para começar os testes de experimentos. Entre esses, estão o Projeto, criado e liderado por um único indivíduo responsável; a fatia, que é uma unidade de isolamento para os experimentos; os agregados, que fornecem recursos para experimentos GENI, sendo considerado, por exemplo, um agregado o rack GENI em uma universidade; e, finalizando, a Application Programming Interface (API) de gerenciamento de agregação, no qual membros podem solicitar recursos de agregados usando uma API padrão de gerenciamento de agregação. A API usa documentos de especificação de recursos para descrevê-los, comumente referido como Resource Specification (RSpec). Os RSpec são documentos eXtensible Markup Language (XML) que descrevem os recursos disponibilizados para os testes. Membros enviam para os agregados um pedido RSpec que descreve os recursos solicitados e o agregado retorna um manifesto RSpec propondo os recursos disponíveis.

Para facilitar o manuseio das funcionalidades providas pela plataforma GENI, foi utilizado o framework baseado em linguagem java conhecido como jFed [79], que permite que os usuários provisionem e gerenciem experimentos, e pode ser utilizado desde que o usuário possua uma conta no GENI. Assim, o usuário entra com o arquivo de seu certificado e sua senha na federação para o uso do programa. Esta plataforma é executada localmente na máquina do usuário e se apresenta na forma de Graphical User Interface (GUI)s e CLIs desenvolvidas para realizarem uma série de funções, as quais vão desde a criação de topologias, agendamento e gerenciamento de experimentos, uma sonda para auxiliar desenvolvedores a testar as suas implementações de APIs, até um testador automatizado para realizar extensivos testes da implementação de APIs e no funcionamento das próprias plataformas de testes [80].

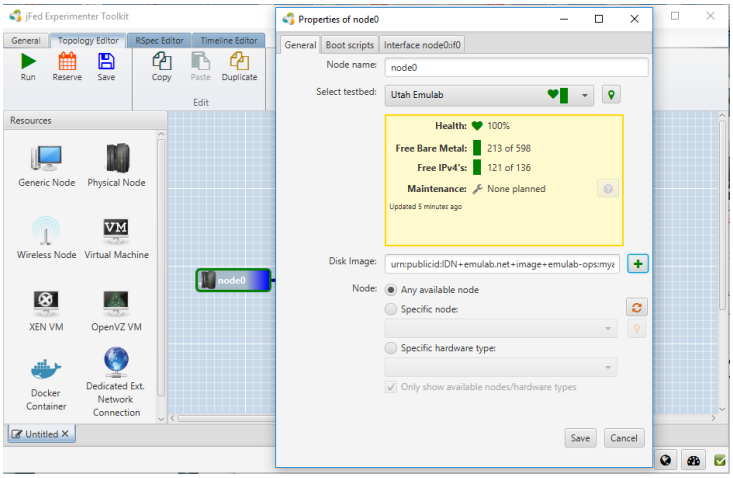
No jFed é possível escolher entre diferentes tipos de recursos aquele que melhor atender as necessidades do experimento elaborado pelo usuário, conforme mostrado na figura 3.5a, e ainda possibilita que estes recursos sejam configurados conforme desejado, como na Figura 3.5b.

Na política atual do GENI, os campus/universidades devem implantar tecnologias como os racks GENI, OpenFlow/SDN e, possivelmente, Worldwide Interoperability for Microwave Access (WiMAX). Atualmente projetos de *racks* podem ser caracterizados em [81]:

- ExoGENI: Uma solução de topologias de redes virtuais flexíveis e de alto custo, incluindo o OpenFlow, que também oferece uma plataforma poderosa para aplicativos



(a) Recursos jFed



(b) Configuração de recursos jFed

Figura 3.5: Utilização do jFed.

de nuvem de vários sites. Geralmente, são implantados como uma parte integrada de uma rede de campus.

- InstaGENI: Apresenta custo intermediário, podendo ser implantada em um grande número de campus, oferecendo suporte a aplicativos de nuvem na Internet, juntamente com redes OpenFlow e Virtual Local Area Network (VLAN).

- OpenGENI: Uma solução expansível de *racks*, implantados em hardware da Dell de custo de médio porte que oferece suporte a aplicativos de nuvem, juntamente com redes OpenFlow e VLAN.

- CiscoGENI: Solução de *rack* que combina software ExoGeni com servidores das séries Cisco UCS-B e C. Esses *racks* suportam OpenFlow, topologias virtuais e aplicativos de nuvem;

- CienaGENI: Solução, ainda em desenvolvimento, de *rack* que combina software ExoGeni com switches Ciena.

3.4.3 OFELIA

O projeto OFELIA [72] começou no outono de 2010 como um projeto de pesquisa financiado pela União Européia no Framework Programme Seven (FP7); foi criado inicialmente com o objetivo de construir, e na segunda fase de interconexão, um conjunto de instalações em campus (ilhas) executando switches habilitados para OpenFlow montando um ambiente experimental, permitindo a pesquisadores não somente experimentação, mas também o controle das redes de forma precisa e dinâmica.

O projeto OFELIA desenvolveu o arcabouço de controle OFELIA Control Framework (OCF) que é utilizado no projeto FIBRE, o OCF foi originalmente criado para cuidar do controle e gerenciamento das plataformas de testes do projeto OFELIA, mas atualmente é utilizado em várias plataformas ao redor do mundo que focam no uso do OpenFlow e no estudo de SDN.

Para monitoração dos experimentos, o OFELIA propõe algumas ferramentas, sendo elas o Cbench[2] e o OFLOPS[3].

Para acessar a plataforma OFELIA, o pesquisador deve criar uma conta OFELIA, configurar uma conexão de rede virtual privada, criar um projeto, definir os recursos necessários e finalmente desenvolver o experimento; esses procedimentos foram realizados para os experimentos em ambiente FIBRE, uma vez que o mesmo se utiliza do OCF para acesso ao ambiente.

[2]Ferramenta de benchmark para controladores OpenFlow
[3]Arcabouço de teste de desempenho para switches OpenFlow

Capítulo 4

A Proposta de Análise

A experimentação com redes OpenFlow apresenta muitas possibilidades que diferem muito em custo, desempenho, flexibilidade, facilidade de implementação, etc. Cada uma das características abordadas impacta de forma diferente nos testes realizados. O uso de uma solução mais veloz no processamento pode fazer diferença em testes que precisem medir o tempo de processamento entre a descoberta de uma falha em uma porta e comutação para a backup, por exemplo. Nesse caso, o uso de um emulador, como o Mininet, pode alterar de forma significativa os resultados.

O intuito desta análise é fornecer insumos para comparar os impactos de cada ambiente de experimentação, não apenas avaliando desempenho mas objetivando verificar sob quais cirscuntâncias cada ambiente diverge, dando parâmetros consistentes para pesquisadores que forem iniciar trabalhos com OpenFlow e precisam escolher o ambiente de testes adequado. Não foram utilizados nos experimentos desta dissertação os simuladores NS-3 e Estinet, o primeiro por possuir algumas limitações didáticas e técnicas, como por exemplo, um módulo OpenFlow que não simula de forma tão fidedigna uma rede OpenFlow real, não possuindo suporte ao tráfego TCP entre o switch e o controlador e ao Spanning Tree Protocol (STP), entre outras limitações. No caso do Estinet, por se tratar de um simulador proprietário, o que inviabiliza seu trabalho acadêmico foi um dos principais fatores, além da escassez de material disponível na internet para estudo e pesquisa e ainda ser pouco conhecido e utilizado para simulações em redes em geral.

A escolha do ambiente de testes precisa ser feita com base nos impactos que o ambiente terá sobre o resultado do experimento. Muitos trabalhos não passam por essa escolha e podem ter resultados invalidados por conta do uso de um ambiente não adequado. Por esse motivo, a avaliação dos ambientes OpenFlow para experimentação é essencial, pois

resulta em uma maior assertividade dos testes realizados quando escolhido de forma adequada para cada solução. Por tal motivo, os experimentos foram realizados em ambientes distintos como emuladores, *testbeds* privados e públicos. Uma vez que o mesmo experimento é realizado nos diferentes ambientes, torna-se possível avaliar o impacto de tal escolha no resultado da experimentação.

Neste capítulo, é apresentada a descrição detalhada dos ambientes de teste utilizados, levando em consideração as características mais impactantes, como o poder de processamento, a quantidade de memória, a capacidade de armazenamento dos equipamentos, o sistema operacional utilizado, a largura de banda e etc.

4.1 Avaliação Prática de Ambientes de Experimentação baseados em OpenFlow

O intuito dos testes realizados por este trabalho é avaliar o comportamento de diversas soluções existentes e prover insumos baseados nos resultados obtidos dos experimentos realizados, aos pesquisadores, para construção de ambientes OpenFlow para testes de novas soluções. Portanto, essa análise requer a avaliação de desempenho de hosts, switches OpenFlow e controlador. Os critérios de avaliação utilizados se baseiam em experimentos que têm enfoque no desempenho do dispositivo OpenFlow. O objetivo principal dos testes é avaliar qual é o impacto do ambiente ou o impacto das ferramentas utilizadas no ambiente ao longo dos resultados do experimento, atentando para os custos despendidos para a concepção dos ambientes utilizados.

Nessa avaliação, optou-se por utilizar pequenos cenários para comparar os testes em todas as plataformas, de forma a evitar a comparação entre ambientes com topologias diferentes. Considerando o switch OpenFlow e o controlador, foi medida a habilidade do dispositivo para tratar requisições simultâneas e distintas, elevado tráfego de dados, diferentes protocolos de transporte e requisições solicitadas pelo controlador. Também foi avaliada a capacidade do host de gerar e receber mensagens. Esta análise de desempenho aponta a maneira de medir a eficiência de cada plataforma de experimentação. Esta análise, no entanto, apenas mostra os impactos em cenários pequenos, o que significa que maiores desvios são esperados em cenários maiores. Essas escolhas são insumos que apontam o caminho para distinguir a eficiência de cada plataforma de experimentação estudada.

Foram então, considerados alguns parâmetros de análise em todos os ambientes de

experimentação utilizados nesta dissertação, de forma a mensurar qualitativamente os ambientes e assim enriquecer de insumos os pesquisadores para a escolha apropriada de ambiente para uso.

São avaliados parâmetros de desempenho do switch no qual o plano de dados é o foco da análise, avaliação do hardware utilizado para o controlador que influenciará no comportamento do plano de controle, e para os hosts que influenciará na geração de tráfego e criação de pacotes.

Itens de interesse para pesquisadores como a celeridade para configurar ambientes, facilidade de uso e acesso dos ambientes, flexibilidade em realizar mudanças no ambiente, capacidade de repetibilidade dos testes no ambiente, no qual verifica-se a variabilidade nos resultados ao longo de experimentações no decorrer do dia, e os custos de montagem do ambiente, que muitas vezes é um fator decisivo na escolha do ambiente, são aferigados ao longo desta dissertação.

4.2 Topologia dos Testes

Para realização dos testes, foi projetada uma topologia para atender emuladores, testbeds e ambientes físicos igualmente, levando em consideração os limites financeiros existentes para realização desta dissertação mas que permitisse a implantação de um ambiente mínimo para experimentação que servisse de base para entendimento dos efeitos dos resultados em topologias maiores. Consequentemente, o tamanho do cenário de teste foi dimensionado baseado nas plataformas privadas, que apresentam maiores gastos financeiros para obtenção do hardware, limitando a quantidade de hosts e switches em todos os ambientes. A idéia é analisar resultados observando os fatores que podem causar divergências nos dados obtidos em cada ambiente.

Dadas todas essas considerações, a topologia utilizada consiste de 2 hosts, 1 controlador Ryu e um switch OpenFlow, conforme ilustra a Figura 4.1.

Esta é a topologia mínima para uma avaliação da rede baseada em OpenFlow, que atende aos requisitos dos testes planejados e, ao mesmo tempo, é de fácil implementação. Usando duas máquinas como cliente, é possível realizar diferentes testes de comunicação entre os pares de clientes, focando tanto no plano de dados quanto no plano de controle de cada ambiente. Como não é de interesse manter o switch com pouco tráfego, é utilizado somente um switch OpenFlow para cada ambiente de experimentação. Porém, objetivando conhecer a sobrecarga de processamento imposta pelo uso do software OvS,

múltiplas instâncias de switch OpenFlow foram configuradas numa mesma máquina física em um dos cenários de testes, verificando até que ponto o processamento do hardware adotado consegue sustentar bons resultados.

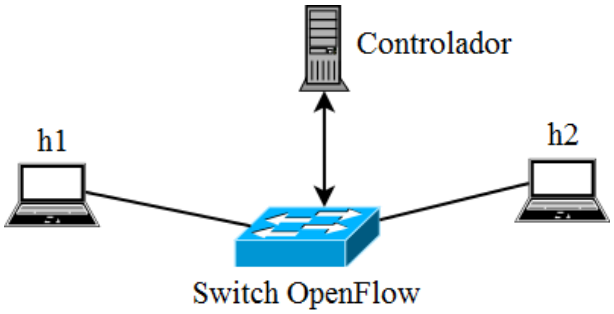


Figura 4.1: Topologia básica de avaliação.

Um fator chave a considerar é o controlador a ser usado no teste. Naturalmente, o ambiente deve ser capaz de dar suporte semelhante a qualquer um dos controladores existentes. Assim sendo, optou-se pelo uso do Ryu pela sua simplicidade e bom desempenho, já que a análise de diferentes controladores não é o foco deste trabalho, mas sim o ambiente de experimentação que dá suporte a um controlador qualquer. O Ryu é escrito em Python, linguagem de alto nível e de fácil implementação [24], e é um controlador OpenFlow *open source* popular. Além do citado, sua escolha neste trabalho levou em consideração seu constante desenvolvimento, com documentação atualizada e de fácil acesso através de uma plataforma Git.

4.3 Configuração dos testes

Os testes duraram 20 segundos, tempo suficiente para realizar as ações de criação de pacotes, encaminhamento de tráfego e monitoração de dispositivos. Os resultados são apresentados com um intervalo de confiança de 95%. Os testes foram realizados inicialmente com foco no protocolo OpenFlow versão 1.0 e, posteriormente, na versão 1.3, a fim de tornar plausível a análise de desempenho de ambas as versões do protocolo OpenFlow [82]. Embora nem todos os ambientes deem suporte ao OpenFlow 1.3, entende-se que essa especificação adicionou significativas funcionalidades ao protocolo [61], como a possibilidade de múltiplas tabelas de fluxos, tendo se tornado uma versão mais popular que a 1.0 [83].

Os testes foram realizados em diferentes ambientes de experimentação OpenFlow:

- Emulação com Mininet versão 2.2.1 e 2.2.2 - em ambos os casos considerando o

protocolo OpenFlow versão 1.0 através do OvS 2.0.2 e o protocolo OpenFlow versão 1.3 através do OvS 2.3.2. Outra característica analisada foi comparar a utilização do controlador Ryu configurado internamente no próprio ambiente emulado do Mininet e o controlador Ryu instalado numa máquina física externa conectada ao Mininet;

- Plataforma de teste privada usando o OvS instalado em desktop - neste caso foi feita a instalação do OvS num mesmo hardware modificando a quantidade de memória disponível entre 4 GB Random Access Memory (RAM) e 8 GB RAM, além de posteriormente a instalação do OvS em outro hardware com melhor capacidade de processamento e memória. Assim como no caso do Mininet, os testes foram realizados nas versões de OvS 2.0.2 e OvS 2.3.2. De forma complementar, foi analisado o desempenho de múltiplos OvS 2.3.2 configurados num mesmo hardware;

- Montagem em plataforma de teste privada usando um switch comercial, especificamente o Pronto (Pica8 - P3295) [84];

- Avaliação em plataforma pública FIBRE com suporte ao OpenFlow 1.0, focando a análise tanto no comportamento ao utilizar somente as placas Networking Field-Programmable Gate Array (NetFPGA), quanto ao utilizar o switch Pronto ;

- Avaliação em plataforma pública GENI, com foco em servidores físicos e em servidores virtuais XEN.

Foi utilizado dois hosts em cada um dos ambientes de experimentação citados, no qual os hosts funcionaram como cliente e servidor criando a possibilidade de testes relacionados a vazão e criação de pacotes através do uso de ferramentas apropriadas para o experimento instaladas nessas máquinas.

A estrutura física desempenha um papel fundamental nos resultados. Máquinas compostas por melhores processadores e memória tendem a realizar resultados superiores nas emulações/testes. Por esse motivo, buscou-se usar a mesma configuração de hardware no ambiente de instalação da máquina virtual Mininet, no switch OvS utilizado na plataforma de teste privada e no hardware utilizado para o controlador Ryu. Qualquer mudança neste hardware dificulta a avaliação justa dos experimentos. A utilização de hardware mais robusto foi adotada somente nas plataformas de testes privadas com instalação de OvS em desktop para de fato mensurar a influência do hardware nos resultados obtidos em testes similares.

Outra característica foi manter o mesmo sistema operacional nas máquinas que possuiam interface gráfica e na máquina física utilizada para instalar o software de virtualização onde foi colocado o Mininet, desta forma restringindo a influência do sistema operacional nos testes realizados.

4.3.1 Configuração Mininet

Os testes baseados em emulação foram feitos utilizando o Mininet 2.2.1 e 2.2.2 (Máquina virtual obtida no site mininet.org), instalado na plataforma *VirtualBox*, conforme orientação obtida no site oficial dos desenvolvedores deste emulador. O fato de usar um software de virtualização para implementar o Mininet, pode impactar negativamente ao comparar seus resultados com ambientes que possuem máquinas exclusivamente dedicadas para uso de experimentos.

No *VirtualBox* foi usado 2G RAM e 4vcpu, localizado em desktop com 4GB RAM, 55GB HD e sistema operacional XUBUNTU 14.04.1, 32bits i686, Intel(R) Core(TM) i7-2600 3.40GHZ (1 placa com 4 núcleos e 8 threads).

Conforme já citado, para realizar testes nas versões do protocolo OpenFlow 1.0 e 1.3 foram utilizadas duas versões distintas de OvS, sendo a versão 2.0.2 atendendo somente ao OpenFlow versão 1.0 e a versão OvS 2.3.2 atendendo ao OpenFlow versão 1.3. A escolha pela versão de OvS 2.3.2 ao longo dos experimentos, não somente em ambiente Mininet, como em outros ambientes de experimentação, foi devido limitações do ambiente Mininet 2.2.1, que possui a configuração de Kernel 3.13.0-24-generic e versões de OvS acima da 2.3.2 necessitam de Kernel a partir da versão 4.0.x.

As configurações de ambos ambientes Mininet são:

- Mininet 2.2.1: Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)
- Mininet 2.2.2: Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

O hardware utilizado para os testes de uso do controlador Ryu externo ao ambiente Mininet possuía as seguintes configurações:

- Intel ® Core i5-M480 2.67 GHz (1 placa com 4 Cores e 8 threads), 4 GB RAM, 500GB of HD, XUBUNTU 14.04.01, 32 bit, i686.

Detalhes do código para implementação do ambiente Mininet pode ser verificado no Apêndice A.

4.3.2 Configuração plataforma de testes privados

Diferentes cenários referente a testes de plataforma privada foram realizados e dessa forma hardwares distintos foram utilizados.

Cenário 1:

- Switch Comercial Pronto Pica8 - P3295 e hosts baseados na placa Mini-TX dual-core Intel® Atom Processor N2800 (2 núcleos e 4 threads), 2 GB RAM, XUBUNTU 14.04.01, 32 bits, i686, que desoneraram os custos financeiros para realização das avaliações. Estes hosts foram utilizados para testes referente a análise de jitter e vazão. Nos testes no qual a máquina cliente necessitava gerar grande volume de pacotes diferentes, foi utilizado um hardware mais robusto com a seguinte configuração: Intel ® Core i5-M480 2.67 GHz (1 placa com 4 Cores e 8 threads), 4 GB RAM, 500GB of HD, XUBUNTU 14.04.01, 32 bit, i686.

Cenário 2:

- Testes com OvS 2.0.2 e testes com OvS 2.3.2 instalado em desktop com 55GB HD e sistema operacional XUBUNTU 14.04.1, 32bits i686, Intel(R) Core(TM) i7-2600 3.40GHZ (1 placa com 4 núcleos e 8 threads). Alterou-se a memória dessa máquina entre 4 e 8 GB de RAM para avaliar o impacto causado pela memória. Os hosts utilizados seguiram a mesma premissa do Cenário 1.

Em ambos os cenários citados, foi utilizado o controlador Ryu em desktop com 4GB RAM, 55GB HD, SO: XUBUNTU 14.04.1, 32bits i686, Intel(R) Core(TM) i7-2600 3.40GHZ (1 placa com 4 núcleos e 8 threads).

Cenário 3:

Para este terceiro cenário, foram utilizadas quatro máquinas mais robustas em todo o contexto do experimento, isto é, nos clientes (h1 e h2), Ryu e OvS, de forma a ser realizada a comparação de desempenho dos resultados apresentados no cenário 2, com uso de máquinas mais simples, com o cenário 3 de hardware mais sofisticado. Segue a configuração máquinas utilizadas:

- Desktop com 1TB HD e sistema operacional XUBUNTU 14.04.1, 32bits, Intel(R) Core(TM) i7-7700 3.60GHZ, 16 GB RAM.

Cenário 4:

Outro cenário de análise foi verificar o comportamento de várias instâncias de OvS numa mesma máquina física, conforme Figura 4.2. Esta pode ser um alternativa interessante para pesquisadores que necessitam fazer testes com topologias que possuem diversos switches OpenFlow, porém não possuem hardware suficiente para tal implementação. Dessa forma, tal cenário serve como insumo para indicar os limites de desempenho, perda de pacotes , processamento e outros, ao colocar diversas instâncias de OvS numa única máquina. Foram realizados testes com duas até quatro instâncias de OvS 2.3.2 em desktop com 55GB HD, sistema operacional XUBUNTU 14.04.1, 32bits i686, Intel(R) Core(TM) i7-2600 3.40GHZ (1 placa com 4 núcleos e 8 threads) e 8 GB de RAM. As configurações das máquinas de usuários e a máquina do Ryu apresentam configuração igual a do cenário 1, de forma a poder comparar os seus resultados com o uso de uma instância de OvS, com este cenário 4.

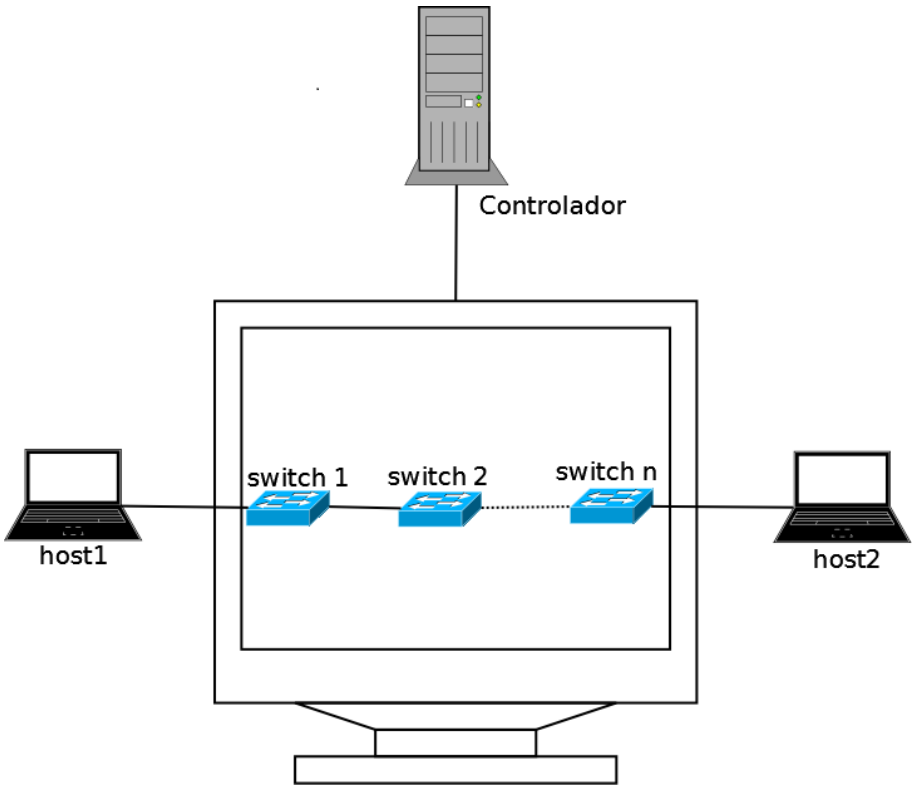


Figura 4.2: Ambiente com múltiplas instâncias de OvS em mesma máquina física.

