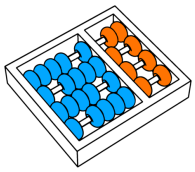
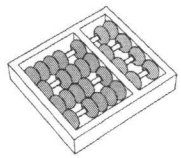


Raphael Vicente Rosa

"Uma Arquitetura para Aprovisionamento de Redes Virtuais Definidas por Software em Redes de Data Center"

CAMPINAS
2014



Universidade Estadual de Campinas
Instituto de Computação

Raphael Vicente Rosa

"Uma Arquitetura para Aprovisionamento de Redes Virtuais Definidas por Software em Redes de Data Center"

Orientador(a): **Prof. Dr. Edmundo Roberto Mauro Madeira**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Raphael Vicente Rosa, sob orientação de Prof. Dr. Edmundo Roberto Mauro Madeira.

Edmundo Madeira

Assinatura do Orientador(a)

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

R71a

Rosa, Raphael Vicente, 1988-
Uma arquitetura para aprovisionamento de redes virtuais definidas por software em redes de data center / Raphael Vicente Rosa. – Campinas, SP : [s.n.], 2014.

Orientador: Edmundo Roberto Mauro Madeira.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

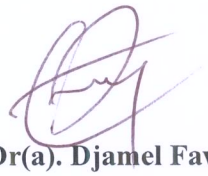
1. Redes de computadores. 2. Arquitetura de redes de computadores. 3. Virtualização de redes. 4. Centros de processamento de dados. I. Madeira, Edmundo Roberto Mauro,1958-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: An architecture for virtual networks defined by software embedding in data center networks
Palavras-chave em inglês:
Computer networks
Architecture of computer networks
Networks virtualization
Data processing service centers
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Edmundo Roberto Mauro Madeira [Orientador]
Djamel Fawzi Hadj Sadok
Nelson Luis Saldanha da Fonseca
Data de defesa: 05-06-2014
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Defesa de Dissertação de Mestrado em Ciência da Computação, apresentada pelo(a) Mestrando(a) **Raphael Vicente Rosa**, aprovado(a) em **05 de junho de 2014**, pela Banca examinadora composta pelos Professores Doutores:



Prof(a). Dr(a). Djamel Fawzi Hadj Sadok
Titular

Prof(a). Dr(a). Nelson Luis Saldanha da Fonseca
Titular

Prof(a). Dr(a). Edmundo Roberto Mauro Madeira
Presidente

Instituto de Computação
Universidade Estadual de Campinas

Uma Arquitetura para Aprovisionamento de Redes Virtuais Definidas por Software em Redes de Data Center

Raphael Vicente Rosa[1]

05 de junho de 2014

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira (Supervisor/*Orientador*)
- Prof. Dr. Djamel Fawzi Hadj Sadok
  Centro de Informática - UFPE
- Prof. Dr. Nelson Luis Saldanha da Fonseca
  Instituto de Computação - UNICAMP
- Prof. Dr. Maurício Ferreira Magalhães
- Prof. Dr. Luiz Fernando Bittencourt

[1]Financial support: CNPq scholarship (process 138210/2011-0) 2011–2013

Abstract

Nowadays infrastructure providers (InPs) allocate virtualized resources, computational and network, of their data center to service providers (SPs) in the form of virtual data centers (VDCs). Aiming maximize revenues and thus efficiently use the resources of their virtualized data centers, InPs handle the problem to optimally allocate multiple VDCs. Even if the allocation of virtual machines in servers can be made using well known techniques and algorithms already existent, cloud computing applications still have performance limitations imposed by the bottleneck of network resources underutilization, which are explicitly defined by bandwidth and latency constraints. Based on Software Defined Network paradigm we apply the Network-as-a-Service model to build a data center network architecture well-suited to the problem of virtual networks embedding. We build services over the control plane of the RouteFlow platform that perform the allocation of virtual data center networks optimizing the utilization of network infrastructure resources. This task is performed by the algorithm proposed in this dissertation, which is based on aggregated information from a virtual routing plane using the BGP protocol and a folded-Clos physical network topology based on OpenFlow 1.3 devices. The experimental evaluation shows that the proposed algorithm performs efficient load balancing on the data center network and altogether yields better utilization of the physical resources. The proposed bandwidth allocation strategy exhibits simplicity and flexibility to attend different traffic communication patterns while yielding an elastic load balanced network. Finally, we argue that the algorithm and the architecture proposed can be extended to achieve performance, scalability and many other features required in data center network architectures.

Resumo

Atualmente provedores de infraestrutura (Infrastructure Providers - InPs) alocam recursos virtualizados, computacionais e de rede, de seus data centers para provedores de serviços na forma de data centers virtuais (Virtual Data Centers - VDCs). Almejando maximizar seus lucros e usar de forma eficiente os recursos de seus data centers, InPs lidam com o problema de otimizar a alocação de múltiplos VDCs. Mesmo que a alocação de máquinas virtuais em servidores seja feita de maneira otimizada por diversas técnicas e algoritmos já existentes, aplicações de computação em nuvem ainda tem o desempenho prejudicado pelo gargalo do subaproveitamento de recursos de rede, explicitamente definidos por limitações de largura de banda e latência. Baseado no paradigma de Redes Definidas por Software, nós aplicamos o modelo de rede como serviço (Network-as-a-Service - NaaS) para construir uma arquitetura de data center bem definida para dar suporte ao problema de aprovisionamento de redes virtuais em data centers. Construímos serviços sobre o plano de controle da plataforma RouteFlow os quais tratam a alocação de redes virtuais de data center otimizando a utilização de recursos da infraestrutura de rede. O algoritmo proposto neste trabalho realiza a tarefa de alocação de redes virtuais, baseado na agregação de informações de um plano virtual executando o protocolo BGP eficientemente mapeado a uma topologia física de rede folded-Clos definida por *switches* com suporte a OpenFlow 1.3. Em experimentos realizados, mostramos que o algoritmo proposto neste trabalho realiza a alocação de redes virtuais de data center de forma eficiente, otimizando o balanceamento de carga e, consequentemente, a utilização de recursos da infraestrutura de rede de data centers. A estratégia de alocação de largura de banda utilizada demonstra flexibilidade e simplicidade para atender a diferentes padrões de comunicação nas redes virtuais ao mesmo tempo que permite elasticidade ao balanceamento de carga na rede. Por fim, discutimos como a arquitetura e algoritmo propostos podem ser estendidos para atender desempenho, escalabilidade, e outros requisitos de arquiteturas de redes de data center.

À minha família, em especial, aos meus pais, Paulo e Helena.

Agradecimentos

Agradeço em especial à minha família, Paulo, Helena, Paula, Heitor e Julia, os quais têm dado apoio incondicional ao longos dos anos a toda minha jornada de formação acadêmica e profissional. Uma família unida e alegre que mantém os alicerces de muitos anos de boas prosas, conselhos com as melhores lembranças de minha vida. Aos meus pais, devo tudo nesta vida, os quais tem minha eterna gratidão por todo esforço que fizeram e continuam a fazer na criação de seus filhos. De todos sinto muita saudade!

Agradeço a meus amigos, companheiros de estudos, disciplinas, boas trocas de ideias no laboratório e também em bares e festas. Todo o trabalho desempenhado nesta dissertação tem uma grande contribuição de meus colegas do Laboratório de Redes de Computadores (LRC). Aos amigos de todos os cantos do Brasil, estes mesmo que longe são companheiros de boas diversões e sem dúvida irão fazer parte de muitas jornadas e conquistas que teremos juntos ao longo dos anos.

Em particular, agradeço a uma grande amiga, Luiza Prado, peça chave em minha motivação e continuidade de estudos ao longo de todo o mestrado. Pessoa com quem conversei bastante sobre estudos, trabalho e vida, que me ensinou muito sobre tudo isso e que contribuiu enormemente para abrir meus horizontes nessa jornada de estudos na UNICAMP.

Agradeço a todos os funcionários e professores do Instituto de Computação com quem tive o prazer de conversar, trocar ideias e aprender bastante! Em especial agradeço a meu orientador Edmundo Madeira, pelo trabalho realizado, assim como a grande dedicação de coorientação do amigo Christian Esteve Rothenberg, que me abriu os horizontes para novas visões de pesquisas no mestrado e doutorado.

E sem dúvidas, penso que a grande recompensa não se dá somente por este trabalho, mas pela amizade e companheirismo que ganhei de vários amigos por diferentes cursos e atividades pela UNICAMP, Barão Geraldo e Campinas. Essa sem dúvida é a melhor experiência que se pode guardar pelo tempo de pesquisas e desenvolvimento dos trabalhos de mestrado, que deixa boas lembranças e perspectivas de um novo caminho a ser trilhado.

"Tendo amor e saúde da vida não reclamo, amo a vida que levo e levo a vida que amo!"

Tião Carreiro

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Acrônimos

API	Application Programming Interface
ASN	Autonomous System Number
BGP	Border Gateway Protocol
DC	Data Center
DCN	Data Center Network
DCTCP	Data Center TCP
ECMP	Equal-Cost MultiPath
EoR	End-of-Row
GRE	Generic Routing Encapsulation
InP	Infrastructure Provider
IP	Internet Protocol
IPv6	Internet Protocol Version 6
LAN	Local Area Network
LLDP	Link Layer Discovery Protocol
LXC	LinuX Container
MAC	Media Access Control
MPLS	MultiProtocol Label Switching
MPTCP	MultiPath TCP

NaaS	Netwok-as-a-Service
NIC	Network Interface Card
ONF	Open Network Foundation
PIF	Physical Interface
QoS	Quality-of-Service
SDN	Software Defined Network
SLA	Service Level Agreement
SN	Substrate Network
SP	Service Provider
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
ToR	Top-of-Rack
UDP	User Datagram Protocol
VDC	Virtual Data Center
VDCN	Virtual Data Center Network
VIF	Virtual Interface
VLAN	Virtual Local Area Network
VLB	Valiant Load Balancing
VLiM	Virtual Link Mapping
VM	Virtual Machine
VN	Virtual Network
VNE	Virtual Network Embbeding
VNoM	Virtual Node Mapping
VPN	Virtual Private Network
VXLAN	Virtual Extensible Local Area Network
WAN	Wide Area Network

Capítulo 1

Introdução

Provedores de infraestrutura (Infrastructure Providers - InPs) têm virtualizado os data centers que possuem criando uma diversidade de recursos (ex: máquinas virtuais, switches e roteadores virtuais) que são utilizados pelos próprios InPs e por diversos provedores de serviços (Service Providers - SPs). Neste modelo de negócio, um data center virtualizado por um InP pode prover recursos a diferentes SPs, os quais constroem seus respectivos data centers virtuais (Virtual Data Centers - VDCs) sobre o compartilhamento de infraestruturas físicas de data centers (DCs) [5]. A virtualização de servidores permite o particionamento de recursos físicos em múltiplas máquinas virtuais (Virtual Machines - VMs) isoladas e atualmente pode ser feita por diversas tecnologias (ex: VMWare, Xen, KVM). Da mesma forma, a virtualização de rede define a alocação de recursos de rede (ex: largura de banda, *switches*, endereços), e tem sido amplamente implementada em redes de data center [45]. No entanto, a repartição e uso eficiente de recursos de rede, como largura de banda, ainda é um desafio que diversas propostas de arquiteturas de redes de data center vem tentando solucionar [57].

Um data center virtualizado provê recursos computacionais e de rede a conjuntos de VMs (*tenants*) formando VDCs e permitindo que eles apliquem suas próprias políticas, definam seu esquema de endereçamento, gerenciem seu próprio conjunto de VMs, etc. Por esse aspecto, um SP compartilhando uma infraestrutura de rede de data center (Data Center Network - DCN) pode possuir diversos requisitos em suas redes virtuais, tais como isolamento de recursos, gerenciamento flexível de alocações de tráfego, tolerância a falhas, balanceamento de carga e implantação de novas aplicações. Estes requisitos muitas vezes necessitam ser alterados dinamicamente conforme a demanda de planos de serviços oferecidos a clientes. Tanto os requisitos citados quanto a flexibilidade de suas implementações não são suportados por redes de data center tradicionais por causa da sua grande dependência de tecnologias atreladas às pilhas de protocolos TCP/IP (ex: Virtual Local Area Networks - VLANs), o que cria diversas limitações como: não isolamento

de desempenho; riscos de segurança aumentados; pobre implementação de aplicações; flexibilidade limitada de gerenciamento; e não suporte a inovações de rede [5].

Em pesquisas recentes, o desempenho de serviços de rede é peça fundamental no processo de alocação de aplicações de clientes com requisitos de comunicação em computação em nuvem [1]. Além disso, é previsto que o tráfego em redes de data center advindo da computação em nuvem triplique no período 2012-2017, sendo parte maior desta contribuição o tráfego interno, denominado leste-oeste [14]. Agregando estes fatores aos requisitos de mapeamento de redes virtuais previamente mencionados, vemos que DCNs impõem um enorme gargalo às aplicações de computação em nuvem, o que ao contrário, deveria permitir uma visão lógica como uma única entidade virtual. Nesse aspecto, recursos de rede devem refletir demandas práticas, e não ficarem sujeitas às restrições *one-size-fits-all* de arquiteturas de DCNs existentes [52]. Isso possibilita um ganho sem precedentes para InPs e SPs, de modo que a programabilidade, automação e controle da rede sejam realizados de modo a permitir a construção de DCNs escaláveis, flexíveis e inovadoras que realmente se adaptem à realidade dos requisitos dos modelos de negócios atualmente visto na computação em nuvem [2].

Inspirados pela proposta [56], seguindo o paradigma de Redes Definidas por Software (Software Defined Networks - SDNs) [20], neste trabalho propomos utilizar o conceito de Rede como Serviço (Network-as-a-Service - NaaS) [29] aplicado a uma nova concepção de construção de redes virtuais de data center. Propomos uma arquitetura que possibilita o tratamento de alocações dinâmicas de redes virtuais de data center com requisitos de largura de banda e resiliência. Consideramos um cenário onde um InP possua a necessidade de compartilhar sua infraestrutura física de data center para diversos SPs os quais já possuam suas demandas de alocações bem definidas de máquinas virtuais em servidores. Portanto, tratamos a alocação de enlaces virtuais para interconectar essas VMs possibilitando balanceamento de carga à infraestrutura de rede do InP.

Utilizando a plataforma RouteFlow [50] definimos um plano virtual executando o protocolo de roteamento de borda (Border Gateway Protocol - BGP) com suporte a múltiplos caminhos considerando as premissas de [33] para operar em DCNs com topologia folded-Clos. No plano de dados utilizamos uma infraestrutura física com suporte a versão 1.3 do padrão OpenFlow [39]. E no plano de controle da plataforma, construímos serviços que agregam informações dos planos virtual e de dados, e logo, possibilitam que estas sejam utilizadas em conjunto para tratar o mapeamento de redes virtuais. O algoritmo proposto neste trabalho aloca largura de banda em enlaces da topologia física do InP através do mapeamento de rotas da topologia virtual às topologias virtuais criadas para cada SP. Políticas, contendo requisitos, como largura de banda, resiliência e regras de endereçamento, representam demandas de SPs para alocação de redes virtuais. Estas são utilizadas pelo algoritmo de mapeamento para construir grafos de redes virtuais de SPs

por meio da configuração de rotas na topologia física de rede do data center. Tais rotas são definidas para prover a SPs isolamento de recursos e suporte às funcionalidades que o padrão OpenFlow 1.3 oferece, como *group tables* e *meter tables*.

A arquitetura proposta para a alocação de redes virtuais proposta neste trabalho é avaliada juntamente com o algoritmo proposto, nos aspectos de *overhead* de protocolo de roteamento, balanceamento de carga e resiliência. A arquitetura e algoritmo criados são comparados com propostas recentes de arquiteturas para mapeamento de redes virtuais em data centers. Logo, a principal contribuição deste trabalho é a definição de uma arquitetura para dar suporte ao mapeamento de redes virtuais como serviço baseada em abstrações de padrões de comunicação entre sistemas finais a fim de prover alocação de recursos de DCNs de forma eficiente. Neste sentido este trabalho se distingue das soluções existentes pelos seguintes aspectos: *(i)* propõe, constrói e avalia uma arquitetura para mapeamento de redes virtuais de data centers seguindo o modelo NaaS e o paradigma de SDNs; *(ii)* propõe a configuração de uma topologia virtual a qual utiliza o protocolo BGP de forma eficiente para o roteamento em data centers com topologia folded-clos; *(iii)* estende a plataforma RouteFlow para dar suporte ao padrão OpenFlow 1.3 e para prover suporte a diferentes aplicações de políticas de redes virtuais, tais como reservas de largura de banda e rotas por múltiplos caminhos; *(iv)* e define um algoritmo de mapeamento de redes virtuais que propicia reservas de largura de banda a redes virtuais de SPs de forma dinâmica atendendo requisitos de largura de banda e resiliência de maneira eficiente.

Os capítulos desta dissertação estão divididos da seguinte maneira:

- O Capítulo 2 apresenta os principais aspectos teóricos envolvidos no desenvolvimento do trabalho realizado. Procuramos nesse capítulo descrever os conceitos básicos relacionados a redes de data center, abordar as principais definições e características de Redes Definidas por Software, descrever o modelo de Rede como Serviço, os conceitos sobre virtualização de redes assim como o problema de alocação de redes virtuais e, de maneira mais específica, os desafios da alocação de redes virtuais de data center;

- O Capítulo 3 define e detalha a arquitetura construída neste trabalho para tratar o problema de alocação de VDCNs. Buscamos nesse capítulo elaborar uma descrição detalhada sobre a plataforma RouteFlow, explicando seu funcionamento e seus componentes, além de como cada um deles foi modificado para atender aos requisitos de nossa proposta. Do mesmo modo, descrevemos os serviços criados sobre o plano de controle da plataforma assim como definimos e explicamos os mecanismos e algoritmos de mapeamento de redes virtuais criados com base nesses serviços.

- No Capítulo 4 detalhamos como foram elaborados os experimentos executados para avaliar a arquitetura e os algoritmos desenvolvidos neste trabalho. Descrevemos o

ambiente de testes criado e por fim mostramos os resultados obtidos de cada conjunto de experimentos realizados. Além disso, realizamos uma discussão minuciosa sobre os resultados apresentados. Visamos, portanto, a apresentação dos resultados através da comparação dos requisitos de análise do problema de alocação de VDCNs mostrados no Capítulo 3. Com isso, conseguimos mostrar as principais características de nosso trabalho e explicar como ele pode ser estendido para atender aos principais requisitos gerais de arquiteturas de DCNs.

- No Capítulo 5 são apresentados os trabalhos relacionados ao foco deste trabalho, a alocação de redes virtuais de data center. Buscamos para isso, abordar e detalhar as principais características dos mecanismos e arquiteturas de alocação de VDCNs atualmente vistos na literatura. E com isso, de forma comparativa, contendo os principais requisitos de alocação de VDCNs, apresentamos uma tabela com esses trabalhos relacionados e a nossa proposta;

- O Capítulo 6 conclui esta dissertação, de modo a enfatizar suas principais contribuições assim como definir os possíveis trabalhos futuros a serem feitos para a criação de novos projetos relacionados às SDNs e DCNs.

Capítulo 2

Conceitos Teóricos

Nesta seção buscamos explicar os principais conceitos relacionados ao trabalho desenvolvido. Dentre os tópicos descritos estão: Redes de Data Center, Redes Definidas por Software, Rede como Serviço, Virtualização de Redes de Data Center, este último tendo como ênfase os assuntos sobre Alocação de Redes Virtuais (Virtual Network Embedding - VNE).

2.1 Redes de Data Center

Procuramos descrever Redes de Data Center com o objetivo de mostrar as raízes dos problemas comumente vistos neste ambiente que desperta a atenção de diversas pesquisas na atualidade. Elucidando os componentes, topologias e arquiteturas, visamos em DCNs os principais questionamentos acerca de formas de gerenciamento de tráfego, principalmente aqueles relacionados à alocação de redes virtuais e balanceamento de carga.

Um data center é formado por servidores (máquinas físicas com CPU, memória e sistemas de armazenamento), dispositivos de rede (ex: roteadores, *switches* e cabeamento de rede), sistemas de distribuição de energia e sistemas de refrigeração. Uma rede de data center é a infraestrutura de comunicação utilizada no data center, sendo constituída pela topologia de rede, equipamentos de comutação e roteamento, e os protocolos de rede (ex: Ethernet, IPv6) [5]. Em uma infraestrutura de DCN, os servidores físicos são tipicamente montados em conjunto dentro de um *rack* e interligados através de um *switch* de acesso. Este *switch*, denominado topo de *rack* (Top-of-Rack - ToR), interliga-se a um ou mais *switches* de agregação (End-of-Row - EoR), agregando *clusters* de servidores. Este segundo nível de domínio de comutação pode, potencialmente, abranger mais de 10 mil servidores individuais. Finalmente, um terceiro nível de *switches* categorizado como central (Core) realiza a interligação dos diferentes *switches* EoR e seus respectivos *clusters* de servidores. A abordagem de topologia em camadas é um fundamento básico de data

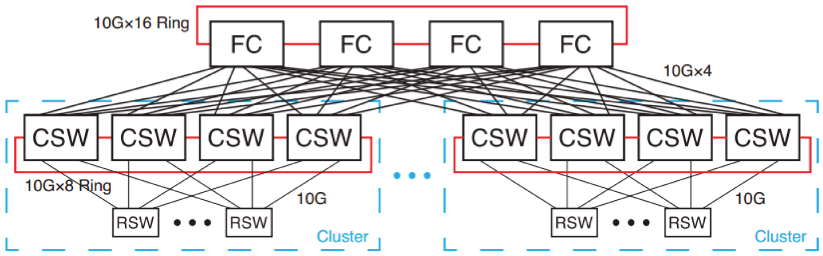


Figura 2.1: Arquitetura de data center (4-post) da empresa Facebook contendo RSWs (*rack* switches), CSWs (cluster switches) e FCs (FatCat aggregation switches) [19]

centers para prover escalabilidade, alto desempenho, flexibilidade, resiliência e facilidade de manutenção [57]. Na Figura 2.1 podemos ver um exemplo claro de uma infraestrutura de DCN utilizada pela empresa Facebook com a utilização de 3 camadas de dispositivos de rede para interligação de seus servidores.