A implementação lógica de vários switches OvS pode ser verificada em detalhes no Apêndice B.

4.3.3 Configuração da plataforma de testes públicos: FIBRE

Os testes utilizando o FIBRE contaram com equipamentos OpenFlow baseados em placas NetFPGA e switches Pronto. Os hosts criados possuem sistema operacional GNU/-Linux Debian 6.0 e são executados em máquinas virtuais criadas pela própria plataforma, com configuração Linux 3.2.0-4-amd64 SMP Debian 3.2.65-1+deb7u2 x86_64, com 120 MB de memória, 8 GB de HD e um único processador do tipo Intel(R) Xeon(R) CPU E5-2650 2.00Ghz.

Para utilização deste ambiente foi necessário a criação de uma conta associada à Rede de Nacional de ensino e Pesquisa (RNP), que foi essencial no apoio aos testes realizados. Além disso, é necessária a utilização do software OpenVPN, software de código aberto para criar redes privadas virtuais do tipo ponto-a-ponto. A criação de um slice para poder realizar os experimentos citados nesta seção pode ser visto em detalhes no Apêndice C.

O foco dos testes nesta plataforma pública foi verificar se essas funcionam de forma tão eficiente quanto as plataformas privadas. Dentro dessa plataforma pública, buscou-se comparar o comportamento dos diferentes switches OpenFlow disponíveis, no caso switches Pronto e switches baseados em NetFPGA.

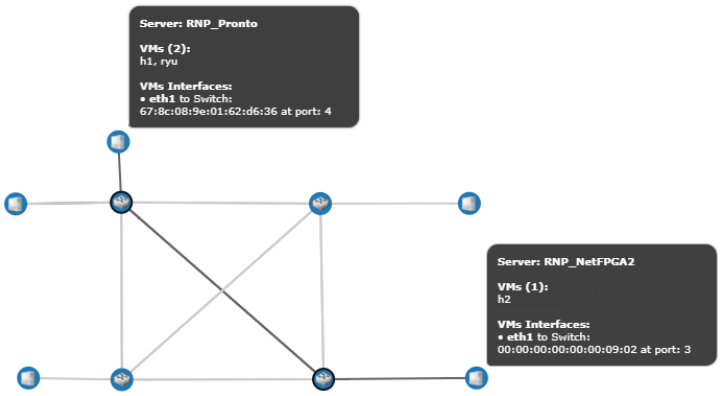
Dessa forma, dois cenários de testes foram elaborados. No primeiro caso, com foco no switch Pronto como pode ser visto na Figura 4.3a. No segundo caso, com foco somente nas máquinas com placas NetFPGA conforme Figura 4.3b.

Ao conseguir a reserva de uma fatia do ambiente provido pela ilha RNP, uma ou mais VLANs, são disponibilizadas para o experimento, por este motivo foi necessário configurar manualmente a VLAN nas interfaces ethernet de cada máquina virtual criada, conforme passo a passo apresentado no Apêndice B.

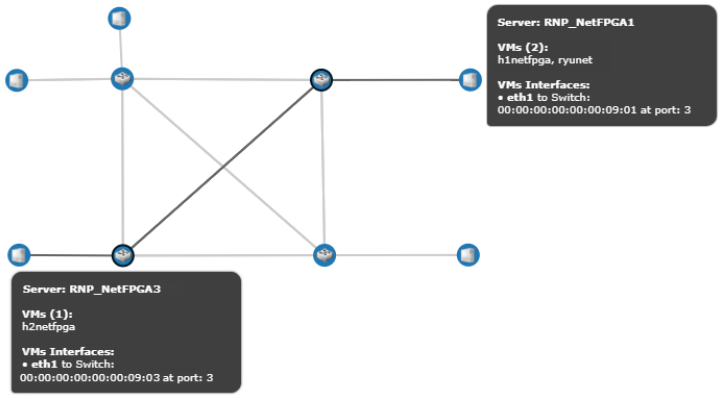
4.3.4 Configuração plataforma de testes públicos: GENI

No ambiente de experimentação GENI, foi necessário que a orientadora deste trabalho solicitasse um projeto, no qual ela fosse o líder e o autor deste trabalho um dos membros. O projeto foi intitulado de OpenFlowPerformance e duas fatias foram criadas, com os nomes de "ambfisico"e "ambxen", ambas utilizando agregados distintos da Universidade de Utah Downtown Data Center (UtahDDC).

As duas fatias criadas dentro do projeto podem ser vistas na Figura 4.4. Seguindo as premissas dos outros experimentos, foram utilizadas quatro máquinas, sendo uma para



(a) Ambiente FIBRE com switch Pronto.



(b) Ambiente FIBRE com máquinas NetFPGA.

Figura 4.3: Topologia ambiente FIBRE.

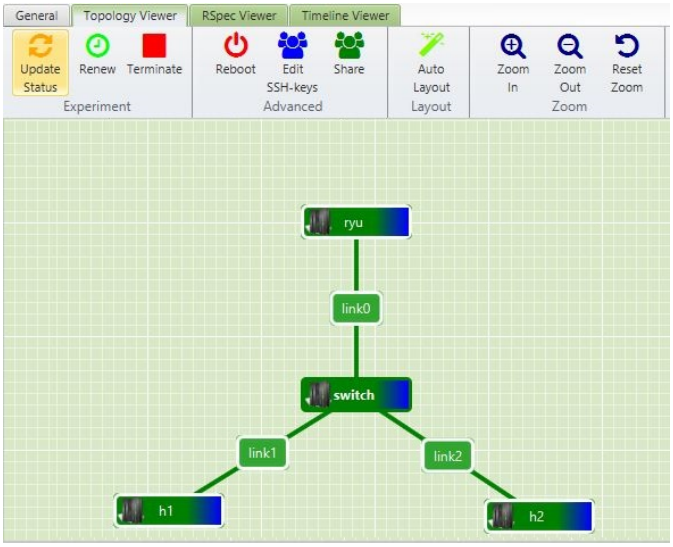
controlador, outra para instalação do switch OvS 2.0.2 e duas máquinas simulando o uso por usuários. A distinção destas fatias foi o tipo de máquinas provisionado, sendo um ambiente composto de máquinas físicas e o outro composto de máquinas virtuais hospedadas no hipervisor Xen.

O ambiente de máquinas virtuais no hipervisor XEN foi elaborado na plataforma InstaGENI Utah DDC e possuia a seguinte configuração:

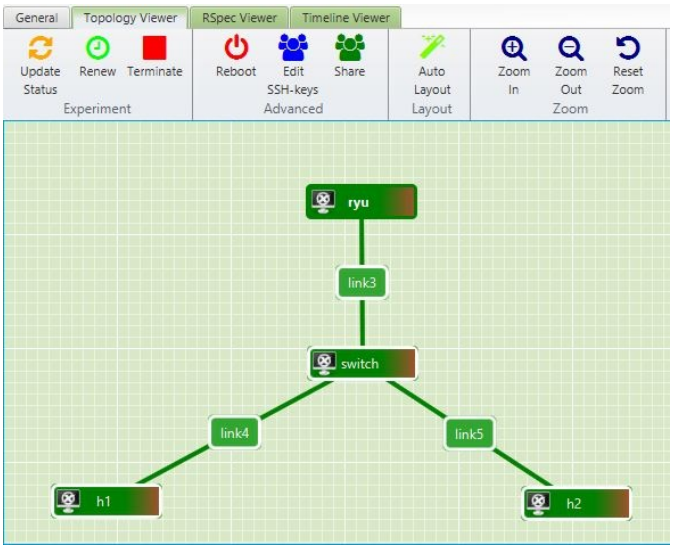
- Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-100-generic x86_64), 2GB de RAM, 16GB de HD, 2 CPU do tipo Intel(R) Xeon(R) CPU E5-2450 2.10GHz, links de comunicação de 1Gbps. Uma capacidade de processamento maior por máquina foi selecionada pelo usuário, porém a instância não possuía o recurso disponível.

O ambiente de máquinas físicas foi elaborado na plataforma Utah Emulab e possuia a seguinte configuração:

- Ubuntu 14.04.1 LTS 64bits (GNU/Linux 3.13.0-117-generic x86_64), 2GB de RAM, 16GB de HD, 2 CPU do tipo Intel(R) Xeon(R) CPU 3.00GHz. Neste caso foi necessário configurar os links de comunicação para 1Gbps, pois o padrão de configuração era de 100Mbps.



(a) Cenário de máquinas físicas.



(b) Cenário de máquinas virtuais.

Figura 4.4: Cenários utilizados.

4.3.5 Configurações Gerais

O aplicativo usado no controlador simula um switch padrão, definindo entradas de fluxo com base apenas no MAC de destino. Não faz parte do objetivo deste trabalho testar

a eficiência do controlador, por isso não foram feitos testes com diferentes controladores.

A aplicação de switch padrão utilizada para testes em OpenFlow versão 1.0 e versão 1.3 pode ser analisada em detalhes no Apêndice A.

Em todos os ambientes, para simulações que necessitam de limitação da largura de banda, são utilizados scripts bash configurados e armazenados dentro das máquinas clientes com as premissas necessárias para limitação da banda de tráfego, conforme pode ser visto no Apêndice B.

Capítulo 5

Resultados Obtidos

Neste capítulo, são detalhadas as experimentações realizadas para avaliação dos diversos ambientes OpenFlow avaliados neste trabalho, com seus respectivos resultados. A estrutura do capítulo separa os testes por módulos de análise, focando principalmente nos objetivos de comparação entre os ambientes, detalhando nos resultados as diferenças detectadas, seja devido à versão do protocolo OpenFlow utilizado, o hardware empregado ou o tipo de configuração implementada.

Foram realizados testes de carga nos quais o foco era a vazão e o impacto dos protocolos de transporte [85], da utilização de ferramentas de monitoração nas máquinas clientes e do tamanho máximo de segmento Maximum Segment Size (MSS). Também foi analisada a criação e uso de enlaces do plano de dados para as faixas de 10 Mbps e 1 Gbps, a capacidade das máquinas clientes na geração de dados e o comportamento do ambiente de experimentação para tamanhos de MSS diferentes[1]. Tamanho menores de segmento de pacote, ocasiona a geração mais intensa de pacotes, pelo cliente, para preencher um mesmo enlace que necessitaria de menos pacotes em caso de elevado tamanho de segmento. Dessa forma a geração de mais pacotes, acarreta mais uso de processamento, elemento de avaliação neste trabalho.

Um outro tipo de teste foi a geração de pacotes diferentes para verificar o comportamento do plano de controle devido a geração de diversos eventos de packet_in pelos switches, gerando grande carga no controlador. Para este tipo de teste, que foi feito utilizando pacotes Address Resolution Protocol (ARP), foi necessário adaptar a API de um

[1]Foi utilizado 536 e 1460 B como tamanho de segmento, pois seguindo as premissas da Request for Comments (RFC) 879 [86], o tamanho máximo de segmento corresponde a 1460 bytes, oriundos dos 1500 bytes do Maximum Transmission Unit (MTU) subtraído dos valores de 20 bytes do cabeçalho IP e 20 bytes do cabeçalho TCP; e como MSS TCP padrão é estipulado o valor de 536 B considerando os mesmos valores subtraídos já citados anteriormente ao conceituar o valor mínimo de MTU igual a 576 B.

switch padrão existente no controlador Ryu, de forma a definir as entradas de fluxo com base não apenas no MAC de destino, mas também no MAC de origem, como pode ser visto no Apêndice A. Os testes foram realizados sem o conceito de multithread, dessa forma somente um processador de cada máquina cliente foi usado para a criação dos pacotes, isto foi considerado pelo fato das máquinas virtuais oriundas da plataforma FIBRE possuírem somente um processador.

Por fim, testes de tráfego de carga User Datagram Protocol (UDP) [87] foram realizados para verificar o atraso na entrega de dados em uma rede OpenFlow.

5.1 Software Utilizados

Para realizar os testes, foi necessária a utilização de alguns softwares de funções específicas, instalados nas máquinas pertinentes e no servidor do controlador conforme necessário:

- Gerador de tráfego Iperf (V2.0.5) [88]: Software utilizado para testar a largura de banda, podendo realizar injeção de pacotes para medir o desempenho de redes de computadores
- Gerador de pacotes Scapy (V2.2.0) [89]: Ferramenta de manipulação de pacotes de rede, escrita em Python, que provê classes para forjar ou decodificar pacotes de diversos tipos de protocolos; enviá-los e capturá-los pela rede; comparar requisições e respostas; e traçar rotas de verificação do tráfego.
- Wireshark [90] e Tcpdump [91]: São analisadores de protocolos de rede, sendo o primeiro com interface gráfica e o segundo por linha de comando, que permite que o usuário capture o tráfego de uma rede de computadores em tempo de execução usando a interface de rede do computador.

5.2 Testes em OpenFlow versão 1.0 e ambientes distintos de experimentação

O primeiro teste realizado considera ambientes de experimentação OpenFlow que utilizem a versão 1.0 do protocolo e avalia o impacto de processamento dos dispositivos sobre os resultados do teste de rede, como a plataforma de teste pública FIBRE e o ambiente de emulação Mininet 2.2.1 fazem parte desta primeira análise, foi ocultado a

representação dos resultados da outra plataforma pública GENI e do emulador Mininet 2.2.2 neste primeiro momento.

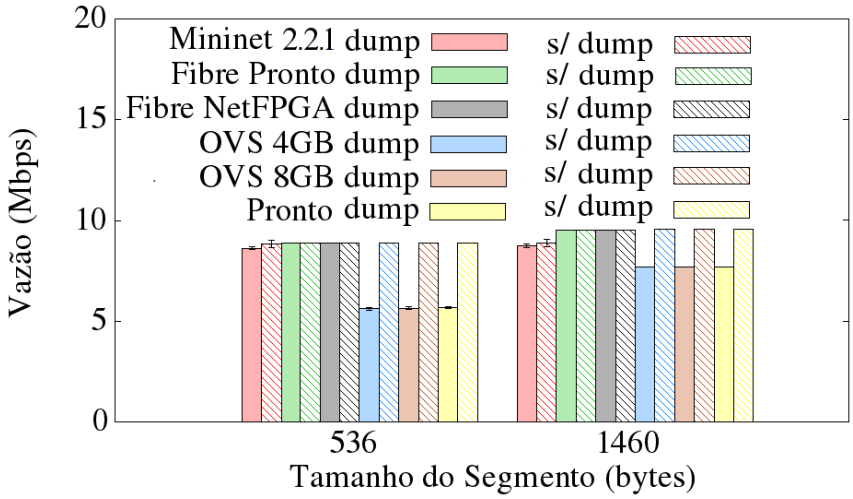
Para a realização dos experimentos, gerou-se um tráfego TCP com o Iperf entre as máquinas h1 e h2, representadas na Figura 4.1. Para avaliar a capacidade de processamento tanto das máquinas hosts como dos switches, variou-se o tamanho do segmento entre 536 e 1460 B, pois segmentos menores levam a uma geração mais intensa de pacotes. Variou-se a banda do enlace entre 10 Mbps e 1 Gbps para variar o número de pacotes gerados/encaminhados.

Outra análise realizada foi o impacto do uso do tcpdump como ferramenta de monitoração. Em testes de avaliação, é desejável usar softwares para monitoramento, mas não é desejável que esses softwares prejudiquem os resultados. Contudo, o uso de analisadores de pacotes é mais um fator que gera sobrecarga na máquina e que tem influência no resultado. O tcpdump foi colocado em ambas as máquinas clientes h1 e h2. A aplicação usada no controlador é a que simula um switch padrão conforme detalhado no Apêndice A, definindo entradas de fluxos com base apenas no MAC de destino.

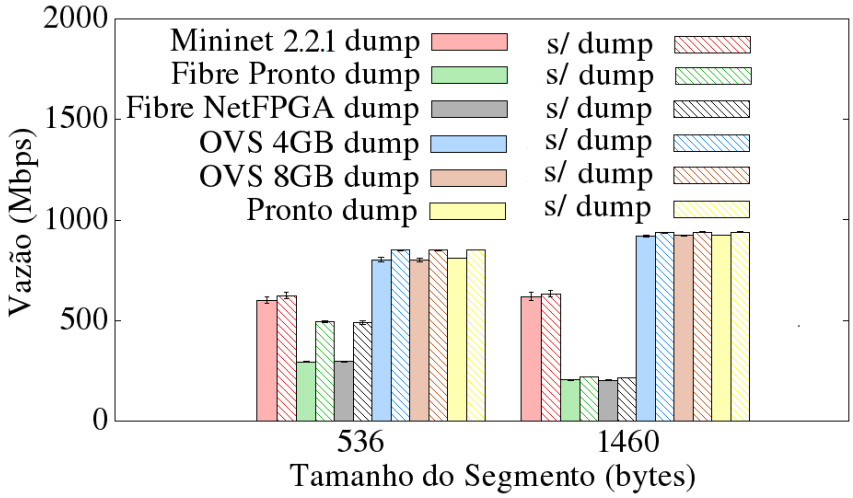
No caso das simulações realizadas em ambiente de laboratório real, representado pelo switch Pronto e OvS instalado em desktop físico, a utilização do tcpdump influencia consideravelmente na vazão obtida, apresentando maior impacto proporcional em enlaces de menor capacidade. Essa característica detectada está relacionada ao hardware utilizado para os hosts no qual foi necessário fazer uso de um arquivo bash para a aplicação de traffic control e assim limitar a interface para uso de no máximo 10Mbps, porém em contrapartida a execução deste script impactou no desempenho da máquina na geração de tráfego. De fato, o equipamento utilizado como host, que foi escolhido visando um custo financeiro mais baixo, acabou proporcionando um processamento aquém do desejável.

Em contrapartida, nos ambientes Mininet e FIBRE, o comportamento foi inconsistente com o esperado. É importante salientar que ambos são ambientes onde não é possível ter controle sobre sua composição, uma vez que a plataforma FIBRE é compartilhada com outros pesquisadores em quantidade não verificável, que podem usar o ambiente no mesmo momento, impossibilitando controlar a banda disponível e o ambiente Mininet pode sofrer influências de outras tarefas do sistema operacional, o que pode ter ocasionado o resultado similar do experimento com e sem o uso da ferramenta de monitoração.

O mesmo experimento sendo realizado em momentos distintos podem resultar em valores diferentes conforme pode ser visto na Figura 5.2, que apresenta o resultado no ambiente FIBRE para links de 1 Gbps quando realizado os testes para o Simpósio Bra-



(a) 10 Mbps



(b) 1 Gbps

Figura 5.1: Observação da influência dos valores de tamanho máximo de segmento e uso de monitoração via tcpdump.

sileiro de Redes de Computadores (SBRC) 2017 [92], representado pela legenda "Fibre SBRC", e os resultados quando foram novamente realizados, especificamente para esta dissertação, tratando-se de instantes de tempo distintos.

Na Figura 5.1a, o uso de tcpdump em nada influenciou o ambiente FIBRE e, na Figura 5.1b pacotes com tamanho de segmento máximo menor, inesperadamente, apresentaram maior vazão. O ambiente FIBRE que se assemelha com um ambiente de uso real, no qual existe pouca controlabilidade dos clientes que o utilizam e a largura de banda e o atraso geralmente variam devido a outro tráfego não controlado. É importante notar que é possível reservar todo o link, mas depende de uma autorização especial de gerentes da plataforma de testes. O fato de executar os testes de maneira sequencial, ou seja, ini-

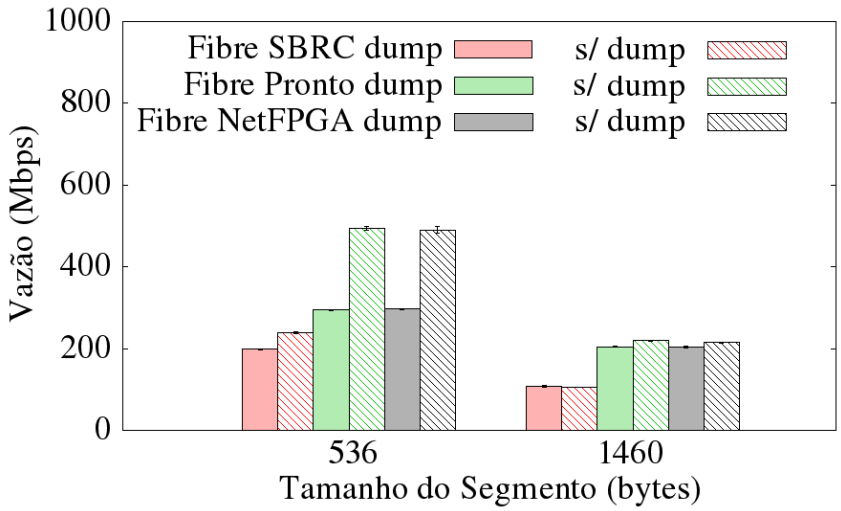
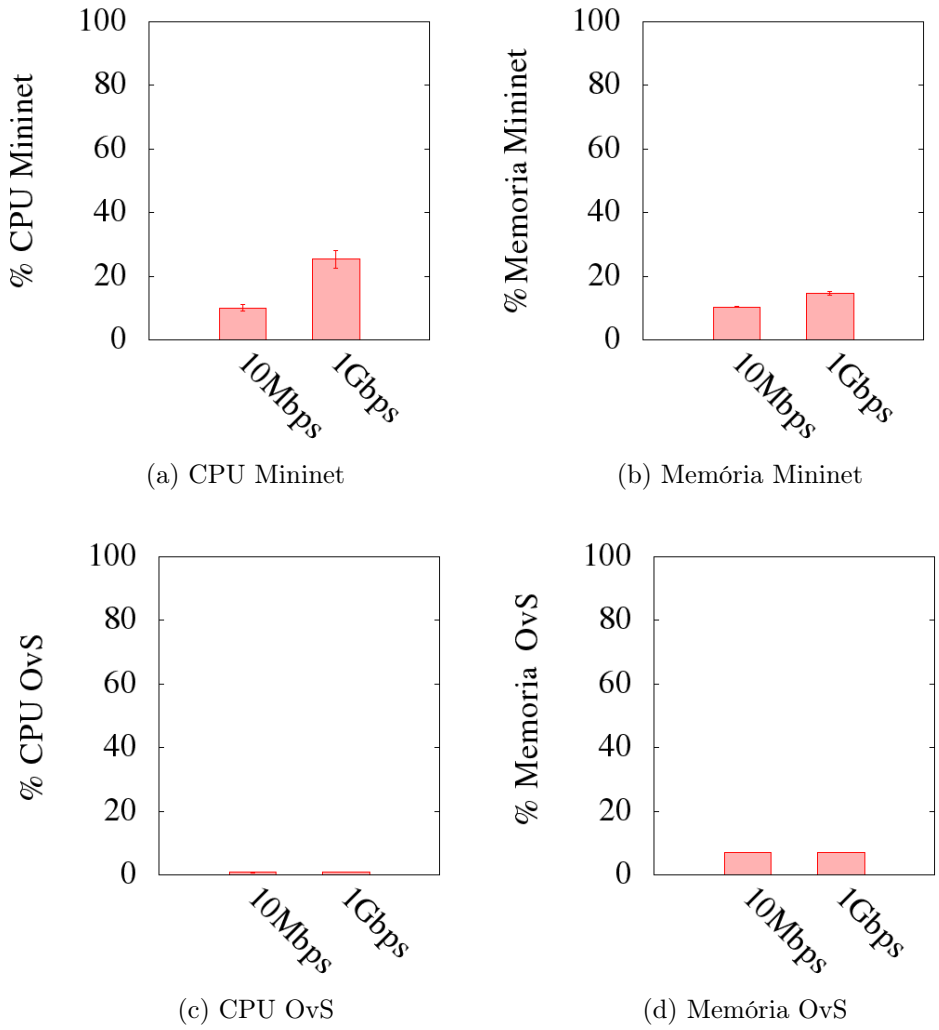


Figura 5.2: Influência nos resultados dos testes, em momentos distintos da plataforma FIBRE.

ciando o experimento com o tamanho máximo de segmento menor, e variando largura de banda, e uso de tcpdump e somente posteriormente a mesma sequência de testes porém com uso de tamanho máximo de segmento maior, pode ter ocasionado o resultado encontrado. Assim, ao utilizar este tipo de ambiente de experimentação é importante executar os testes de forma mesclada, para aumentar a probabilidade que os diferentes cenários sejam expostos às mesmas condições externas e de preferência por um longo período de tempo, como por exemplo um mês, tornando mais equânime os resultados obtidos, no qual a característica de instabilidade da plataforma pode ser analisada em mais detalhes.

No Mininet, o valor de tamanho de segmento máximo e o uso de tcpdump, considerando a margem de erro, em nada influenciaram na vazão dos testes de Iperf. No entanto, não conseguiu alcançar o uso do link de 1 Gbps, o que pode ser explicado devido alguma limitação de software, como uma possível deficiência na emulação do canal ao configurar o enlace para funcionamento em 1 Gbps em apenas 20 segundos de emulação. Conforme mostrado na figura 5.3a e 5.3b, através do uso do comando TOP do Linux utilizado para monitorar a carga do sistema, o uso de processamento no Mininet é maior que no caso de OvS instalado em máquina física, porém não chega ao total de 100%. Eliminando a possibilidade do processamento ser o limitador do uso do link de 1 Gbps.

Todavia, ao se estressar o ambiente Mininet sem limitar a sua largura de banda, observou-se a limitação causada pelo uso da ferramenta de monitoração tcpdump, sendo que obteve-se vazão de 9 Gbps sem uso de tcpdump e de 4.5 Gbps com o uso de tcpdump. A criação de enlaces pelo Mininet é feita através do módulo "TCLink", e os experimentos realizados nesta dissertação indicam que este módulo de *traffic control* não consegue



(a) CPU Mininet (b) Memória Mininet

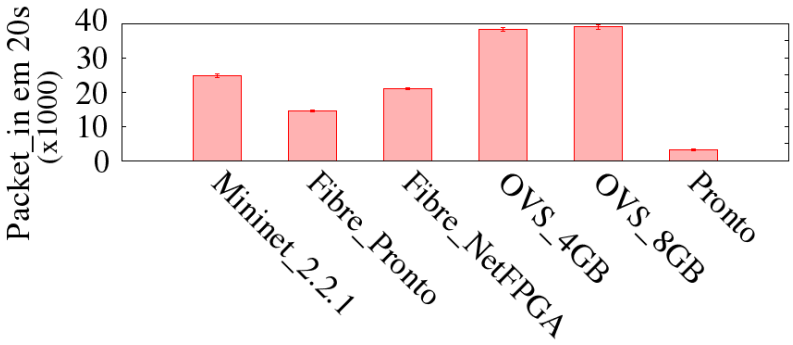
(c) CPU OvS (d) Memória OvS

Figura 5.3: Análise de CPU e Memória.

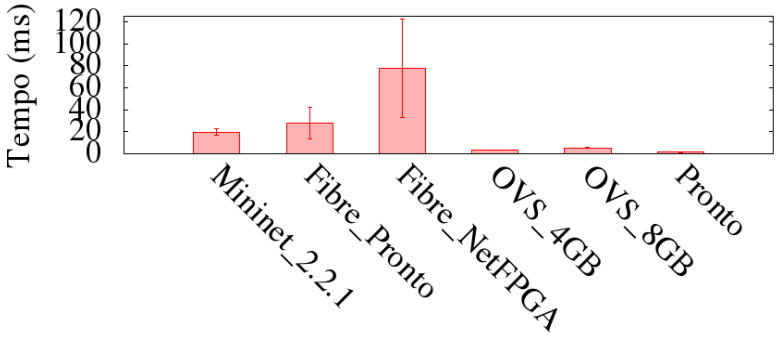
limitar corretamente o tamanho do enlace, com destaque para valores a partir de 1 Gbps, sendo aconselhado para melhor estabilidade deste módulo do emulador a realização de cada teste por mais de um minuto. Esta análise é um indicativo importante, pois, no Mininet, a capacidade de processamento é dividida pelos elementos que se colocam na emulação. Se fossem usados mais hosts ativos ou mais switches; links com 1 Gbps, ou até menos, seriam bastante afetados, pois teria em conjunto com o módulo TCLink a questão do processamento afetando o desempenho, causando uma perda considerável na capacidade do enlace.

A próxima análise realizada foi relativa aos impactos sobre o plano de controle dependendo da plataforma de experimentação que está em uso. Especificamente, buscou-se aumentar o número de eventos de packet_in, que são gerados todas as vezes que o switch recebe um pacote para o qual não existe entrada de fluxo. A quantidade de eventos ge-

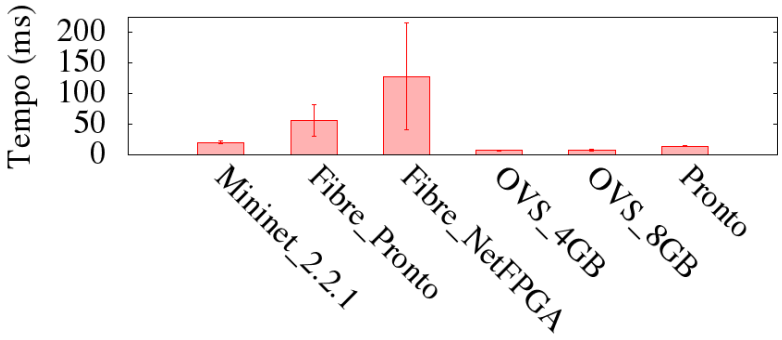
rados é impactada pela capacidade da máquina de gerar pacotes com endereços MAC diferentes, pela capacidade de processamento do switch e pela capacidade de resposta do controlador. Os pacotes são gerados no host 1 utilizando a biblioteca SCAPY em todos os ambientes, gerando pacotes ARP para o host 2, variando o MAC de origem. Para medir a quantidade de packet_in gerados e devidamente processados, foi habilitado o uso de wireshark em h1, h2 e controlador.



(a) Eventos de packet_in processados pelo switch.



(b) Tempo de processamento do plano de controle.



(c) Tempo de requisição ARP entre h1 e h2.

Figura 5.4: Análise do processamento de eventos de packet_in nos diversos ambientes de teste.

O host 1 de configuração mais robusta conforme citado na seção 4.3.2 cenário 1 (Intel ® Core i5-M480 2.67 GHz (1 placa com 4 Cores e 8 threads), 4 GB RAM, 500GB of

HD, XUBUNTU 14.04.01, 32 bit, i686), montado para os testes de OvS em desktop e switch Pronto foi o que gerou maior quantidade de pacotes ARP, em média 39000 pacotes ARPs, representado na Figura 5.5 com H1_Robusto, indicando que um desktop com configurações mais robustas favorece experimentos que dependem do desempenho de máquinas que geram pacotes. Nesta mesma figura com a legenda de H1_Mini-itx é apresentado o total de pacotes ARP gerados pelas Mini-itx, num valor aproximado de somente 7500 pacotes, por este motivo não sendo utilizado este computador para tal experimento de geração de pacotes ARP.

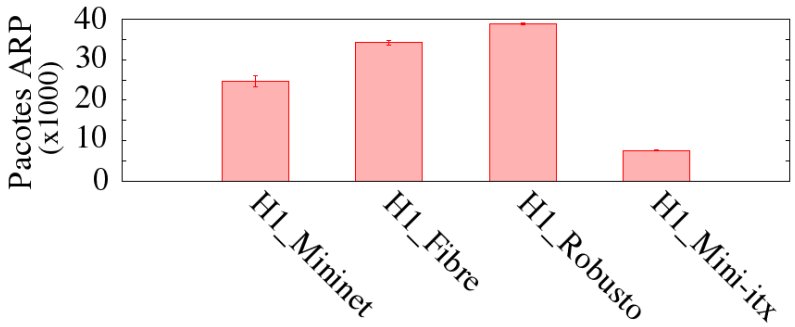


Figura 5.5: Total de pacotes ARP gerados pelos clientes (h1).

Conforme apresentado na Figura 5.4a, nos testes de OvS em desktop, os pacotes ARP ocasionaram a mesma quantidade de packet_in para o controlador Ryu. Para esse teste, o ambiente OvS em desktop obteve o melhor resultado. Porém no caso dos testes em switch Pronto, observou-se um comportamento anômalo, pois os pacotes ARP gerados em h1 não criaram a mesma quantidade de packet_in pelo switch. Assim, conclui-se que houve perda de pacotes ARP, que não foram convertidos em eventos de packet_in. Isso, contudo, não representa um problema do ambiente de teste privado onde era esperado um excelente desempenho, mas sim um problema de firmware de um switch comercial que não estava em sua versão mais atual. A máquina virtual do Mininet criou em média 25000 packet_in provenientes de 25000 chamadas ARP, não havendo perda de pacotes ao longo do processo.

O host h1, montado via plataforma FIBRE, criou em média 34000 pacotes ARP, porém devido a perda de pacotes ARP a quantidade de packet_in proveniente do switch Pronto para o controlador Ryu foi inferior assim como nos provenientes do experimento com foco nas máquinas NetFPGA, indicando que estes ambientes impactaram negativamente o processamento no plano de controle.

Tais testes foram mensurados através da análise do arquivo wireshark gerado no host h1, verificando a quantidade de pacotes ARP gerados (com MAC de origem distintos) e

no arquivo wireshark gerado no controlador, verificando a quantidade de packet_in que chegaram no mesmo. Como no caminho entre ambos só há o switch, a quantidade de packet_in menor que a quantidade de pacotes ARP gerados indica a perda de pacotes. Tal perda ocasionada pelo baixo desempenho do elemento OpenFlow analisado em tratar grande quantidade de requisições.

A Figura 5.4b mostra a análise do tempo de processamento entre chegada dos eventos de packet_in no controlador e a geração do evento flow_mod pelo controlador, que configuraa uma nova entrada de fluxo no switch. Nesse teste, observa-se um desempenho inferior do ambiente FIBRE com o plano de controle. Este desempenho inferior fica em destaque no caso de uso de ambiente FIBRE com placas NetFPGA, em que o controlador ao receber grande quantidade de packet_in, apresentou elevados valores de CPU chegando ao fim da simulação com uma média de 97% de uso, porém como no ambiente FIBRE é utilizado os mesmos servidores de virtualização para gerar máquinas Pronto e máquinas NetFPGA, o resultado inferior na análise das máquinas NetFPGA é um indicativo da alta variabilidade que esta plataforma de teste pública apresenta pelo fato de ser compartilhada.

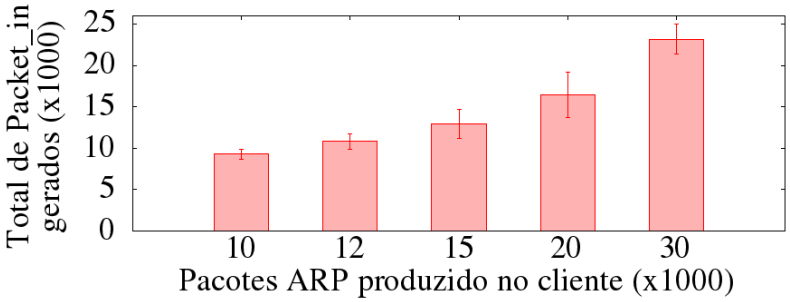
Embora o Mininet possua todos os elementos da topologia em um único componente e tenha processado um número de packet_in superior aos ambientes FIBRE simulados, ele apresenta, na média, um tempo de processamento no plano de controle melhor que o FIBRE. Os ambientes de testes privados apresentaram um tempo menor que o Mininet e o FIBRE, mesmo nos casos dos OvS em desktop (OVS_4GB e OVS_8GB), nos quais transitou um número elevado de pacotes no plano de controle. Devido a perda de pacotes ARP, no caso do teste com switch Pronto, a quantidade de pacotes no plano de controle foi reduzida, o que proporcionou um processamento rápido.

Mediu-se, como mostrado na Figura 5.4c, o tempo entre a requisição ARP feita por h1, com MACs distintos, até a mensagem chegar em h2 e seu retorno até h1. Nesse caso, o ambiente FIBRE foi o que apresentou maiores tempos. Uma vez que o ambiente FIBRE pode estar sendo compartilhado por outros pesquisadores, tal situação pode ter impactado o tempo de processamento do elemento OpenFlow para criar um número de entradas de fluxo elevada.

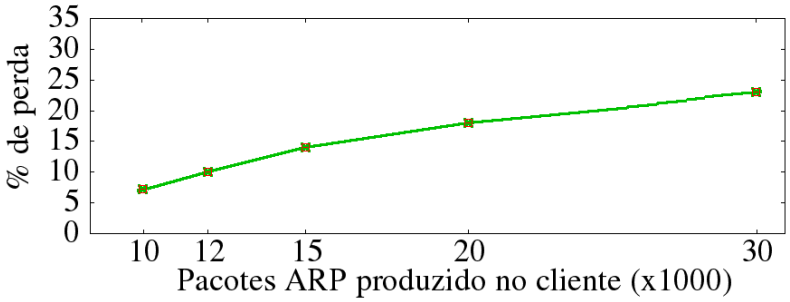
Devido o comportamento instável detectado durante os testes na plataforma pública FIBRE, foi realizado testes extras com foco no plano de controle para verificar seu comportamento.

Conforme Figura 5.6, foi concebido cinco cenários específicos de geração de pacotes

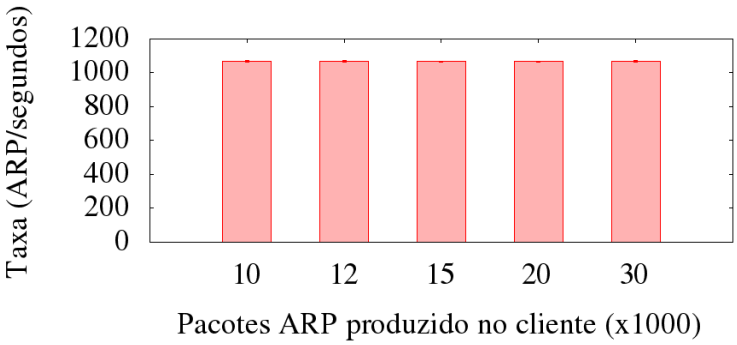
ARPs a partir do cliente h1 com endereçamento físico de origem distintos. No primeiro cenário eram gerados 10 mil pacotes, em seguida, 12, 15, 20 e 30 mil pacotes.



(a) Eventos de packet_in processados pelo switch.



(b) Porcentagem de perda dos pacotes ARPs gerados pelo cliente.



(c) Taxa de geração de pacotes ARPs por segundo nos cenários apreciados.

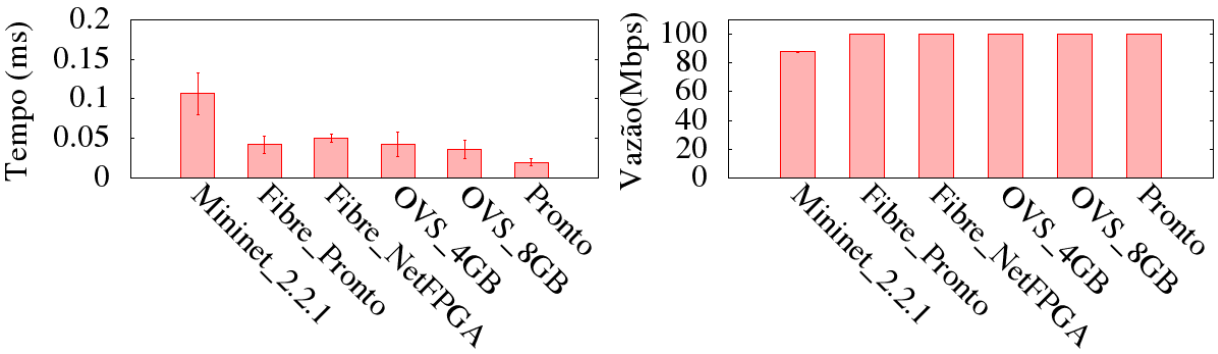
Figura 5.6: Análise do comportamento dos comutadores OpenFlow na plataforma FIBRE.

Na Figura 5.6a é possível detectar que a quantidade de pacotes criados por h1 não gera a mesma quantidade de packet_in e este valor se torna mais discrepante para cenários de maior geração de pacotes ARP, o que indica que os dispositivos OpenFlow da plataforma FIBRE não conseguem responder a contento para demandas elevadas de eventos de packet_in.

Na Figura 5.6b é apresentada a porcentagem desta perda de pacotes, onde detecta-se com clareza o aumento na perda quanto maior foi a geração de pacotes pelo cliente h1, e

por fim na Figura 5.6c mostra que a taxa de pacotes por segundos gerados pelo cliente h1 para todos os cenários foram equivalentes, concluindo-se que o acumulo de pacotes que chegaram ao elemento OpenFlow foi o motivador no impacto de perda de pacotes pois o mesmo não conseguiu tratar a contento posto que a taxa se manteve por longo tempo quanto maior era o cenário de geração de pacotes ARP.

Por fim, é verificado as comunicações do tipo UDP, com análise sobre a variação do atraso entre os pacotes sucessivos de dados (*jitter*), conforme a Figura 5.7a e a vazão conforme a Figura 5.7b. Os testes utilizaram tamanho máximo de segmento de 1460 bytes, enlaces de 1 Gbps e taxa de envio de 100 Mbps. Nesta análise, fica claro como a plataforma Mininet possui desempenho inferior comparado com os outros cenários quando se trata de comunicações UDP, tanto no que se refere ao *jitter*, quanto na vazão obtida.



(a) Observação de JITTER

(b) Vazão

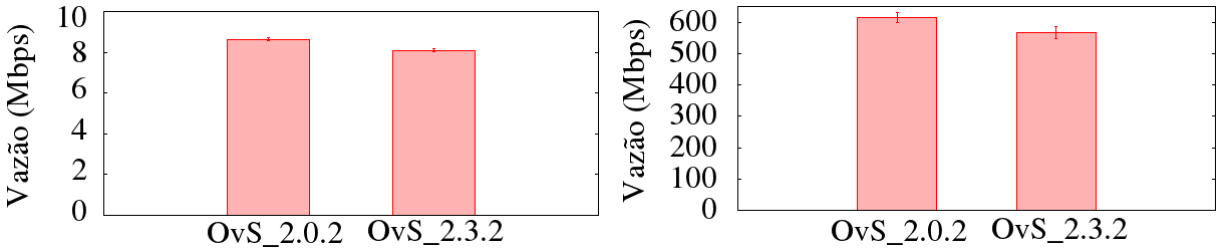
Figura 5.7: Observação da influência dos resultados em conexões UDP.

5.3 Comparação entre ambientes utilizando o Mininet

A próxima sequência de resultados, que tem como destaque o ambiente Mininet, compara as questões de utilização de switches OvS de diferentes versões, no qual o foco está na alternância do protocolo OpenFlow entre 1.0 e 1.3, analisando o desempenho do plano de dados e de controle. Além disso, foi comparado o uso do controlador instalado na máquina virtual do Mininet ou seu uso de forma externa, numa máquina conforme citado em 4.3.1.

As Figuras 5.8a e 5.8b mostram o comportamento observado de versões distintas de switch OvS (2.0.2 "OpenFlow 1.0"e 2.3.2 "OpenFlow 1.3"). Para isto, é apresentada tal análise em ambiente Mininet 2.2.1, com uso de controlador interno. Foi utilizado o gerador

de tráfego TCP Iperf, tamanho de máximo segmento de 536 bytes, banda de enlace de 10 Mbps e 1 Gbps e ferramenta de monitoração tcpdump.

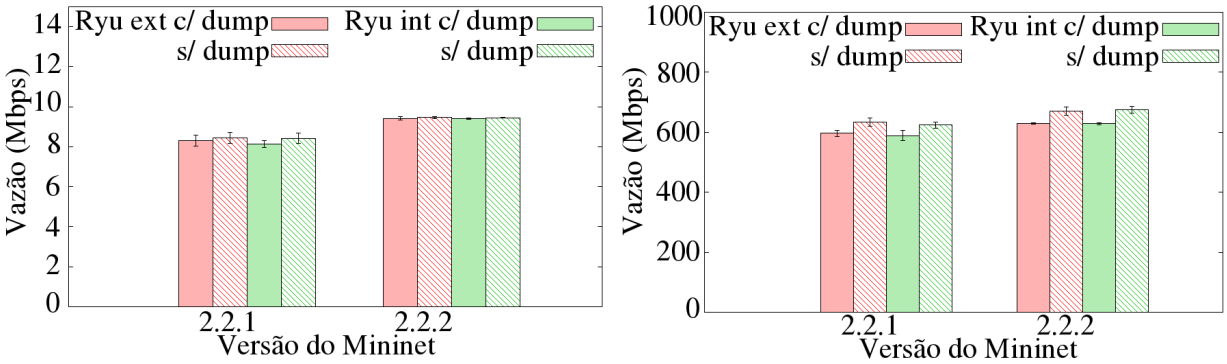


(a) 10 Mbps (b) 1 Gbps

Figura 5.8: Observação do comportamento do uso de versões de switches OvS distintos.

Detecta-se que o plano de dados apresenta melhor desempenho em ambientes com o protocolo OpenFlow 1.0, embora o resultado de vazão não tenha sido satisfatório. Este comportamento se repetiu para casos com máximo segmento de 1460 bytes, sem uso de ferramenta de monitoração e com uso do controlador em máquina externa.

A próxima análise mantém o foco no plano de dados, porém concentra-se em duas variáveis, a versão de Mininet e o possível impacto no plano de dados do uso do controlador de modo interno, isto é, instalado no próprio emulador Mininet, e configurado numa máquina externa ligada ao Mininet. As Figuras 5.9a e 5.9b apresentam os resultados desta análise, a qual foi realizada com máximo segmento de 1460 bytes e OpenFlow 1.3.



(a) 10 Mbps (b) 1 Gbps

Figura 5.9: Observação do comportamento das versões 2.2.1 e 2.2.2 do emulador Mininet.

O resultado deixa claro que o Mininet 2.2.2 apresenta melhor desempenho no plano de dados do que a versão 2.2.1. Resultados similares foram obtidos ao longo do desenvolvimento deste trabalho em experimentos realizados com OpenFlow 1.0 e tamanho de

segmento de 536. Outra característica a se destacar é sobre o método de uso do controlador, pois, independentemente da escolha, o mesmo não influencia nos resultados obtidos.

O melhor desempenho do uso do Mininet 2.2.2 também ficou nítido na capacidade do host h1 em gerar pacotes através da biblioteca Scapy como visto na Figura 5.10.

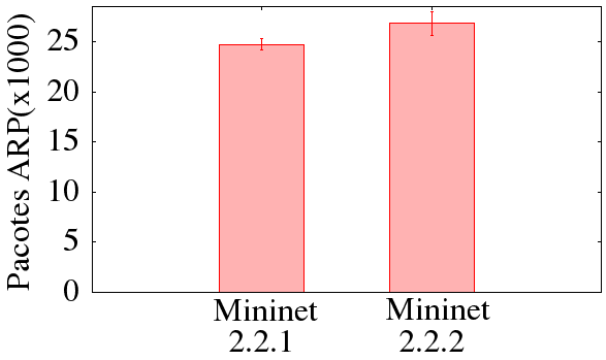


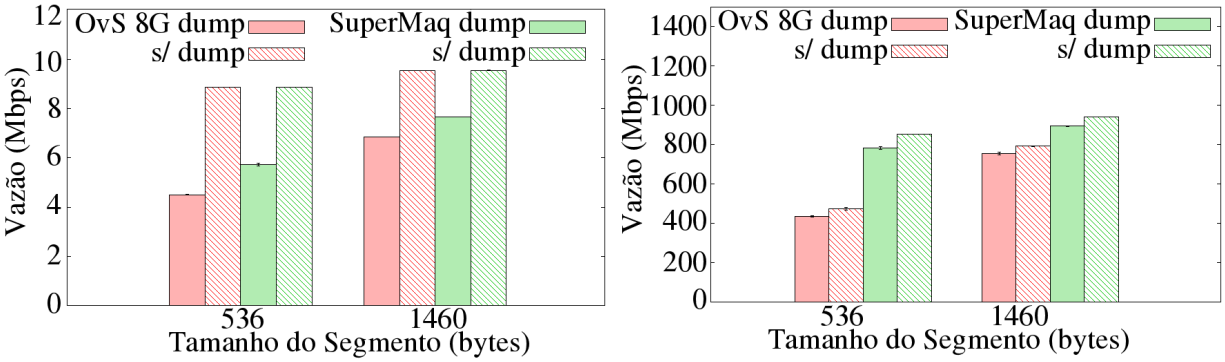
Figura 5.10: Criação de pacotes pelo host h1 em ambientes Mininet de diferentes versões.