Conforme a arquitetura, DCNs podem possuir diferentes tipos de topologias. Uma topologia Clos é definida por múltiplos estágios de *switches*, de forma que cada *switch* de um estágio esteja conectado a todos os *switches* dos outros estágios, o que provê uma grande diversidade de caminhos à rede. Folded-Clos e FatTree (Figura 2.2) são tipos especiais de topologia Clos com suas respectivas particularidades, tais como, requisitos de quantidade de portas em *switches*, custo de cabeamento e largura de banda agregada. Além disso, novas propostas de topologias foram feitas nos últimos anos tais como, BCube [24] (Figura 2.2) e Jellyfish [55]. Em [42] é apresentada uma comparação detalhada sobre as particularidades de diferentes arquiteturas e topologias de DCNs.

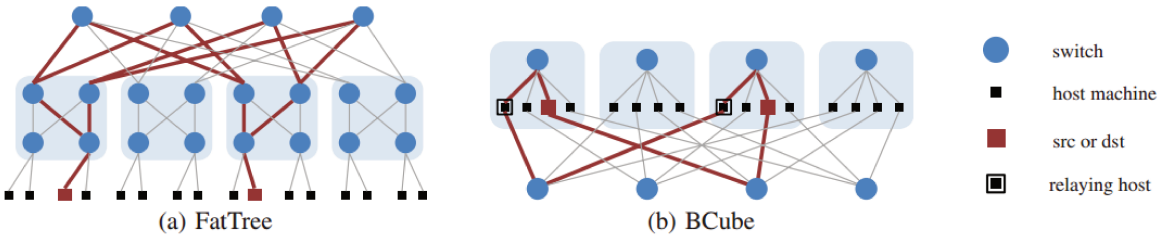


Figura 2.2: Exemplo de topologias de DCNs: FatTree e BCube [46]

Além dos dispositivos de rede citados, atualmente, *switches* virtuais (ex: Open vSwitch) foram incorporados a servidores e hypervisors (ex: Xen, KVM) permitindo programabilidade na comunicação de servidores e máquinas virtuais entre si e a rede externa por meio de diferentes tecnologias de rede (ex: Nicira NVP, Cisco Nexus 1000V). Esta existência de indireção na comunicação entre máquinas virtuais por toda a DCN cria uma enorme

transparência entre diferentes *tenants*, além de criar uma nova camada de comutação entre máquinas virtuais pela criação de redes sobrepostas (*Overlay Networks*), o que tem facilitado tarefas de migração, atribuição de endereços, isolamento de performance e controle de VMs em todo o data center [40]. Na Figura 2.3, é mostrada a arquitetura do switch virtual Open vSwitch sendo representado como domínio de gerenciamento (Management Domain), com VMs e suas interfaces virtuais (VIF) sendo controladas em níveis de espaço de usuário e de *kernel* do sistema operacional juntamente com interfaces físicas (PIF) e interfaces externas.

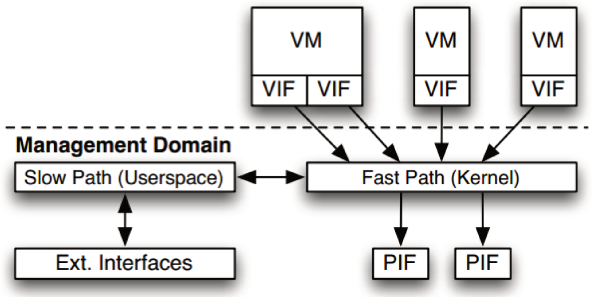


Figura 2.3: Arquitetura de *switch* virtual *Open vSwitch* [40]

DCNs podem ser dimensionadas de diferentes formas. Quando a largura de banda oferecida aos servidores de um *rack* se iguala à largura de banda de conectividade oferecida por switches ToR e EoR diz-se que a rede é bem provisionada. Ou seja, a largura de banda agregada em toda a rede é capaz de atender a comunicação fim a fim de todos os servidores se comunicando em suas capacidades máximas de utilização de largura de banda. Já quando a largura de banda requerida pelos servidores é menor do que aquela oferecida pelos switches, existe *oversubscription* na rede. Isto significa que há um fator de *oversubscription* que determina a razão entre as larguras de banda requerida e oferecida na rede. Por exemplo, *racks* contendo switches ToR conectados a 10 servidores cada um com interfaces 1 Gbps e interligados cada um a switches EoR por 4 interfaces com capacidade de 1Gbps cada, determinam um fator de *oversubscription* entre ToRs e EoRs igual a 2,5. Esta proporcionalidade pode ser mantida ou aumentada conforme a largura de banda oferecida na interconexão dos *switches* EoR e Core. Partindo do exemplo anterior, se cada *switch* EoR se comunica aos switches Core por uma interface de rede a 1Gbps, então o fator de *oversubscription* da rede passa a valer 10. Ou seja, os fatores de *oversubscription* são multiplicados a cada camada de comunicação [57].

2.1.1 Problemas e Pesquisas

Grandes companhias como Google, Microsoft, Amazon e Facebook, utilizam data centers para armazenamento e processamento em larga escala. Com o surgimento da computação em nuvem estas empresas tem desempenhado um papel fundamental no futuro da indústria de Tecnologia da Informação operando um mercado da ordem de bilhões de dólares e investindo cada vez mais em serviços e tecnologias para data centers. Apesar das arquiteturas atuais utilizarem tecnologias de virtualização de servidores (ex: VMware, Xen), no que diz respeito às DCNs elas ainda confiam largamente nas tradicionais pilhas de protocolos TCP/IP o que resulta em uma série de limitações [5], como:

- Não isolamento de desempenho: aplicações de computação em nuvem, como serviços web, tem rigorosos requisitos de latência e largura de banda, os quais não são garantidos pelas tradicionais tecnologias de rede de melhor serviço, dificultando assim a previsibilidade de QoS para estas aplicações [6, 9]. Por exemplo, a alocação estática de serviços atrelada à fragmentação de recursos de rede de um data center impossibilitam garantias mínimas de latência e vazão na comunicação de servidores;

- Riscos de segurança: em tecnologias empregadas em DCNs não há suporte a restrições de utilização de largura de banda para aplicações, o que pode resultar vulnerabilidades a ataques internos como negação de serviço;

- Implementação limitada de novas aplicações: muitas empresas utilizam protocolos e faixas de endereçamento próprios de suas aplicações. Somente com a alteração dos protocolos destas aplicações seria possível migrá-las para um ambiente de data center preso a características limitadas de configuração de rede;

- Flexibilidade de gerenciamento limitada: em um data center onde aplicações compartilham recursos de servidores e de rede, gerentes de aplicações muitas vezes tem a necessidade de gerenciar a malha de rede de suas aplicações no intuito de prover balanceamento de carga e diagnósticos de falhas. No entanto, arquiteturas tradicionais de data centers não permitem esta flexibilidade de gerenciamento da própria interconexão de rede de uma *tenant*;

- Ausência de suporte à inovação de rede: introduzir mudanças nas redes tradicionais de data centers pode ser difícil, tais como a atualização de serviços de rede ou a introdução de novos protocolos de rede, o que em longo prazo, reduz a eficácia do capital investido em redes de data center, tornando portanto a aplicação de técnicas de *scale-up* a equipamentos de rede cada vez menos eficientes;

- Eficiência energética: alocações de cargas de trabalho proporcionais ao consumo energético precisam ser feitas para tornar arquiteturas de data centers eficientes

do ponto de vista ecológico. Isto pode ser feito garantindo agilidade às aplicações, possibilitando que a rede aja como um fator determinante para tornar a alocação de recursos o mais flexível possível, e não como um empecilho, entravando a capacidade crescente da dependência inerente de aplicações distribuídas ao total aproveitamento da comunicação oferecida pela topologia de rede.

2.2 Virtualização de Redes

Historicamente, a virtualização de recursos computacionais (ex: CPU, memória, disco) tem permitido a usuários e aplicações se libertarem das limitações impostas por recursos físicos. Por exemplo, uma aplicação, executada em ambiente virtualizado, que constantemente necessita de acesso a uma grande quantidade de memória, tem como perspectiva a existência de um grande banco dedicado de memória física quando na verdade está acessando recursos compartilhados com outras aplicações. Logo, um ponto chave a respeito desta solução é a virtualização de componentes de hardware a qual preserva as abstrações dos recursos que estão sendo virtualizados [12].

A virtualização de redes existe há um bom tempo na forma de diversas tecnologias já bem definidas e empregadas comercialmente, tais como VLAN, *MultiProtocol Label Switching* (MPLS), *Virtual Private Network* (VPN) e redes sobrepostas. Virtualização de redes é um conceito promissor para as tecnologias da Internet do Futuro [58]. Introduzida como uma forma de testes de novos protocolos e serviços de rede, atualmente é vista como uma forma de tratar a resistência a mudanças fundamentais da Internet, pois permite principalmente que testes e experimentos com novos protocolos sejam feitos utilizando redes legadas. Ela tem como entidade primária a Rede Virtual (Virtual Network - VN), a qual é constituída por nós e enlaces virtuais de rede sob um substrato de rede (Substrate Network - SN). Nós virtuais são interconectados por meio de enlaces virtuais formando assim uma topologia virtual. Múltiplas topologias virtuais podem coexistir sobre uma mesma infraestrutura física de rede [21]. Técnicas de virtualização de redes possuem diferentes características de acordo com: a tecnologia empregada na virtualização, a qual vai permitir a disponibilização de atributos sobre a plataforma de rede virtualizada; a camada de virtualização, que conforme seu nível permite maior flexibilidade; o domínio da arquitetura, o qual dita os serviços que são oferecidos às redes virtualizadas; e a granularidade da virtualização, que determina até onde a rede virtualizada pode se auto gerenciar [13].

2.2.1 Mapeamento de Redes Virtuais

O problema de instanciar redes virtuais em um substrato de rede é um desafio de alocação de recursos sob o aspecto da virtualização de redes, e é comumente definido como o problema de Virtual Network Embedding (VNE), como visto na Figura 2.4. Este problema pode ser decomposto em duas funções, uma de mapeamento de nós virtuais (Virtual Node Mapping - VNoM) e outra de mapeamento de enlaces virtuais (Virtual Link Mapping - VLiM). A execução de um algoritmo de mapeamento de resolução do problema de VNE pode ser definida ocorrendo em um único estágio ou por partes. No primeiro caso, ocorre o VNoM e o VLiM de forma integrada, ou seja, a medida que nós virtuais vão sendo mapeados, os enlaces destes nós são continuamente mapeados originando o mapeamento de novos nós virtuais. Já em etapas, geralmente o mapeamento primário ocorre com o VNoM e em seguida o VLiM realiza a interconexão dos nós virtuais. Além disso, esses mapeamentos podem ser coordenados ou não, por exemplo, se o VNoM realiza mapeamento para nós virtuais que dispõem de uma maior quantidade de mapeamentos de enlaces virtuais facilita a tarefa de VLiM [10].

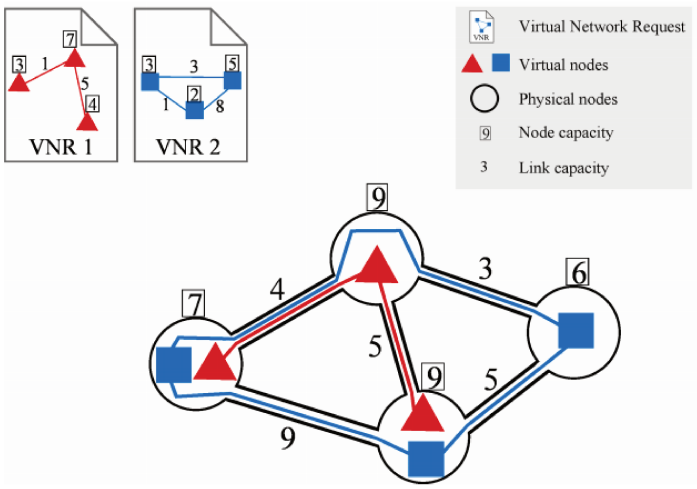


Figura 2.4: Exemplo de mapeamento de redes virtuais VNR1 e VNR2 [21]

Algoritmos de resolução do problema de VNE podem ser agregados em diferentes classes, relacionadas à forma de abordagem ao problema. Estas classes definem se o algoritmo é: centralizado ou distribuído; estático ou dinâmico; e conciso ou redundante. Logo, um algoritmo de VNE pode ser centralizado, dinâmico e redundante, ou seja, pode pertencer a qualquer classe oriunda das múltiplas combinações dos parâmetros anteriormente mencionados. A centralização de informações ou a distribuição das tarefas de mapeamento definem se um algoritmo é centralizado ou distribuído. Já outra abordagem, considerando

possíveis modificações tanto na VN quanto no SN nos mapeamentos realizados, caracteriza um algoritmo de VNE dinâmico, enquanto que mapeamentos que não consideram qualquer reconfiguração nas VNs ou no SN, o definem como estático. Em termos de suporte a resiliência, a opção de realizar o mapeamento de uma VN por múltiplos caminhos, buscando tolerância a falhas, define um algoritmo de VNE como redundante, e no caso contrário como conciso [21].

No que diz respeito a estratégias de otimização para mapeamento de redes virtuais, há na literatura trabalhos que buscam abordar o problema de VNE com soluções exatas, utilizando heurísticas e empregando meta-heurísticas [45, 60]. A primeira abordagem busca em técnicas de programação linear inteira realizar a formulação conjunta dos subproblemas de mapeamento de nós e enlaces virtuais, o que em situações práticas pode tornar este problema NP-completo. No entanto, há técnicas (ex: *branch and bound*, *branch and cut*, *branch and price*) que somente são escaláveis em tempo de resolução para pequenas instâncias deste problema. Heurísticas permitem que o tempo de execução de algoritmos de resolução do problema VNE seja diminuído em função do comprometimento da qualidade de solução encontrada. Já a utilização de meta-heurísticas procura encontrar soluções candidatas que estejam próximas a uma melhor escolha e melhorá-las com critérios de alguma característica ou medida de qualidade. Exemplos de meta-heurísticas são algoritmos genéticos e *simulated annealing*. Em [21] várias métricas relacionadas ao problema VNE são definidas e utilizadas para avaliar os trabalhos atualmente existentes na literatura. Estas métricas podem ser definidas em classes, as quais estão listadas abaixo.

- Qualidade de Serviço: abrange métricas que vão desde o comprimento dos caminhos do mapeamento e fator de utilização dos recursos do substrato de rede até fatores mais específicos, como *throughput*, atraso e latência das redes virtuais.

- Gastos de recursos: estes estão bem associados aos requisitos de InPs. Por exemplo, a taxa de aceitação de mapeamento de redes virtuais define o quão bem é para um InP aplicar um algoritmo de VNE em sua infraestrutura. Outra métrica interessante, a taxa de utilização do substrato de rede dividido pela quantidade de recursos utilizados pelas VNs, indica o quão bem estão sendo aproveitados os recursos de um InP.

- Resiliência: define as métricas associadas à sobrevivência de redes virtuais, tais como a quantidade de caminhos redundantes, o custo da resiliência inserida nos mapeamentos, a probabilidade de bloqueio de recuperação de VNs e a quantidade de migrações necessárias para a recuperação de uma VN.

- Outras: podem incluir variáveis mais específicas, como a quantidade de mensagens

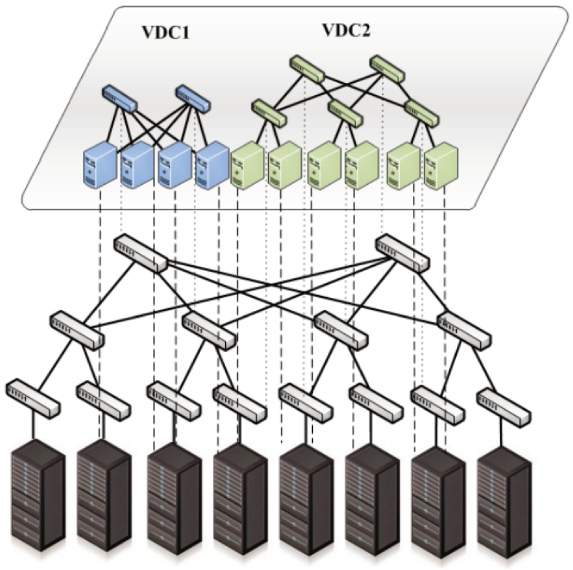


Figura 2.5: Exemplo de definição de redes virtuais VDC1 e VDC2 sobre infraestrutura de data center compartilhada [5]

de coordenação necessária para a execução de um algoritmo ou mesmo o tempo de execução de um algoritmo de VNE conforme o tamanho das VNs ou do SN. Outra métrica importante, é a quantidade de recursos ativos do substrato utilizado, o que avalia soluções que visam economias de energia.

2.2.2 Virtualização de Redes de Data Center

Um data center virtualizado possui alguns ou todo o seu hardware virtualizado (e.g. servidores, roteadores, switches e enlaces), enquanto que um data center virtual é um conjunto de recursos virtualizados (e.g. VMs, *switches* virtuais, roteadores virtuais) conectados por enlaces virtuais. Logo, um VDC é um subconjunto de recursos definidos por um data center virtualizado. Neste cenário, uma rede virtual é definida como um conjunto de recursos de redes virtualizados (e.g. switches, roteadores) conectados por enlaces virtuais. Ou seja, a virtualização de redes, similar à virtualização de servidores, permite a criação de múltiplas redes virtuais sobre um mesmo substrato de rede físico, as quais são implementadas e gerenciadas de forma independente [5]. A Figura 2.5 mostra um exemplo de duas redes virtuais de dois VDCs definidos por VDC1 e VDC2, as quais compartilham um mesmo substrato de rede.

Trabalhos recentes sobre virtualização de redes de data center levantam uma série de desafios a serem pesquisados como técnicas de virtualização, esquemas de endereçamento, isolamento de performance, escalabilidade e tolerância a falhas. Em tais propostas, a

utilização dos recursos da DCN ocorre de duas formas, por disputa (ex: Oktopus [3] e Proteus [59]) ou compartilhamento (ex: Seawall [54], Netshare [31]). No primeiro caso, ocorre alguma forma de disputa de recursos de rede por multiplexação estatística e requisitos mínimos de largura de banda a VMs, sem fornecer garantias determinísticas de recursos de rede (ex: latência, largura de banda). Já no segundo caso, ocorre a alocação de garantias mínimas de largura de banda a conjuntos de VMs, os quais são definidos de diferentes formas, tais como heurísticas e análises temporais de padrões de comunicação entre VMs. Com relação à caracterização das arquiteturas de propostas atualmente vistas em pesquisas e soluções proprietárias, elas se definem de duas maneiras, fim a fim e ou baseada na rede. A primeira propõe a virtualização de rede utilizando *switches* virtuais de servidores do data center, de modo que a comutação entre o tráfego de máquinas virtuais seja feita pela programação de tais *switches* em *hypervisors*. Já a segunda, caracteriza a virtualização de DCNs pela criação de caminhos virtuais programados nos *switches* da DCN (ex: Gatekeeper [47], SecondNet [25]) ou pela indireção de endereços através de um diretório central de mapeamento de endereços de aplicação e de localização (ex: VL2 [22], Portland [38]).

2.3 Redes Definidas por Software

Redes Definidas por Software [20] constituem uma nova abordagem para redes de comunicação que tem como atributos principais: a separação dos planos de dados e controle, uma interface entre tais planos, um plano de controle centralizado logicamente e a repartição e virtualização da rede subjacente. O controle logicamente centralizado é realizado por meio de um sistema operacional de rede, comumente denominado controlador, o qual constrói e apresenta um mapa completo da rede de forma que serviços e aplicações possam ser implementados sobre ele. A essência do paradigma de SDNs explica que primitivas básicas para distribuição de estado devam ser implementadas uma única vez por meio de diversas funções de controle (ex: roteamento, engenharia de tráfego, controle de acesso), sobre um espectro de granularidades (ex: fluxos individuais ou agregados) em uma variedade de contextos (ex: Local Area Networks - LANs, Data Centers, Wide Area Network - WAN). Tais primitivas devem utilizar técnicas bem conhecidas de propostas gerais vindas da literatura de sistemas distribuídos e não de algoritmos mais especializados, encontrados em protocolos de roteamento e outros mecanismos de controle de rede [11]. A Figura 2.6 apresenta um exemplo de arquitetura genérica de SDNs, mostrando as camadas de aplicação (*Application Layer*), controle (*Control Layer*) e de infraestrutura (*Infrastructure Layer*) com seus principais componentes, respectivamente definidos por aplicações de nuvem/negócios, controladores de rede e interface com plano de controle, e plano de dados com interface ao plano de controle (ex: OpenFlow).

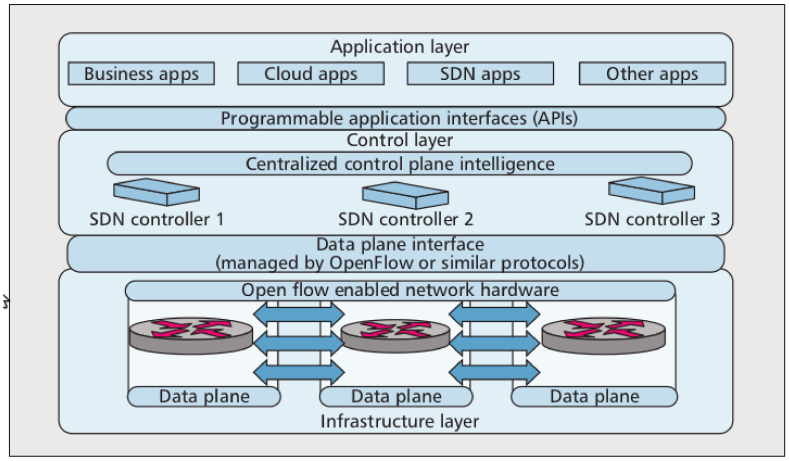


Figura 2.6: Arquitetura genérica de SDN [52]

2.3.1 OpenFlow

O padrão OpenFlow [39] surgiu da necessidade de construção de redes virtualizadas programáveis para realização de experimentos envolvendo novas propostas de protocolos de rede. Por meio de especificações bem definidas pela organização *Open Network Foundation* (ONF) ele define uma Interface de Programação de Aplicação (Application Programing Interface - API) para um controlador de rede poder interagir com elementos da rede. Estes, por meio de regras definidas por campos de cabeçalhos de pacotes (ex: endereços IP, MAC e portas TCP/UDP) são programados para realizar ações sobre fluxos de pacotes associados a tais regras, tais como encaminhamento, descarte, reescrita e encaminhamento ao controlador. Fluxos significam um conjunto de pacotes que possuem as mesmas definições em seus campos de cabeçalho e de acordo com estas associações podem possuir diferentes granularidades. O padrão OpenFlow segue o paradigma de SDNs e diferentemente da ideia tradicional de comutação de pacotes provê certos benefícios, tais como: um controlador centralizado logicamente pode supervisionar todas as decisões em nível de fluxos, evitando a necessidade de carregar configurações de políticas globais de rede em cada um de seus elementos; o controlador pode ver todos os fluxos, potencialmente permitindo o gerenciamento otimizado de forma global do tráfego de rede; *switches* com suporte a OpenFlow são relativamente simples, de baixo custo e com futuro garantido, devido à ausência de dependências de *firmware* e *hardware* [16]. A Figura 2.6 apresenta uma separação clara entre os planos de dados e controle, evidenciando o padrão OpenFlow como API *SouthBound* para comunicação entre tais planos.