Esta característica melhor no ambiente Mininet 2.2.2 pode estar estreitamente relacionada ao fato do seu módulo Kernel ser de versão superior, o que leva a melhor eficiência do sistema e um aproveitamento superior do hardware disponível para a máquina virtual do Mininet.

5.4 Comparação entre ambientes de testes privados

O foco principal nesta parte da análise é explorar as plataformas de testes privadas em tipos diferentes de hardware, com comparações de desempenho, versão de OvS e de protocolo OpenFlow, e comportamento do plano de controle, de forma a mostrar como o hardware pode impactar os resultados do trabalho de um pesquisador.

As Figuras 5.11a e 5.11b apresentam os resultados de testes realizados em OvS 2.3.2, com tamanho do segmento entre 536 e 1460 B, banda do enlace de 10 Mbps e 1 Gbps e uso de ferramenta de monitoração. No quesito hardware, são utilizados o cenário 2 (OvS em máquina de 8 GB de RAM) e cenário 3 (OVS em máquina com clock e processador superiores aos utilizados no cenário 2, com 16 GB de RAM) descritos em 4.3.2, no qual pode perceber que tratam-se de cenários com processadores de valores distintos, com o cenário 3 sendo superior, conforme detectado em sites de benchmark como o “http://cpu.userbenchmark.com/”. Como no cenário 3 as máquinas utilizadas são mais robustas, as figuras expostas neste tópico referente ao cenário 3, aparecem com a legenda de "SuperMaq".



(a) 10 Mbps

(b) 1 Gbps

Figura 5.11: Observação da influência do hardware em plataforma de teste privado.

O resultado deste teste deixa claro que, com máquinas que possuem melhores configurações de CPU, o plano de dados apresenta melhor desempenho, em especial em situações de grande carga no switch, conforme esperado.

Quanto melhor o processador, menor é o impacto do uso do software de captura de pacotes. Quando o mesmo não é utilizado, em enlaces de baixa capacidade, os resultados se equivalem, não valendo o investimento em máquinas mais caras. Em caso de uso de enlaces com elevada capacidade (1 Gbps), máquinas mais robustas se destacam positivamente. Estes mesmos testes realizados em switch OvS 2.0.2 mostraram resultados similares.

A Figura 5.12 refere-se a testes com o protocolo UDP, nos quais o destaque é a análise do *jitter*. São realizados testes em switch OvS 2.0.2 e 2.3.2, com cenários distintos de hardware, de maneira similar à análise anterior. Conforme observado, o uso de máquinas melhores ocasionou valores de jitter menores, demonstrando que é necessário o investimento em hardware para testes nos quais se avalie aplicações de tempo real, como jogos on line, video ao vivo, entre outros. Outra conclusão demonstrada no teste é a não interferência da versão de protocolo OpenFlow utilizado no resultado do *jitter*.

Outra importante análise realizada é com relação ao funcionamento do plano de controle em plataformas privadas com hardwares distintos. Foram realizados testes no cenário 2 e 3, nas duas versões de switch OvS, para evidenciar o comportamento do protocolo OpenFlow 1.0 e 1.3. A Figura 5.13 evidencia o resultado do processamento entre a chegada de inúmeros eventos de packet_in e a geração dos fluxos correspondentes a cada um deles em direção ao switch.

Detecta-se dois aspectos distintos neste teste; primeiramente, o controlador processa mais rápido quando é utilizado a versão 1.0 do protocolo OpenFlow, independente do tipo

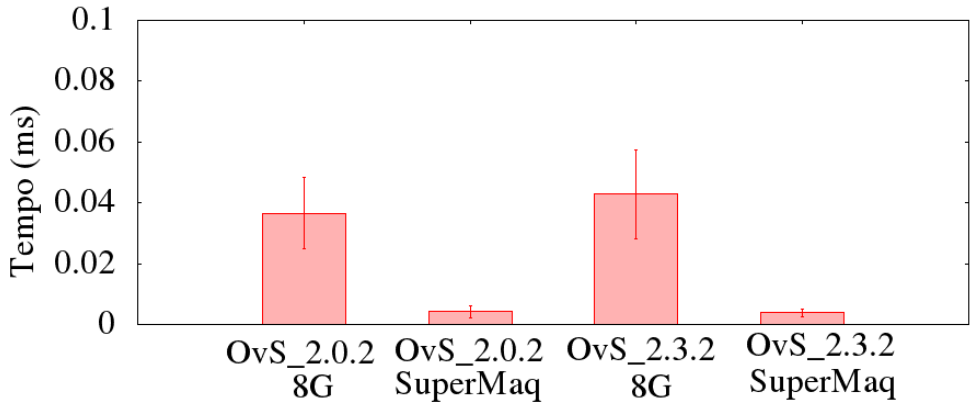


Figura 5.12: Observação do *Jitter* em plataformas privadas distintas.

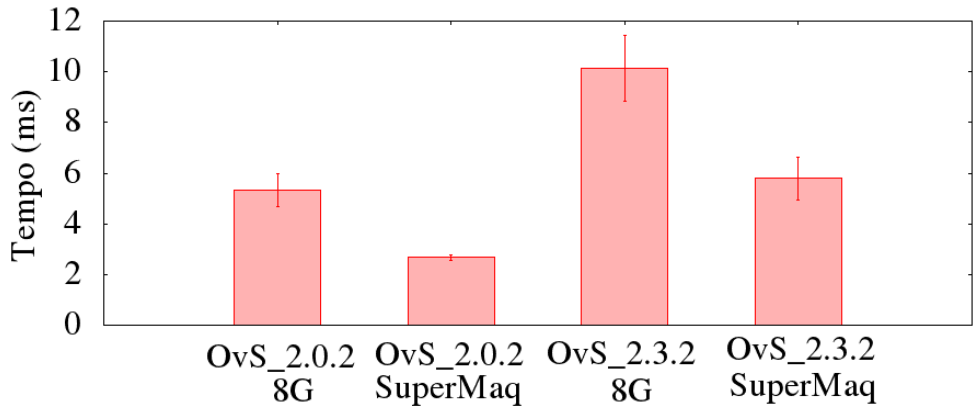


Figura 5.13: Observação do tempo de processamento no plano de controle para máquinas de configurações distintas.

de hardware utilizado, esta constatação fica mais nítida na Figura 5.14 no qual foi utilizado a mesma versão de OvS, no caso a versão 2.3.2, para tratar de requisições relacionadas ao OpenFlow 1.0 e ao OpenFlow 1.3, indicando que a diferença no tempo de processamento de packet_in pelo controlador é devido à complexidade do protocolo OpenFlow 1.3 que possui mais campos em seu cabeçalho para análise, perante o 1.0 e não está relacionada com a influência do OvS utilizado.

A segunda observação com relação a análise dos testes com foco no plano de controle, é a a constatação que ao utilizar o controlador em máquina com melhor configuração de memória e CPU, o tempo de processamento é praticamente metade do valor apresentado do cenário comparado. Assim, esse se mostrou um resultado significativo. Em testes de medição de tempo, no qual o tempo de resposta do controlador tem forte influência (muitas operações do controlador), o experimentador deve investir em uma máquina mais potente para o controlador.

Este resultado se torna mais expressivo ao analisar o funcionamento dos hosts nesses

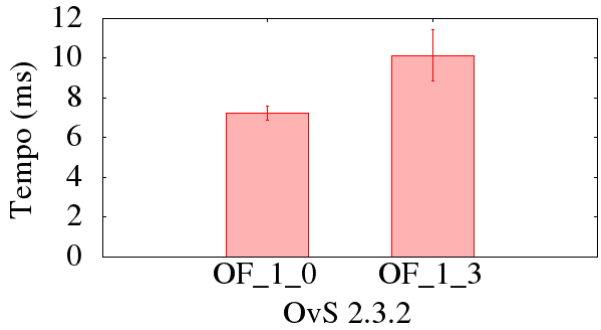


Figura 5.14: Comparação entre versões distintas do protocolo OpenFlow.

dois cenários. A Figura 5.15 mostra a quantidade de pacotes gerados pelos hosts nos dois cenários. A quantidade de pacotes ARP gerados através da biblioteca Scapy com as máquinas melhores é significativamente maior do que a gerada com o host do cenário 2. Isso significa que, no teste com o cenário 3, obteve-se um número de eventos de packet_in maior e, ainda assim, a capacidade de processamento do switch e do controlador do cenário 3 foi muito superior. De fato, ao longo do trabalho, verificou-se que o uso de máquinas de configurações mais modestas em h1 e h2 impactou severamente na geração de tráfego para análise do desempenho do plano de dados.

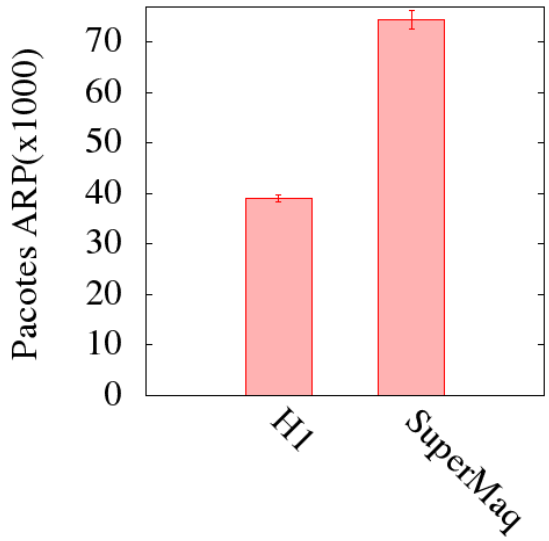
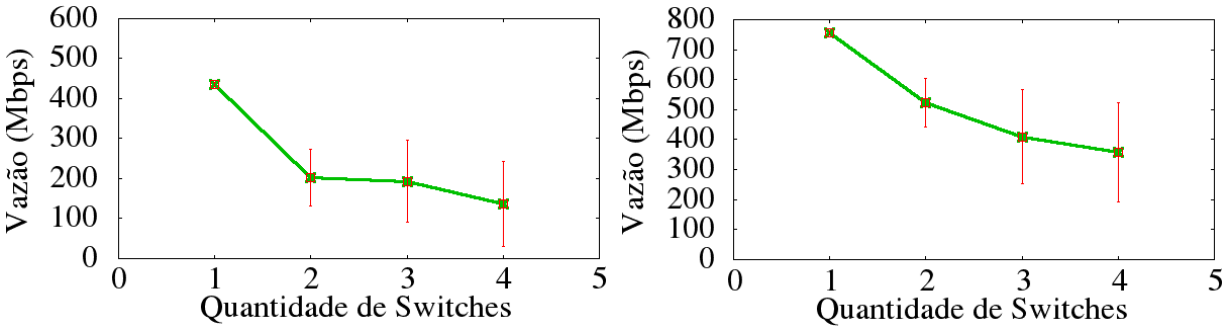


Figura 5.15: Total de pacotes ARP gerados por hardwares distintos.

5.5 Comparação no uso de múltiplos switches em uma única máquina

O prosseguimento deste trabalho busca verificar o comportamento de múltiplas instâncias de switch OvS em uma única máquina física, pois busca-se analisar a possibilidade de aumentar o tamanho da rede, mantendo-se resultados satisfatórios, sem aumentar os custos de montagem do ambiente. Os testes são realizados em versão 2.3.2 do switch OvS, em hardware definido no cenário 4 da subseção 4.3.2. As Figuras 5.16a e 5.16b avaliam o impacto de processamento no plano de dados ao conduzir os experimentos até a utilização de 4 instâncias de OvS; foi utilizado o Iperf para a geração de tráfego em h1, variação de máximo segmento em 536 e 1460 bytes, uso de links de 1 Gbps e ferramenta de monitoração tcpdump em h1 e h2.



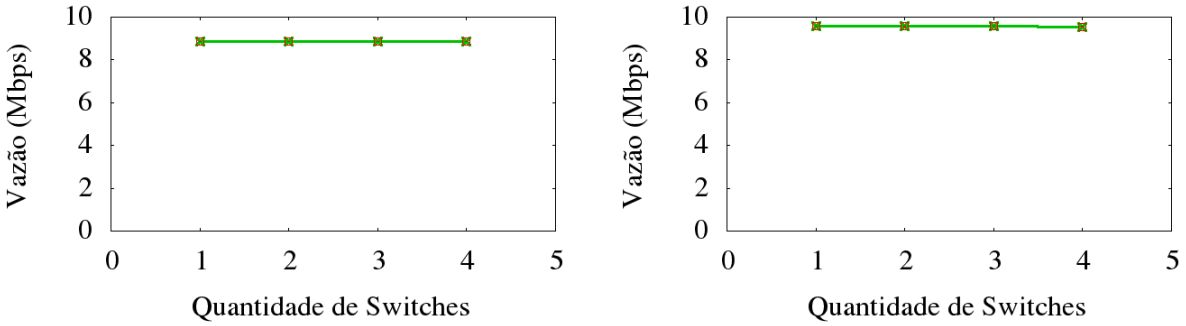
(a) Segmento de 536 bytes

(b) Segmento de 1460 bytes

Figura 5.16: Observação do plano de dados em links de 1 Gbps devido ao uso de múltiplos switches em mesma máquina.

Visivelmente, observa-se que, ao aumentar a quantidade de instâncias OvS, o tráfego final detectado diminui e o intervalo de confiança aumenta, sendo um indicativo da variação dos resultados a cada simulação cada vez que insere um novo switch na máquina. Este resultado direciona a um pesquisador utilizar múltiplas instâncias apenas em estudos no qual a largura de banda não seja o item prioritário, já que não é interessante usar múltiplas instâncias por máquina, e sum uma instância por máquina, distribuindo a plataforma de teste por várias máquinas. Isto foi corroborado nas simulações com largura de banda de 10 Mbps, no qual o uso de múltiplas instâncias de OvS não impactou na vazão. Conforme representado nas Figuras 5.17a e 5.17b que avaliam o impacto de processamento no plano de dados ao conduzir os experimentos até a utilização de 4 instâncias de OvS; com variação de máximo segmento em 536 e 1460 bytes e uso de links de 10 Mbps.

O teste seguinte, cujos resultados estão na Figura 5.18, foram feitos baseados na



(a) Segmento de 536 bytes (b) Segmento de 1460 bytes

Figura 5.17: Observação do plano de dados em links de 10 Mbps devido ao uso de múltiplos switches em mesma máquina.

geração de 100 Mbps de tráfego sobre protocolo UDP, utilizando Iperf. Nesse caso, mediu-se a variação do atraso entre os pacotes sucessivos de dados e observou-se que o impacto se torna maior somente após a inserção do terceiro switch. Nesse teste, não foi observado nenhum impacto na vazão.

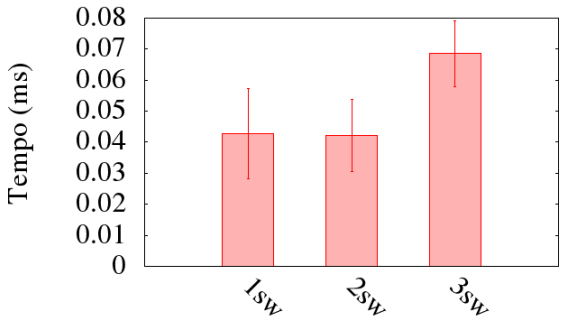
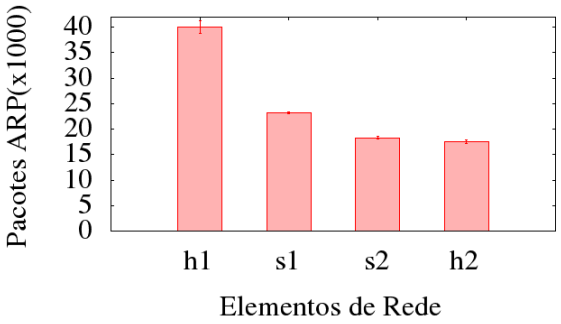


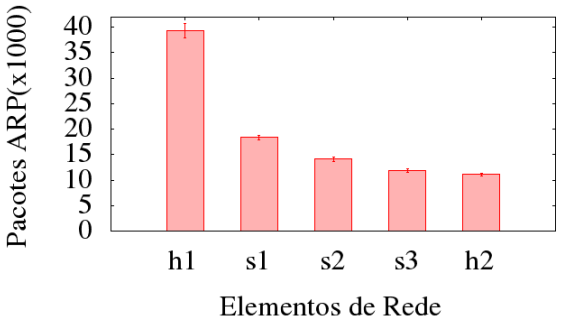
Figura 5.18: Observação do *Jitter* ao aumentar as instâncias de OvS.

O próximo teste visa a avaliação da capacidade de processamento de cada switch ao se utilizar múltiplos switches em uma mesma máquina. Aqui, observou-se um resultado divergente dos cenários nos quais apenas uma instância do OvS estava configurado em uma máquina física. Nesses cenários, os pacotes ARPs criados através da biblioteca Scapy pelo cliente h1 foram convertidos em eventos de packet_in em mesma quantidade. Ao se utilizar mais de switch por máquina, obteve-se o resultado apresentado nas Figuras 5.19a, 5.19b e 5.19c. A quantidade de pacotes ARP oriundas de h1 e detectadas em h2 é menor, e tal impacto é mais explícito cada vez que aumenta a quantidade de instâncias de OvS instalada. Assim, testes que utilizem maior quantidade de pacotes e que sejam influenciados por perdas não podem fazer uso de mais de switch por máquina.

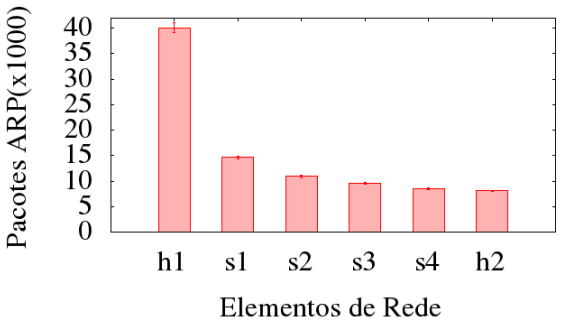
Simultaneamente a esta análise, foi verificado o impacto sobre o plano de controle



(a) Pacotes ARP com 2 instâncias de OvS.



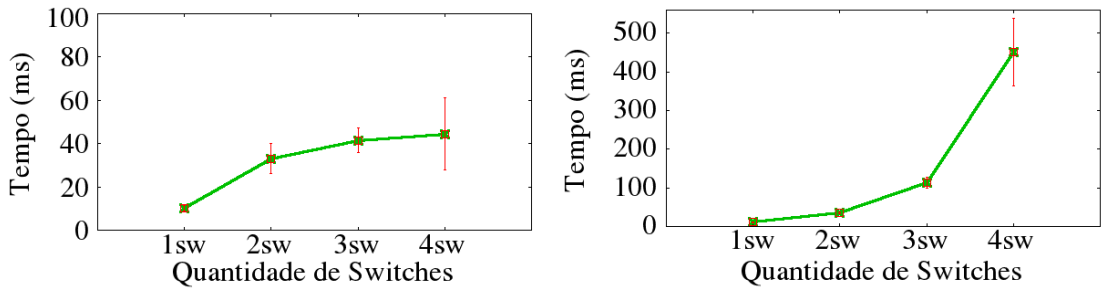
(b) Pacotes ARP com 3 instâncias de OvS.



(c) Pacotes ARP com 4 instâncias de OvS.

Figura 5.19: Observação do impacto de múltiplas instâncias na quantidade de pacotes ARP em h2.

referente aos eventos de *packet_in* que foram reportados ao controlador, conforme a Figura 5.20a, e o tempo total de requisição ARP entre h1 e h2, conforme a Figura 5.20b. Claramente, é observado que o aumento de instâncias OvS numa mesma máquina degrada o desempenho do plano de controle, como esperado, já que o controlador passa a atender mais switches. É importante entender que a tabela de fluxo em switches OpenFlow serve como a memória que contém todos os fluxos ativos e é semelhante à Forwarding Information Base (FIB) em roteadores convencionais. Após a recepção de um *flow_mod* enviado pelo controlador, que modifica o conjunto de entradas de fluxo de um switch, o switch atualiza a tabela e (se necessário) reordena as entradas de fluxo. Este processo, de inserção de fluxo na tabela de fluxo, incorre em atrasos no sistema, e cresce à medida que o número de entradas de fluxo aumenta. Observa-se que o atraso fim a fim aumentou muito mais do que o atraso gerado pelo plano de controle, o que demonstra que o hardware não é capaz de responder rapidamente quando vários switches estão na mesma máquina.



(a) Tempo de processamento do plano de controle em caso de múltiplas instâncias.

(b) Tempo de requisição ARP entre h1 e h2 em ambientes com múltiplas instâncias.

Figura 5.20: Análise do impacto no processamento do plano de controle para máquinas com múltiplas instâncias de OvS.

5.6 Comparação no uso de plataforma de testes públicas

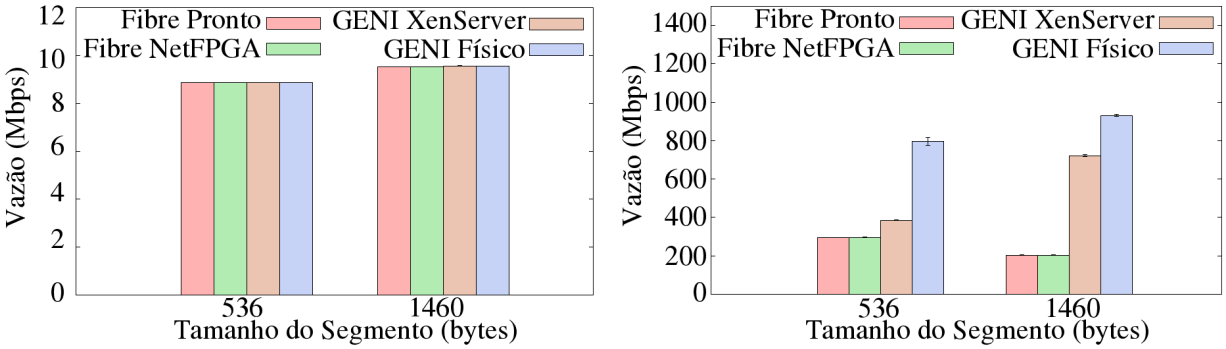
Os testes foram realizados nas plataformas públicas FIBRE e GENI; como o FIBRE só atende ao protocolo OpenFlow versão 1.0, os testes na plataforma GENI foram realizados com o OvS 2.0.2, que utiliza o protocolo OpenFlow 1.0.

Os testes com FIBRE são divididos na análise do uso do Pronto e do uso das placas NetFPGA[2]; na plataforma GENI os testes são divididos em uso de laboratório com máquinas físicas e no ambiente com uso de máquinas virtuais criadas sob um hypervisor XEN. Nas Figuras 5.21a e 5.21b, são apresentados os resultados dos testes feitos a partir da geração de um fluxo TCP com o Iperf entre as máquinas h1 e h2, variação de segmento entre 536 e 1460 B, banda do enlace de 10 Mbps e 1 Gbps e ferramenta de monitoração tcpdump.

O ambiente GENI possui desempenho no plano de dados, assumindo tráfego TCP, superior ao FIBRE em enlaces de 1 Gbps, com maior destaque quando é utilizado o ambiente de servidores físicos do GENI. De fato, são observados resultados bem distintos nos testes, conforme mostra a Figura 5.21b. Os ambientes GENI utilizados eram providos pela mesma universidade(UtahDDC), porém por agregados diferentes, sem similiariedade de processadores, que devem possuir demandas diversas e assim apresentaram nos experimentos desta simulação comportamentos heterogêneos.

Foi feita também uma análise com tráfego UDP à 100 Mbps, taxa que deveria estar

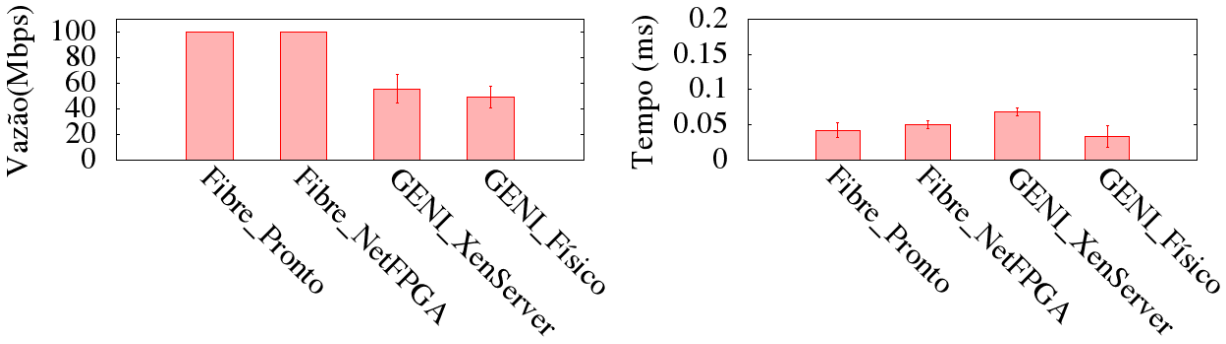
[2]Em testes padrão no FIBRE, é comum utilizar as duas plataformas de hardware, Pronto e NetFPGA, indistintamente e em conjunto. Contudo, foi observada uma diferença de comportamento nos switches que pode influenciar os resultados, motivando essa validação.



(a) 10 Mbps (b) 1 Gbps

Figura 5.21: Observação do plano de dados de cenários equivalentes em plataforma de testes públicas distintas.

disponível em todas as plataformas de hardware dos testbeds públicos testados, de acordo com a Figura 5.21b. Contudo, diferentemente do esperado de acordo com o teste com tráfego TCP, a vazão do tráfego UDP foi boa no ambiente FIBRE, mas insatisfatória no ambiente GENI, conforme Fígura 5.22a. Porém, na Figura 5.22b, observa-se que o *jitter* no ambiente GENI com servidores físicos se destaca dos demais pelo menor valor médio detectado.



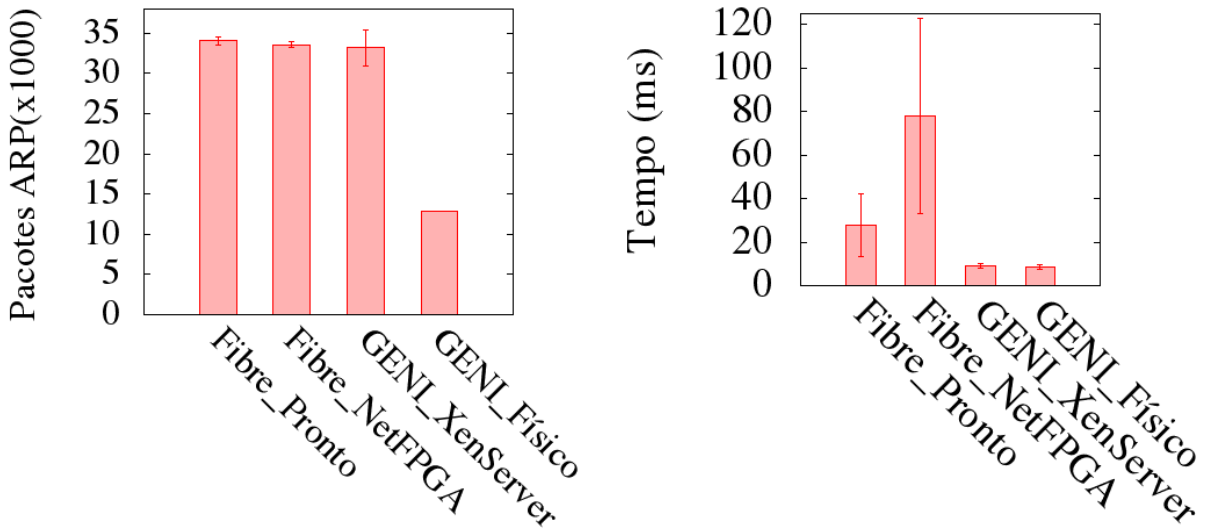
(a) Vazão (b) Jitter

Figura 5.22: Observação do plano de dados em tráfego UDP de cenários equivalentes em plataforma de testes públicas distintas.

Para finalizar os resultados deste trabalho, foi analisado o comportamento do plano de controle de plataformas públicas. Nesse caso, foi repetido o procedimento de gerar pacotes ARP em um host, direcionados ao segundo host, utilizando a ferramenta SCAPY. Cada pacote ARP, ao ser processado pelo switch, geraria um evento de *packet_in* para o controlador. Assim, primeiramente, é apresentada, na Figura 5.23a, a capacidade de geração de pacotes ARP pelos hosts em ambas as plataformas. Foi verificado que não

há perda de pacotes no ambiente GENI, conforme ocorreu no ambiente FIBRE, como já foi explanado na Seção 5.2. Porém, a capacidade de gerar pacotes em h1, quando esse está implementado nas máquinas físicas do GENI, foi insatisfatória, quando comparada às demais. Mesmo existindo dois processadores nas máquinas físicas do GENI, o script elaborado não faz uso de multithreads e então utiliza somente um processador. Essa foi uma escolha feita para gerar justiça na comparação entre os ambientes, pois como as máquinas no ambiente FIBRE possuem somente um processador. Pelo resultado apresentado nas máquinas físicas do ambiente GENI, conclui-se que o seu processador seja de capacidade inferior aos dos demais ambientes, sendo necessário o uso de multithreads nesse contexto para evitar uma grande perda de desempenho.

No que tange ao tempo de processamento do plano de controle, fica nítido pela Figura 5.23b que o ambiente GENI com máquinas virtuais apresenta melhor resultado. Esta constatação está relacionada ao fato de existirem dois processadores dedicados por máquina, conseguindo tratar as múltiplas requisições que chegam ao controlador, não sobrecarregando nenhum deles. A comparação com o GENI_Físico não é justa, já que esse plano de controle teve uma carga de pacotes inferior.



(a) Criação de pacotes ARP.

(b) Tempo de processamento no plano de controle.

Figura 5.23: Comportamento do plano de controle das plataformas públicas analisadas.

Capítulo 6

Análise de Resultados

A proposta deste capítulo é fazer uma análise geral dos resultados apresentados no Capítulo 5. Para isto, são apresentados três seções com abordagens distintas. A primeira realiza uma comparação das capacidades, apresentando o que pode ser afirmado a partir dos resultados obtidos. A segunda seção visa mostrar os custos financeiros associados na implantação dos ambientes de experimentação utilizados, de forma a conscientizar o leitor das expectativas de gastos caso o mesmo decida usar algum dos ambientes apresentados. Por fim, a terceira seção faz uma análise qualitativa das soluções usadas como ambientes de experimentação, verificando os itens que compõem os ambientes OpenFlow.

6.1 Comparação de Capacidades

O cenário utilizado como ambiente de experimentação neste trabalho foi de uma topologia básica atendendo ao conceito de experimento, no qual o pesquisador manipula e controla uma ou mais variáveis independentes e observa a variação nas variáveis dependentes concomitantemente à manipulação das variáveis independentes. O propósito é manipular e medir as variáveis no experimento e captar causalidade (relação entre causa e efeito).

O pesquisador precisa ter controle total sobre as variáveis, pois o ambiente de experimentação é criado e conduzido pelo pesquisador. Já nos experimentos de campo, o pesquisador não consegue ter controle absoluto sobre as variáveis, pois são projetos conduzidos em uma situação real, no qual há a presença de muitos fatores que interferem na pesquisa e que podem fugir do controle do pesquisador. Por essa razão, o pesquisador precisa se adequar ao ambiente utilizando técnicas para atenuar os efeitos de fatores que atrapalham as observações.

Como parte dos experimentos deste trabalho utilizam plataformas públicas manipuladas concomitantemente por terceiros, estas podem ter sua repetibilidade afetada.

Uma das premissas dos testes realizados era a de manter uma topologia padrão em todos os experimentos. Para isso, foi necessário a escolha de um cenário enxuto, implementável em todas as ferramentas, mas que ao mesmo tempo atendesse a critérios mínimos de análise para dar embasamento para escolhas de ferramentas para outros cenários.

Assim, o uso de dois clientes, um controlador e um elemento OpenFlow é suficiente para verificar as questões mais básicas relacionadas ao plano de controle e ao plano de dados. Dessa forma, a escolha por uma topologia menor não desmerece os resultados obtidos, nem o embasamento científico em considerar estes resultados para entender algumas limitações do funcionamento de cada ferramenta em topologias maiores.

Como exemplo, foi observada uma grande flutuação nos resultados na plataforma pública FIBRE, devido ao compartilhamento com outros usuários. Ao utilizar mais nós, espera-se que essa flutuação continue a ser observada e, até mesmo, aumentada. De fato, ao utilizar a federação de ilhas e ao usar mais enlaces de diferentes ilhas, cria-se a possibilidade de compartilhamento com uma quantidade maior de pesquisadores, criando mais flutuações no resultado.

O pesquisador pode ter seu ambiente preparado para o teste em específico, em equipamento real, com garantia de fidelidade, assim o pesquisador tem acesso a plataforma porém não tem o detalhamento dos links e da transição de dados, isto pode afetar em caso de uso de topologias em estilo de árvore com muitos nós conectados típicas de um data center, pois a visibilidadde do pesquisador pode não ser suficiente para o mesmo utilizar, por exemplo, ferramentas de monitoração em elementos da plataforma, e afetar a interpretação dos dados detectados. Por isso, mesmo com a possibilidade de vários recursos físicos que uma plataforma de teste pública possa oferecer ao pesquisador, ela também pode ser um problema por causa de algumas limitações no que tange a repetibilidade.

A plataforma GENI tem similaridade com a plataforma FIBRE e seu intuito é montar um grande laboratório virtual em larga escala para experimentações em rede de computadores, cuja maior importância é prever e criar novas possibilidades para Internet do Futuro. Da mesma forma que no ambiente FIBRE, qualquer que seja a implementação em um nó GENI, será permitido que múltiplos pesquisadores simultaneamente compartilhem a mesma infraestrutura e isto possa causar as limitações anteriormente citadas e que foram constatadas nos resultados provenientes deste trabalho nas plataformas públicas, o que leva os pesquisadores a utilizarem estes ambientes cientes de imprevisibilidade dos

seus resultados.

Quando se trata de emuladores, os mesmos não necessitam de grandes mudança em questão de código ou configuração para um usuário, já que o ambiente emulado pode fazer uso de recursos reais sem maiores alterações, além de solucionar vários problemas de facilidade ao usuário e possuírem a flexibilidade como uma característica atraente para pesquisadores. O Mininet entra neste grupo de sistemas de emulação e, como tal, opera em tempo real. Porém o mesmo pode não operar de maneira tão fidedigna ao que se espera.

No Mininet, o encaminhamento de pacotes na rede virtual compartilha recursos de CPU e memória e a capacidade agregada oferecida pelo Mininet é geralmente em torno de alguns gigabits dependendo da máquina física. Para experiências com CPU limitada, também é necessário alocar cuidadosamente os recursos da CPU para os hosts virtuais. Quando o tamanho da rede de destino aumenta ou a utilização de mais nós é necessária, os recursos de CPU atribuídos a cada aplicativo executando um host virtual podem ficar severamente sub-provisionados e, portanto, afetar a precisão de tempo dos experimentos e o resultado do experimento.

O uso do Mininet, neste trabalho, apesar da plataforma extremamente simples, apresentou um resultado mediano, com uma percepção clara de como o compartilhamento dos recursos da máquina física afetou os testes. A grande questão relativa ao Mininet é como saber se os recursos estão comprometidos a ponto de comprometer o teste ou não. Assim, ele torna-se uma plataforma imprecisa, embora muito interessante em termos de simplicidade de uso. Além disso, também se observou que o Mininet não foi fidedigno na criação de enlaces com o tamanho especificado, em curto espaço de tempo, devido o comportamento do módulo TCLink, o que pode gerar distorções no resultado. Nesse sentido, testes com Mininet exigem um estudo detalhado das expectativas de uso dos enlaces e de quais máquinas demandarão grande capacidade de processamento, de forma a dimensionar quantas máquinas físicas executando o Mininet de forma integrada serão necessárias, porém este emulador continua sendo a melhor opção para aprendizado do funcionamento de ambientes OpenFlow com foco nas Redes Definidas por Software.

No caso das plataformas privadas, existe a segurança dos recursos estarem completamente dedicados ao pesquisador. O uso deste hardware dedicado para a função traz ganhos de desempenho ao sistema, porém acrescenta algumas limitações, por exemplo, o custo de aquisição do hardware e os custos (financeiros ou não) com a manutenção do hardware, caso haja falha. Além disto, modificações de topologia e de configurações po-

dem ser custosas em termos de homem-hora, já que não são feitas de forma automatizada pela plataforma. Por exemplo, para mudar uma topologia no Mininet durante um teste, basta adicionar algumas linhas no script, enquanto em um testbed privado, isso demanda uma ação física. Para mudar o sistema operacional de um host em um testbed público, basta trocar de máquina virtual, enquanto em um testbed privado seria necessário reinstalar a máquina. Assim, existem custos associados ao uso dos testbed privados, além dos problemas de escalabilidade e custo.

Nos resultados deste trabalho, observou-se que os testes em plataforma privadas com uso de computadores pessoais apresentaram resultados consistentes. Além disso, ficou claro que, quanto melhor o hardware utilizado, mais preciso são os resultados. Essa aquisição de melhores máquinas, contudo, encarece os custos da pesquisa. Porém, resta claro que, se financeiramente possível, trata-se do melhor caminho a seguir para o desenvolvimento de novos sistemas em ambientes OpenFlow.

6.2 Custo Financeiro das Soluções

Além de observar fatores relacionados ao desempenho, é importante também relacionar os custos financeiros envolvidos para implementação dos ambientes de teste, uma vez que sempre existem limitações financeiras nos projetos. Foi realizado um levantamento, aproximado, nos preços dos elementos utilizados para montagem dos ambientes de experimentação analisados, considerando o switch Pronto Pica8 P-3295, os computadores baseados em placas Mini-ITX DN2800MT, as placas HP Ethernet 1Gb 4-port 331T Adapter, que foram usadas na emulação de switches em computadores com processador Intel Core i7 2600 e memória de 8GB de RAM; e os computadores mais robustos, com 16GB de RAM e processadores Intel Core i7-7700 3.60GHZ.

Os custos apresentados são baseados na aquisição de equipamentos para implantar o ambiente de teste assumindo uma topologia com um host por switch e um único controlador, como mostrado na topologia da Figura 6.1 e seus respectivos custos na Figura 6.2. É importante notar que o caso de uso de computadores pessoais com melhores configurações é nomeado na figura como OvS_SuperMaq. Os ambientes FIBRE e GENI requerem um único computador para conectar-se na plataforma de teste. Portanto, representam a solução mais econômica, já que um computador ou celular com acesso à Internet está geralmente disponível. Mais do que isso, fica claro o potencial desse tipo de infraestrutura para aulas, além do seu uso comum para pesquisas de novas soluções, já que ambientes de

grande porte para diversos usuários simultâneos costumam ser muito custosos para universidades. Contudo, laboratórios de informática com um computador por usuário costumam ser comuns nesses ambientes, o que já viabilizaria o uso das plataformas públicas para o ensino de redes.

O custo do ambiente Mininet é determinado pela aquisição de um computador com configurações de memória entre 4 GB ou 8 GB de RAM e Intel Core i7 2600. Como no ambiente FIBRE, este computador geralmente está disponível e não requer orçamento extra. No entanto, este computador tem mais requisitos de processamento e memória do que o computador para acessar o FIBRE e GENI.

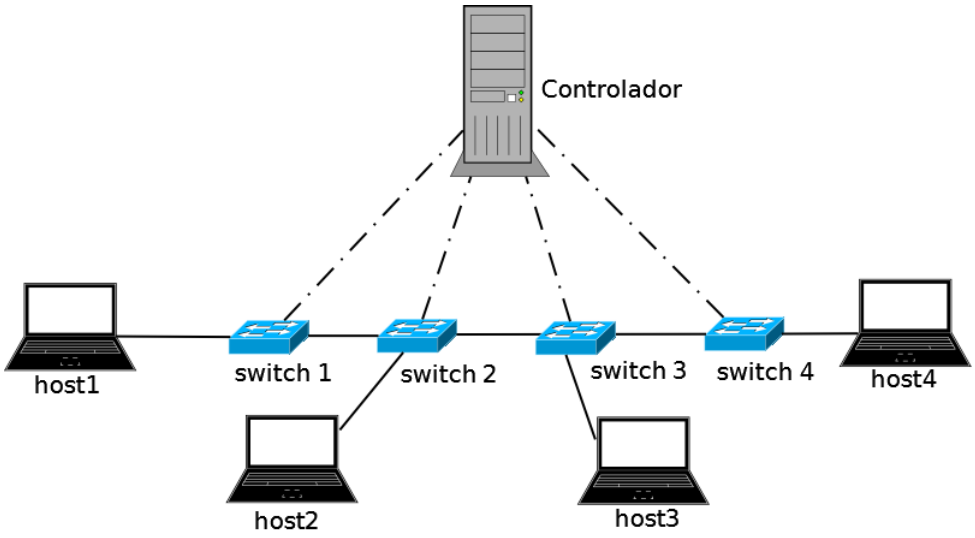


Figura 6.1: Topologia planejada para mensurar os custos envolvidos de aquisição dos equipamentos.

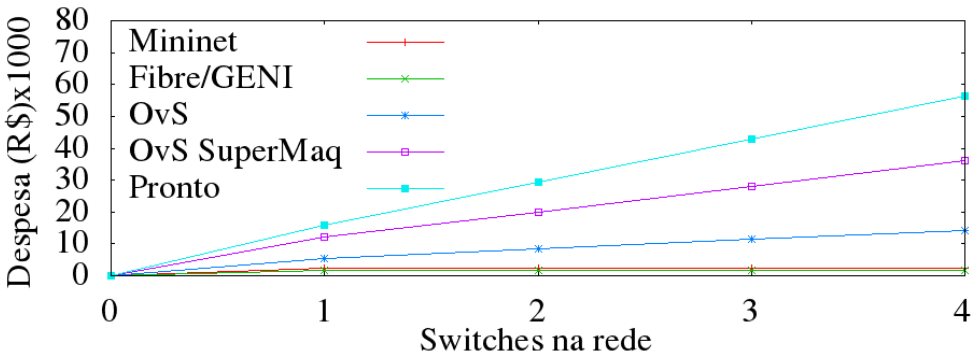


Figura 6.2: Demonstrativo de gastos por quantidade de switches com um host conectado por switch no ambiente de experimentação.

As plataformas de testes privadas que utilizam o switch emulado OvS tem seu custo de aquisição determinado pelo computador pessoal para o controlador e um por switch usando uma placa Ethernet com 4 portas. Além disso, requer a aquisição dos hosts, que

foram avaliados assumindo o uso de uma placa Mini-ITX por host implantado. O teste privado usando um switch comercial é semelhante, com exceção do custo do switch que neste caso não é o custo de um computador pessoal, mas de um hardware de switch de elevado valor.

No caso do teste privado com máquinas mais robustas, todos os elementos, sejam o switch, o controlador ou os hosts, foram elaborados utilizando o mesmo tipo de computador pessoal, considerando-se nessa análise computadores com processador Intel Core i7-7700 3.60GHZ, 16GB de RAM, 1T de HD e duas placas HP Ethernet 1Gb 4-port 331T Adapter, para totalizar até 8 entradas ethernet por computador.

A Figura 6.2 apresenta a evolução do custo de cada solução em função do número de switches a serem utilizados. Para os ambientes públicos e Mininet, o aumento no número de switches não afeta o custo. Evidentemente, o aumento do número de switches é limitado em ambos os casos, seja pelo número máximo de switches disponíveis na plataforma, como no caso do FIBRE, ou pelo processamento e capacidade de memória da máquina, no caso do Mininet.

Com base nessa análise, cabe observar que as plataformas privadas com uso de OvS em computadores pessoais são adequadas para testes realizados em topologias com poucos links ativos em cada switch e, consequentemente, menos tráfego. Como resultado, se obtém um custo financeiro reduzido em comparação com as plataformas privadas que utilizam switches reais, ao mesmo tempo que se obtém a repetibilidade dos testes, o que não é, de forma geral, possível nos testbeds públicos.

6.3 Análise Qualitativa

A Tabela 6.1 visa comparar os diversos parâmetros analisados nos ambientes de experimentação averiguados. Outro foco é apresentar o desafio tanto no que tange a implantação quanto a obtenção dos elementos que possibilitam a análise dos ambientes. Os parâmetros foram classificados entre alto, médio e baixo de forma a mensurar a aptidão de cada parâmetro da tabela ao seu respectivo ambiente de experimentação. Em conjunto, foram apresentadas algumas linhas informando a disponibilidade ou não de itens possíveis de existir em ambientes de experimentação. A seguir, são apresentados alguns comentários sobre cada item elencado na tabela:

1. Desempenho do plano de dados: avalia o comportamento do plano de dados dos

switches nos ambientes testados. O FIBRE e alguns cenários do ambiente Mininet 2.2.1 e GENI apresentam comportamento inferior em comparação com o teste privado.

2. Desempenho do plano de controle: Avalia o comportamento da interação dos switches com o controlador, levando em consideração a capacidade de receber novos pacotes de dados e criar packet_in para enviar aos controladores.

3. Desempenho do cliente (host): avalia o comportamento do host na geração de tráfego. Para as plataformas privadas, é considerado nesta tabela o Mini-itx com baixo custo, mas desempenho baixo e a máquina robusta com elevado custo, porém desempenho elevado.

4. Controle de vazão: a possibilidade de controlar o desempenho dos ambientes de teste é avaliada de forma a não afetar o resultado dos testes de vazão. No Mininet, no FIBRE e no GENI, esse controle é afetado por fatores externos, como compartilhar o sistema operacional com outra máquina, no caso do Mininet, ou o uso por outros pesquisadores, caso das plataformas públicas.

5. Celeridade para implantação: avalia a agilidade para configurar ambientes

6. Complexidade para medições: Avalia a dificuldade que os ambientes apresentam para obter os resultados dos testes.

7. Custos de implantação: avalia as despesas financeiras para a implementação dos ambientes.

8. Usa outras versões do OpenFlow: avalia a possibilidade de testar diferentes versões do protocolo OpenFlow.

9. Permite controle in band ou out band: Avalia se o plano de controle usa o mesmo enlace que o plano de dados.

10. Repetibilidade: Avalia se há repetibilidade nos resultados dos testes.

11. Flexibilidade: Avalia o grau de facilidade para mudanças de topologia nos ambientes implantados. Considera-se o Mininet como ambiente com maior possibilidade de flexibilidade.

Cabe observar que, em termos de desempenho, é sabido que o Mininet apresenta várias falhas ao se aumentar a escala da rede, de forma que alguns desses parâmetros podem mudar de forma negativa nessas situações.

Tabela 6.1: Comparação qualitativa dos ambientes de experimentação, considerando a topologia com um switch, dois hosts e um controlador.

Parâmetros	**Mini-net2.2.1**	**Mininet 2.2.2**	**FIBRE**	**GENI**	**Mini-itx + Desktop**	**Mini-Itx + Switch Pronto**	**OvS SuperMaq**
Desempenho do plano de dados	Médio	Alto	Baixo	Alto	Alto	Alto	Alto
Desempenho do plano de controle	Alto	Alto	Baixo	Alto	Alto	Baixo	Alto
Desempenho do cliente (host)	Médio	Médio	Médio	Médio	Baixo	Baixo	Alto
Controle de vazão	Baixo	Baixo	Baixo	Baixo	Alto	Alto	Alto
Celeridade para implantação	Alto	Alto	Baixo	Baixo	Médio	Médio	Médio
Complexidade para medições	Baixo	Baixo	Alto	Alto	Médio	Médio	Médio
Usa outras versões do OpenFlow	Sim	Sim	Não	Sim	Sim	Sim	Sim
Permite controle in band	Não	Não	Não	Sim	Sim	Não	Sim
Permite controle out band	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Repetibilidade	Sim	Sim	Não	Não	Sim	Sim	Sim
Flexibilidade	Alto	Alto	Médio	Médio	Baixo	Baixo	Baixo

A Tabela 6.2 visa apresentar os tipos de testes que podem ser realizados pelos ambientes apresentados neste trabalho. Com base nos resultados obtidos pelos testes realizados, foi preenchida a tabela levando em consideração os tipos de testes comuns a serem realizados pelos pesquisadores.