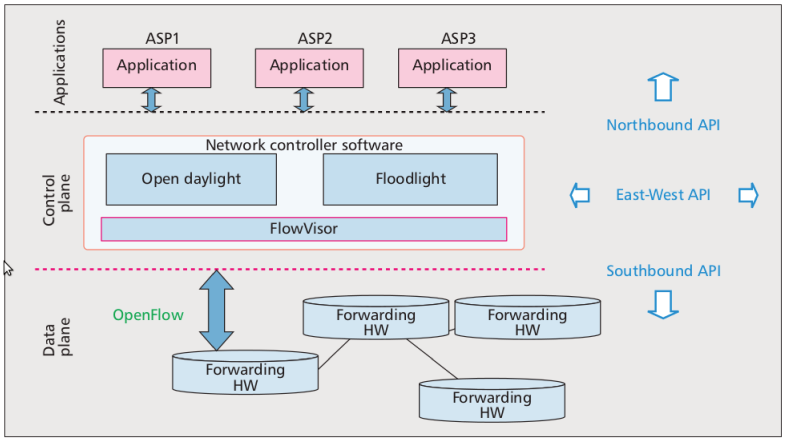


Figura 2.7: APIs de SDN com referência ao padrão OpenFlow como *Southbound* API [28]

Resultado de seguidas pesquisas almejando a separação, principalmente, dos planos de dados e de controle, tais como a proposta de arquitetura 4D em 2005 [23], o padrão OpenFlow se consolidou em 2008 como a tecnologia precursora do conceito de SDN. A partir daí, seguindo o modelo deste padrão, inúmeras propostas de controladores de rede foram surgindo em diferentes linguagens de programação, como NOX em C++, POX e Ryu em Python, Beacon, Floodligth e OpenDaylight em Java e Trema em C/Ruby. Muitas destas propostas, sendo de código aberto, propiciam um crescimento de propagação do conceito de SDN, justamente devido a sua capacidade de inovação em controlar redes de computadores e levam ao surgimento de inúmeras outras tecnologias também de código aberto e que seguem o paradigma de SDNs, tais como ferramentas de testes, simulação, comutação, roteamento e aplicações de segurança. O trabalho [34] descreve minuciosamente quais as principais tecnologias atuais que suportam o padrão OpenFlow, as diferentes características advindas da evolução de suas especificações, suas principais tecnologias habilitadoras, e os trabalhos relacionados às suas diferentes frentes de pesquisa, como segurança, tolerância a falhas e desempenho. Neste trabalho são apresentadas as principais características das especificações mais recentes do padrão OpenFlow e como se dá o funcionamento de um *switch* que segue tais especificações.

Em termos gerais, um *switch* com suporte a OpenFlow 1.3 possui um modelo de especificação conforme a Figura 2.8. Um *switch* OpenFlow possui uma ou mais tabelas de fluxos (*flow tables*) e uma tabela de grupos (*group table*), as quais realizam análises de pacotes (pesquisa de campos em seus cabeçalhos), e encaminhamento dos mesmos. Um canal OpenFlow seguro entre o *switch* e o controlador de rede permite que o primeiro seja programado pelo segundo por meio do protocolo OpenFlow. Usando este protocolo, o controlador pode adicionar atualizar ou remover entradas de fluxos (*flow entries*) em *flow tables*, tanto reativamente quanto proativamente. Cada *flow table* pode conter *flow*

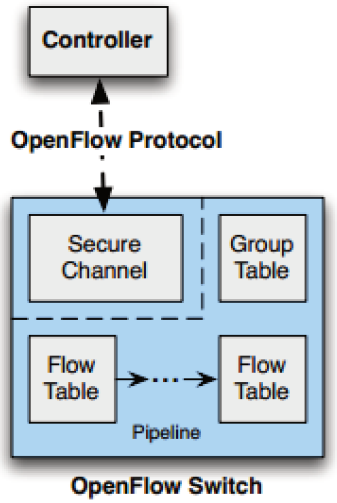


Figura 2.8: Principais componentes de *switch* OpenFlow [39]

entries as quais consistem de entradas de campos de cabeçalhos de pacotes (*match fields*), contadores (*counters*) e um conjunto de instruções para serem aplicadas aos pacotes que tiverem *match* com os *match fields*.

Como visto na Figura 2.9, a operação de *matching* segue um *pipeline* e começa na primeira *flow table* e pode continuar nas *flow tables* seguintes. *Flow entries* podem ter prioridades, uma vez que a primeira de uma *flow table* corresponda a um pacote, as instruções associadas a ela são executadas. Caso não haja nenhuma *flow entry* para um pacote, uma entrada padrão para esta operação pode ser ou não programada no *switch* o que pode fazer com que o pacote seja enviado ao controlador, descartado ou encaminhado para a próxima *flow table*.

Instruções relacionadas a uma *flow entry* podem conter ações e regras de modificações no *pipeline* de processamento de pacotes. Ações definidas em instruções podem realizar encaminhamento e/ou modificação de pacotes ou o processamento em *group tables*. Uma *group table* contém entradas especificando uma lista de conjunto de ações definindo *action buckets* com semânticas específicas dependendo do tipo do grupo. Ações associadas a um ou mais *action buckets* são aplicadas aos pacotes direcionados a entradas de grupos da *group table*. A *Meter Table* de um *switch* define entradas que podem ser relacionadas a *flow entries* permitindo operações simples de qualidade de serviço como limitação de largura de banda, as quais podem ser implementadas em combinação com operações de filas definindo assim operações mais complexas, como serviços diferenciados (ex: *DiffServ*).

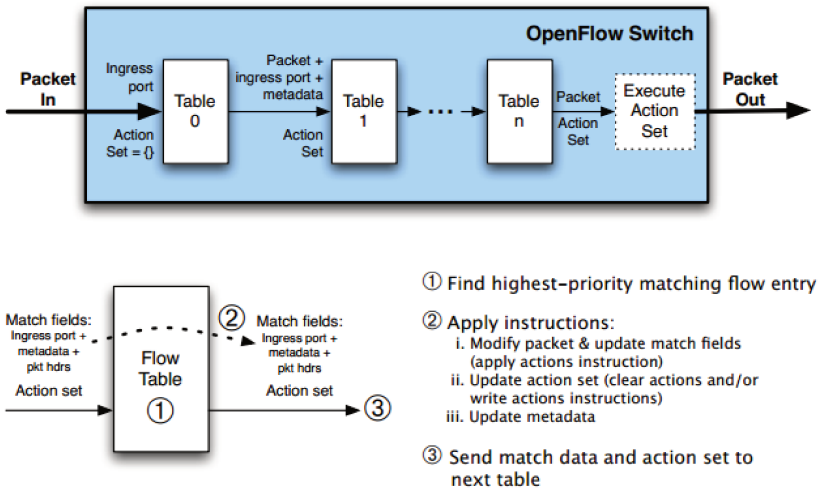


Figura 2.9: Modelo de processamento de pacotes por *switch* OpenFlow [39]

2.3.2 Ryu

Ryu[1] é um sistema operacional de rede escrito na linguagem de programação Python que está sendo desenvolvido rapidamente com técnicas ágeis de programação e com o suporte da empresa de telecomunicação NTT. Abordado em uma seção Developer Track no evento *Open Network Summit* em 2013, Ryu é um arcabouço (*framework*) baseado em componentes que possui uma API simples o qual permite usuários criarem aplicações de controle e gerenciamento de redes. Este sistema tem suporte a vários protocolos de gerenciamento de redes, como OpenFlow (versões 1.0, 1.2, 1.3 e extensões Nicira), Netconf, OF-config, etc. O foco de desenvolvimento do Ryu é criar um sistema operacional para SDNs que tenha uma alta qualidade, suficiente para ser usado em grandes ambientes de produção.

Em termos gerais, um controlador de rede com suporte a OpenFlow opera seguindo os seguintes moldes, possui em sua arquitetura um núcleo responsável por realizar e gerenciar a comunicação entre *switches* (*datapaths*) e o controlador, além de responsabilidades como tratamento de eventos da rede, fornecimento de uma API de alto nível para desenvolvimento de aplicações e gerenciamento de aplicações de rede sobre o controlador. Dentre as aplicações necessárias ao funcionamento de um controlador, está a descoberta de topologia de rede, a qual busca, em termos gerais, enviar pacotes de protocolos de descoberta de redes (ex: Link Layer Discovery Protocol - LLDP) aos *switches* associados ao controlador a fim de descobrir os enlaces da rede e, portanto, a topologia da rede. Outras aplicações já desenvolvidas sobre controladores são: simples comutação de pacotes em

[1]https://github.com/osrg/ryu

camada 2, *firewall*, tunelamento com *Generic Routing Encapsulation* (GRE) e atribuição dinâmica de VLANs.

2.3.3 Mininet

Mininet [27] é um emulador baseado em contêineres que permite a reprodução de experimentos de rede empregando mecanismos de isolamento, aprovisionamento e monitoramento de recursos utilizando o padrão OpenFlow. A emulação baseada em contêineres permite executar código real em uma rede emulada utilizando técnicas de virtualização em nível de sistema operacional com os devidos cuidados de isolamento e monitoramento de recursos. Mininet, atualmente na versão 2.0, tem como origem a proposta Mininet-HiFi, a qual tem como principais características:

- Realismo funcional: parte do princípio que o sistema deve ter a mesma funcionalidade que um hardware real implementado em um ambiente real, e logo, executar o mesmo código deste ambiente;

- Realismo de temporização: o comportamento de temporização do sistema deve ser indistinguível ou muito perto do hardware real e o sistema deve detectar quando esta característica é violada;

- Realismo de tráfego: o sistema deve ser capaz de gerar e receber tráfego de rede real da Internet, de usuários e sistemas e da rede local;

- Flexibilidade de topologia: deve ser fácil realizar a criação e execução de qualquer experimento com qualquer topologia de rede;

- Fácil replicação: deve ser fácil e simples duplicar o ambiente de um experimento assim como a sua execução; Baixo custo: não deve haver custo na reprodução da execução de um experimento.

Com base nessas características, utilizando a plataforma Mininet se torna fácil realizar a criação, customização e compartilhamento de SDNs. Atualmente, fazendo uso de Open vSwitches para a definição de *datapaths*, Mininet consegue emular de centenas a milhares de nós em uma topologia, além de realizar diferentes experimentos de rede com tráfego e comportamento reais. Muitos experimentos já foram executados na Mininet para a reprodução de trabalhos consolidados em redes de computadores, tais como Hedera, Data Center TCP (DCTCP), MultiPath TCP (MPTCP) e Jellyfish [27].

2.3.4 RouteFlow

RouteFlow [37] é uma plataforma que segue o paradigma de SDNs ao centralizar logicamente o controle da rede, unificar o estado de informação e desacoplar as lógicas de encaminhamento e configuração de equipamentos de rede. Ela busca reproduzir em um plano virtual de controle um mapeamento dos recursos de uma infraestrutura física de rede com suporte ao padrão OpenFlow e provê a característica de plataforma como serviço à rede permitindo o mapeamento flexível de recursos de uma topologia virtual sobre uma infraestrutura física de rede a qual pode ser distribuída e compartilhada. Constituída por três planos, virtual, físico e de controle, a plataforma RouteFlow possui respectivamente em cada um destes planos os seguintes componentes principais: rfclient, rfproxy e rfserver (vide Figura 2.10).

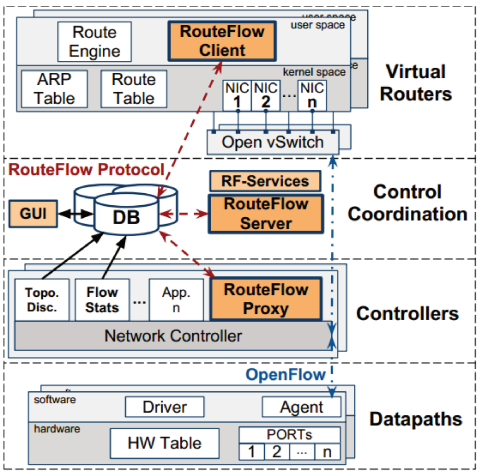


Figura 2.10: Arquitetura da plataforma RouteFlow [50]

O plano virtual é constituído de roteadores virtuais interconectados por um *switch* virtual programável com suporte a OpenFlow. Estes são executados em nível de virtualização de sistema operacional em LinuX Containers (LXCs), os quais executam cada um duas aplicações, rfclient e uma suíte de roteamento de domínio público (e.g. Quagga). A aplicação rfclient realiza a captura de rotas computadas pela suíte de roteamento e as envia ao plano de controle. O plano físico é formado por uma topologia de rede contendo *switches* com suporte a OpenFlow. Estes se comunicam com um controlador de rede o qual tem sobre ele executada a aplicação rfproxy, responsável por realizar a interface entre o plano de controle e o controlador de rede. No plano de controle está a aplicação rfserver e o banco de dados MongoDB. Este armazena todo o estado de configuração do mapeamento entre topologias física e virtual a partir da inicialização de um arquivo de configuração. O gerenciamento deste mapeamento bem como o controle de todas as men-

sagens entre os planos físico e virtual são realizados pela aplicação rfserver. Esta, como componente central da plataforma, gerencia o estado de todo o mapeamento físico/virtual contido no banco de dados, além de realizar a troca e formatação de mensagens entre as aplicações rfclient e rfproxy, tais como rotas computadas em LXCs no plano virtual.

2.3.5 Relação e Desafios de SDN e Virtualização de DCNs

Virtualização de redes não requer SDN e vice-versa, e de forma simples, a relação entre esses dois conceitos pode ser definida de três formas [20]:

- SDN como tecnologia habilitadora para virtualização: Como no caso da computação em nuvem, onde múltiplas *tenants* compartilham uma mesma infraestrutura de rede, a possibilidade de programação de *switches* da rede ou virtuais (ex: Open VSwitch), permite a criação de redes sobrepostas por meio de técnicas de encapsulamento (ex: VXLAN);

- Virtualização de redes para teste e avaliação de SDNs: O desacoplamento entre os planos de controle e dados permite que tecnologias de SDN sejam testadas por tecnologias de virtualização (ex: Mininet) antes de serem implementadas em ambientes reais;

- Virtualização de SDN: A repartição de fluxos de rede em diferentes espaços de regras de encaminhamento permite a virtualização de equipamentos de rede, sem a necessidade de instanciação de roteadores/*switches* virtuais, ou mesmo a repartição de domínios administrativos de rede como no caso da solução FlowVisor [53].

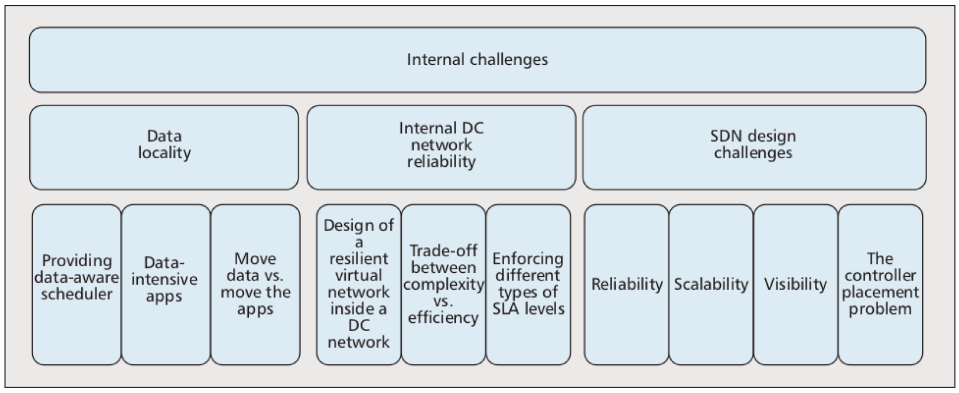


Figura 2.11: Desafios relacionados a SDN e virtualização de redes de data center [52]

No caso da virtualização de redes associada a computação em nuvem pela utilização do paradigma de SDN, os principais desafios, mostrados na Figura 2.11, são definidos segundo [52] em três blocos:

- Desafios de design de SDN: Englobam este bloco confiabilidade, escalabilidade, visibilidade e o problema de localização do(s) controlador(es) de rede;
- Confiabilidade da DCN: Nesse caso, como definidos nos desafios de DCNs, estão associados a este tópico, o design de redes virtuais confiáveis, *trade-offs* entre eficiência e complexidade, e implantação de diferentes níveis de contratos de acordos de serviços (Service Level Agreements - SLAs);
- Localidade dos dados: aprovisionamento de escalonadores cientes das condições de tráfego, requisitos de latência de aplicações de alto desempenho, e o *trade-off* entre mobilidade de dados e/ou de aplicações.

2.4 Rede como Serviço

Apesar de progressos significativos ocorrerem em virtualização de redes ainda existem muitos desafios que possam permitir este conceito ser amplamente utilizado no contexto de Internet do Futuro. Em um amplo cenário, considerando a existência de múltiplos InPs, SPs necessitam descobrir quais os recursos de redes disponibilizados em cada InP. E assim, os recursos apropriados para atender a serviços de usuários finais devem ser selecionados por SPs de forma a constituírem redes virtuais. Logo, uma interação flexível e efetiva entre InPs, SPs e usuários finais existindo de forma colaborativa pode permitir que redes virtuais sejam alocadas pela integração de tecnologias e sistemas heterogêneos [18].

Em uma situação hipotética, os recursos definidos por uma infraestrutura de rede podem ser encapsulados em serviços. SPs podem acessar tais serviços por meio do paradigma de IaaS e assim compor serviços de rede fim a fim. Aplicações de usuários finais, podem utilizar toda a plataforma de rede subjacente disponibilizada por InPs acessando os serviços de rede criados por SPs. Esta constituição de redes virtuais sobre a infraestrutura de InPs oferecidas por meio de serviços de rede de SPs é o que caracteriza a essência do paradigma de Network-as-a-Service [15]. A virtualização de redes orientada a serviço provê formas de abstrair os recursos de rede a aplicações. Por exemplo, expor a heterogeneidade de protocolos, equipamentos e tecnologias de rede a aplicações sem a virtualização de redes tornaria muito complexo o gerenciamento dos recursos disponíveis na rede.

O fornecimento de serviços como endereçamento específico, roteamento, reserva de largura de banda, isolamento de rede, por meio de uma plataforma de NaaS, faz com

que ela necessite operar com alguns requisitos mínimos, como uma visibilidade ampla da topologia, a existência de dispositivos de rede que possam realizar o encaminhamento de pacotes de maneira customizada e a capacidade de processamento de pacotes dentro da rede. Em redes definidas por software esses requisitos podem ser preenchidos, quando utilizada como mecanismo para virtualização de redes. A aplicação do conceito de NaaS a SDNs está em progresso e atualmente segue os moldes do modelo computacional de IaaS aplicado a computação em nuvem [7].

Capítulo 3

Arquitetura Proposta

Neste capítulo procuramos descrever a arquitetura de rede de data center proposta neste trabalho. Inicialmente buscamos moldar quais são os seus princípios básicos, como a busca de separação dos planos de controle e de dados, além do estabelecimento de um plano de conhecimento da rede elaborado para dar ao plano de controle a visibilidade necessária à tarefa de mapeamento de redes virtuais. Buscamos também enfatizar a necessidade de abstrações de regras de encaminhamento no plano de dados, as quais possam traduzir de forma coerente as necessidades de estabelecimento de rotas na infraestrutura de rede do data center para as redes virtuais a serem mapeadas. E no âmbito de fornecermos um modelo eficiente de roteamento para redes de data center, nos baseamos no estabelecimento de um plano virtual especializado para realizar esta tarefa com base em propostas já existentes e consolidadas na literatura.

Em seguida nos dedicamos a traduzir os conceitos propostos para a arquitetura construída neste trabalho. Logo, para isso, enfatizamos as razões das escolhas das tecnologias utilizadas neste trabalho, tendo como base as definições de cada uma delas bem como suas utilizações em soluções já propostas na literatura de DCNs e SDNs. Em seguida, descrevemos as mudanças realizadas nas tecnologias utilizadas, como a elaboração de um plano de controle especializado em mapeamento de redes virtuais construído sobre a plataforma RouteFlow. Descrevemos nessa etapa os requisitos buscados na elaboração de cada uma das modificações realizadas e como elas contribuem para o desenvolvimento deste trabalho como um todo.

Com base nos requisitos de mapeamento de redes virtuais, nos dedicamos, ao final deste capítulo, a mostrar e descrever os algoritmos criados para prover suporte e realizar o mapeamento de redes virtuais em DCNs. Enfatizamos nesta etapa a importância das informações fornecidas pelo plano de controle da arquitetura construída neste trabalho, e o modo como cada parte desta arquitetura contribui para a alocação e instanciação eficiente de redes virtuais em DCNs. Por fim explicamos as principais etapas de funcionamento

dos algoritmos criados, com exemplos de sua execução, e os motivos das escolhas das estruturas de dados criadas e utilizadas por eles na tarefa de alocação de redes virtuais.

3.1 Modelo de Arquitetura

Buscando os requisitos de operação de DCNs, como escalabilidade, flexibilidade e desempenho, tratamos nesta seção de descrever os princípios da arquitetura criada neste trabalho. Como requisito base, buscamos propor uma arquitetura que fornecesse as informações necessárias para permitir ao plano de controle construir abstrações de rotas e de topologia de rede contendo recursos que possam ser úteis ao aprovisionamento de redes virtuais contendo requisitos de largura de banda e resiliência.

3.1.1 Abstrações, Requisitos e Desafios

No sentido de demonstrar como a arquitetura proposta neste trabalho se compatibiliza à tarefa de alocação de redes virtuais, levantamos nesta seção as abstrações requeridas para que o plano de controle consiga proporcionar requisitos de largura de banda e resiliência às redes virtuais. Estas abstrações ao mesmo tempo que constituem os alicerces da arquitetura, também definem desafios relacionados a constituição e configuração de redes virtuais em DCNs os quais estão associados aos conceitos de SDN utilizados neste trabalho.

Em termos das abstrações consideramos que elas definam os seguintes itens abaixo.

- Configuração de encaminhamento: consideramos que existam no plano de dados elementos de rede capazes de suprimir os requisitos de alto nível de configuração de redes virtuais necessários ao plano de controle. Dessa forma, tomamos como referência conceitos de SDNs, onde a separação dos planos de dados e de controle se faz presente, concedendo uma visão logicamente centralizada de toda a rede. Isto permite que equipamentos de rede possam ser programados conforme comandos que satisfaçam, por exemplo, limitadores de largura de banda ou encaminhamento por múltiplas portas. Em termos simples, consideramos que uma topologia de rede operando sob condições e requisitos de SDNs exista, esteja sob operação de controlador(es) de rede, e que ele(s) tenha(m) a capacidade de comunicação com o plano de controle por meio de aplicações que traduzam comandos de alto nível para comandos de configuração para equipamentos da infraestrutura de rede.

- Definição de rotas/endereços: propomos a dissolução da associação entre rotas e endereços. Ou seja, que estas estejam definidas por algum algoritmo/mecanismo, o qual abstraia a topologia de rede do plano de dados e que, de forma eficiente,

consiga manter atualizado o estado das rotas da topologia de rede conforme necessário a requisitos de consistência de rotas no plano de controle. Logo, estes mecanismos/algoritmos devem ter uma associação flexível a considerar a de existência de equipamentos de rede e sistemas finais, podendo computar rotas conforme requerido pelo plano de controle. Essa permissividade de rotas e endereços torna a arquitetura livre de qualquer problema de escalabilidade de quaisquer esquemas de endereçamento.

- Visão ampla de rede, objetivos de alto nível e controle direto da rede: De maneira simples, objetivos como requisitos de largura de banda e/ou resiliência devem ser concedidos por meio de definições de políticas de rede, as quais levam à execução de algoritmos de construção de topologias virtuais na forma de grafos que representam tanto recursos de rotas quanto da topologia física de rede. Por meio deles, faz-se necessária a tradução das informações que eles representam para configurações da infraestrutura de rede do plano de dados através de uma interface de controle direto sobre a rede, a qual existe como forma de contato com controlador(es) de rede definido(s) para tradução destas mensagens em comandos que se ajustem às configurações necessárias de serem feitas em equipamentos de rede. Além disso a presença de abstrações para os recursos, rotas e grafos de topologias (físicas e virtuais), são necessários para uma visão ampla de rede, de modo que estes estejam consistentes com o estado de toda a arquitetura.

Com base na elaboração dos itens acima, vemos que as abstrações definidas conduzem a existência natural de requisitos para que de fato existam em uma arquitetura de DCNs. Logo, abaixo definimos os requisitos gerados por estas abstrações e como cada um deles produz desafios que serão levados em consideração na prototipação da arquitetura construída neste trabalho.

- Plano de dados com suporte a SDNs: Este é sem dúvida o primeiro requisito que é levantado quando uma abstração de encaminhamento por configurações e requisitos de alto nível se fazem necessárias. Logo, a utilização de alguma interface *southbound* para se comunicar com equipamentos de rede com suporte a alguma tecnologia de SDN impõe o principal requisito das abstrações de configuração de encaminhamento. Este requisito levanta consigo os seguintes desafios: design de uma interface de comunicação com equipamentos de rede sobre controlador(es) de rede a qual não só mantenha consistência sobre o estado da infraestrutura física de rede para o plano de controle, como também possa fornecer a interface de tradução de mensagens de controle em configurações de equipamentos de rede.

- Plano de controle logicamente centralizado: A visão ampla de rede, e consequentemente o suporte a SDNs, resulta em um plano de controle centralizado logicamente, o qual estabelece neste requisito seus maiores desafios: escalabilidade para DCNs devido a grande quantidade de equipamentos de rede; flexibilidade a fim de que exista a permissividade de implantação de configurações dinamicamente sobre a infraestrutura de rede; o desempenho deve suprir as demandas de operação da infraestrutura de rede conforme a demanda de recursos físicos e virtuais; tolerância a falhas é um ponto chave, visto que a centralização em si traz a característica de tornar o plano de controle um ponto único de falhas, logo, há a necessidade de sua separação lógica; definição de estruturas lógicas que interpretem consistentemente não só os elementos dos planos de dados como também seus recursos, de modo que estes possam ser traduzidos possivelmente a outras estruturas que representem redes virtuais e seus atributos.

- Dimensionamento de plano de conhecimento de rotas: para que ocorra a separação e não dependência de esquemas de endereçamento e topologia física de rede, estes dois componentes de uma DCN devem fazer correspondência por algum mecanismo que não limite a flexibilidade da rede, e logo, ofereça a ela a possibilidade de implantação de qualquer algoritmo de computação de rotas e tradução/resolução de endereços de rede. Isto traz a possibilidade de inovações de rede, e não dependência de aplicações às características de endereçamento da rede, no entanto traz também como desafios, a construção de uma identificação que seja suficiente para representar a topologia de rede e como a comunicação entre os diferentes sistemas finais que nela se localizam possam se comunicar, seja em domínio das camadas de enlace ou roteamento. E essa característica também deve suprimir a dependência de equipamentos de rede executarem protocolos de roteamento comumente conhecidos, deixando esta tarefa para ser construída em algum plano de conhecimento de rotas sobre o plano de controle da arquitetura.

3.2 Proposta de Arquitetura

Da maneira como as abstrações foram elaboradas e os desafios estabelecidos para os requisitos necessários à definição da arquitetura (vide Figura 3.1), chegamos a conclusão de construí-la com três planos: virtual, de controle e de dados. Em cada um deles definimos as principais funções e interfaces entre si de modo que as necessidades de provisão de redes virtuais sejam satisfeitas com base nos requisitos levantados anteriormente. Abaixo descrevemos as principais características de cada um dos planos elaborados, como eles interagem entre si, e as tecnologias utilizadas em cada um deles assim como o porquê da

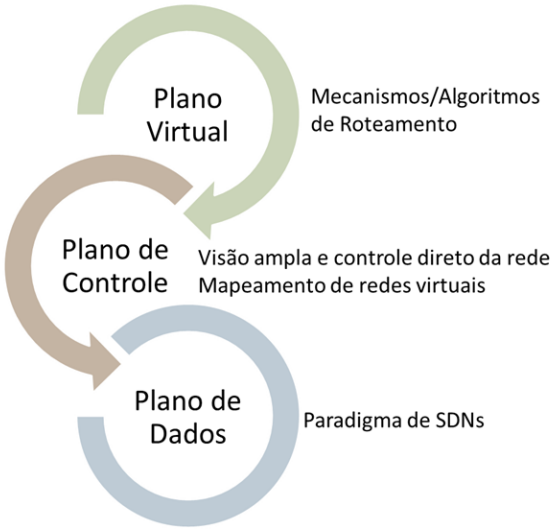


Figura 3.1: Arquitetura Proposta

utilização de cada uma delas.

Plano de Dados: Neste plano definimos a existência de equipamentos de rede com suporte ao padrão OpenFlow 1.3. A razão desta escolha está associada às possíveis configurações de encaminhamento que este padrão possui, como restrições de largura de banda por *meter tables* e encaminhamento de pacotes por múltiplas interfaces utilizando *group tables*. Estas duas características, por exemplo, podem respectivamente abstrair as necessidades de limitação e reserva de largura de banda por redes virtuais e encaminhar tráfego em técnicas de espalhamento de pacotes semelhantes a Equal-Cost MultiPath (ECMP). Os equipamentos de rede tem como tarefa principal estarem associados a um ou mais controlador(es) de rede sendo passíveis de programação e configuração de quaisquer requisitos necessários à constituição de redes virtuais na infraestrutura física de rede. Além disso, estabelecemos uma topologia de rede folded-Clos para os equipamentos de rede. O motivo desta escolha se deve à existência de múltiplos caminhos entre sistemas finais, escalabilidade modular pela adição de mais estágios ou densidade de portas em elementos de rede, e largura de banda agregada não bloqueante, ou seja, caso não haja *oversubscription* na rede, quaisquer dois sistemas finais podem utilizar toda a largura de banda de suas interfaces para se comunicarem.

Plano Virtual: Inicialmente justificamos a existência deste plano por causa da desagregação de funções de equipamentos do plano de dados, visto que eles serão *switches* programáveis com suporte a OpenFlow. Além disso, desvencilhamos a função de cálculo de rotas e do plano de controle pois, como ponto central de falhas, é necessário apenas que rotas sejam calculadas e enviadas a ele ou ao plano de dados, definindo as duas in-

terfaces deste plano. Consideramos para isso que existam elementos neste plano os quais conheçam a topologia física de rede e de alguma forma representem-na por alguma forma de mapeamento que possibilite a definição de rotas necessárias a toda a arquitetura. E como condição necessária, que este mapeamento seja flexível, a ponto de não estar associado a configurações de endereços de rede ou protocolos/mecanismos de roteamento. Esta configuração então existe na forma de uma topologia virtual, onde elementos virtuais de rede conectados por um *switch* virtual façam a correspondência à topologia do plano de dados e possam consistentemente prover rotas a equipamentos de rede do plano de dados, seja diretamente ou através do plano de controle.

Plano de Controle: Como ponto central da arquitetura, este plano delineia as principais abstrações que são necessárias ao aprovisionamento de redes virtuais de data center e, logo, define como esta sua principal tarefa. Ao definirmos uma interface de controle sobre equipamentos do plano de dados através de uma aplicação executada sobre controlador(es) de rede, conseguimos assim programar as redes virtuais para que de fato elas encaminhem tráfego. Além disso, esta mesma aplicação fornece ao controle da rede as informações de recursos disponíveis no plano de dados, como elementos de rede existentes, topologia, largura de banda de enlaces, e estatísticas de tráfego na rede. Isto se justifica pois, a visão logicamente centralizada de controlador(es) de rede possibilita uma visão ampla de rede necessária à tarefa de provisão de redes virtuais. Por outro lado, com o plano virtual este plano estabelece a relação de estabelecimento de comunicação e programação do *switch* virtual que representa a interligação dos equipamentos virtuais de rede e consequentemente a representação virtual do plano de dados. Por este aspecto, cabe ao plano de controle determinar esta topologia virtual por meio da programação deste *switch* virtual conforme o estado da topologia do plano de dados, e ao mesmo tempo receber as rotas virtuais calculadas, utilizando-as para a programação de equipamentos de rede no plano de dados. Assim, consideramos que estruturas genéricas correspondam às topologias base, física e virtual, representando respectivamente os planos de dados e virtual, para armazenar as informações que estes planos enviam ao plano de controle, tais como rotas e topologia de rede. Além disso, estabelecemos que exista neste plano algoritmos que, com base nas informações de toda a arquitetura, realizem a alocação de redes virtuais de forma eficiente, determinando assim o controle direto na rede com base na definição de políticas que representem objetivos de alto nível, tais como largura de banda, resiliência por uso de múltiplos caminhos e esquemas de endereçamento.

3.3 Definição da Arquitetura

A partir dos objetivos definidos na proposta da arquitetura, nesta seção descrevemos a arquitetura construída neste trabalho. Logo, identificamos as tecnologias utilizadas, as

razões de suas escolhas, assim como a associação entre os componentes desenvolvidos para cada um dos planos (dados, controle, virtual) e suas características de operação.

3.3.1 Modificações na Plataforma RouteFlow

Com base na necessidade de uma arquitetura que desse suporte a criação e instanciação de redes virtuais em DCNs, vimos no controle de rede logicamente centralizado da plataforma RouteFlow as características que permitem o controle direto da rede por meio da unificação do estado das informação dos planos virtual e de dados, além do desacoplamento das lógicas de encaminhamento e configuração de equipamentos de rede. Logo, escolhemos esta plataforma principalmente pelo fato de que ela se encaixa perfeitamente nos objetivos e planos definidos pela arquitetura proposta. Ou seja, possui planos virtual, de dados, e de controle, os quais permitem abstrações de cálculo de rotas, proporcionam controle direto sobre a rede seguindo o paradigma de SDNs, e centralizam logicamente o controle com uma visão ampla da rede onde objetivos de alto nível podem ser facilmente implementados.

Utilizamos a plataforma RouteFlow para propor uma arquitetura para DCNs (vide Figura 3.2) com topologia virtual definida por roteamento com protocolo BGP, plano de dados com suporte a OpenFlow 1.3, e plano de controle com total gerenciamento sobre as topologia física e virtual do data center para prover suporte ao algoritmo de construção e alocação de redes virtuais proposto neste trabalho. A arquitetura criada permite que *group tables* e *meter tables* sejam definidas pelo plano de controle para serem programadas por controlador(es) de rede no plano de dados, de modo que fique transparente para ambos os planos, físico e virtual, quais os detalhes de implementação das funcionalidades de cada um. Nos tópicos seguintes, descrevemos as modificações na plataforma RouteFlow feitas para sua adequação aos requisitos de mapeamento de elementos da infraestrutura física de um InP em uma topologia virtual para operação de uma DCN. Estas descrições estão separadas de acordo com os planos da arquitetura nas seguintes subseções: plano físico, plano virtual e plano de controle.

Plano Físico

Neste plano utilizamos uma topologia de rede folded-Clos contendo *switches* com suporte a OpenFlow 1.3. Em um controlador de rede com suporte a esta mesma versão de OpenFlow construímos a aplicação rfproxy. Esta, auxiliada pela aplicação de descoberta de topologia, captura eventos de adição e remoção de *switches* e enlaces na topologia física e os encaminha ao plano de controle para a definição da topologia física disponível aos mapeamentos. Funcionalidades do padrão OpenFlow 1.3 são utilizadas para dar suporte aos mapeamentos de redes virtuais, tais como a utilização de *group tables* para encami-

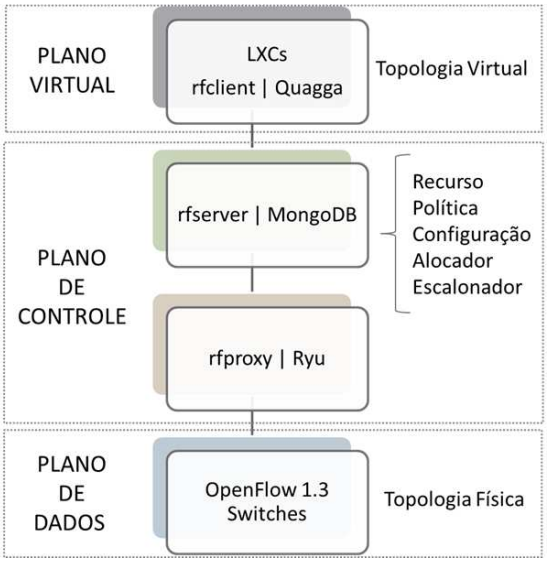


Figura 3.2: Arquitetura Construída

nhamento de dados semelhante ao mecanismo ECMP, e reservas de largura de banda definidas por *meter tables*.

A programação de rotas de topologias virtuais de SPs na topologia física ocorre pela identificação única de uma rede de um SP por *tags* MPLS. Em switches Core e EoR, o tráfego é encaminhado somente por *matches* destas *tags*. Já em ToRs, quatro tabelas são programadas. A tabela 0 trata *matches* em *tags* MPLS com tráfego destinado ao interior do *rack* e realiza a ação de retirada desta camada MPLS e o encaminhamento para a tabela 1. Nesta, o *match* ocorre por endereços de rede, e tem como ações a reescrita de endereços MAC nos pacotes e o encaminhamento para a próxima tabela. Na tabela 2, o *match* com os enredeços MAC de servidores gera a ação de encaminhamento de pacotes para as devidas portas em que eles estão conectados. Por fim, a tabela 3 trata o tráfego a ser encaminhado para fora do *rack*. Logo, esta tem como ação a marcação da tag MPLS relativa a rota que foi definida a um determinado SP no plano de controle. Todas as rotas de uma política são programadas em switches com um tempo de *hard timeout* o qual define o tempo de permanência de uma topologia virtual no plano de dados. Este tempo é configurado pelo plano de controle, e nos switches do plano de dados, quando expirado, define a exclusão das rotas por ele definidas. Regras padrão de encaminhamento as tabelas seguintes são programadas com menor prioridade em todas as tabelas descritas anteriormente, para que caso um pacote não tenha *match* na tabela 0 ele possivelmente possa ter *match* nas tabelas seguintes. Caso não ocorra nenhum *match* nas tabelas descritas, então o pacote é encaminhado a última tabela, a qual corresponde a regras de tráfego de controle, as quais determinam o encaminhamento de pacotes ao controlador e em seguida ao plano

virtual. Limitadores de largura de banda são programados em ToRs delimitando em *racks* o tráfego de entrada e saída por *meter tables* associadas respectivamente às regras das tabelas 2 e 3.

Plano Virtual

Encontramos na proposta [33] as características de configuração do protocolo BGP para o roteamento intra-AS em DCNs que foram ao encontro das funcionalidades da plataforma RouteFlow. Essa proposta define o uso do protocolo BGP com suporte a múltiplos caminhos sobre uma topologia de rede folded-Clos tendo cada um de seus roteadores configurados com base em um número de sistema autônomo (Autonomous System Number - ASN) e posicionamento dos *switches* na rede. Dentre as vantagens desta abordagem podemos citar: design prático de roteamento para grandes data centers; protocolo de simples implementação com baixa complexidade de código e fácil suporte operacional; minimização do domínio de falha de equipamentos e protocolo de roteamento; diminuição de custos operacional e de capital; e escalabilidade para *scale-out* da topologia de rede.

Neste plano a aplicação rfclient foi melhorada com a característica de leitura de rotas *multipath* na tabela de encaminhamento de LXCs. A suíte de roteamento Quagga[1] foi utilizada para realizar o roteamento entre os LXCs do plano virtual por meio do protocolo BGP. Este foi configurado seguindo as vantagens de sua utilização em uma topologia folded-Clos [33]. Na topologia virtual agregamos elementos que continham o mesmo ASN em uma camada da topologia física folded-Clos contribuindo para uma redução de criação de LXCs. Isto foi possível, pois considerando que elementos de uma mesma camada da topologia folded-Clos não estão interconectados entre si, possuem rotas equidistantes a quaisquer outros elementos da topologia, e logo computam rotas com mesmo AS-PATH para todos os destinos. Por exemplo, conforme a Figura 3.3, vimos que a proposta [33] determinada um mesmo ASN para todos os roteadores Core, logo, agregamos todos eles em um único roteador virtual com suas respectivas portas mapeadas a cada um dos diferentes *switches* Core da topologia física. Da mesma forma, *switches* EoR, que possuem um mesmo ASN, também foram agregados em um mesmo roteador virtual. E por fim, como um único ASN é definido para cada *switch* ToR, eles não puderam ser agregados.

Seguindo os moldes dessa proposta, realizamos o mapeamento das topologias base, física e virtual, as quais representam os planos de dados e virtual da seguinte forma: agregamos todos os *switches* Core em um único roteador virtual; os *switches* EoR que se interconectam com um mesmo conjunto de *switches* ToR foram agregados em um mesmo roteador virtual EoR; e como definido na proposta [33], *switches* ToR não são agregados, pois cada um possui um único ASN. Além disso, neste plano definimos que inicialmente

[1] http://git.savannah.gnu.org/cgit/quagga.git

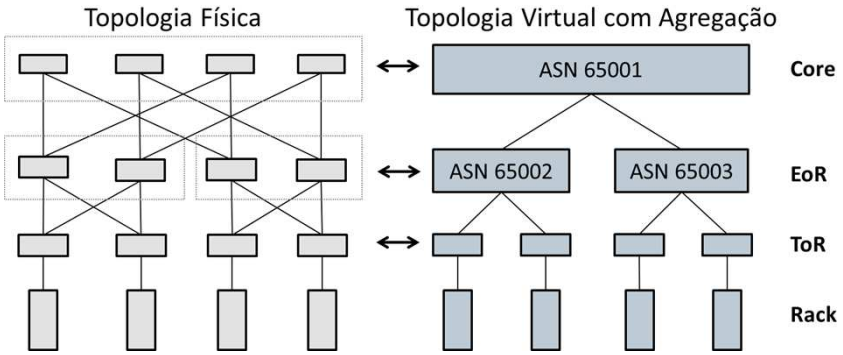


Figura 3.3: Mapeamento de topologias física e virtual com agregação

todas as mensagens de controle são transferidas entre os planos virtual e de dados por meio do controlador de rede. Uma vez mapeadas as topologias base, física e virtual, rotas são definidas no *switch* do plano virtual para que todas as mensagens de controle permaneçam na topologia virtual, e portanto, não sobrecarreguem o controlador de rede.

Plano de Controle

Neste plano é executada a aplicação rfserver bem como o banco de dados o qual mantem o estado do mapeamento entre os planos físico e virtual. Nenhuma modificação foi feita com relação a forma de construção do mapeamento estático das topologias física e virtual ou de sua manutenção. No entanto, foram desenvolvidas novas funcionalidades sobre o plano de controle as quais foram incorporadas a aplicação rfserver. Estas funcionalidades se dividem em cinco componentes (vide Figura 3.4): **Recurso**; **Politica**; **Configuração**; **Alocador**; e **Escalonador**.