1. Vazão: Considera a característica do ambiente para testes em que a vazão é o foco a ser analisado. Foi considerado se o ambiente se comportou com um desempenho satisfatório para esta avaliação. O ambiente Mininet e FIBRE não apresentaram bom comportamento, portanto não são recomendáveis para este tipo de teste.

2. Atraso: Considera o comportamento dos ambientes para testes de atraso. O interesse é a diferença de tempo entre o envio e recebimento de um sinal. Para esta análise considerou-se os testes de pacotes ARP no qual a requisição tinha origem no host 1 com destino o host 2. O ambiente FIBRE com placas NetFPGA apresentou comportamento muito variável, não sendo indicado para estas análises. O ambiente Mininet, com uma topologia pequena, apresentou um resultado mediano em termos de atraso, mas há que se considerar que a taxa de eventos *packet in* era inferior a das outras plataformas. É natural assumir que esse comportamento tende a piorar ao se aumentar a sobrecarga da máquina com mais nós ou mais tráfego.

3. Jitter/UDP: Considera testes em que o ambiente não deve amplificar o *jitter*. Esta avaliação considera os testes realizados para tráfego UDP. Dessa forma, testes que sejam sensíveis ao *jitter*, como jogos e vídeos, são afetados. Os ambientes Mininet foram os que apresentaram maiores valores de Jitter em comparação com os outros ambientes, por isso não devem ser recomendados.

4. Novos recursos (Prototipagem): verifica a possibilidade de que os ambientes sejam úteis para protótipo de novos recursos em ambientes OpenFlow. Todos os ambientes utilizados, considerando suas limitações, permitem tal implementação.

5. Enlaces de longa distância: Avalia-se, com base nas características dos ambientes testados, a possibilidade de realizar testes de longa distância. Somente o ambiente FIBRE e GENI possuem esta característica, porque são ambientes que envolvem a interconexão de ilhas em diferentes regiões do Mundo.

6. Escalabilidade: Avalia-se a possibilidade de expandir os testes com os ambientes usados, sem despesas financeiras adicionais. Levando esta premisa em consideração além do ambiente Mininet, que funciona como um emulador, somente as plataformas públicas permitem esta expansão. Outros ambientes exigem a compra de novos equipamentos. Contudo, o Mininet permite essa expansão com perda de credibilidade dos resultados, já que se observou uma banda total dos enlaces limitada à valores em torno de 9 Gbps e também que o processamento dos nós, quando em excesso,

interfere nos testes. Assim, essa plataforma, de maneira geral, não é interessante para testes de escalabilidade.

Tabela 6.2: Analisar quais tipos de testes podem ser realizados em cada ambiente de experimentação baseado nos resultados obtidos.

Testes	**Mininet 2.2.1**	**Mininet 2.2.2**	**FIBRE**	**GENI**	**OvS Desk-top**	**Switch Pronto**	**OvS Super-Maq**
Vazão	Não	Não	Não	Sim	Sim	Sim	Sim
Atraso	Não	Não	Não	Sim	Sim	Sim	Sim
Jitter/UDP	Não	Não	Sim	Sim	Sim	Sim	Sim
Novos recursos (Prototipa-gem)	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Enlaces de longa distância real	Não	Não	Sim	Sim	Não	Não	Não
Escalabili-dade	Não	Não	Sim	Sim	Não	Não	Não

Cabe sempre observar, contudo, que dependendo do tipo de ambiente, podem ocorrer variações no comportamento que influenciem os resultados obtidos. Tais variações são comuns em ambientes como FIBRE e GENI, que compartilham recursos com outros pesquisadores, e Mininet, dado o compartilhamento dos recursos da máquina física.

Na Figura 6.3, é apresentado um fluxograma de estrutura de decisão baseados em premissas que foram avaliadas nesta dissertação, de forma a ajudar o pesquisador na escolha do ambiente que melhor atenda suas necessidades. O fluxograma se inicia pela premissa de quantidade de nós utilizados. Esta escolha é uma forma de ajudar os pesquisadores a se planejarem na escolha de ambientes com topologia maiores, uma vez que as experimentações nesta dissertação foram baseadas em quatro nós, até o máximo de sete nós no caso dos testes de múltiplas instâncias de OvS numa mesma máquina física.

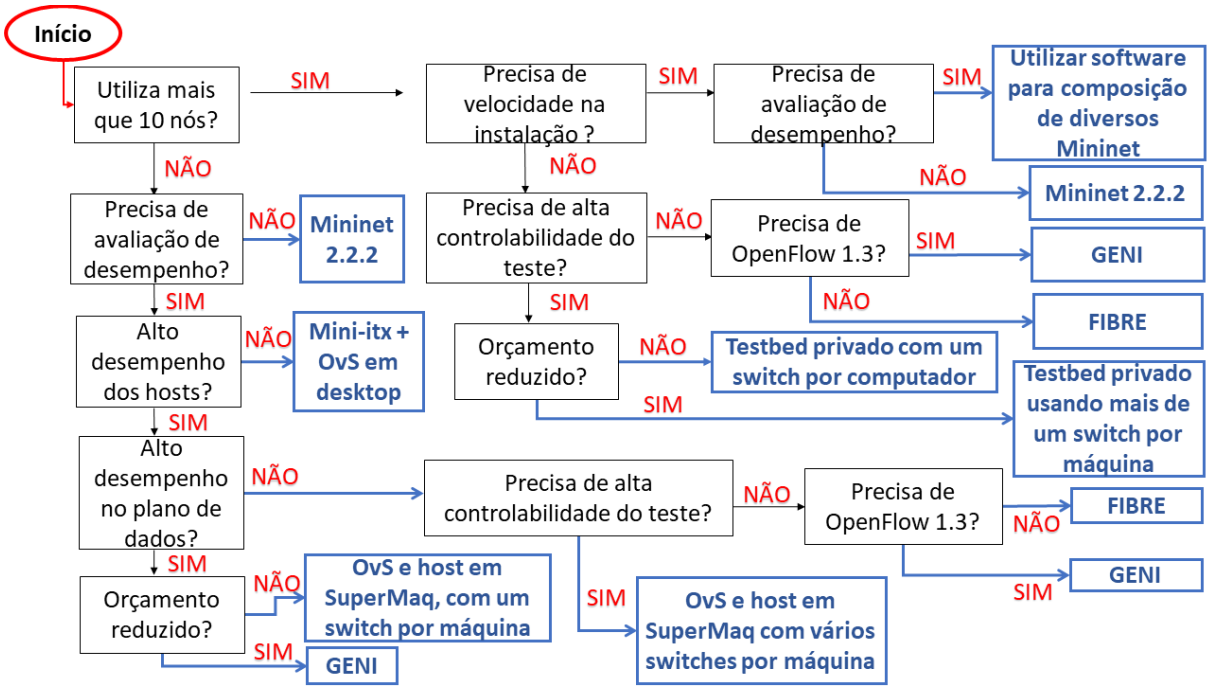


Figura 6.3: Fluxograma de decisão de ambiente de experimentação.

Capítulo 7

Trabalhos Relacionados

Este Capítulo apresenta os principais trabalhos acadêmicos que inspiraram e foram fundamentais para orientar o trabalho de análise do comportamento de ambientes OpenFlow, no qual o conceito de SDN está diretamente associado.

As SDNs apresentam um modelo de arquitetura inovador capaz de entregar provisionamento automatizado, virtualização e programação de rede para data centers e redes corporativas, ganhando continuamente terreno no mercado corporativo e em provedores de serviços em nuvem para data center. De fato, diversos artigos têm sido publicados apresentando meios de utilização do OpenFlow [93] [94], o que demonstra a variedade de campos onde a separação entre o plano de dados do plano de controle tem grande utilidade.

Alguns trabalhos analisam o desempenho das SDN [95] verificando o tipo de projeto e capacidade da infraestrutura em tarefas comuns, pois consideram que a flexibilidade habilitada pelos sistemas baseados em SDN induzem em penalidade de desempenho. Em [96], o foco é na avaliação do desempenho do plano de controle da SDN. Já em [97], é proposta uma ferramenta para avaliar o desempenho de uma arquitetura SDN específica, ou seja, várias implementações de OpenFlow e medições de desempenho sem comparação com outras soluções SDN ou com sistemas de rede não SDN. Algumas análises [98] avaliam a velocidade de encaminhamento do OpenFlow, mas focada principalmente nos efeitos que o controlador possui no plano de encaminhamento.

No meio acadêmico, diversas pesquisas para aprimorar o conceito e entendimento da utilidade da tecnologia SDN são realizadas, abrangendo diferentes enfoques, como análises dos tipos de controladores, no qual pode ser citado o Beacon [22], NOX [99] e Onix [100]; do protocolo OpenFlow [101] e dos equipamentos de diferentes fornecedores que suportam

este protocolo [8].

Alguns trabalhos focam suas análises de desempenho em comparações sobre o comportamento dos switches quando utilizados com o protocolo OpenFlow e no modo *legacy*, avaliando se as implementações do protocolo OpenFlow, são, de fato, realizadas em hardware ou em instâncias de um software switch adaptadas ao mecanismo interno do dispositivo [34].

Outros trabalhos [62] se concentram na influência da heterogeneidade das implementações de switch no desempenho de uma SDN, no qual a capacidade das tabelas de fluxo e a compatibilidade das regras diferem entre os modelos de switch, assim como a colocação de mesmas regras pode ocorrer de forma diferente e a otimização da colocação das regras durante o tempo de execução [102], resultando em um comportamento imprevisível, onde a velocidade de encaminhamento pode cair drasticamente.

O tema do desempenho de SDN atrai muitos pesquisadores e alguns aspectos já foram investigados na literatura. No entanto, vários autores geralmente se concentram em partes selecionadas da arquitetura SDN, como o desempenho do plano de controle, plano de dados, ou um controlador [103] [104] [102] e perdem o conceito geral do sistema.

Além disso, muitos autores [95] não investigam os switches físicos, mas teorizam seus resultados com base em simuladores ou emuladores. O conceito relativamente novo de SDN resultou em múltiplos produtos de hardware que oferecem o suporte para o protocolo OpenFlow. Esta variedade resultou em implementações diferentes e, portanto, o desempenho oferecido pode diferir entre os vendedores ou mesmo entre os modelos de switch do mesmo fornecedor.

Outras avaliações de desempenho de equipamentos OpenFlow focam em uma camada de abstração de hardware convertendo dispositivos não habilitados para OpenFlow em dispositivos habilitados [105], com uso de plataforma de teste OpenFlow Operations Per Second (OFLOPS). Algumas avaliações apresentam somente a análise pontual do dispositivo OpenFlow instalado num desktop com sistema operacional Linux [104], mensurando como é o comportamento do plano de dados neste cenário, sem comparação com outros meios possíveis de instalação de um dispositivo OpenFlow. Sua configuração é semelhante à abordagem NFV, onde o hardware de rede é substituído por servidores. A análise é realizada para caso de fluxo único e múltiplo, de modo que o desempenho de encaminhamento pode ser examinado com (para fluxo único) e sem a presença do tamanho limitado da tabela de encaminhamento (para fluxo múltiplos). Além disso, os autores testam os atrasos do processamento de pacotes para regras de combinação exata e curingas, bem

como para diferentes tipos de tabelas de fluxo e seus tamanhos. Embora os resultados sejam relevantes para uma combinação de SDN e NFV, eles não têm o contexto de um *data Center*, onde geralmente são usados hardwares de switches.

Da mesma forma, outros trabalhos [95] não examinam hardware dos switches para investigar o desempenho. Eles indicam que o SDN possui uma penalidade de desempenho, no entanto, não está necessariamente relacionado ao nível de complexidade da infra-estrutura SDN, apenas comparam o desempenho de uma configuração SDN com o desempenho de dispositivos de rede não SDN. Eles usam um software switch SDN e o ProGFE baseado em Linux, que é uma alternativa ao OpenFlow.

Muitos autores usam técnicas de simulação para resolver modelos de desempenho. Em alguns trabalhos [106], os autores investigam vários parâmetros relacionados ao desempenho de um sistema SDN. O sistema é modelado e examinado no simulador OFSim. Os autores dividem o processo de encaminhamento de pacotes em quatro etapas: plano de dados, plano de dados para controle, plano de controle e processamento de plano de controle para dados. Os autores identificam parâmetros correlacionados com o desempenho e examinam as dependências entre desempenho alcançado, tamanhos de tabela e várias taxas de mensagem (packet_in, flow_mod). Os autores também observam que o gargalo de desempenho pode estar localizado nos switches existentes e o atraso na instalação da entrada da tabela de fluxo. Não há a validação do simulador em oposição a uma configuração com switches físicos.

De maneira muito similar outros trabalhos envolvem o emulador Mininet [59], analisando a sua usabilidade [107] ou verificando as suas limitações em diferentes ambientes [36]. O desempenho do Mininet pode não ser confiável para grandes redes com carga de tráfego pesada. Para permitir experimentos de grande rede, surgiu a MaxiNet [108] que amplia a abordagem de emulação baseada em contêiner para executar Mininet distributivamente em um ambiente de cluster. A MaxiNet amplia o famoso ambiente de emulação Mininet para abranger a emulação em várias máquinas físicas permitindo emular redes muito grandes em SDN. A MaxiNet é executada em um conjunto de várias máquinas físicas chamadas *Workers*. Cada um desses *Workers* executa uma emulação Mininet e só emula uma parte de toda a rede. Switches e hosts estão interligados usando túneis Generic Routing Encapsulation (GRE) em diferentes *Workers*. Todos os pacotes gerados pelas aplicações e que atravessam esses links devem ser encaminhados em todos os túneis GRE de acordo. A largura de banda e a latência dos enlaces entre os *Workers* dependem da conexão de rede física, que pode ser severamente limitada. O MaxiNet fornece uma

API centralizada para controlar a emulação. Esta API é invocada em um Worker especializado chamado Frontend. O Frontend divide e distribui a rede virtual para os Workers e mantém uma lista de qual nó reside em qual Worker. Desta forma, é possível acessar todos os nós através do Frontend.

Outra utilização de emulação Mininet executada em um cluster é o Distributed OpenFlow Testbed (DOT) [109]. Trata-se de um emulador altamente escalavel para SDN, podendo emular grandes implementações, distribuindo a carga de trabalho em um cluster. Estudos de plataforma de testes baseadas em emulação com o Mininet[110] estão sempre em destaque, dada a popularidade da ferramenta no aprendizado de SDN.

Em contrapartida ao uso do Mininet através de cluster, outros trabalhos [111] desenvolveram uma técnica para construir um Mininet de tempo virtual, chamado VT-Mininet, para melhorar a precisão da temporização da emulação. Esta tecnologia desenvolveu um sistema de tempo virtual leve em um contêiner Linux e integrou o sistema com o Mininet, de modo que todos os contêineres tenham seus próprios relógios virtuais em vez de usar o relógio do sistema físico que reflete a execução serializada de múltiplos contêineres. O intuito é que com a noção de tempo virtual, todos os contêineres percebam o tempo virtual como se estivessem executando de forma independente e concorrente. Como resultado, as interações entre os contêineres e o sistema físico são artificialmente dimensionadas, fazendo com que uma rede pareça ser mais rápida do ponto de vista das aplicações dentro dos contêineres do que realmente é.

Outros trabalhos [60], propõem um método alternativo para a decomposição de experiências de rede em grande escala entre instâncias Mininet distribuídas através de uma simulação simbiótica, que integra simulação e emulação em tempo real. O conceito apresentado é que a emulação fornece simulação com informações de fluxo de tráfego geradas pelas aplicações em execução nas máquinas virtuais e a simulação fornece emulação com atualizações periódicas das filas de rede cruzadas pelo tráfego de aplicações reais.

As plataformas de testes públicas têm crescido e estimulado inúmeros pesquisadores de rede para executar seus protótipos e experimentos. Com isso, pesquisas abrangentes sobre os desafios encontrados nestas plataformas são apresentadas [112], porém, comparações que envolvam os diversos ambientes possíveis de simulações em OpenFlow com foco em desempenho, capacidade, custos e facilidades de uso, entre outros, ainda são pouco exploradas.

Neste trabalho, avaliam-se diferentes meios de soluções de implementações de switch OpenFlow para definir quais são mais próprias para a realização de experimentos de novas

ferramentas, ressaltando as suas limitações de desempenho e de custos. São feitas medidas que avaliam ambientes baseados em switches OpenFlow comerciais, implementados em computadores pessoais, emulados e ainda construídos em testbeds públicos.

Diferentemente dos outros trabalhos, a proposta não é meramente avaliar o plano de dados ou o plano de controle OpenFlow, mas fornecer informações que sejam úteis na definição da plataforma de teste adequada a um experimento específico.

Capítulo 8

Conclusão

Neste trabalho, o comportamento de diferentes plataformas de experimentação baseadas em OpenFlow foi avaliado. O objetivo foi avaliar as características, limitações e desempenho de cada plataforma, a fim de oferecer aos pesquisadores entradas consistentes da maturidade de cada ambiente, permitindo uma escolha mais embasada do ambiente que se adequa às suas necessidades de teste.

A escolha do ambiente é uma das premissas para se alcançar um resultado mais fiel. Uma vez que o desempenho do ambiente afeta diretamente o resultado da validação de novas propostas, testes que avaliem o tempo de processamento do ambiente, vazão, atraso, corretamente são essenciais para uma avaliação mais assertiva.

A utilização do ambiente Mininet pode ser uma excelente escolha para o início de aprendizagem sobre o funcionamento do OpenFlow e do conceito de SDN, tendo a vantagem de ser uma máquina virtual obtida diretamente em sites especializados e facilmente colocada em qualquer computador com um software de virtualização. Porém, esta pode ser uma solução muito limitada se o usuário visa realizar testes de desempenho. Até o momento, o Mininet não fornece desempenho e qualidade fiéis a uma rede real. Isto se deve aos recursos que são tratados pelo kernel da máquina servidora no qual ele foi instalado, uma vez que a largura de banda total é limitada por restrições de CPU e como detectado ao longo dos experimentos, pelo módulo TCLink quando emulado para valores elevados de largura de banda.

Como observado, o Mininet 2.2.1 apresentou, na maioria dos casos, um desempenho muito inferior quando comparado aos demais ambientes de teste, por isso é importante o pesquisador estar atento de qual versão do emulador utilizar. O Mininet 2.2.2, lançado em março de 2017, é principalmente uma correção de erros e uma versão de compati-

bilidade para o Mininet 2.2.1. Assim, vale a máxima de sempre utilizar a versão mais nova do software, mesmo que isso implique na necessidade de se executar todos os testes novamente.

Soluções que fazem uso de desktops físicos para implementação de software de switch OpenFlow apresentam bom desempenho e tornando-se uma boa opção para testes e estudos de novas soluções em SDN. O custo de utilização desse mecanismo de implementação apresenta valores financeiros abaixo do que os conseguidos com a utilização de um switch físico habilitado para OpenFlow e, como verificado nos experimentos, a variação de memória no desktop utilizado não afeta de forma consistente os resultados obtidos.

Porém, a CPU pode ser um grande gargalo de desempenho para o OvS. Nos casos em que a carga no OvS é alta, o emulador baseado em OvS, será mais lento do que o hardware. Isto deixa claro que a diferença de desempenho entre os cenários de testes privados foi causada pela limitação de processamento do hardware utilizado. Contudo, dependendo do porte do teste, esta pode ser a solução viável mais próxima da realidade que pode caber em um investimento inicial para montagem da rede de teste. Se a escalabilidade for necessária, contudo, o uso de testbed privado não é uma boa opção. Além disso, este tipo de solução não emula distâncias longas e a variação natural do tráfego da Internet, por exemplo.

O ambiente FIBRE apresentou aspectos positivos e negativos. O ambiente utiliza tanto switches OpenFlow comerciais quanto switches emulados em computadores com placas NetFPGA, mas tem alta variabilidade no desempenho observado, pois o ambiente é compartilhado entre pesquisadores, característica similar a plataforma pública GENI, porém esta plataforma GENI apresentou resultados melhores do que o ambiente FIBRE principalmente nos experimentos no qual era necessário elevada vazão de dados. Não existe controle de banda mínima ou de disponibilidade de recursos nos servidores de máquinas virtuais criadas no FIBRE, podendo ser desconfortável não saber quando um recurso estará ou não disponível ao tentar criar certa topologia, além do fato de cada máquina virtual criada pelo usuário nesta plataforma possuir somente um processador o que afeta nos resultados dos testes. Esta influência do processador ficou nítida nos testes realizados no ambiente FIBRE com foco no processamento de packet_in, no qual a máquina utilizada para funcionar como controlador apresentou picos de uso nos casos de elevado número de packet_in e consequentemente os resultados ficaram aquém do esperado com elevada perda de pacotes.

Porém, percebe-se um constante avanço das universidades que são membros deste pro-

jeto em aperfeiçoá-lo e torná-lo cada vez mais propício para aprendizado e utilização em pesquisas de novas tecnologias. Além disso, as plataformas públicas tem nós espalhados por diversos locais do Brasil e do mundo, tornando o cenário mais real para experimentação. Além disso, apresenta custo de implementação praticamente zero, tornando-se uma boa ferramenta, dependendo dos requisitos do teste.

Portanto, existem várias possibilidades de ambientes de experimentação para aplicações baseadas em OpenFlow, mas nem todos os ambientes são propícios para qualquer tipo de teste, até mesmo a escolha de qual versão do protocolo OpenFlow utilizar pode afetar o resultado dos seus testes, tendo esta premissa, se os testes a serem realizados são compatíveis com versões antigas do protocolo OpenFlow, isto é, menos complexas de análise pelo controlador, a mesma deve ser adotada. Além disso, ficam claras as restrições de cada um dos ambientes e quais são os impactos dele sobre os resultados observados. Assim, as conclusões obtidas em cada tipo de ambiente devem sempre ser balizadas pelas restrições específicas do cenário.

8.1 Trabalhos Futuros

Como trabalhos futuros, pretende-se alinhar a utilização e influência do kernel nas máquinas servidoras onde foi instalado o switch OvS de forma a poder realizar cenários com novas versões de OvS. Utilizar geradores de pacotes com maior capacidade e menor custo computacional, realizar simulações com inserção de vários fluxos simultâneos de forma a estressar o cenário de testes e assim verificar o impacto na memória das máquinas que funcionam como switch e possíveis limitações que podem ser causadas nos testes.

Com relação as plataformas de testes, vislumbra-se o aprimoramento no uso da plataforma GENI, fazendo uso de outras funcionalidades que a mesma possui e não foram empregadas nos experimentos. Além disso, é possível também verificar a compatibilidade de uso do protocolo OpenFlow em outras plataformas de experimentação que fazem parte do projeto GENI, com destaque para PlanetLab e ORBIT.

No caso do ambiente FIBRE, a ilha RNP que apoiou neste trabalho, passou por um processo de evolução, substituindo seus servidores NetFPGA por servidores WhiteBox, tornando os mesmo foco de trabalho futuro para pesquisas relacionadas a esta dissertação. Outra plataforma que será analisada é a Orbit Management Framework (OMF) da Universidade Federal Fluminense (UFF), a qual é uma implementação originalmente desenvolvida para testes sem fio, mas que pode ser usada para realização de testes de redes

cabeadas.

Como os testes não vislumbraram o uso de múltiplos controladores, este será um dos focos de futuros trabalhos, de forma a aprimorar o entendimento do plano de controle verificando como os switches se comunicariam com múltiplos controladores que poderiam assumir diferentes papéis, e como o hardware utilizado para a implementação destes controladores impactariam no resultado destas comunicações.

Outros trabalhos a serem realizados trata-se da utilização de softwares que fazem uso de vários Mininets, como o caso do Maxinet, testes que aumentem a escalabilidade dos ambientes e o uso de simuladores como o OMNet++.

Referências

[1] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, vol. 2, pp. 2–6, 2012.

[2] "Data blast," https://datablast.wordpress.com/2014/02/, accessed: 2017-09-20.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM*, pp. 69–74, 2008.

[4] M. R. Nascimento, C. E. Rothenberg, R. R. Denicol, M. R. Salvador, and M. F. Magalhaes, "Routeflow: Roteamento commodity sobre redes programáveis," *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos-SBRC*, 2011.

[5] S. Seetharaman, "Openflow/sdn tutorial ofc/nfoec," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference*. IEEE, 2012, pp. 1–52.

[6] "Fibre infrastructure," http://fibre.org.br/infrastructure/resources/, accessed: 2017-11-22.

[7] "Geni map," http://www.geni.net/about-geni/what-is-geni/, accessed: 2017-11-22.