- **Recurso** realiza o armazenamento e gerenciamento de informações dos planos virtual e de dados. Para isso faz a criação de classes as quais representam os recursos de cada plano. As classes *TopologiaFísica* e *TopologiaVirtual* tem como atributos componentes que representam respectivamente os planos físico (ex: switches, enlaces) e virtual (ex: LXCs, interfaces, rotas). Estas duas classes são herdeiras da classe *Topologia* a qual constrói grafos utilizados para representá-las. Neste componente também define-se a classe *Topologias* a qual possui funções de instanciação de topologias físicas e virtuais, gerenciamento do mapeamento entre elas, preenchimento de suas configurações tais como rotas, enlaces e *switches* ou LXCs, além de funções de análise das configurações e estados das topologias física e virtuais. Ou seja, com base em modificações na aplicação rfserver todas as trocas de mensagens entre as topologias base, física e virtual, as quais representam os respectivos planos

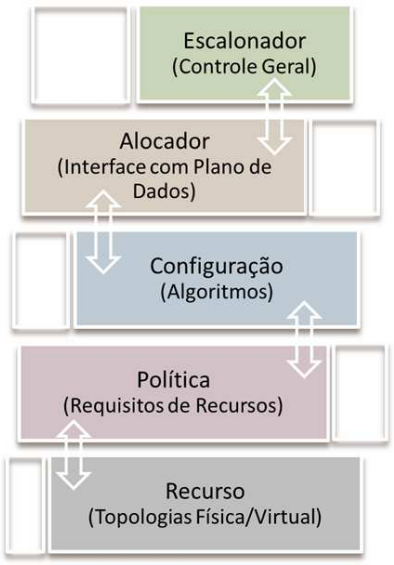


Figura 3.4: Componentes do Plano de Controle

de dados e virtual, tem suas informações armazenadas em classes que as representam, construídas a partir do componente Recurso. A definição de topologias de data centers virtuais é feita pela construção de objetos que representam topologias de data centers virtuais de SPs, contendo *switches*, rotas e enlaces. Além disso, a classe *Servidores* trata toda a representação de VMs em *racks* de servidores.

- **Política** é responsável pelo gerenciamento de demandas de alocação de redes virtuais com requisitos de largura de banda, resiliência e esquemas de endereçamento entre VMs e servidores que irão constituir a topologia virtual do VDC estabelecido a um SP. Ou seja, a partir da determinação do mapeamento de VMs em servidores interconectados a ToRs, previamente estipulada por SP e InP, uma política representa este mapeamento. Ela contém os endereços das VMs, servidores e ToRs desta demanda, assim como a matriz de tráfego entre eles. A topologia de rede virtual de uma política, criada pelo mapeamento desta no plano de dados pelo componente *Configuração* e representada pela instância de uma *TopologiaVirtual* do componente *Recurso*, é armazenada e associada a um identificador MPLS único desta política, utilizado na programação de rotas no plano de dados. Dentro deste componente a classe denominada *Políticas* realiza o cadastramento, gerenciamento e armazenamento das políticas criadas bem como de suas respectivas topologias virtuais construídas e mapeadas.

- **Configuração** contém algoritmos de aprovisionamento que realizam tanto a associação entre topologias quanto a passagem de rotas da topologia base virtual do InP

a topologias virtuais de SPs criando redes de data center virtualizadas. Além disso dentro deste componente existem algoritmos que auxiliam nas funções principais de mapeamento, os quais lidam com modificações de recursos das topologias base para melhor atender as demandas de alocação. Basicamente os algoritmos definidos neste componente, com base em requisitos de políticas, alocam rotas, enlaces e largura de banda da topologia física base. Como resultado obtêm-se um grafo o qual define uma topologia virtual a qual atende os atributos da política de requisição.

- **Alocador** realiza toda a interface de configuração de topologias virtuais no plano físico. É por meio dele que são configuradas topologias virtuais constituídas por políticas, que alocações são feitas utilizando o componente Configuração e que topologias virtuais são mapeadas na topologia física do InP com o envio de mensagens à aplicação rfproxy do plano físico. Além disso, ele também realiza a configuração de rotas no *switch* virtual para o gerenciamento do tráfego de controle do plano virtual do InP.

- **Escalonador** é o componente responsável por gerir todo o funcionamento dos componentes anteriores. Por meio dele são criadas políticas, constituídas e configuradas topologias virtuais e físicas e onde demandas de provisão de topologias virtuais são definidas em políticas para serem alocadas e desalocadas dinamicamente. Ou seja, este componente realiza toda a interface de comunicação com a aplicação rfserver, trata todos os parâmetros de configuração e orquestra as operações de todos os outros componentes. Além disso, nele exitem rotinas que permitem a análise de todas as topologias mapeadas para obtenção de dados para inferências estatísticas sobre o estado de toda a DCN.

3.3.2 Execução da Arquitetura

O funcionamento da arquitetura proposta ocorre da seguinte forma. Os elementos da plataforma são inicializados nesta ordem (vide Figura 3.5): LXCs e aplicações rfclient; *switch* do plano virtual; banco de dados; aplicação rfserver; controlador de rede e aplicação rfproxy; *switches* da topologia física. A aplicação rfserver realiza a inicialização de todos os componentes descritos na subseção anterior, tratando-os como serviços da plataforma RouteFlow.

O componente Escalonador (vide Figura 3.4), após aguardar a convergência de rotas do plano virtual e a descoberta de topologia do plano de dados, cria uma política padrão a qual é definida pelo mapeamento das topologias base, física e virtual. Neste primeiro momento, os objetos que representam estas topologias são instanciados para serem utilizados nos mapeamentos posteriores. Além disso, o objeto que representa servidores é

Figura 3.5: Execução da arquitetura construída

criado para definir a instanciação de políticas. A partir de então, três *threads* (vide Figura 3.6) são criadas para realizarem as tarefas de: (*i*) criação de demandas de alocação (ex: geradas por OpenStack, Hadoop) e definição de políticas; (*ii*) alocação de políticas para criação de grafos de topologias virtuais; e (*iii*) desalocação de políticas conforme tempo de alocação atingido.

A primeira *thread*, com base em uma matriz de tráfego entre VMs, aloca estas em *racks* de servidores utilizando o objeto *Servidores*, criando uma matriz de tráfego entre ToRs. Estas informações são agregadas juntamente com requisitos de largura de banda, resiliência e endereçamento, definindo uma política que é colocada em uma fila para ser alocada. A *thread* de alocação verifica esta fila e realiza a alocação de políticas na topologia física base por meio do algoritmo de mapeamento proposto neste trabalho, e logo, cria a topologia virtual associada a esta política. Em função do tempo de alocação definido para cada política, a *thread* de desalocação retira o mapeamento de topologias virtuais da topologia física base.

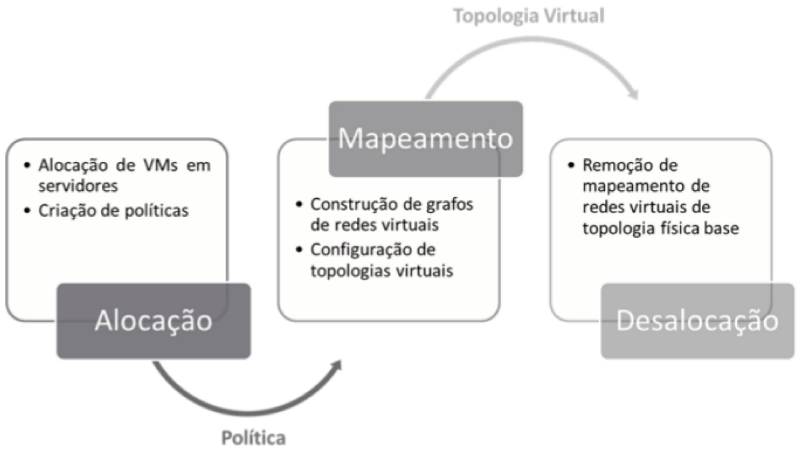


Figura 3.6: Execução das *threads* do plano de controle

Portanto, temos a seguinte relação de comportamento entre os componentes do plano de controle. *Recurso* sendo manipulado por *Escalonador* conforme informações de desco-

berta de topologia do plano de dados e constituição de rotas do plano virtual, constrói respectivamente as topologias base física e virtual que representam estes planos. Uma vez que as *threads* são iniciadas, a alocação de VMs em servidores definida pelo objeto *Servidores* ocorre e determina a construção de uma *Política*. Políticas vão sendo criadas conforme requisições de mapeamento são tratadas pela *thread* de criação de demandas. *Política* armazena os requisitos de uma rede virtual, constituídos de largura de banda (definição de matriz de tráfego entre VMs e ToRs), fator de resiliência (expresso em porcentagem) e esquema de endereçamento (endereços MAC, IP, IPv6 de VMs, servidores, ToRs, e portas de aplicações em sistemas operacionais). Na *thread* de alocação, *Configuração* tratada pelo *Escalonador* faz a construção da topologia virtual sobre *Recurso* conforme requisitos de *Política*. Ao final, *Alocador*, regido pelos requisitos de endereçamento e largura de banda definidos na topologia virtual construída, envia mensagens ao plano de dados para configurar a rede virtual constituída. Por fim, a *thread* de desalocação remove redes virtuais com tempo de mapeamento expirado utilizando os componentes *Recurso* e *Configuração*.

3.4 Algoritmos do Plano de Controle

Os algoritmos principais do plano de controle, localizados dentro do componente *Configuração*, são: (*i*) o algoritmo 1 (*Suporte*), que auxilia o gerenciamento de recursos do plano físico para que (*ii*) o algoritmo 2 (*Mapeamento*) realize a alocação de topologias virtuais de forma consistente. Este último é auxiliado pelo algoritmo 3 (*SelecionaRotas*) o qual proporciona a ele as rotas advindas do plano virtual necessárias ao mapeamento das redes virtuais.

Visto que a principal tarefa do componente *Configuração* é traduzir requisitos de políticas em alocações de recursos, a definição clara de quais operações podem ser realizadas sobre as topologias física e virtual se define como a principal característica dos algoritmos apresentados nesta seção. Logo, procuramos inicialmente mostrar quais estruturas de dados relacionadas a recursos do plano de dados são levadas em consideração no mapeamento de redes virtuais e como estas se associam aos requisitos de políticas de rede.

3.4.1 Algoritmo de Suporte

Em cada um dos *switches* da topologia física base do plano de controle, dois atributos se destacam. O primeiro define uma tabela (*bw_port_table*) de porcentagem de largura de banda em cada porta de um *switch*. Cada vez que um enlace adjacente a um *switch* é adicionado à topologia física base ou que uma rota é mapeada a este enlace, o percentual

de largura de banda disponível neste enlace é calculado e armazenado na *bw_port_table* dos *switches* adjacentes, na forma {*porta adjacente a enlace : porcentagem de largura de banda disponível*}. O outro atributo diz respeito à tabela (*bw_table*) de porcentagem de largura de banda disponível para endereços de destino das rotas de um *switch*. Cada vez que uma rota é mapeada a um enlace, os *switches* adjacentes atualizam suas tabelas de largura de banda conforme o endereço de destino da rota mapeada, constituindo entradas na forma {*endereço de destino : [porta de destino de rota a endereço : porcentagem de largura de banda disponível para a rota]* }.

É importante ressaltar que uma *bw_table* compreende a largura de banda fim-a-fim disponível para uma rota, enquanto que a *bw_port_table* representa a largura de banda disponível local, ou seja, somente em enlaces adjacentes a *switches*. *Switches* ToR tem em suas *bw_tables* os valores de largura de banda disponíveis para cada um dos servidores que estão interconectados a eles, definindo assim a porcentagem de largura de banda disponível em um *rack*. Além disso, Consideramos como estrutura principal de identificação das porcentagens de largura de banda em *bw_tables* os prefixos de rede que indicam servidores de um *rack* interconectado a um ToR. No entanto, qualquer informação que represente a identificação de ToRs e o conjunto de servidores conectados a ele pode ser utilizada como estrutura de dados em *bw_tables*.

Tendo como base estes atributos, o ponto chave do algoritmo de suporte está em atualizar as estruturas *bw_table* de todos os *switches* da topologia física base para que elas estejam consistentes com o estado das politicas mapeadas, e logo, sejam utilizadas pelo algoritmo de mapeamento proposto neste trabalho. Ou seja, este algoritmo faz com que todos os *switches* da topologia física base tenham informações atualizadas sobre a largura de banda disponível em cada *rack* da DCN assim como em cada uma das portas que contém rotas a estes *racks*. Esta atualização ocorre somente na topologia física base, toda vez que uma política é criada, alocada e desalocada.

O algoritmo acima se baseia em uma busca em largura, onde os nós de entrada são cada um dos *switches* ToRs com as informações de seus prefixos de rede e suas estruturas completas *bw_tables*. Nas linhas 1 a 3, vemos esta entrada sendo realizada e colocada na fila *filaSwitches*. A partir de então, cada um destas entradas é retirada da fila, tem sua *bw_table* atualizada para o menor valor de largura de banda que possa ser igualmente distribuído em todas as portas que contém rotas ao prefixo de rede do registro retirado da fila. Em seguida, todos os enlaces adjacentes a este *switch* retirado da fila que interligam outros *switches* tem suas entradas analisadas para que os elementos adjacentes sejam adicionados à fila de *switches* e a busca em largura aconteça na rede. A estrutura *SwitchesEmFila* armazena as *bw_tables* de *switches* que estejam na fila pois, estas são atualizadas nas linhas 6 a 9 do algoritmo 1 para que não ocorram inconsistências causadas pela atualização não simultânea destas estruturas decorrente da própria execução de uma busca em largura.

Algoritmo 1 : Executado para atualizar *bw_tables* de *switches* da topologia física base

Entrada: topologia física base (topo_phy_base)
Saída: *bw_tables* atualizadas de todos *switches* de topologia física base (topo_phy_base)

```
1: para todo switch ToR em topo_phy_base faça
2:     filaSwitches.adicionaItem(ToR, PrefixoDeRedeDeToR, ToR.bw_table, ToR.portas_a_servidores)
3: fim para
4: enquanto filaSwitches não vazia faça
5:     (switch, PrefixoDeRede, bw_tableDeSwitch, porta) ← filaSwitches.retiraItem()
6:     se (switch, PrefixoDeRede, porta) presente em SwitchesEmFila então
7:         bw_tableDeSwitch ← SwitchesEmFila(switch, PrefixoDeRede, porta)
8:         SwitchesEmFila.removeItem(switch, PrefixoDeRede, porta)
9:     fim se
10:    SwitchesVisitados.adicionaItem(switch, PrefixoDeRede, porta)
11:    bw_usage_list ← lista de todos os valores de largura de banda em switch.bw_table com destino a PrefixoDeRede
12:    bw_usage_mean ← Maior valor a ser alocado igualmente em todas entradas de bw_usage_list para PrefixoDeRede
13:    se porta contida em enlaces a outros switches então
14:        switch_bw_table[PrefixoDeRede][porta] ← min(bw_usage_mean,switch._bw_port_table[porta])
15:    fim se
16:    para todo enlace adjacente a switch não contendo porta faça
17:        se (enlace.switchDestino,PrefixoDeRede,enlace.portaDestino) não presente em SwitchesEmFila e SwitchesVisitados então
18:            filaSwitches.adicionaItem(enlace.switchDestino,PrefixoDeRede,switch_bw_table,enlace.portaDestino)
19:        fim se
20:        se (enlace.switchDestino,PrefixoDeRede,enlace.portaDestino) presente em SwitchesEmFila então
21:            SwitchesEmFila(switch,PrefixoDeRede,porta) ← switch_bw_table
22:        fim se
23:    fim para
24: fim enquanto
```

Ou seja, caso dois *switches* ToR adicionem à fila um mesmo *switch* EoR, este terá sua *bw_table* atualizada duas vezes para dois prefixos de rede diferentes provenientes destes ToRs, e logo, caso a última atualização não considere a atualização anterior inconsistências existirão na *bw_table* do EoR.

3.4.2 Algoritmo de Mapeamento

Assim como o algoritmo 1, este algoritmo é definido por uma busca em largura na topologia física base. Os nós de entrada são os *switches* ToR que uma política possui definidos pela matriz de tráfego de VMs mapeadas em *racks* de servidores. A saída do algoritmo é uma topologia virtual construída com base em informações de rotas da topologia virtual base e disponibilidade de largura de banda da topologia física base. O mapeamento ocorre por anotações de largura de banda, definidas pelo identificador único de uma política, feitas em enlaces da topologia física base. Estas marcações são realizadas de acordo com rotas selecionadas da topologia virtual perante requisitos de largura de banda e resiliência definidos nas políticas de mapeamento.

Como especificado na descrição do plano de controle da arquitetura proposta, uma política constitui-se da definição de VMs alocadas em servidores, que por sua vez são referenciados pelos ToRs aos quais estão interconectados. Políticas também possuem requisitos de largura de banda, os quais são definidos por quaisquer padrões de tráfego que

representem a comunicação entre VMs. Estas alocações de largura de banda são agregadas conforme o mapeamento de VMs em servidores, para que componham uma matriz de tráfego entre ToRs, referenciando as VMs, servidores e seus respectivos endereços de rede (ex: MAC, IP, IPv6). Adicionalmente políticas também possuem requisitos de resiliência, os quais são expressos em porcentagens. Estas representam a quantidade de múltiplos caminhos que serão escolhidos na seleção de rotas no mapeamento de largura de banda às redes virtuais. Por exemplo, caso uma política tenha fator de resiliência igual a 0.5, e um *switch* tenha rotas definidas em 4 portas para um determinado prefixo de rede a ser alocado a esta política, quando a seleção de rotas ocorrer 2 (0.5 x 4) rotas serão selecionadas, caso a resiliência fosse igual a 1, as 4 portas seriam utilizadas para mapear as rotas desta política. Consideramos que esta tarefa de divisão de largura de banda é feita de forma igualitária entre todas as portas com rotas a um determinado prefixo de rede.

Descrevendo o algoritmo 2, vemos que ele realiza duas tarefas principais, a seleção de rotas, a marcação destas em enlaces da topologia física base, e a construção de um grafo representando a topologia virtual segundo os requisitos de largura de banda e resiliência de uma política. Logo, com base na matriz de tráfego entre ToRs de uma política, todos os *switches* com prefixos de rede comunicantes com um determinado conjunto de VMs alocadas a servidores de um ToR são colocados na fila *filaSwitches* conforme visto nas linhas 2 a 6. Ou seja, todos os *switches* ToR que requisitam largura de banda na comunicação entre si definida pela matriz de tráfego de uma política, tem uma entrada representada na estrutura *filaSwitches*, na forma (lxc, ToR, prefixo de rede de destino). Esta estrutura constitui-se de um identificador do LXC, elemento da topologia virtual mapeado ao ToR, do qual serão obtidas as rotas a serem inseridas ao ToR na topologia virtual a ser construída.

A tarefa de mapeamento ocorre pela anotação de requisições de largura de banda em enlaces da topologia física base. Rotas são selecionadas pelo algoritmo 3 segundo duas técnicas, uma desenvolvida neste trabalho (*SelecRotas Agreg*) e outra comumente vista na literatura (*SelecRotas Tradicional*). Conforme as rotas selecionadas, a largura de banda é igualmente dividida entre elas para serem alocadas aos enlaces que interligam estas rotas de um *switch* a outro. O algoritmo 2 segue percorrendo todos os *switches* da topologia física base (linhas 8 a 27), obtendo rotas dos LXCs mapeados a ele (linhas 14), selecionando estas rotas conforme os mecanismos do algoritmo 3 (linhas 14 e 15) e anotando as larguras de banda requisitadas nos enlaces que estas rotas selecionadas representam (linhas 16 a 26). Por esta tarefa, este algoritmo constrói uma topologia virtual a qual representa os requisitos de comunicação e largura de banda da matriz de tráfego da política de entrada. Caso todas as requisições de largura de banda sejam corretamente alocadas o algoritmo 2 retorna um grafo representando a topologia virtual

Algoritmo 2 : Executado para mapear topologias virtuais na topologia física base

Entrada: topologia física base (topo_phy_base), topologia virtual base (topo_virt_base) e política
Saída: topologia virtual

para todo *switch* ToR em matriz de tráfego de política **faça**
  para todo PrefixoDeRede em matriz de tráfego de política ≠ faixa de endereços de rede de *switch* ToR **faça**
    lxc ← lxc de topo_virt_base mapeado a ToR
    filaSwitches.adicionaItem(lxc, ToR, PrefixoDeRede)
    PropriedadesSwitches(lxc, ToR, PrefixoDeRede) = Largura de banda de ToR a PrefixoDeRede em matriz de tráfego de política
  fim para
fim para
enquanto filaSwitches não vazia **faça**
  (lxc, *switch*, PrefixoDeRede) = filaSwitches.retiraItem()
  se PrefixoDeRede não presente em entradas de SwitchesEmFila **então**
    SwitchesVisitados.adicionaItem(lxc, *switch*, PrefixoDeRede)
  fim se
  larguraDeBandaRequisitada = PropriedadesSwitches(lxc, *switch*, PrefixoDeRede)
  switch_rotas_selecionadas ← SelecionaRotas(*switch*,lxc,PrefixoDeRede,larguraDeBandaRequisitada) # Algoritmo 3
  larguraDeBandaRotas←larguraDeBandaRequisitada dividida pela quantidade de switch_rotas_selecionadas
  para todo rota em switch_rotas_selecionadas **faça**
    se larguraDeBandaRotas[rota] alocada em enlace de topo_phy_base definido por rota **então**
      Adicione *switch*, rota em *switch* e enlace em TopologiaVirtual
      Defina como lxcDestino e switchDestino os respectivos lxc e *switch* de destino do enlace definido por rota
      filaSwitches.adicionaItem(lxcDestino, switchDestino, PrefixoDeRede)
      PropriedadesSwitches(lxcDestino, switchDestino, PrefixoDeRede) ← larguraDeBandaRotas[rota]
    else
      Mapeamento ← Falso
      Pare os laços de execução
    fim se
  fim para
fim enquanto
se Mapeamento ≠ Verdadeiro **então**
  Desfaça mapeamentos até então feitos de politica em topo_phy_base
fim se

Algoritmo 3 SelecionaRotas: Executado para selecionar rotas para mapeamento

Entrada: (*switch*, lxc, PrefixoDeRede, larguraDeBandaRequisitada)
Saída: rotas selecionadas para mapeamento

switch_rotas ← lxc.get_rotas(PrefixoDeRede)
Seleciona switch_rotas com maior capacidade em switch.bw_port_table que satisfaçam critério de resiliência de política
se Opção SelecRotas Agreg **então**
  Calcula média, desvio padrão, maior e menor valor de entradas de switch.bw_port_table com portas definidas por switch_rotas
  Seleciona combinações de rotas de switch_rotas que satisfazem larguraDeBandaRequisitada dividida entre elas **e** que definam valores de switch._bw_port_table maiores ou iguais a média subtraída do dobro do desvio padrão de switch.bw_port_table
  Para as combinações de rotas selecionadas na etapa anterior selecione aquela que possui a menor diferença entre os valores máximo e mínimo caso seus valores de largura de banda sejam selecionados e aplicados a switch.bw_port_table
  Retorne conjunto de rotas selecionadas na etapa anterior
fim se
se Opção SelecRotas Tradicional **então**
  Retorna rota de switch_rotas que satisfaz larguraDeBandaRequisitada em entrada PrefixoDeRede de switch.bw_table
fim se

construída conforme os requisitos de política de entrada. A estrutura *PropriedadeSwitches* armazena a largura de banda selecionada a uma rota, a qual é atualizada caso *switches* estejam na fila, para que não ocorram erros de alocação de largura de banda decorrentes

dos caminhos selecionados pela busca em largura efetuada pelo algoritmo 2.

Na opção *SelecRotas Agreg* do algoritmo 3, os critérios de resiliência de políticas são definidos por porcentagens, as quais expressam a quantidade de rotas que serão selecionadas para a avaliação de disponibilidade de caminhos. Além disso, nesta opção, a utilização do parâmetro de seleção de rotas, média subtraída do dobro do desvio padrão, permite que as rotas selecionadas estejam dentro de dois valores de desvio padrão da média dos valores de *bw_port_tables*. Isto garante que haja balanceamento de carga nas portas de um *switch*, ao contrário da opção *SelecRotas Tradicional*, onde somente uma rota com menor disponibilidade de largura de banda é selecionada e nenhum critério de balanceamento de carga analisado.

Logo, a opção *SelecRotas Agreg* garante que a largura de banda requisitada a ser mapeada a enlaces pela seleção de rotas, seja igualmente distribuída entre todas as portas que estas rotas representam. Como visto nas linhas 4, 5 e 6 do algoritmo 3, determinamos a existência de balanceamento de carga pois, as rotas selecionadas na opção *SelecRotas Agreg* são aquelas que definem a menor diferença da média dos valores de porcentagem de largura de banda contidos em *bw_port_tables* e *bw_tables*. Daí a importância dos valores destas tabelas estarem consistentes com o estado da topologia física base, conforme políticas tenham suas topologias virtuais construídas, mapeadas e desalocadas. Logo, segue assim a justificativa de relevância do algoritmo 1 em atualizar essas tabelas.

Tabela 3.1: Exemplo *Bw_table* 1

Prefixo de Rede	Portas	% de largura de banda
10.0.1.0/24	1	0.5
	2	0.5
	3	0.4
	4	0.4

Tabela 3.2: Exemplo *Bw_table* 2

Prefixo de Rede	Portas	% de largura de banda
10.0.2.0/24	1	0.8
	2	0.7
	3	0.3
	4	0.2

Nas Tabelas 3.1 e 3.2 acima, temos os exemplos de duas entradas em uma *bw_table*, fazendo referências aos prefixos de rede 10.0.1.0/24 e 10.0.2.0/24. Supondo que o algoritmo 3 precise fazer a seleção de rotas para a definição do mapeamento de uma rede virtual em um *switch* que possua tais entradas em sua *bw_table*, onde a largura de banda represente 80% da largura de banda de cada enlace deste *switch*, temos dois exemplos sobre a escolha a ser feita pela opção *SelecRotas Agreg*:

- Caso o prefixo de destino seja 10.0.1.0/24 todas as rotas que definem as entradas para as portas 1, 2, 3 e 4 serão escolhidas. Pois, caso esta largura de banda seja

alocada em todos os enlaces adjacentes a estas portas, teremos a subdivisão igual de 20% desta demanda a todas as portas, obtendo uma nova entrada para este prefixo de rede igual a [0.3; 0.3; 0.2; 0.2]. Esta entrada possui a menor diferença entre os valores máximo e mínimo caso seus valores de largura de banda sejam selecionados e aplicados a *bw_port_table*. Lembrando que esta é igual a *bw_table* deste *switch* por causa da execução do algoritmo 1, o qual atualiza todas as entradas de *bw_tables* em toda os *switches*.

- Caso o prefixo de destino seja 10.0.2.0/24 as rotas que definem as entradas para as portas 1 e 2 serão escolhidas. Como descrito anteriormente, isto ocorre pois, de todas as opções de rotas a serem selecionadas, o valor desta entrada na *bw_table* caso esta opção seja escolhida é igual a [0.4; 0.3; 0.3; 0.2], o que representa a menor diferença entre seus valores máximo e mínimo, igual a 0.2. Valores maiores poderiam ser obtidos, como por exemplo, caso todas as portas fossem escolhidas esta diferença seria de 0.6 (ex: [0.6; 0.5; 0.1; 0.0]), e caso a porta 1 fosse a única escolhida, a diferença seria de 0.5 (ex: [0.0; 0.7; 0.3; 0.2]). Além disso, a própria seleção de combinações de rotas também refina este valor, pois a média e desvio padrão dos valores da Tabela 3.2 são iguais a 0.5 e 0.2549. Logo, a seleção das rotas para as portas 1 e 2 também é uma combinação que satisfaz a primeira condição onde seus valores subtraídos da demanda de largura de banda (valores iguais a [0.4; 0.3]) são maiores que a média subtraída de duas vezes o desvio padrão das entradas da *bw_table*.

No exemplo acima, podemos ver como a opção *SelecRotas Agreg* do algoritmo 3 seleciona rotas para satisfazer o balanceamento de carga nas portas de cada *switch*. Caso a opção *SelecRotas Tradicional* fosse utilizada, o mapeamento seria negado no primeiro caso, e no segundo caso usaria toda a largura de banda disponível na porta 1. Nenhuma destas escolhas leva em consideração requisitos de balanceamento de carga na rede. É importante notar, que a seleção de rotas pela opção *SelecRotas Agreg* procura igualar ao máximo os valores de largura de banda utilizados em cada porta de um *switch*, considerando para isso a opção de seleção de rotas que define a menor diferença entre os valores de uma *bw_table* caso a demanda seja alocada com a melhor escolha de rotas.

Capítulo 4

Análise Experimental

Neste capítulo descrevemos como foram organizados e realizados os experimentos feitos para a avaliação da arquitetura criada e dos algoritmos propostos neste trabalho. Inicialmente descrevemos as ferramentas utilizadas na arquitetura, suas principais funções e como elas se encaixam nos requisitos da implementação da arquitetura. Adiante procuramos abordar as características do ambiente de testes montado, e como ele reflete os interesses a serem medidos e analisados para a avaliação da arquitetura e dos algoritmos. Assim, apresentamos os experimentos realizados, os resultados obtidos bem como as associações entre eles e suas análises. Por fim apresentamos uma amostragem do desempenho da arquitetura construída, discriminando os tempos de mapeamento, configuração e desalocação.

4.1 Ferramentas Utilizadas

O *testbed* utilizado para a avaliação da arquitetura proposta neste trabalho foi montado utilizando a plataforma RouteFlow com as respectivas modificações mencionadas na seção 3.3 do capítulo 3. No plano de dados construímos a aplicação rfproxy sobre o controlador de rede Ryu[1] com as devidas modificações para suporte a OpenFlow 1.3. Este controlador de rede foi escolhido, pois até onde conhecemos ele era o único que possuía suporte completo a OpenFlow 1.3 quando a arquitetura foi construída. Utilizamos o *switch* ofsoftswitches13[2] tanto no plano de dados quanto no plano virtual. Este *switch* possui suporte completo a OpenFlow 1.3 e está em amplo desenvolvimento com uma comunidade ativa e colaborativa.

Realizamos a construção de duas topologias folded-Clos utilizando a plataforma Mi-

[1]https://github.com/osrg/ryu
[2]https://github.com/CPqD/ofsoftswitch13

ninet[3], com 12 e 48 *switches*. No plano virtual realizamos a construção do mapeamento com agregação para as topologias físicas criadas. Utilizamos a suíte de roteamento de código aberto Quagga[4] nos LXCs do plano virtual. O protocolo de roteamento BGP foi especificamente configurado para cada um dos LXCs conforme o mapeamento de topologias física e virtual apresentado na elaboração da arquitetura. Ou seja, realizamos a configuração de um único roteador virtual Core representando todos os 4 *switches* Core de uma topologia física folded-Clos com 12 *switches*, da mesma forma um único roteador virtual Core para os 16 *switches* da topologia física folded-Clos com 48 *switches*. Nesse caso, ambos roteadores virtuais Core foram configurados com um único ASN e tiveram suas portas mapeadas a cada uma das portas de todos os *switches* Core da topologia do plano de dados. Da mesma forma, foi realizado este procedimento para os *switches* EoR, agregados em um único roteador virtual com um mesmo ASN, todos aqueles que se interligam a um mesmo conjunto de *switches* ToR.

4.2 Avaliações Analíticas

Inicialmente fizemos um estudo analítico sobre a abordagem de construção da arquitetura com a utilização do protocolo BGP em roteadores virtuais definidos pelo mapeamento com agregação de switches da topologia física. Nesse aspecto, buscamos observar uma comparação entre as topologias folded-Clos com e sem agregação, referenciadas respectivamente por Clos e ClosAgreg. Na Tabela 4.1 vemos que a topologia virtual com agregação determina um número menor de conexões entre roteadores virtuais, menor quantidade de conexões TCP para as sessões BGP e, consequentemente, menor número de mensagens trocadas em cada rodada de atualizações do protocolo BGP.

Estes resultados indicam que a topologia virtual é mais eficiente do que uma topologia física tanto em quantidade de elementos computadores de rotas (roteadores virtuais ou não) quanto em quantidade de conexões entre pares de roteadores executando o protocolo BGP. Logo, mensagens de controle do protocolo BGP são menos frequentes entre os pares de roteadores. Outro fator importante, é o fato destas mensagens estarem confinadas somente no plano virtual. Ou seja, não correm riscos de serem perdidas ou recebidas fora de ordem, em virtude de apenas serem encaminhadas por um único *switch* virtual o qual define rotas específicas entre cada uma das interfaces que interconectam as portas dos roteadores virtuais.

Por outro aspecto, buscamos avaliar a quantidade de rotas a serem programadas em *switches* do plano de dados pela configuração das redes virtuais mapeadas. Logo, tratamos de verificar primeiramente, a quantidade de entradas em cada tabela de cada classe de

[3]https://github.com/mininet/mininet
[4]https://github.com/opensourcerouting/quagga

Tabela 4.1: Comparação entre topologias virtuais sem e com agregação

Topologia	Switches Plano de Dados	Roteadores Virtuais	Sessões BGP	Mensagens de Controle
Clos	12	12	16	32
ClosAgreg	12	7	6	12
Clos	48	48	64	128
ClosAgreg	48	9	8	16
Clos	96	96	256	512
ClosAgreg	96	11	34	68
Clos	128	128	1024	2048
ClosAgreg	128	35	68	136

switches (ToR, EoR, Core) da topologia física. Conforme especificado na subseção 3.3.1, no tópico "Plano Físico", os *switches* EoR e Core encaminham tráfego conforme *tags* MPLS definidas especificamente para cada rede virtual. Enquanto que os *switches* ToR possuem definições específicas para suas tabelas 0 a 3, além da constituição de limitadores de largura de banda pela configuração de *meter tables*.

Com base nas informações de como as tabelas dos elementos de rede do plano de dados são constituídas, abaixo definimos as Tabelas 4.2 e 4.3 para especificar a quantidade de entradas em tabelas de *switches* conforme a quantidade de VMs em *racks* e a quantidade de redes virtuais configuradas no plano de dados, respectivamente. Desejamos com isso comparar o modelo de configuração de rotas definido para a arquitetura proposta com o modelo de rotas caso sejam configuradas rotas com endereços IP e MAC para interconexão de servidores caso seja necessário mapear a uma rede virtual que interligue todos eles na DCN. Por este aspecto, na Tabela 4.2 consideramos a existência de uma rede virtual e uma topologia folded-Clos com 48 *switches* sendo 16 deles do tipo ToR. Já na Tabela 4.3 consideramos a existência de uma topologia folded-Clos com 48 *switches* sendo 16 deles do tipo ToR, 20 servidores por *rack* e 20 VMs alocadas em cada servidor.

Tabela 4.2: Quantidade Regras em Tabelas de ToRs por # de VMs

# Regras em Tabelas	# VMs por *rack*			
	20	100	400	2000
0	1	1	1	1
1	20	20	20	20
2	20	100	400	2000
3	15	15	15	15

Tabela 4.3: Quantidade Regras em Tabelas de ToRs por # de Redes Virtuais

# Regras em Tabelas	# de Redes Virtuais			
	1	100	1000	10^6
0	1	100	1000	10^6
1	20	20	20	20
2	400	400	400	400
3	15	1500	15000	$15 * 10^6$

Conseguimos ver na Tabela 4.2 que a quantidade de regras instaladas em *switches* que a quantidade de regras nas tabelas 1 e 3 permanecem constantes, iguais respectivamente a 1 e 15. Isto pois, como existe somente uma rede virtual configurada na rede, cada ToR define na tabela 0 uma regra com a *tag* MPLS que identifica a entrada de tráfego no *rack* para esta rede virtual, enquanto que as 15 regras da tabela 3 identificam *tags* MPLS as quais identificam regras associadas ao envio de tráfego com destino aos outros *racks* da rede, no caso 15 ToRs. As regras na tabela 1 identificam rotas a endereços de rede de servidores, como consideramos a existência de 20 servidores por *rack*, esta é a mesma quantidade de entradas nesta tabela. E por fim, a tabela 2 define a quantidade de entradas com destinos a endereços MAC, os quais representam entradas com destinos a VMs, as quais variam linearmente conforme o crescimento da quantidade de VMs em cada *rack*.

Na Tabela 4.3 avaliamos a quantidade de regras em ToRs conforme a quantidade de redes virtuais mapeadas englobando todos os servidores e VMs de toda a topologia folded-Clos com 48 *switches* definida por 20 servidores por *rack* e 20 VMs por servidor. Logo, vemos que as entradas para as tabelas 1 e 2 são constantes pois, o número de servidores e VMs em cada *rack* é constante. Já as outras tabelas, 0 e 1, representam respectivamente a quantidade de regras definidas por *tags* MPLS que representam a entrada e saída de tráfego em *racks*. Logo, variam linearmente conforme a quantidade de redes virtuais configuradas no plano de dados, ou seja, a tabela 0 representa regras de redes virtuais que destinam tráfego ao interior de *racks*, enquanto que a tabela 3 representa regras que destinam tráfego a todos os outros *racks* da rede, sendo portanto, múltipla de 15.

Abaixo mostramos exemplos de definições de regras apresentadas nas tabelas 0 a 3 de um *switch* ToR. Vemos na primeira entrada que a tabela 0 possui como *match* a *tag* MPLS igual a 234 e esta regra realiza as ações de retirar a camada MPLS dos pacotes, aplicar o limitador de largura de banda identificado por *meter* 1 e encaminhar à tabela 1. Nesta o tráfego com endereço IP de destino igual a 192.168.1.2 tem os campos de endereço MAC definidos e segue a tabela 2. Conforme os endereços MAC fonte 02:a1:a1:a1:a1:14 e destino 0a:00:00:00:00:26 o tráfego nesta tabela tem como destino a porta 4 sendo feita

a reescrita de endereços MAC para aquele relativo a VM de destino. A tabela 3 define o encaminhamento de tráfego ao prefixo de rede 192.168.3.0/24, aplicando o limitador de largura de banda *meter 2*, a *tag* MPLS 235 e os endereços MAC fonte e destino especificados, e por fim, enviando os pacotes a uma entrada na *group table* com o destino *group 10000000*.

```
{table="0", match="oxm{mpls_label="234", eth_type="0x8847"}",
dur_s="9", dur_ns="155000", prio="24", idle_to="0", hard_to="0",
cookie="0x0", pkt_cnt="0", byte_cnt="0", insts=[meter{meter="1"},
apply{acts=[mpls_pop{eth="0x0800"}, goto{table="1"}]}}

{table="1", match="oxm{ipv4_dst="192.168.1.2", eth_type="0x800"}",
dur_s="8", dur_ns="931000", prio="32", idle_to="0", hard_to="0",
cookie="0x0", pkt_cnt="0", byte_cnt="0",
apply{acts=[set_field{field:eth_dst="0a:00:00:00:00:26"},
set_field{field:eth_src="02:a1:a1:a1:a1:14"}]}, goto{table="2"}]}

{table="2", match="oxm{eth_dst="0a:00:00:00:00:26",
eth_src="02:a1:a1:a1:a1:14", eth_type="0x800"}", dur_s="8",
dur_ns="994000", prio="10", idle_to="0", hard_to="0",
cookie="0x0", pkt_cnt="0", byte_cnt="0",
apply{acts=[set_field{field:eth_dst="0a:00:00:00:00:55"},
set_field{field:eth_src="02:a1:a1:a1:a1:14"}, out{port="4"}]}]}

{table="3", match="oxm{eth_type="0x800", ipv4_dst="192.168.3.0",
ipv4_dst_mask="255.255.255.0"}", dur_s="9", dur_ns="155000",
prio="24", idle_to="0", hard_to="0", cookie="0x0", pkt_cnt="0",
byte_cnt="0", insts=[meter{meter="2"},
apply{acts=[mpls_psh{eth="0x8847"}, set_field{field:mpls_label="235"},
set_field{field:eth_dst="02:b1:05:05:05:05"},
set_field{field:eth_src="02:a1:a1:a1:a1:14"},
group{id="10000000"}]}]}
```

Abaixo seguem exemplos de entradas de configurações para tabelas *group* e *meter*. A primeira define o limitador *meter* 1 o qual realiza o descarte *drop* de pacotes caso a taxa 500kbps seja ultrapassada. Já a entrada seguinte, define o identificador 10000000 para o registro que realiza o encaminhamento de pacotes através das portas 5, 6, 7 e 8, definidas em cada um dos *buckets* desta entrada do tipo *select*, a qual tem por função espalhar o tráfego igualmente entre todos estes *buckets*.

```
{meter= "1", flags="1", bands=[{type = drop,
rate="500", burst_size="0"}]}

{type="sel", group="10000000", buckets=[{w="1", wprt="0", wgrp="0",
acts=[out{port="5"}]}, {w="1", wprt="0", wgrp="0", acts=[out{port="7"}]},
{w="1", wprt="0", wgrp="0", acts=[out{port="8"}]}, {w="1", wprt="0",
 wgrp="0", acts=[out{port="6"}]}]}
```

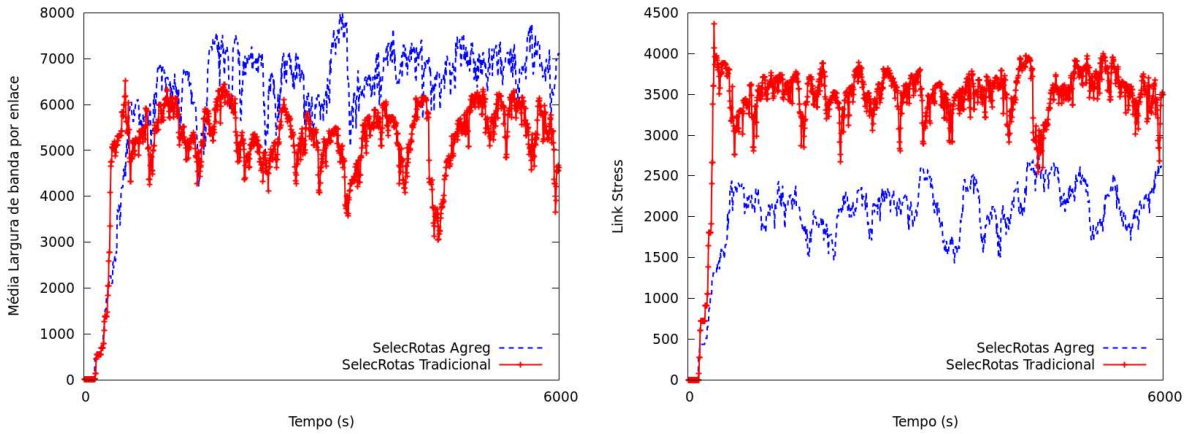
4.3 Experimentos e Resultados

Adiante tratamos de avaliar o algoritmo *SelecRotas Agreg* proposto neste trabalho e compará-lo ao algoritmo *SelecRotas Tradicional*. Para os parâmetros de avaliação, consideramos na topologia física e no plano de controle que cada enlace entre *swiches* possui 10.000 unidades de largura de banda, e entre *switches* e servidores 1.000 unidades de largura de banda. Além disso, consideramos que nas topologia físicas com 12 e 48 *switches* existam respectivamente 20 e 40 servidores em cada *rack*, podendo cada servidor possuir no máximo 20 VMs alocadas. Realizamos a criação de demandas de mapeamento em um processo de Poisson com VMs sendo alocadas de forma ordenada em servidores de ToRs conforme a disponibilidade de recursos em servidores.

Consideramos os seguintes parâmetros para a realização do experimento: chegada de demandas por processo de Poisson com média de 30 requisições por minuto. Cada requisição possui, respectivamente para as topologias com 12 e 48 *switches*: fatores de resiliência iguais a 100% para a opção *SelecRotas Tradicional* no algoritmo 3; quantidade de máquinas virtuais uniformemente distribuída entre 10 e 30 e entre 30 e 70; demanda de tráfego entre VMs uniformemente distribuída entre 1 e 10 unidades de largura de banda; redes virtuais mapeadas com tempo de permanência na rede uniformemente distribuído entre 180 e 300 e entre 540 e 660 segundos; e tempo total de experimento definido em 6000 e 18000 segundos. Neste sentido, avaliamos a largura de banda e sua variação (*link stress*) nos enlaces da rede, para entendermos o balanceamento de carga na topologia física base.