[8] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[9] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[10] N. McKeown, "How sdn will shape networking," *Open Networking Summit*, 2011.

[11] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[12] K. Kaur, J. Singh, and N. S. Ghumman, "Mininet as software defined networking testing platform," in *International Conference on Communication, Computing & Systems (ICCCS)*, 2014, pp. 139–42.

[13] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Infinite CacheFlow in software-defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 175–180.

[14] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, and M. F. Magalhães, "Openflow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes," *Cad. CPqD Tecnologia, Campinas*, vol. 7, no. 1, pp. 65–76, 2010.

[15] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, "Sane: A protection architecture for enterprise networks." in *USENIX Security Symposium*, vol. 49, 2006, p. 50.

[16] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 1–12.

[17] "Open networking foundation," https://www.opennetworking.org/, accessed: 2017-09-20.

[18] T. D. Nadeau and K. Gray, *SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies*. "O'Reilly Media, Inc.", 2013.

[19] "Nox controller," https://github.com/noxrepo/nox, accessed: 2017-05-20.

[20] "Pox controller tutorial," http://sdnhub.org/tutorials/pox/, accessed: 2017-05-20.

[21] E. Ng, Z. Cai, and A. Cox, "Maestro: A system for scalable openflow control," *Rice University, Houston, TX, USA, TSEN Maestro-Techn. Rep, TR10-08*, 2010.

[22] D. Erickson, "The beacon openflow controller," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 13–18.

[23] "Project floodlight," http://www.projectfloodlight.org/floodlight/, accessed: 2017-05-20.

[24] "Ryu sdn framework," https://osrg.github.io/ryu-book/en/Ryubook.pdf, accessed: 2016-05-20.

[25] "Opendaylight," https://www.opendaylight.org, accessed: 2017-05-20.

[26] "Open network operating system," http://onosproject.org/, accessed: 2017-05-20.

[27] "Simple network access control," https://github.com/bigswitch/snac, accessed: 2017-05-20.

[28] "Trema - full-stack openflow framework in ruby and c," http://trema.github.io/trema/, accessed: 2017-05-20.

[29] "Open networking foundation. release openflow 1.1," https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.1.0.pdf, accessed: 2017-12-10.

[30] "Open networking foundation. release openflow 1.2," https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.2.pdf, accessed: 2017-12-10.

[31] "Open networking foundation. release openflow 1.3," http://www.cs.yale.edu/homes/yu-minlan/teach/csci599-fall12/papers/openflow-spec-v1.3.0.pdf, accessed: 2017-12-10.

[32] "Open networking foundation. release openflow 1.4," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, accessed: 2017-12-10.

[33] "Open networking foundation. release openflow 1.5," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf, accessed: 2017-12-10.

[34] L. C. Costa, A. B. Vieira, E. de Britto, D. F. Silva, G. Gomes, L. H. Correia, and L. F. Vieira, "Avaliação de Desempenho de Planos de Dados OpenFlow," 2016.

[35] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *Journal of Network and Computer Applications*, vol. 67, pp. 1 – 25, 2016.

[36] F. Keti and S. Askar, "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments," in *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*. IEEE, 2015, pp. 205–210.

[37] "Ns-3 project," http://www.nsnam.org/index.html, accessed: 2017-05-20.

[38] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 60.

[39] S. Chick, P. Sánchez, D. Ferrin, and D. Morrice, "Simulation of large-scale networks using ssf."

[40] G. F. Riley, "The georgia tech network simulator," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*. ACM, 2003, pp. 5–12.

[41] C. D. Carothers, D. Bauer, and S. Pearce, "Ross: A high-performance, low-memory, modular time warp system," *Journal of Parallel and Distributed Computing*, vol. 62, no. 11, pp. 1648–1669, 2002.

[42] D. Klein and M. Jarschel, "An openflow extension for the omnet++ inet framework," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, pp. 322–329.

[43] M. A. Salih, J. Cosmas, and Y. Zhang, "Openflow 1.3 extension for omnet++," in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1632–1637.

[44] "Nsnam webpage," https://www.nsnam.org/doxygen/classns3_1_1_open_flow_switch_net_device.html#details, accessed: 2017-05-20.

[45] M. P. Fernandez, "Comparing openflow controller paradigms scalability: Reactive and proactive," in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*. IEEE, 2013, pp. 1009–1016.

[46] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of software defined networking (sdn) controllers," in *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*. IEEE, 2014, pp. 1–7.

[47] S. A. Shah, J. Faiz, M. Farooq, A. Shafi, and S. A. Mehdi, "An architectural evaluation of sdn controllers," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3504–3508.

[48] L. J. Chaves, I. C. Garcia, and E. R. M. Madeira, "Ofswitch13: Enhancing ns-3 with openflow 1.3 support," in *Proceedings of the Workshop on ns-3*. ACM, 2016, pp. 33–40.

[49] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of sdn/openflow controllers," in *Proceedings of the 9th central & eastern european software engineering conference in russia*. ACM, 2013, p. 1.

[50] M.-C. Chan, C. Chen, J.-X. Huang, T. Kuo, L.-H. Yen, and C.-C. Tseng, "Opennet: A simulator for software-defined wireless local area network," in *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*. IEEE, 2014, pp. 3332–3336.

[51] "S3f/s3fnet: Simpler scalable simulation framework," https://s3f.iti.illinois.edu/, accessed: 2017-09-20.

[52] S.-Y. Wang, C.-L. Chou, and C.-M. Yang, "Estinet openflow network simulator and emulator," *IEEE Communications Magazine*, vol. 51, no. 9, pp. 110–117, 2013.

[53] "Estinet webpage," http://www.estinet.com/ns/, accessed: 2017-05-20.

[54] S.-Y. Wang, C. Chou, C. Huang, C. Hwang, Z. Yang, C. Chiou, and C. Lin, "The design and implementation of the nctuns 1.0 network simulator," *Computer networks*, vol. 42, no. 2, pp. 175–197, 2003.

[55] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[56] J. Ortiz, J. Londoño, and F. Novillo, "Evaluation of performance and scalability of mininet in scenarios with large data centers," in *Ecuador Technical Chapters Meeting (ETCM), IEEE*, vol. 1. IEEE, 2016, pp. 1–6.

[57] "Mininet webpage," http://mininet.org/overview/, accessed: 2017-05-20.

[58] "Ovs webpage," http://openvswitch.org/, accessed: 2017-05-20.

[59] B. Lantz, B. Heller, and N. 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*. ACM, 2010, p. 19.

[60] R. Rong and J. Liu, "Distributed mininet with symbiosis," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[61] E. L. Fernandes and C. E. Rothenberg, "Openflow 1.3 software switch," *Salao de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos SBRC*, pp. 1021–1028, 2014.

[62] P. Rygielski, M. Seliuchenko, S. Kounev, and M. Klymash, "Performance analysis of sdn switches with hardware and software flow tables," in *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2016)*, 2016.

[63] M. Blott, J. Ellithorpe, N. McKeown, K. Vissers, and H. Zeng, "Fpga research design platform fuels network advances," *Xilinx Xcell Journal*, vol. 4, no. 73, pp. 24–29, 2010.

[64] G. Pongrácz, L. Molnár, Z. L. Kis, and Z. Turányi, "Cheap silicon: a myth or reality? picking the right data plane hardware for software defined networking," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 103–108.

[65] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254–265, 2011.

[66] K. Kannan and S. Banerjee, "Compact tcam: Flow entry compaction in tcam for power aware sdn," in *International Conference on Distributed Computing and Networking*. Springer, 2013, pp. 439–444.

[67] P. Goransson and C. Black, *Software Defined Networks: A Comprehensive Approach*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2014.

[68] M. Appelman and M. de Boer, "Performance analysis of openflow hardware," *University of Amsterdam, Tech. Rep*, pp. 2011–2012, 2012.

[69] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[70] M. Ott, I. Seskar, R. Siraccusa, and M. Singh, "Orbit testbed software architecture: Supporting experiments as a service," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on*. IEEE, 2005, pp. 136–145.

[71] T. Salmito, L. Ciuffo, I. Machado, M. Salvador, M. Stanton, N. Rodriguez, A. Abelem, L. Bergesio, S. Sallent, and L. Baron, "Fibre-an international testbed for future internet experimentation," in *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC 2014*, 2014, pp. p–969.

[72] M. Suñé, L. Bergesio, H. Woesner, T. Rothe, A. Köpsel, D. Colle, B. Puype, D. Simeonidou, R. Nejabati, M. Channegowda *et al.*, "Design and implementation of the ofelia fp7 facility: The european openflow testbed," *Computer Networks*, vol. 61, pp. 132–150, 2014.

[73] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "Geni: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[74] I. M. Moraes, D. M. Mattos, L. H. G. Ferraz, M. E. M. Campista, M. G. Rubinstein, L. H. M. Costa, M. D. de Amorim, P. B. Velloso, O. C. M. Duarte, and G. Pujolle, "Fits: A flexible virtual network testbed architecture," *Computer Networks*, vol. 63, pp. 221–237, 2014.

[75] I. MACHADO, L. CIUFFO, D. MARQUES, T. SALMITO, M. STANTON, A. ABELEM, J. F. de REZENDE, M. SALVADOR, S. SALLENT, L. BERGESIO *et al.*, "Building an infrastructure for experimentation between brazil and europe to enhance research collaboration in future internet," in *proc. of TERENA Networking Conference, Dublin, Ireland*, 2014.

[76] "Fibre testbed," http://fibre.org.br/, accessed: 2016-08-20.

[77] "Flowvisor - stanford openflow webpage," https://openflow.stanford.edu/display/DOCS/Flowvisor, accessed: 2016-08-20.

[78] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, vol. 1, p. 132, 2009.

[79] B. Vermeulen, W. Van de Meerssche, and T. Walcarius, "jfed toolkit, fed4fire, federation," in *GENI Engineering Conference (GEC)*, 2014.

[80] T. Wauters, B. Vermeulen, W. Vandenberghe, P. Demeester, S. Taylor, L. Baron, M. Smirnov, Y. Al-Hazmi, A. Willner, M. Sawyer *et al.*, "Federation of internet experimentation facilities: architecture and implementation," in *European Conference on Networks and Communications (EuCNC 2014)*, 2014.

[81] "Geni rack projects," http://groups.geni.net/geni/wiki/GENIRacksHome, accessed: 2017-09-20.

[82] V. Šulák, P. Helebrandt, and I. Kotuliak, "Performance analysis of openflow forwarders based on routing granularity in openflow 1.0 and 1.3," in *Open Innovations Association (FRUCT), 2016 19th Conference of.* IEEE, 2016, pp. 236–241.

[83] O. N. Fundation, "The benefits of multiple flow tables and ttps," ONF Technical Report, Tech. Rep., 2015.

[84] "Pica 8 - open networking," http://www.pica8.com/documents/pica8-datasheet-48x1gbe-p3290-p3295.pdf, accessed: 2016-05-20.

[85] J. Postel, "Transmission control protocol specification," *RFC 793*, 1981.

[86] ——, "Rfc 879: The tcp maximum segment size and related topics," *Internet Engineering Task Force (IETF)*, 1983.

[87] J. Postel and U. D. Protocol, "Rfc 768," *User datagram protocol*, pp. 1–3, 1980.

[88] "Iperf.fr," https://iperf.fr/, accessed: 2016-05-20.

[89] "Scapy," http://www.secdev.org/projects/scapy/, accessed: 2016-05-20.

[90] "Wireshark," https://www.wireshark.org/, accessed: 2016-05-20.

[91] "Tcpdump," http://www.tcpdump.org, accessed: 2016-05-20.

[92] R. A. OLIVEIRA, Y. LOPES, D. C. Muchaluat-Saade, and N. C. FERNANDES, "Analisando o comportamento de ambientes de experimentação baseados em openflow," in *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos 2017-XXII Workshop de Gerência e Operação de Redes e Serviços (WGRS)*, 2017.

[93] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving Energy in Data Center Networks." in *NSDI*, vol. 10, 2010, pp. 249–264.

[94] S. Das, G. Parulkar, N. McKeown, P. Singh, D. Getachew, and L. Ong, "Packet and circuit network convergence with OpenFlow," in *Optical Fiber Communication Conference*. Optical Society of America, 2010, p. OTuG1.

[95] A. Gelberger, N. Yemini, and R. Giladi, "Performance analysis of software-defined networking (sdn)," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*. IEEE, 2013, pp. 389–393.

[96] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks." *Hot-ICE*, vol. 12, pp. 1–6, 2012.

[97] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, A. W. Moore *et al.*, "Oflops: An open framework for openflow switch evaluation." in *PAM*, vol. 7192. Springer, 2012, pp. 85–95.

[98] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an openflow architecture," in *Proceedings of the 23rd international teletraffic congress*. International Teletraffic Congress, 2011, pp. 1–7.

[99] 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*, vol. 38, no. 3, pp. 105–110, 2008.

[100] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A Distributed Control Platform for Large-scale Production Networks." in *OSDI*, vol. 10, 2010, pp. 1–6.

[101] D. F. Macedo, D. Guedes, L. F. Vieira, M. A. Vieira, and M. Nogueira, "Programmable networks-from software-defined radio to software-defined networking," *IEEE Surveys & Tutorials*, vol. 17, no. 2, pp. 1102–1125, 2015.

[102] M. Kuźniar, P. Perešíni, and D. Kostić, "What you need to know about sdn flow tables," in *International Conference on Passive and Active Network Measurement*. Springer, 2015, pp. 347–359.

[103] D. Y. Huang, K. Yocum, and A. C. Snoeren, "High-fidelity switch models for software-defined network emulation," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 43–48.

[104] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow switching: Data plane performance," in *ICC, 2010 IEEE*. IEEE, 2010, pp. 1–5.

[105] M. Zal and J. Kleban, "Performance Evaluation of OpenFlow Devices," 2014.

[106] X. Kong, Z. Wang, X. Shi, X. Yin, and D. Li, "Performance evaluation of software-defined networking with real-life isp traffic," in *Computers and Communications (ISCC), 2013 IEEE Symposium on*. IEEE, 2013, pp. 000 541–000 547.

[107] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using Mininet for emulation and prototyping software-defined networks," in *COLCOM, 2014 IEEE on*. IEEE, 2014, pp. 1–6.

[108] P. Wette, M. Draxler, A. Schwabe, F. Wallaschek, M. H. Zahraee, and H. Karl, "Maxinet: Distributed emulation of software-defined networks," in *Networking Conference, 2014 IFIP*. IEEE, 2014, pp. 1–9.

[109] A. R. Roy, M. F. Bari, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dot: distributed openflow testbed," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 367–368.

[110] B. Lantz and B. O'Connor, "A mininet-based virtual testbed for distributed sdn development," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 365–366.

[111] J. Yan and D. Jin, "Vt-mininet: Virtual-time-enabled mininet for scalable and accurate software-define network emulation," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015, p. 27.

[112] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and J. Liu, "A Survey on Large-Scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges," *IEEE Surveys & Tutorials*, 2016.

APÊNDICE A – Exemplos do Código

A.1 Código para implementação de Ambiente Mininet.

```
1  from mininet.net import Mininet
2  from mininet.topo import Topo
3  from mininet.clean import Cleanup
4  from mininet.topo import MinimalTopo
5  from mininet.link import TCLink
6  from mininet.node import Ryu, RemoteController
7  import sys,os,time
8
9  class mytopo(Topo):
10         def __init__ (self, banda, **opts):
11                 Topo.__init__(self, **opts)
12                 h1 = self.addHost("h1")
13                 h2 = self.addHost("h2")
14                 s1 = self.addSwitch("s1")
15                 if banda != 2000:
16                         l1 = self.addLink(h1, s1,bw=banda)
17                         l2 = self.addLink(h2, s1, bw=banda)
18                 else:
19                         l1 = self.addLink(h1, s1)
20                         l2 = self.addLink(h2, s1)
21
22
23 if __name__ == "__main__":
24         topos = {'mytopo': (lambda:mytopo())}
25         print sys.argv #0 - nome do script / 1 - tipo de teste /
               2 - Tamanho do pacote / 3 - Banda do UDP com
```

```
        letrinha / 4 - banda do enlace em Mbps / 5 - tempo de
         teste / 6 - rodada / 7 - nome do teste / 8 - COMDUMP
         ou SEMDUMP
26      comando = "mkdir "+sys.argv[7]
27      os.system(comando)
28      clean = Cleanup()
29      clean.cleanup()
30      os.system("killall -9 ryu-manager")
31      os.system("killall -9 ryu-manager")
32      comando = "ryu-manager <aplicação.py> > "+sys.argv[7]+"/
           result_controller"
33      for i in range(1,9):
34              comando += '-'+sys.argv[i]
35      comando += " &"
36      os.system(comando)
37      time.sleep(5)
38      topologia = mytopo(float(sys.argv[4]))
39      if sys.argv[4] != 2000: #Nao desabilita o tc
40              net = Mininet(topo=topologia, link=TCLink,
                   controller=RemoteController)
41      else:
42              net = Mininet(topo=topologia, controller=
                   RemoteController)
43
44      ryu=RemoteController('ryu', ip='127.0.0.1', port=6633)
45      net.addController(ryu)
46      net.start()
47      h1, h2 = net.hosts[0], net.hosts[1]
48      l1,l2 = net.links[0],net.links[1]
49
50      if sys.argv[8] == 'COMDUMP':
51              comando = "tcpdump -i h1-eth0 -vvv > "+sys.argv
                   [7]+" arquivo_teste_cliente"
52              for i in range(1,9):
53                      comando += '-'+sys.argv[i]
```

```
54                  comando += " &"
55                  comando1 = "tcpdump -i h2-eth0 -vvv > "+sys.argv
                        [7]+"/arquivo_teste_servidor"
56                  for i in range(1,9):
57                          comando1 += '-'+sys.argv[i]
58                  comando1 += " &"
59
60                  h1.cmd(comando)
61                  h2.cmd(comando1)
62
63          os.system("killall -9 iperf")
64          parameter = "-M "+str(sys.argv[2])
65          result_iperf= net.iperf(fmt=parameter, seconds=int(sys.
                argv[5]), l4Type=sys.argv[1], udpBw=str(sys.argv[3]))
66          comando = sys.argv[7]+"/result_iperf"
67          for i in range(1,9):
68                  comando += '-'+sys.argv[i]
69
70          f=open(comando,'w')
71          f.write(str(result_iperf))
72          f.close()
73          net.stop()
```

Ao utilizar um controlador externo pequenas modificações foram realizadas, mais precisamente no IP de acesso ao controlador, no qual não se utiliza o IP de loopback e sim da máquina externa onde se encontra o controlador.

```
1           ryu=RemoteController('ryu', ip='192.168.0.102', port
                =6633)
```

Outra modificação é com relação ao script referente a aplicação a ser utilizada em testes de OpenFlow 1.0 e OpenFlow 1.3.

A.2 Código para implementação de switch padrão OpenFlow 1.0.

```
1  from ryu.base import app_manager
2  from ryu.controller import ofp_event
3  from ryu.controller.handler import MAIN_DISPATCHER
4  from ryu.controller.handler import set_ev_cls
5  from ryu.ofproto import ofproto_v1_0
6  from ryu.lib.mac import haddr_to_bin
7  from ryu.lib.packet import packet
8  from ryu.lib.packet import ethernet
9  from ryu.lib.packet import ether_types
10
11
12 class SimpleSwitch(app_manager.RyuApp):
13     OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION]
14
15     def __init__(self, *args, **kwargs):
16         super(SimpleSwitch, self).__init__(*args, **kwargs)
17         self.mac_to_port = {}
18
19     def add_flow(self, datapath, in_port, dst, actions):
20         ofproto = datapath.ofproto
21
22         match = datapath.ofproto_parser.OFPMatch(
23             in_port=in_port, dl_dst=haddr_to_bin(dst))
24
25         mod = datapath.ofproto_parser.OFPFlowMod( datapath=
               datapath, match=match, cookie=0, command=ofproto.
               OFPFC_ADD, idle_timeout=0, hard_timeout=0, priority=
               ofproto.OFP_DEFAULT_PRIORITY, flags=ofproto.
               OFPFF_SEND_FLOW_REM, actions=actions)
26         datapath.send_msg(mod)
27
28     @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
29     def _packet_in_handler(self, ev):
```

```
30          msg = ev.msg
31          datapath = msg.datapath
32          ofproto = datapath.ofproto
33
34          pkt = packet.Packet(msg.data)
35          eth = pkt.get_protocol(ethernet.ethernet)
36
37          if eth.ethertype == ether_types.ETH_TYPE_LLDP:
38              # ignore lldp packet
39              return
40          dst = eth.dst
41          src = eth.src
42
43          dpid = datapath.id
44          self.mac_to_port.setdefault(dpid, {})
45
46          self.logger.info("packet in %s %s %s %s", dpid, src, dst
                , msg.in_port)
47
48          # learn a mac address to avoid FLOOD next time.
49          self.mac_to_port[dpid][src] = msg.in_port
50
51          if dst in self.mac_to_port[dpid]:
52              out_port = self.mac_to_port[dpid][dst]
53          else:
54              out_port = ofproto.OFPP_FLOOD
55
56          actions = [datapath.ofproto_parser.OFPActionOutput(
                out_port)]
57
58          # install a flow to avoid packet_in next time
59          if out_port != ofproto.OFPP_FLOOD:
60              self.add_flow(datapath, msg.in_port, dst, actions)
61
62          data = None
```

```
63          if msg.buffer_id == ofproto.OFP_NO_BUFFER:
64              data = msg.data
65
66          out = datapath.ofproto_parser.OFPPacketOut( datapath=
                datapath, buffer_id=msg.buffer_id, in_port=msg.
                in_port, actions=actions, data=data)
67          datapath.send_msg(out)
68
69      @set_ev_cls(ofp_event.EventOFPPortStatus, MAIN_DISPATCHER)
70      def _port_status_handler(self, ev):
71          msg = ev.msg
72          reason = msg.reason
73          port_no = msg.desc.port_no
74
75          ofproto = msg.datapath.ofproto
76          if reason == ofproto.OFPPR_ADD:
77              self.logger.info("port added %s", port_no)
78          elif reason == ofproto.OFPPR_DELETE:
79              self.logger.info("port deleted %s", port_no)
80          elif reason == ofproto.OFPPR_MODIFY:
81              self.logger.info("port modified %s", port_no)
82          else:
83              self.logger.info("Illeagal port state %s %s",
                    port_no, reason)
```

A.3 Código para implementação de switch padrão OpenFlow 1.3.

```
1
2  from ryu.base import app_manager
3  from ryu.controller import ofp_event
4  from ryu.controller.handler import CONFIG_DISPATCHER,
       MAIN_DISPATCHER
5  from ryu.controller.handler import set_ev_cls
6  from ryu.ofproto import ofproto_v1_3
7  from ryu.lib.packet import packet
8  from ryu.lib.packet import ethernet
9  from ryu.lib.packet import ether_types
10
11
12 class SimpleSwitch13(app_manager.RyuApp):
13     OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
14
15     def __init__(self, *args, **kwargs):
16         super(SimpleSwitch13, self).__init__(*args, **kwargs)
17         self.mac_to_port = {}
18
19     @set_ev_cls(ofp_event.EventOFPSwitchFeatures,
           CONFIG_DISPATCHER)
20     def switch_features_handler(self, ev):
21         datapath = ev.msg.datapath
22         ofproto = datapath.ofproto
23         parser = datapath.ofproto_parser
24
25         match = parser.OFPMatch()
26         actions = [parser.OFPActionOutput(ofproto.
               OFPP_CONTROLLER, ofproto.OFPCML_NO_BUFFER)]
27         self.add_flow(datapath, 0, match, actions)
28
29     def add_flow(self, datapath, priority, match, actions,
           buffer_id=None):
```

```
30          ofproto = datapath.ofproto
31          parser = datapath.ofproto_parser
32
33          inst = [parser.OFPInstructionActions(ofproto.
              OFPIT_APPLY_ACTIONS, actions)]
34          if buffer_id:
35              mod = parser.OFPFlowMod(datapath=datapath, buffer_id
                  = buffer_id, priority=priority, match=match,
                  instructions=inst)
36          else:
37              mod = parser.OFPFlowMod(datapath=datapath, priority=
                  priority, match=match, instructions=inst)
38          datapath.send_msg(mod)
39
40      @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
41      def _packet_in_handler(self, ev):
42          if ev.msg.msg_len < ev.msg.total_len:
43              self.logger.debug("packet truncated: only %s of %s
                  bytes", ev.msg.msg_len, ev.msg.total_len)
44          msg = ev.msg
45          datapath = msg.datapath
46          ofproto = datapath.ofproto
47          parser = datapath.ofproto_parser
48          in_port = msg.match['in_port']
49
50          pkt = packet.Packet(msg.data)
51          eth = pkt.get_protocols(ethernet.ethernet)[0]
52
53          if eth.ethertype == ether_types.ETH_TYPE_LLDP:
54              # ignore lldp packet
55              return
56          dst = eth.dst
57          src = eth.src
58
59          dpid = datapath.id
```

```
60          self.mac_to_port.setdefault(dpid, {})
61
62          self.logger.info("packet in %s %s %s %s", dpid, src, dst
                , in_port)
63
64          # learn a mac address to avoid FLOOD next time.
65          self.mac_to_port[dpid][src] = in_port
66
67          if dst in self.mac_to_port[dpid]:
68              out_port = self.mac_to_port[dpid][dst]
69          else:
70              out_port = ofproto.OFPP_FLOOD
71
72          actions = [parser.OFPActionOutput(out_port)]
73
74          # install a flow to avoid packet_in next time
75          if out_port != ofproto.OFPP_FLOOD:
76              match = parser.OFPMatch(in_port=in_port, eth_dst=dst
                    )
77              # verify if we have a valid buffer_id, if yes avoid
                    to send both
78              # flow_mod & packet_out
79              if msg.buffer_id != ofproto.OFP_NO_BUFFER:
80                  self.add_flow(datapath, 1, match, actions, msg.
                        buffer_id)
81                  return
82              else:
83                  self.add_flow(datapath, 1, match, actions)
84          data = None
85          if msg.buffer_id == ofproto.OFP_NO_BUFFER:
86              data = msg.data
87
88          out = parser.OFPPacketOut(datapath=datapath, buffer_id=
                msg.buffer_id, in_port=in_port, actions=actions, data
                =data)
```

```
89              datapath.send_msg(out)
```

A.4 Adaptação de código switch padrão OpenFlow 1.0 considerando MAC origem.