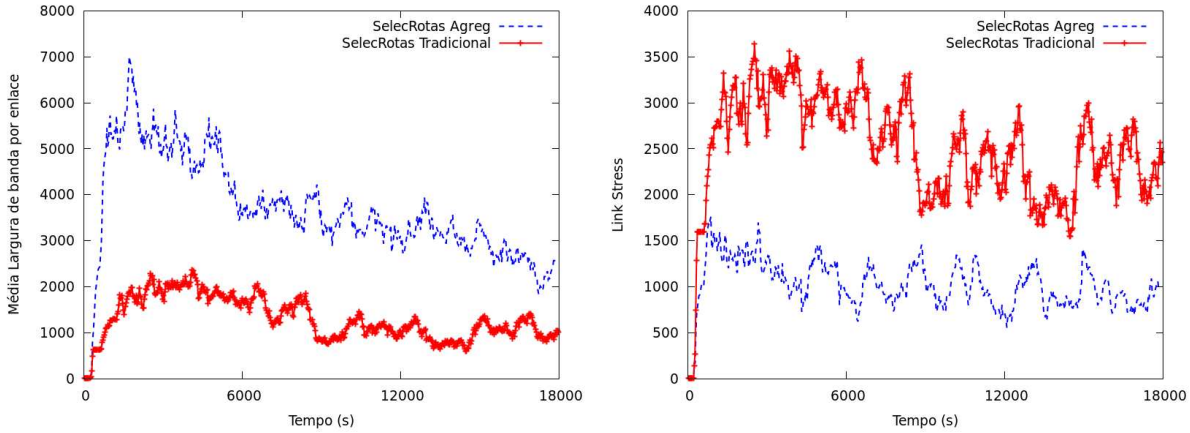
É importante ressaltar que os valores de demandas de alocação de VMs, de taxas de chegada e tempo de alocação de redes virtuais, assim como os valores de largura de banda estipulados para a comunicação entre VMs, não tem como objetivo a obtenção de taxas de alocação eficientes de VMs em servidores de *racks* de ToRs. Estamos ciente de que existem inúmeros algoritmos na literatura que realizam esta tarefa de forma eficiente utilizando diversas técnicas, como algoritmos genéticos e programação linear. No entanto, o algoritmo que aloca VMs em servidores utilizado neste trabalho apenas realiza esta tarefa conforme a ordenação de recursos (CPU e largura de banda) disponíveis em *racks* de servidores. Estes valores apenas representam parâmetros que levam a criação de redes

virtuais as quais são de interesse à alocação do algoritmo proposto neste trabalho. O principal objetivo na criação de demandas de alocações de VMs é a definição de alocações de VMs de uma mesma demanda em *racks* distintos levando assim a composição de redes virtuais pelo algoritmo de mapeamento proposto neste trabalho.



(a) Média de largura de banda utilizada por enlace (b) Link stress

Figura 4.1: Dados de topologia física folded-Clos com 12 switches

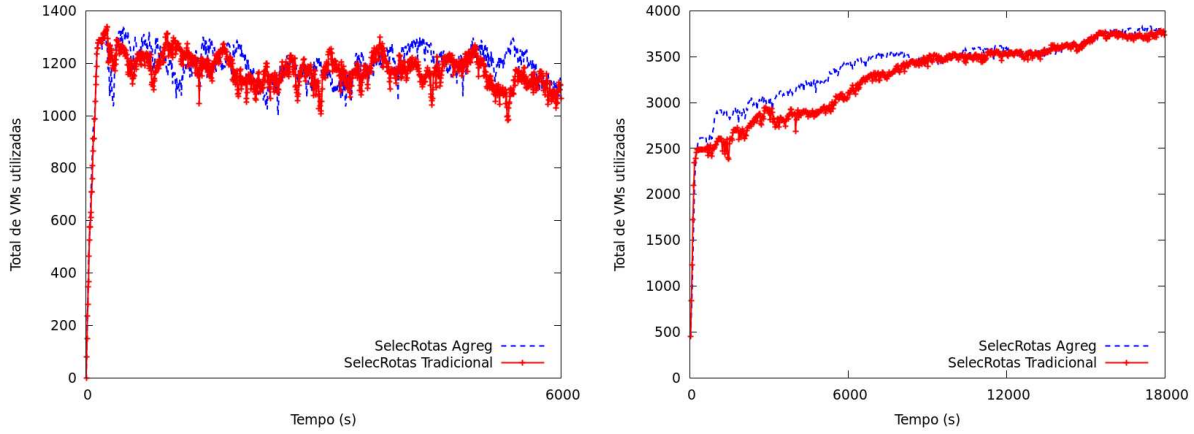


(a) Média de largura de banda utilizada por enlace (b) Link stress: Padrão de comunicação All-to-All

Figura 4.2: Dados de topologia física folded-Clos com 48 switches

Para analisar o balanceamento de carga nas topologias de rede construídas, obtemos nas Figuras 4.1 e 4.2 os valores de média de largura de banda utilizada por enlace e desvio padrão da utilização de largura de banda por enlace. Vemos nas Figuras 4.1(a) e 4.2(a) que os valores de largura de banda na rede para as topologias com 12 e 48 *switches* são

em média maiores quando utilizamos a opção *SelecRotas Agreg* em comparação com a opção *SelecRotas Tradicional* no algoritmo 3. As variações observadas se devem ao comportamento do algoritmo de alocação de VMs em servidores e das taxas de chegada e desalocação de redes virtuais definidas por processos de Poisson. Este comportamento é visto nas Figuras 4.3(a) e 4.3(b), onde oscilações nas quantidades de VMs ocorrem ao longo de todos os experimentos. Notamos também nestas figuras que, como mencionado anteriormente, não procuramos uma alocação ótima de VMs em servidores, e logo, obtemos as alocações totais de VMs abaixo dos valores máximo de *slots* disponíveis, sendo 1600 e 12800 respectivamente para as topologias com 12 e 48 *switches*.



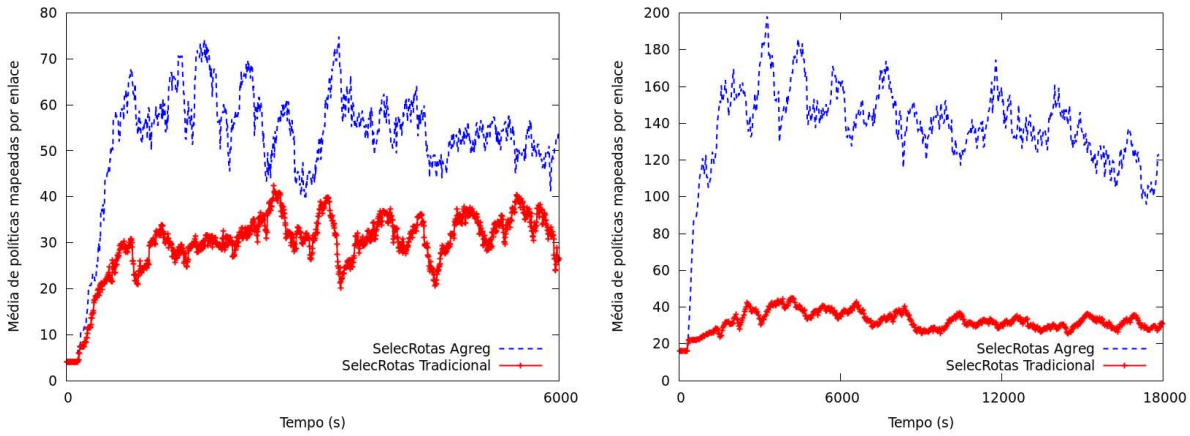
(a) Quantidade de VMs em folded-Clos com 12 switches

(b) Quantidade de VMs em folded-Clos com 48 switches

Figura 4.3: Alocação de VMs em servidores

Além disso, temos nas Figuras 4.1(b) e 4.2(b) que os valores médios de *Link Stress* são menores quando utilizamos a opção *SelecRotas Agreg* em comparação com a opção *SelecRotas Tradicional* no algoritmo 3. Os valores apresentados nestas figuras mostram que a variação de utilização de largura de banda nos enlaces é consideravelmente menor quando utilizamos a opção *SelecRotas Agreg*. Isto indica que os enlaces em média apresentam a mesma alocação de largura de banda para esta opção, tornando assim a rede balanceada com a carga gerada pelos mapeamentos de redes virtuais configurados. Logo, como a opção *SelecRotas Tradicional* apresenta grandes variações de largura de banda nos enlaces da rede, vemos que alguns enlaces estão sendo mais utilizados do que outros. A razão da diferença entre estes valores se deve à utilização de múltiplos caminhos para a seleção de rotas por parte da opção *SelecRotas Agreg* definida com 100% de resiliência para todas as políticas criadas. Este parâmetro garante que a maior quantidade possível de rotas disponíveis, que levem a enlaces com capacidade permitida para a alocação de largura de banda desejada pela política, é selecionada para mapear uma rede virtual.

Em termos de quantidade de políticas alocadas por enlace, vemos nas Figuras 4.4(a) e 4.4(b) que há uma grande disparidade entre a utilização de enlaces por políticas alocadas quando comparadas as opções *SelecRotas Agreg* e *SelecRotas Tradicional*. Ou seja, por fazer uso de múltiplos caminhos e demonstrar balanceamento de carga eficiente na rede a opção *SelecRotas Agreg* mapeia uma maior quantidade de políticas em enlaces da topologia física de rede, fazendo uso de maior largura de banda de uma maior quantidade de enlaces. Já a opção *SelecRotas Tradicional* utiliza de uma menor quantidade de enlaces com políticas mapeadas para encaminhar tráfego, o que expressa um comportamento esperado devido à ausência de utilização de múltiplos caminhos e não balanceamento de carga na rede.

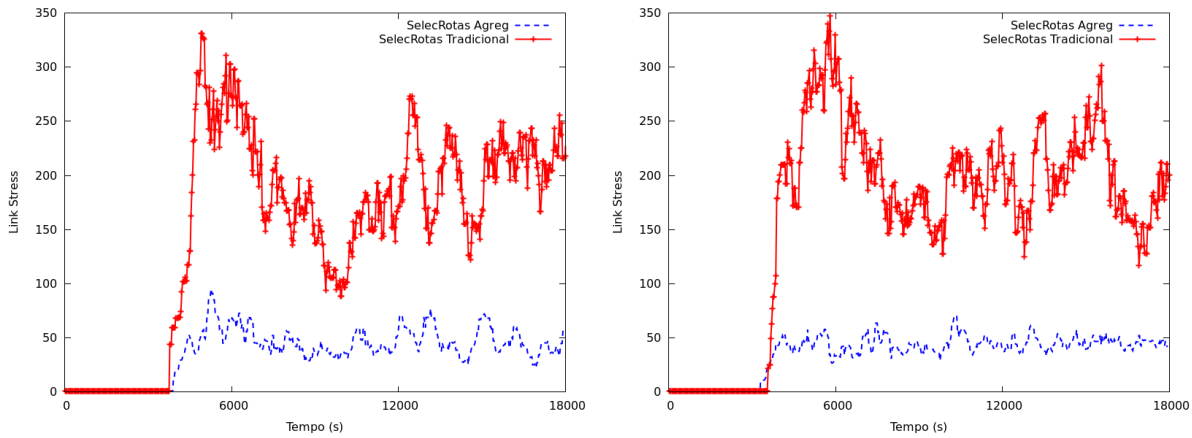


(a) Média de políticas por enlace em folded-Clos com 12 switches

(b) Média de políticas por enlace em folded-Clos com 48 switches

Figura 4.4: Alocação de políticas em enlaces

Para obtermos análises relativas a padrões de tráfego, realizamos um experimento utilizando a topologia folded-Clos com 48 switches com os mesmos parâmetros inicialmente utilizados, mas com diferentes padrões de comunicação entre VMs: *all-to-one* and *one-to-all*. Novamente observamos (vide Figura 4.5) o mesmo comportamento observado no padrão de comunicação *all-to-all* (Figura 4.2). Em ambos experimentos, *all-to-one* (Figura 4.5(a)) e *one-to-all* (Figura 4.5(b)), obtivemos menores valores de *link stress* utilizando a estratégia (*SelecRoutes Agreg*). É importante mencionar que em todos os experimentos, Figuras 4.1, 4.2 and 4.5, foi obtida a mesma média do número de políticas de mapeamento alocadas, assim como taxa zero de requisições negadas.



(a) Link stress: Padrão de comunicação All-to-one (b) Link stress: Padrão de comunicação One-to-all

Figura 4.5: Dados de Link stress para diferentes padrões de comunicação entre VMs.

Buscando obter critérios de desempenho da arquitetura criada e do algoritmo proposto, realizamos experimentos para avaliar o tempo de mapeamento de demandas conforme o tamanho de requisições de redes virtuais. Discriminamos os tempos de operações de: pré configuração e checagem de topologias; mapeamento da rede virtual; e configuração das rotas na topologia física. Realizamos experimentos variando o tamanho das politicas de mapeamento, com matrizes de tráfego entre 1 e 16 *racks*, na topologia com 48 switches.

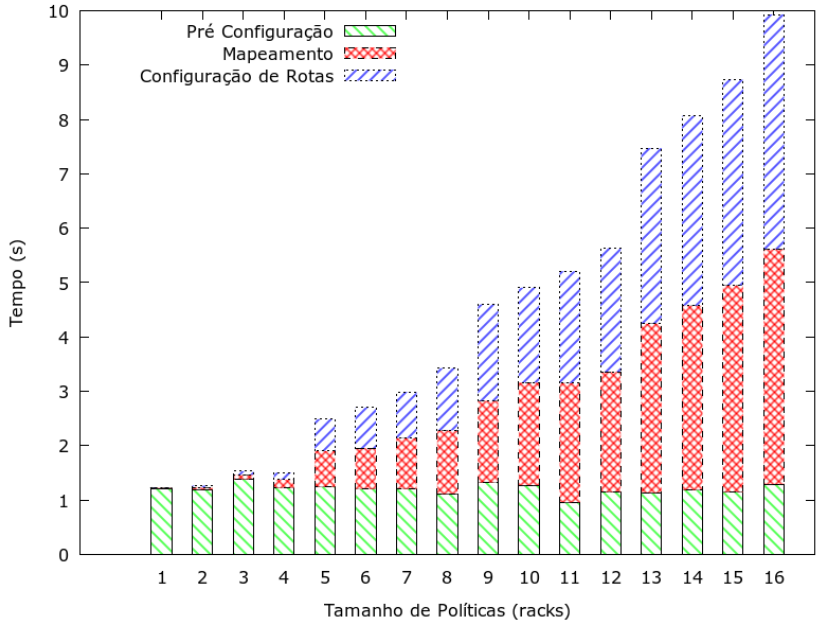


Figura 4.6: Tempos de Mapeamento

Observamos na Figura 4.6 que os tempos de pré-configuração tem definição praticamente constante por não dependerem do tamanho das políticas de mapeamento. Já os tempos de mapeamento e configuração crescem conforme o tamanho das políticas devido a quantidade de rotas a serem analisadas, selecionadas, mapeadas e configuradas nos *racks*. No caso da topologia analisada, notamos que existem pequenos saltos de tempo conforme uma política utiliza *racks* não interconectados aos mesmos switches EoR, o que se deve à necessidade de mapeamento de rotas em switches Core.

4.4 Análise dos Dados e Discussão

Notamos nos resultados obtidos que o algoritmo proposto no nosso trabalho realiza o balanceamento de carga na rede de forma eficaz quando comparado ao algoritmo "SelecRotas Tradicional", comumente visto na literatura (ex: [3] e [59]). A arquitetura construída permite um alto grau de liberdade na criação de VDCs. Toda a agregação de estado dos planos físico e virtual da arquitetura propicia ao plano de controle uma visão não conhecida nos trabalhos relacionados a esta proposta. Primeiro, pois objetivos de alto nível podem ser facilmente implementados tanto em políticas criadas conforme demandas de alocação de redes virtuais quanto na rede interna do próprio InP, seja pela programação de regras na seleção de rotas ou pela definição de configurações na própria execução do protocolo BGP no plano virtual [32]. Do mesmo modo, a permissividade de características do padrão OpenFlow 1.3 provê ao mapeamento de redes virtuais todos os parâmetros necessários para programá-las de diferentes formas, como por exemplo, definindo a seletividade de tráfego de um VDC somente por portas TCP ou diferentes protocolos de rede.

Diferente das propostas existentes na literatura, até onde temos conhecimento, a arquitetura construída neste trabalho é a única que faz utilização do padrão OpenFlow 1.3 e aborda requisitos de balanceamento de carga na rede. Além disso, não só a utilização de múltiplos caminhos, como também de critérios de resiliência de políticas, é uma abordagem até então desconhecida na literatura de DCNs. Agregar informações em grafos de topologias física e virtual em um plano de controle logicamente centralizado, permite uma diversidade de oportunidades de controle da rede e mapeamento de redes virtuais. Logo, tanto questões relacionadas a formas de endereçamento, flexibilidade da rede, desenvolvimento e criação de novas aplicações de rede, entre outras, podem ser feitas com total liberdade sobre a arquitetura proposta. Tal fato permite a abordagem de diversos problemas de DCNs pela extensão deste trabalho, considerando para isso diferentes requisitos de operação destas redes. Logo, fundamentalmente, a arquitetura criada permite que diferentes requisitos de políticas vão ao encontro de objetivos eficientes de compartilhamento de DCNs. Abordamos abaixo discussões em tópicos de interesse deste trabalho

e que se relacionam a sua continuidade para a produção de trabalhos futuros.

Mecanismos/algoritmos de roteamento: O plano virtual utilizado para a obtenção de rotas foi configurado segundo o mapeamento da topologia física do plano de dados realizando a agregação de seus elementos de rede conforme números de sistema autônomo (ASNs) do protocolo BGP atribuídos a conjuntos de elementos de rede, como em *switches* EoR e Core definidos na proposta [33]. Nesta proposta, em relação a utilização do protocolo BGP para data centers, temos dentre as vantagens que podemos citar: design prático de roteamento para grandes data centers; protocolo de simples implementação com baixa complexidade de código e fácil suporte operacional; minimização do domínio de falha de equipamentos e protocolo de roteamento; diminuição de custos operacional e de capital; e escalabilidade para scale-out da topologia de rede. Como visto na Tabela 4.1, utilizando a agregação de elementos do plano de dados, obtivemos no plano virtual menores quantidades de elementos de roteamento no plano virtual, menor quantidade de conexões entre roteadores e, consequentemente, menor *overhead* de mensagens de controle do protocolo BGP. A manutenção das mensagens de controle no plano virtual é outra vantagem obtida na arquitetura proposta neste trabalho, pois essas mensagens não correm o risco de serem descartadas por congestionamentos do plano de dados, além disso o tempo de convergência do protocolo BGP pode ser configurado de forma a responder rapidamente a falhas na rede umas vez que o overhead de mensagens de controle se mantém na topologia virtual e os tempos de envio de mensagens de verificação de sessão do BGP podem ser definidos em cada roteador virtual. Outra vantagem importante é que não somente o protocolo BGP, mas qualquer outro mecanismo pode ser implementado sobre o plano virtual para realizar o cálculo de rotas, para que estas sejam disponibilizadas conforme a topologia física de rede do plano de dados.

Esquemas de endereçamento: Da mesma forma que os mecanismos/algoritmos de roteamento descritos anteriormente, segue a ideia por traz de esquemas de endereçamento da arquitetura proposta. Em nenhum momento definimos algum esquema fixo de tradução de endereços de rede que pudesse ser imposto a sistemas finais da rede. Isto ocorreu porque as rotas calculadas pelo plano virtual somente representam caminhos na topologia física, não estando propriamente atreladas a faixas de endereço IP. Como descrevemos anteriormente na elaboração da arquitetura, fizemos uso de faixas de endereço IP para cada ToR, mas qualquer informação que pudesse ser utilizada para encaminhar pacotes a um *switch* ToR também pode ser utilizada na arquitetura construída, como a proposta Zeppelin [30]. Outro ponto importante deste trabalho, não totalmente explorado, é a capacidade de encaminhamento de tráfego a VMs não somente por endereços MAC mas também por quaisquer campos que possam definir *matches* em regras de fluxos definidas pelo padrão OpenFlow. Isto permitiria por exemplo que não somente *tags* MPLS pudessem ser escolhidas para encaminhar tráfego na rede. Logo, como apresentado nas

Tabelas 4.2 e 4.3, a quantidade de regras a serem programadas conforme o esquema proposto, principalmente nas tabelas 0 e 3 de *switches* ToR, poderiam ser minimizadas. E por fim, a utilização de mapeamentos de redes virtuais por meio da configuração de *switches* virtuais localizados em servidores diminuiria substancialmente a quantidade de regras de fluxos em *switches* ToR, pois a tradução de endereços de rede em termos de parâmetros que representem a localização de VMs na rede ficaria estabelecida em software (*hypervisors*) ao invés de hardware (ToRs).

Balanceamento de carga: Este foi o tópico de maior importância mostrado nos resultados obtidos. Os valores de *Link Stress* obtidos, utilizando o algoritmo de mapeamento de redes virtuais proposto em conjunto com a opção de seleção de rotas *SelecRotas Agreg*, mostraram o quanto os recursos de rede foram eficientemente utilizados, quando comparados à opção *SelecRotas Tradicional*. O critério utilizado como índice de balanceamento de carga na rede, se deu pela abordagem de distribuição de utilização de largura de banda entre portas de *switches* da infraestrutura física de rede por parte das redes virtuais mapeadas sobre ela. A utilização de políticas com critérios de resiliência, onde 100% das rotas disponíveis puderam ser utilizadas, propiciou o mapeamento de redes virtuais em múltiplos caminhos. Enquanto diversas propostas atuais [44, 51] buscam tratar o balanceamento de carga no serviço de comutação de pacotes em equipamento de rede, a abstração de mapeamento de redes virtuais com balanceamento de carga realizada neste trabalho apenas considera que equipamentos de rede comoditizados com suporte a OpenFlow 1.3 estejam presentes no plano de dados da arquitetura construída e, logo, possam fornecer as abstrações principais utilizadas neste trabalho, encaminhamento de tráfego por *group tables* e limitadores de largura de banda por *meter tables*. Além disso, pensamos que a utilização de técnicas de balanceamento de carga em sistemas finais, como MPTCP [46], podem ser melhoradas com a existência das redes virtuais construídas pela arquitetura proposta, pelo fato desta propiciar proativamente os requisitos e recursos necessários a existência dessas soluções de sistemas finais.

Abstrações de políticas: Como visto nos resultados experimentais o algoritmo proposto realiza balanceamento de carga eficiente na rede quando comparado ao algoritmo *SelecRotas Tradicional*, comumente visto na literatura (ex: [3] and [59]). O foco em requisitos de políticas centrados em dados traz a tona os principais aspectos do algoritmo criado. A base de conhecimento da rede estabelecida no algoritmo *Resource Bookkeeping* permite ao grafo da topologia física uma visão consistente dos recursos de rede a serem utilizados pelo *Mapping Algorithm*. Sobre este ponto de vista, a granularidade de tráfego disponível para alocação pela opção *SelecRotas Agreg* poderia ser dinamicamente ajustada para selecionar rotas de forma eficiente a diferentes aplicações, baseando-se em suas características de tráfego e prioridades (ex: Hadoop, backup, video streamming). Estas propriedades definem um campo importante de trabalhos futuros onde estatísticas oriun-

das de medições online sobre tráfego e consumo de energia possam ser correlacionadas ao algoritmo proposto para proverem balanceamento de carga na DCN [49].