Onde lê-se:

```
# install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:
            self.add_flow(datapath, msg.in_port, dst, actions)
```

Leia-se:

```
# install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:
            self.add_flow(datapath, msg.in_port, src, dst,
                actions)
```

E, onde lê-se:

```
def add_flow(self, datapath, in_port, dst, actions):
        ofproto = datapath.ofproto

        match = datapath.ofproto_parser.OFPMatch(
            in_port=in_port, dl_dst=haddr_to_bin(dst))
```

Leia-se:

```
def add_flow(self, datapath, in_port, src, dst, actions):
        ofproto = datapath.ofproto

        match = datapath.ofproto_parser.OFPMatch(
            in_port=in_port, dl_src=haddr_to_bin(src),dl_dst=
                haddr_to_bin(dst))
```

A.5 Adaptação de código switch padrão OpenFlow 1.3 considerando MAC origem.

Onde lê-se:

```
# install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:
            match = parser.OFPMatch(in_port=in_port, eth_dst=dst
                )
            # verify if we have a valid buffer_id, if yes avoid
                to send both
            # flow_mod & packet_out
            if msg.buffer_id != ofproto.OFP_NO_BUFFER:
                self.add_flow(datapath, 1, match, actions, msg.
                    buffer_id)
                return
            else:
                self.add_flow(datapath, 1, match, actions)
```

Leia-se:

```
# install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:
            match = parser.OFPMatch(in_port=in_port, eth_src=src
                , eth_dst=dst)
            # verify if we have a valid buffer_id, if yes avoid
                to send both
            # flow_mod & packet_out
            if msg.buffer_id != ofproto.OFP_NO_BUFFER:
                self.add_flow(datapath, 1, match, actions, msg.
                    buffer_id)
                return
            else:
                self.add_flow(datapath, 1, match, actions)
```

APÊNDICE B – Exemplos de configurações nas máquinas

B.1 Configuração de múltiplas instâncias de OvS em mesma máquina física.

```
ovs-vsctl add-br <bridge name>
ovs-vsctl add-port <bridge name> <port name>
ovs-vsctl set interface <port name> type=patch
ovs-vsctl set interface <port name> options:peer=<peer name>
ovs-vsctl set-controller <bridge name> tcp:<ip controlador>:6633

ifconfig <bridge name> up
ifconfig <bridge name> <ip do bridge name>
```

Ao final da configuração, ao realizar o comando *ovs-vsctl show* para o caso de 3 instâncias de OvS é obtido o seguinte resultado:

```
02bb7d70-b000-47c7-96c8-96b2f7ffa6c4
    Bridge "s3"
        Controller "tcp:10.0.0.6:6633"
        Port "eth2"
            Interface "eth2"
        Port "s3"
            Interface "s3"
                type: internal
        Port "patchs3-s2"
            Interface "patchs3-s2"
                type: patch
```

```
            options: {peer="patchs2-s3"}
    Bridge "s2"
        Controller "tcp:10.0.0.6:6633"
        Port "patchs2-s1"
            Interface "patchs2-s1"
                type: patch
                options: {peer="patchs1-s2"}
        Port "s2"
            Interface "s2"
                type: internal
        Port "patchs2-s3"
            Interface "patchs2-s3"
                type: patch
                options: {peer="patchs3-s2"}
    Bridge "s1"
        Controller "tcp:10.0.0.6:6633"
        Port "patchs1-s2"
            Interface "patchs1-s2"
                type: patch
                options: {peer="patchs2-s1"}
        Port "eth1"
            Interface "eth1"
        Port "eth0"
            Interface "eth0"
        Port "s1"
            Interface "s1"
                type: internal
    ovs_version: "2.3.2"
```

B.2 Configuração de VLAN em máquinas do FIBRE.

```
ifconfig eth<x> up
modprobe 8021q
vconfig add eth<x> <VLAN>
ifconfig eth<x> 0.0.0.0
ifconfig eth<x>.<VLAN> <IP> up
```

B.3 Configuração de QoS nos hosts.

```
tc qdisc del dev eth<x> root
tc qdisc add dev eth<x> root handle 1: htb default 30
tc class add dev eth<x> parent 1: classid 1:1 htb rate <largura de banda>
tc class add dev eth<x> parent 1: classid 1:2 htb rate <largura de banda>
tc filter add dev eth<x> protocol ip parent 1:0 prio 1 u32 match ip dst
<ip host> flowid 1:1
tc filter add dev eth<x> protocol ip parent 1:0 prio 1 u32 match ip src
<ip host> flowid 1:2
tc -s class ls dev eth<x>
```

APÊNDICE C – Criação de Slice em ambiente FIBRE

C.1 Como criar um slice.

Neste Apêndice é apresentado o passo a passo, desde o acesso ao portal do até a criação do slice.

1- Acesso ao Portal:

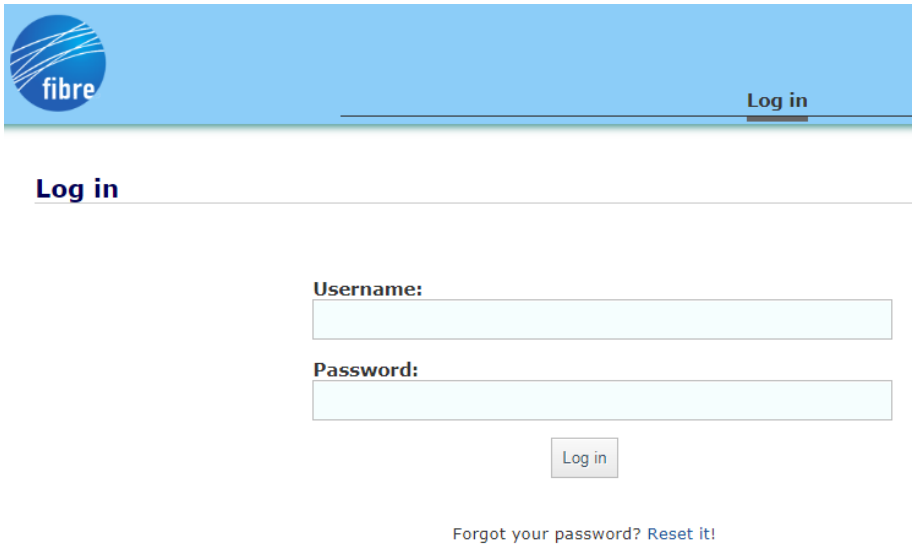
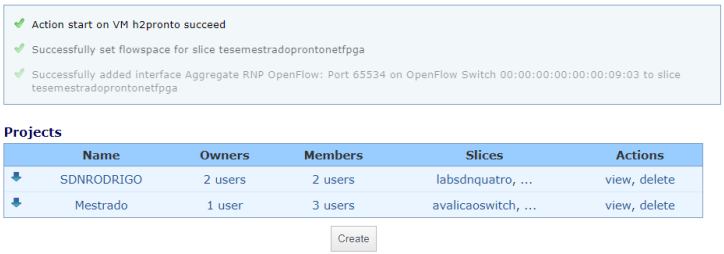
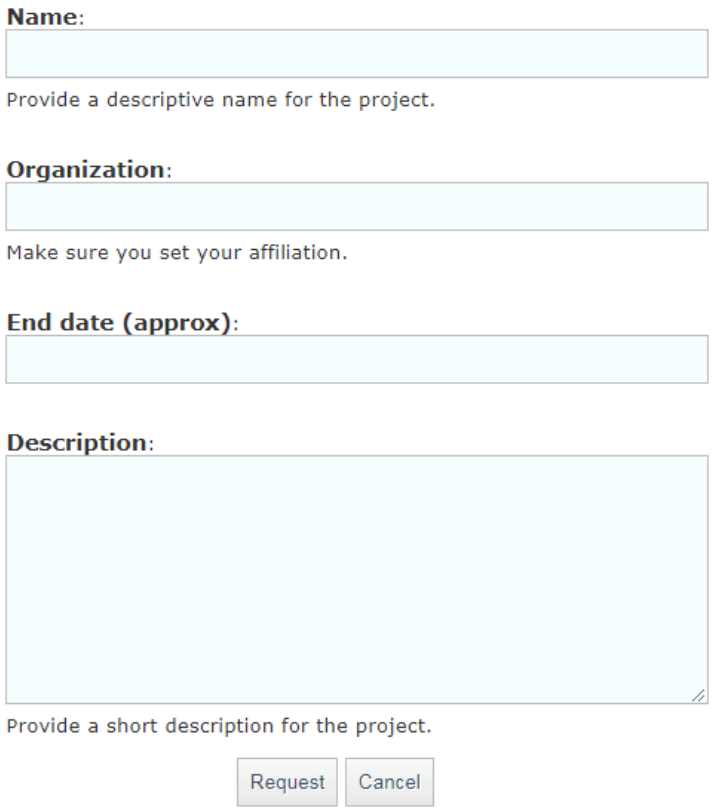


Figura C.1: Página inicial do Portal OFC FIBRE.

2- Solicitação de um projeto, esta etapa depende de aprovação:



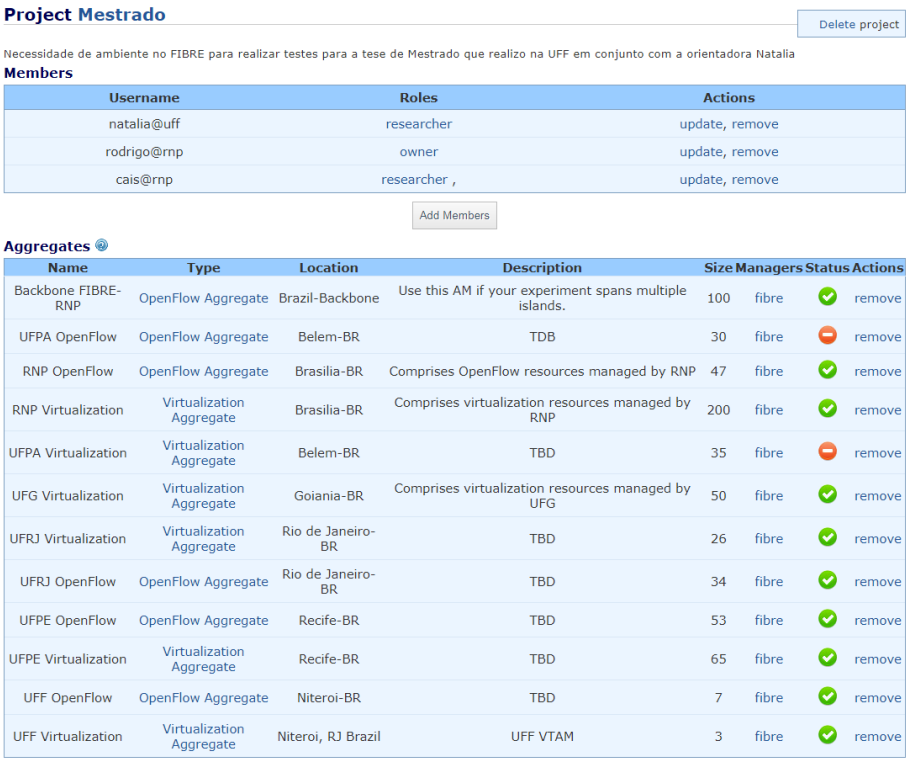
(a) Opção de criar projeto.



(b) Ficha para solicitação de um projeto.

Figura C.2: Como obter um projeto.

3- Adicionar os membros do seu projeto e os agregados, que são as ilhas que irão compor o seu projeto.



Project Mestrado Delete project

Necessidade de ambiente no FIBRE para realizar testes para a tese de Mestrado que realizo na UFF em conjunto com a orientadora Natalia

Members

Username	Roles	Actions
natalia@uff	researcher	update, remove
rodrigo@rnp	owner	update, remove
cais@rnp	researcher ,	update, remove

Add Members

Aggregates

Name	Type	Location	Description	Size	Managers	Status	Actions
Backbone FIBRE-RNP	OpenFlow Aggregate	Brazil-Backbone	Use this AM if your experiment spans multiple islands.	100	fibre	✓	remove
UFPA OpenFlow	OpenFlow Aggregate	Belem-BR	TDB	30	fibre	⊖	remove
RNP OpenFlow	OpenFlow Aggregate	Brasilia-BR	Comprises OpenFlow resources managed by RNP	47	fibre	✓	remove
RNP Virtualization	Virtualization Aggregate	Brasilia-BR	Comprises virtualization resources managed by RNP	200	fibre	✓	remove
UFPA Virtualization	Virtualization Aggregate	Belem-BR	TBD	35	fibre	⊖	remove
UFG Virtualization	Virtualization Aggregate	Goiania-BR	Comprises virtualization resources managed by UFG	50	fibre	✓	remove
UFRJ Virtualization	Virtualization Aggregate	Rio de Janeiro-BR	TBD	26	fibre	✓	remove
UFRJ OpenFlow	OpenFlow Aggregate	Rio de Janeiro-BR	TBD	34	fibre	✓	remove
UFPE OpenFlow	OpenFlow Aggregate	Recife-BR	TBD	53	fibre	✓	remove
UFPE Virtualization	Virtualization Aggregate	Recife-BR	TBD	65	fibre	✓	remove
UFF OpenFlow	OpenFlow Aggregate	Niteroi-BR	TBD	7	fibre	✓	remove
UFF Virtualization	Virtualization Aggregate	Niteroi, RJ Brazil	UFF VTAM	3	fibre	✓	remove

Figura C.3: Membros e agregados "ilhas".

4- Com o projeto obtido, é possível criar uma ou mais fatias (slices) associadas ao projeto:

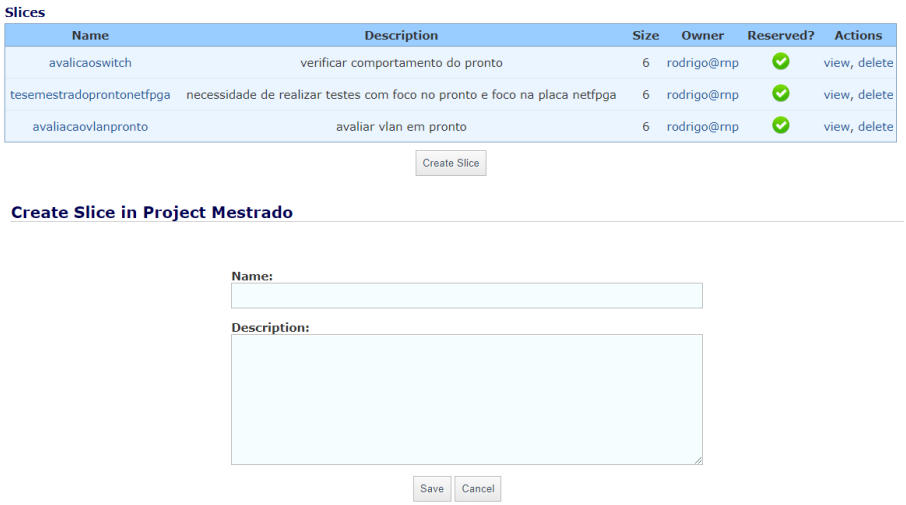


Figura C.4: Criando e nomeando uma fatia.

5- Após a criação da fatia, é necessário adicionar à mesma um ou mais agregados (ilhas) que foram selecionados pelo usuário para compor o projeto:

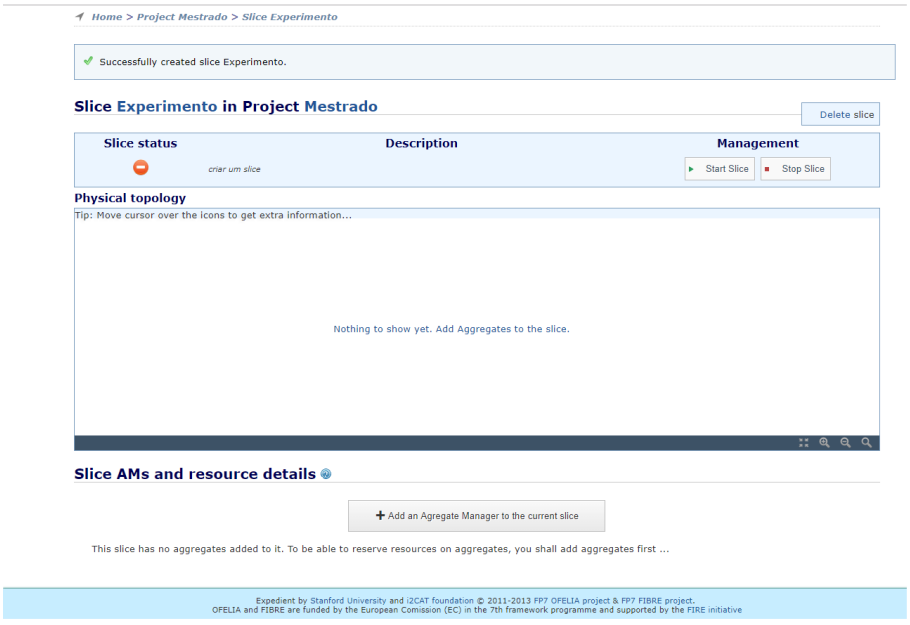


Figura C.5: Tela da fatia criada.

6- Ao selecionar RNP OpenFlow e RNP Virtualization, o usuário tem condições de montar um ambiente de experimentação OpenFlow. Na tela a seguir é possível verificar a caixa de criação de máquinas virtuais, definição da máquina que funcionará como controlador e a solicitação de um flowspace para realização do experimento:

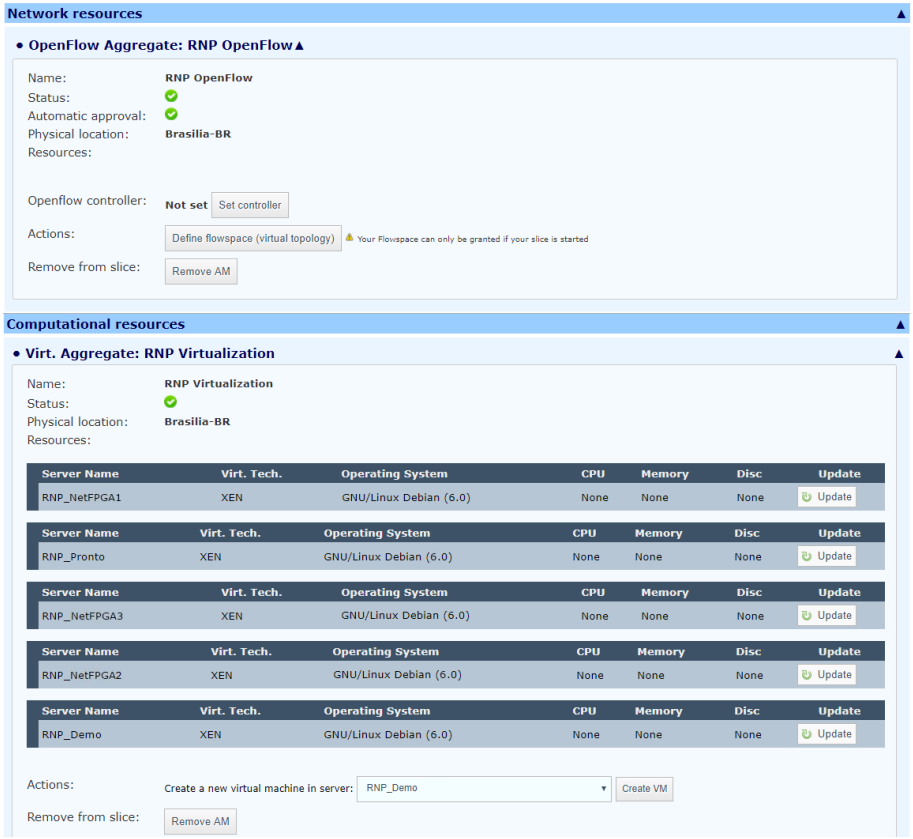


Figura C.6: Itens possíveis de criação para a fatia selecionada.

7- Após criação das máquinas que funcionarão na fatia, é necessário selecionar o flowspace, neste momento é atribuído uma ou mais VLANs, conforme necessário, para a fatia do usuário:

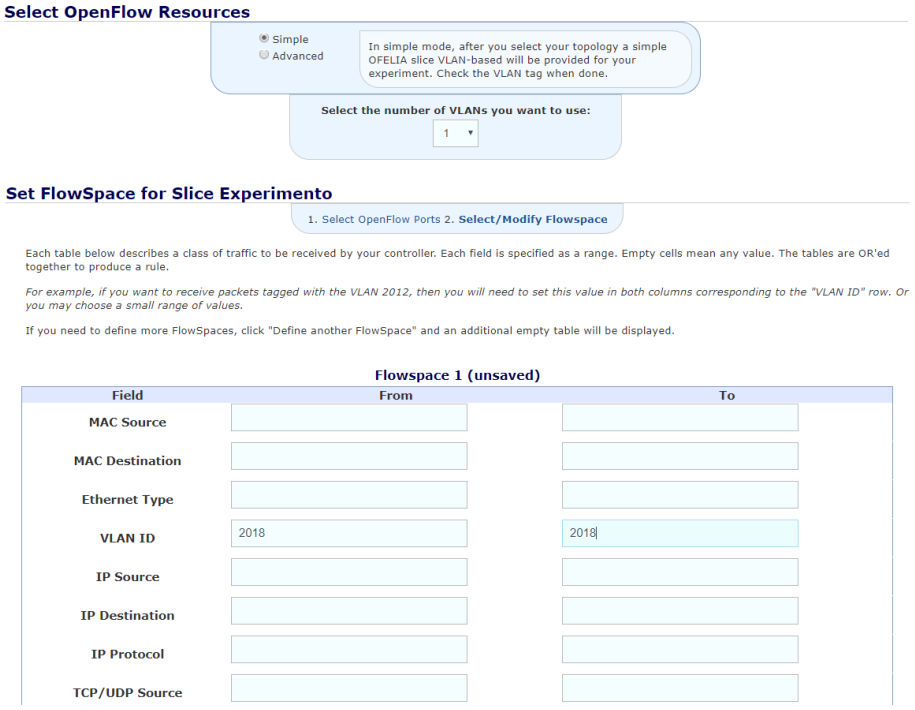


Figura C.7: Configuração do Flowspace.

8- Para finalizar o usuário deve iniciar sua fatia para poder usá-la:

Figura C.8: Iniciando a fatia.