Modelo de alocação de largura de banda: Até então, os trabalhos relacionados (ex: [54, 31, 41, 47, 25, 3, 59, 43]) explicitamente lidam com os *trade-offs* nas opções de alocação e competição de largura de banda entre *tenants* além de técnicas de isolamento de recursos de rede e requisitos de compartilhamento de filas. O esquema de alocação de largura de banda utilizado neste trabalho aborda o suporte de balanceamento de carga a diferentes padrões de tráfego os quais podem ser inseridos em políticas pela definição de padrões de comunicação entre aplicações. O componente *Política* não restringe a implantação de multiplexação estatística em análises de alocações de tráfego para também serem implementados sobre o plano de controle da arquitetura, para promover por exemplo, alocações de tráfego com conservação de recursos (*work conserving*) [43]. Em [26] são discutidos discernimentos que vão ao encontro das abstrações de comunicação entre sistemas finais abordadas neste trabalho. Por exemplo, as definições de políticas propostas podem ser facilmente implementadas de acordo com os modelos VOC [3] ou TAG [35] para abstrações de tráfego de aplicações e seus respectivos atributos (ex: largura de banda, esquema de endereçamento, resiliência, prioridades).

Limitações: Soluções centralizadas frequentemente envolvem aspectos de escalabilidade. Da mesma maneira, em nosso caso, o estado de informações agregadas em um plano de controle centralizado está sujeito a requisitos de desempenho conhecidos, principalmente no que diz respeito a pesquisas atuais da comunidade de SDNs. Em uma outra perspectiva, focamos em mapear com agregação a topologia física folded-Clos em uma topologia virtual com as vantagens identificadas em [33] de modo que fosse possível utilizar a plataforma RouteFlow para orquestrar estas topologias por meio do plano de controle. Um tópico interessante de pesquisa está em conduzir as pesquisas realizadas neste trabalho em quaisquer topologias de data center com múltiplos caminhos, mesmo que estas não tenham balanceamento de carga uniforme entre seus caminhos (ex: Jellyfish). Neste caso, por exemplo, um mecanismo de roteamento baseado em controladores de rede poderia ser utilizado ao invés do plano virtual arquitetado neste trabalho. E finalmente, tolerância a falhas, inserida em reconfigurações de redes virtuais é uma possibilidade de trabalho já observada em redes metropolitanas (Wire Area Networks - WANs), a qual pode se tornar uma simples extensão do algoritmo de mapeamento proposto.

OpenFlow: Diferente das propostas existentes na literatura, a arquitetura desenvolvida neste trabalho assim como a prova de conceito e experimentos desenvolvidos utiliza a versão 1.3 do padrão OpenFlow e lida com requisitos de balanceamento de carga em DCNs por meio da utilização de múltiplos caminhos e granularidades de alocação de tráfego. As características do OpenFlow 1.3 permitem todas as propriedades necessárias para programar os grafos de redes virtuais de diferentes formas, por exemplo, pela definição de

restrições de encaminhamento de tráfego de políticas por regras de fluxos em portas e protocolos TCP/IP. (ex: NVGRE, VXLAN). Da mesma forma, de acordo com estudos recentes em técnicas de espalhamento de pacotes (*packet spraying*) [17], a programação de formas de operação de filas, definidas em especificações de switches OpenFlow 1.3, poderiam também ser utilizadas em diferentes particularidades de encaminhamento de tráfego, tais como isolamento e qualidade de serviço. A avaliação de critérios de desempenho e escalabilidade, como o uso de múltiplos controladores de rede, técnicas de tradução de esquemas de endereçamento heterogêneas e outras especificações do OpenFlow 1.3, são tópicos de pesquisa que estão sendo explorados e emergem como assuntos proeminentes na literatura de SDNs.

Capítulo 5

Trabalhos Relacionados

Pesquisas recentes relacionadas à virtualização de redes de data center têm dado foco à utilização de técnicas de compartilhamento de recursos do data center, formas de encaminhamento de pacotes e de isolamento de performance da rede. É possível notar que estas diferentes abordagens feitas ao problema de alocação de redes virtuais em DCNs, resultam em *trade-offs* relacionados às imposições das configurações de rede consideradas, tais como: topologia de rede, forma de endereçamento, método de compartilhamento ou alocação de largura de banda e técnica de encaminhamento de pacotes.

Em [36] a utilização do *Spanning Tree Protocol* (STP) é feita para prover a utilização de múltiplos caminhos em topologias arbitrárias de DCNs. *Smart Path Assignment in Networks* (SPAIN) faz uso de equipamentos de rede de baixo custo para calcular caminhos disjuntos entre pares de *switches* de borda utilizando VLANs. Agentes instalados em sistemas finais realizam o encaminhamento de tráfego a estes caminhos/VLANs. Um agente, com base em informações de toda a rede, realiza o balanceamento de carga alternando o encaminhamento de tráfego entre diferentes caminhos. Ele também detecta falhas em diferentes caminhos e reencaminha o tráfego para outras VLANs.

Oktopus [3] é uma implementação que busca oferecer a *tenants* (conjuntos de VMs) garantias mínimas de largura de banda por meio de duas abstrações de redes virtuais, *virtual cluster* e *virtual oversubscribed cluster*. Estas são propostas a fim de controlar os *trade-offs* entre garantias de largura de banda para *tenants*, seus respectivos custos e o lucro do provedor de serviço. Um *virtual cluster* proporciona às VMs de uma *tenant* a transparência da interconexão em um mesmo *switch* sem nenhum fator de *oversubscription*. Já *virtual oversubscribed cluster* proporciona a interconexão de VMs por meio de camadas de *switches* com um determinado fator de *oversubscription*, o qual pode ser escolhido pela aplicação.

A proposta SecondNet [25] foca em dar garantias de largura de banda com requisitos de qualidade de serviço entre VMs em um ambiente de múltiplas *tenants* sobre um data center

virtualizado. A técnica definida como *Port-Switching Source Routing* (PSSR) realiza o encaminhamento de pacotes com o cálculo de rotas sendo feito na origem. A reserva de largura de banda é feita em *hypervisors* de servidores. SecondNet permite a adição e remoção de VMs e alterações de requisitos de largura de banda dinamicamente, e com a utilização de migração de VMs, consegue tratar falhas e reduzir a fragmentação de recursos do data center.

Em Gatekeeper [47] é proposta uma abordagem que tem como critérios fundamentais: a escalabilidade do isolamento de performance da rede em termos da quantidade de VMs; previsibilidade de desempenho da rede; robustez contra comportamentos maliciosos de *tenants*; e flexibilidade com relação a garantias mínima e máxima de desempenho da rede oferecidas às *tenants*. A garantia de alocação de largura de banda para múltiplas *tenants* em um data center é feita por meio de interfaces de rede (Network Interface Cards - NICs) virtuais em *switches* lógicos que realizam o ajuste do envio de dados conforme o congestionamento da rede dado por medições em VMs.

O trabalho [8] propõe uma arquitetura de redes virtuais CloudNaaS para implementar e gerenciar aplicações empresariais na nuvem. Nessa arquitetura tais aplicações podem ter requisitos de endereçamento, suporte a *middleboxes*, *broadcasting*, agrupamento de VMs e reserva de largura de banda. Essa proposta considera a existência de suporte ao padrão OpenFlow em sistemas finais e diversas técnicas de encaminhamento de dados para tratar o número de entradas em tabelas de encaminhamento de pacotes nos elementos de rede. Além disso, tem suporte a técnicas de tratamento de falhas online e a alterações em especificações de políticas de rede para rearranjar o mapeamento de VDCs.

Seawall [54] define um esquema que permite a provedores de infraestrutura definir como a alocação de largura de banda será compartilhada entre múltiplos *tenants*. Pesos são assimilados a entidades geradoras de tráfego (ex: servidores, aplicações, VMs) e utilizados para alocação de largura de banda de forma proporcional. A alocação ocorre em cada entidade por meio de um filtro NDIS (Network Driver Interface Specification) responsável por enviar pacotes a uma taxa proporcional ao peso determinado a entidade. Dessa maneira, o isolamento entre múltiplos *tenants* é garantido por Seawall, assim como a inexistência de *tenants* com comportamento malicioso. Além disso, os pesos podem ser dinamicamente assimilados as entidades conforme alterações nos requisitos de largura de banda, como também modificados conforme alterações na rede, como mudanças na topologia ou falhas.

A proposta NetShare [31] trata o problema de alocação de largura de banda por data centers virtuais sem a necessidade de modificações em elementos da rede. As alocações de largura de banda são feitas de forma proporcional entre os *tenants* e de maneira eficiente para o provedor de infraestrutura. NetShare pode ser implementado com três técnicas: *group allocation* para tratar fluxos TCP; *rate throttling* a fim de ajustar taxas de envio de

fluxos UDP; e *centralized allocation* para alocar largura de banda não utilizada a fluxos específicos. O tratamento de falhas é confiado a protocolos de roteamento e o suporte a *multipath* ocorre com o uso do mecanismo ECMP.

Proteus [59] é uma abordagem que busca flexibilizar a utilização de recursos de rede por múltiplos *tenants* por meio de técnicas de definição de perfis de utilização temporal de largura de banda. Esta dimensão temporal de alocação de largura de banda aumenta a eficiência de sua utilização pelo provedor de infraestrutura. A ideia da alocação espacial e temporal de *clusters* virtuais (Time Interleaved Virtual Clusters - TIVCs), definidos como um agregado de VMs e enlaces virtuais com requisitos de largura de banda, ocorre por meio da seleção de perfis de configuração por clientes os quais são gerados conforme características de comunicação dos TIVCs. Nesse trabalho, políticas associadas a conjuntos de fluxos de pacotes de *tenants* são consideradas na alocação de largura de banda feita por elementos de rede e o suporte a *multipath* não é tratado.

ElasticSwitch [43] propõe um modelo de alocação semelhante a Proteus. No entanto, além de realizar a alocação com recursos mínimos de largura de banda (*guarantee partitioning*), ainda propicia que *tenants* utilizem o máximo de largura de banda disponível (*rate allocation*) caso que não esteja sendo usada *work conserving*. Por ser baseada em *switches* virtuais de *hypervisors*, essa solução se torna prática para se adequar a grandes data centers já existentes e se destaca por ser distribuída e não requerer comunicação com um controlador de rede centralizado, portanto, sem problemas de escalabilidade.

Hadrian [4] realiza a alocação de garantias mínimas de largura de banda para comunicação *inter-tenants* considerando custos proporcionais às alocações de cada *tenant*. Argumentando a existência e importância de tráfego entre *tenants*, e alocações de tráfego proporcionais a compartilhamentos de largura de banda por diferentes *tenants*, nesse trabalho é proposto Hadrian, um *framework* para alocação e colocação de máquinas virtuais com base em limites superiores de alocação de largura de banda para *tenants* e garantias mínimas de tráfego para VMs. Dentre as conclusões desse trabalho é mostrado que o compartilhamento robusto e proporcional de largura de banda a *tenants* deve estar relacionado ao custo de alocação de cada *tenant*.

Em [5] são analisadas diferentes técnicas de virtualização de redes de data center bem como as propostas já realizadas sobre este assunto na literatura. Nesse trabalho diferentes parâmetros são levados em consideração para estabelecer comparações entre as soluções e arquiteturas existentes. A Tabela 5.1 abaixo estende estende tais parâmetros para compará-los às principais contribuições deste trabalho.

Na Tabela 5.1, os trabalhos relacionados a esta dissertação são comparados perante diferentes características. Definimos como abstração topológica, a capacidade da solução apresentar formas abstrair os padrões de comunicação entre as entidades (ex: aplicações, VMs, servidores) de uma *tenant* de modo que seja possível construir alguma forma de

Tabela 5.1: Tabela Comparativa de Trabalhos Relacionados e Arquitetura Construída

Proposta	Características										
	Abstração Topológica	Encaminhamento	Largura de banda	Multipathing	Políticas	Balanceamento de carga	QoS	Tolerância a falhas	Escalabilidade	Controle	Topologia
SPAIN [36]	✔	✖	✖	✔	✖	✔	✖	✔	Baixa	Centralizado	Árvore
Oktopus [3]	✖	✖	Garantida	✖	✖	✖	✔	✔	Alta	Centralizado	Qualquer
SecondNet [25]	✖	✔ (PSSR)	Garantida	✖	✖	✖	✔	✔	Alta	Centralizado	Qualquer
Gatekeeper [47]	✖	✖	Garantida	✖	✖	✖	✔	✔	Alta	Distribuído	Core sem congestionamento
CloudNaaS [8]	✖	✔ (OpenFlow)	Garantida	✖	✔	✖	✔	✔	Baixa	Centralizado	Qualquer
Seawall [54]	✖	✖	Compartilhada	✖	✔	✖	✖	✔	Alta	Distribuído	Qualquer
NetShare [31]	✖	✖	Compartilhada	✔	✖	✔	✖	✔	Baixa	Distribuído	Qualquer
Proteus [59]	✖	✖	Garantida	✖	✖	✖	✖	✔	Baixa	Centralizado	Qualquer
ElasticSwitch [43]	✖	✖	Compartilhada	✔	✖	✖	✖	✔	Alta	Distribuído	Qualquer
Hadrian [4]	✖	✖	Garantida	✖	✖	✖	✖	✔	Baixa	Centralizado	Árvore
Este Trabalho [48]	✔	✔ (OpenFlow 1.3)	Garantida	✔	✔	✔	✔	✔	Baixa	Centralizado	Árvore

representação sobre tal comunicação (ex: clusters, grafos). Por encaminhamento tratamos propostas que definem formas diferentes de encaminhamento como *Port-Switching Source Routing* (PSSR) definido em SecondNet [25] ou OpenFlow por CloudNaaS [8]. Nesse aspecto, a maioria das propostas apenas considera que formas de encaminhamento já existam na infraestrutura de rede do data center (ex: VLB, ECMP). A respeito da forma de alocação de largura de banda, as propostas estão subdivididas em garantida e compartilhada, sendo a primeira definida por regras estritas de particionamento de largura de banda entre tenants, e a segunda por alguma forma de compartilhamento, justo ou por multiplexação estatística.

Da mesma maneira que formas de encaminhamento, existem propostas que definem explicitamente premissas de utilização de múltiplos caminhos (*multipathing*). A definição de políticas em alguma forma de aplicar parâmetros, como esquema de endereçamento e diferenciação de tráfego, a *tenants* é abordado para comparar o quão customizáveis podem ser as soluções relacionadas a este trabalho para tratarem a alocação de redes virtuais. O balanceamento de carga, fortemente associado a *multipathing*, define se a solução possui alguma forma de alocar largura de banda conforme a carga de trabalho dos enlaces da rede. A maioria das soluções apenas avalia que técnicas de encaminhamento (ex: VLB, ECMP) ou controle de congestionamento final (ex: DCTCP, MTCP) existam e possam ser utilizadas para balancear o tráfego de rede.

As propostas que tratam qualidade de serviço estão associadas a alocações garantidas de largura de banda, pois definem que estas possam permitir a *tenants* variações de requisitos de diferenciação de tráfego para atender as garantias requisitadas. Com relação a tolerância a falhas, vemos que todas as propostas apresentam alguma forma de suportar esta característica, seja por restabelecimento de caminhos, utilização de *multipathing* ou por possuírem controle distribuído. A característica de controle define se a construção e instanciação de redes virtuais ocorre por uma entidade centralizada, como no caso de nossa proposta, ou distribuído, como no caso de ElasticSwitch onde características de controle são implementadas em *switches* virtuais de sistemas finais. Esta última está bem associada a questão de escalabilidade, onde muitas vezes uma solução centralizada possui baixa escalabilidade ao contrário de uma solução distribuída. No caso da arquitetura desenvolvida neste trabalho, como discutido no capítulo anterior, ela possui limitações de baixa escalabilidade advindas de SDNs, as quais estão sendo pesquisadas pela comunidade acadêmica. No entanto, há soluções centralizadas que possuem alta escalabilidade por fazerem uso de técnicas que não demandam uma grande quantidade de recursos do controle centralizado, como no caso das soluções Oktopus e SecondNet. E por fim, há propostas que restringem a utilização de topologias particulares, como no formato de árvore (ex: Clos, FatTree), e outras que são independentes desta característica.

Capítulo 6

Conclusões

Neste trabalho propomos e construímos uma arquitetura para o aprovisonamento de redes virtuais em redes de data center. Com base nesta tarefa, tivemos como objetivos secundários construir um plano virtual de roteamento eficientemente planejado por meio do uso do protocolo BGP programado em roteadores virtuais, e a definição de um plano de dados com programação eficiente de regras de fluxos. A topologia virtual foi construída com base no mapeamento de elementos de rede do plano de dados, propiciando assim eficiência na determinação da quantidade de roteadores virtuais e, consequentemente, na quantidade de mensagens de controle oriundas do protocolo BGP. No plano de dados realizamos a construção de topologias de rede folded-Clos com elementos de rede com suporte ao padrão OpenFlow 1.3, dando todo suporte às abstrações de alocações de redes virtuais por características como *group* e *meter tables*. No plano de controle da arquitetura, propomos um conjunto de componentes os quais definem uma série de tarefas para a agregação de informações dos planos virtual e de dados, de modo a possibilitarem a inserção de requisitos de redes virtuais para que estas fossem construídas e alocadas almejando o balanceamento de carga da rede de data center.

Com base na construção da arquitetura proposta, buscamos avaliar a alocação de redes virtuais em data centers utilizando o modelo de *Network-as-a-Service* com a aplicação de conceitos de Redes Definidas por Software. Construímos a arquitetura utilizando a plataforma RouteFlow a fim de moldar todo o ambiente de rede, assim como os requisitos de operação de uma arquitetura de data center. Por meio de algoritmos implementados sobre o plano de controle da plataforma RouteFlow, conseguimos mostrar eficiência no aprovisionamento de redes virtuais considerando os fatores de utilização de largura de banda e balanceamento de carga da rede. Como analisados nos experimentos realizados também obtivemos eficiência na utilização de tabelas de equipamentos de rede do plano de dados, além de cálculo eficiente de rotas por meio de uma topologia virtual construída com base na agregação lógica de componentes do plano de dados.

A partir dos resultados obtidos, levantamos e analisamos uma série de questionamentos sobre tópicos de pesquisa relacionados ao trabalho realizado e da literatura de redes de data center associados a SDNs, balanceamento de carga, esquemas de endereçamento, mecanismos de roteamento, os modelos de alocações de largura de banda na rede e as principais abstrações por traz do conceito de políticas utilizado neste trabalho. Além disso, com base no estado da arte em pesquisas relacionadas a DCNs e SDNs expomos as principais limitações da arquitetura construída, mas também como ela pode ter suas funções melhoradas para se adequar aos requisitos de operação de DCNs. Por fim, apresentamos os principais trabalhos relacionados à arquitetura proposta e construída, descrevendo os principais pontos chave associados aos trabalhos presentes na literatura com base principalmente no parâmetro de alocação de redes virtuais em redes de data center.

Vemos neste trabalho que muitos desafios surgem da aplicação de conceitos de SDN em DCNs, principalmente pelo uso do conceito de *Network-as-a-Service*. A avaliação de critérios de desempenho e escalabilidade em DCNs, utilizando novas formas de endereçamento, mapeamento de VMs em servidores, outras funcionalidades do OpenFlow 1.3, são tópicos de pesquisa que já estão em andamento. Futuramente, iremos estender o funcionamento da plataforma RouteFlow para suportar tolerância a falhas em todos os seus planos, o que irá prover à arquitetura proposta um alto grau de flexibilidade e escalabilidade na alocação de redes virtuais. Buscaremos também elaborar novos algoritmos que levem em consideração requisitos de gerenciamento de energia no que diz respeito à utilização de recursos de rede e como estes interesses podem ir de encontro às abordagens que tratam resiliência associada ao mapeamento de redes virtuais em data centers. Entendemos que, por exemplo, a seleção de rotas no mapeamento de redes virtuais poderia ser estabelecida de diversas formas, considerando pesos de enlaces, políticas de roteamento interno, ou mesmo critérios de utilização de energia em enlaces e resiliência de caminhos. Não obstante, vemos que os nós de entrada do algoritmo proposto podem ser estendidos a servidores, tornando esta solução flexível à programação de *switches* virtuais.

Por fim, de forma condizente com os principais trabalhos relacionados a este, vemos que o estado da arte proposto para o aprovisionamento de redes virtuais em DCNs está bem consolidado. No entanto, no que diz respeito a sua elaboração com base em conceitos do paradigma de SDNs, ainda teremos muitas propostas na literatura decorrentes das inúmeras abstrações de requisitos de redes virtuais a serem suportadas no plano de controle da rede. A evolução de SDNs caminha para tratar problemas advindos de sua própria criação, tais como escalabilidade, desempenho e tolerância a falhas. Assim como novas propostas de topologias de rede e arquiteturas de DCNs surgem na literatura, nos próximos anos, com o grande crescimento do tráfego em redes de data center proveniente da computação em nuvem e de dispositivos móveis, nosso trabalho paira como uma contribuição neste vasto cenário de novas tecnologias da Internet e das redes de comunicação.

Referências Bibliográficas

[1] M. Abu Sharkh, A. Ouda, and A. Shami. A resource scheduling model for cloud computing data centers. In *9th International Wireless Communications and Mobile Computing Conference (IWCMC), 2013*, pages 213–218, July 2013.

[2] S. Azodolmolky, P. Wieder, and R. Yahyapour. Cloud computing networking: challenges and opportunities for innovations. *IEEE Communications Magazine*, 51(7):54–62, July 2013.

[3] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. In *Proc. of the ACM SIGCOMM 2011 Conference on Data Communication*, pages 242–253, New York, NY, USA, 2011. ACM.

[4] Hitesh Ballani, Keon Jang, Thomas Karagiannis, Changhoon Kim, Dinan Gunawardena, and Greg O'Shea. Chatty tenants and the cloud network sharing problem. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, pages 171–184, Berkeley, CA, USA, 2013. USENIX Association.

[5] M.F. Bari, R. Boutaba, R. Esteves, L.Z. Granville, M. Podlesny, M.G. Rabbani, Qi Zhang, and M.F. Zhani. Data center network virtualization: A survey. *IEEE Commun. Surveys Tuts.*, 15(2):909–928, 2013.

[6] Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pages 267–280, New York, NY, USA, 2010. ACM.

[7] Theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. Cloudnaas: A cloud networking platform for enterprise applications. In *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, pages 8:1–8:13, New York, NY, USA, 2011. ACM.

[8] Theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. Cloudnaas: A cloud networking platform for enterprise applications. In *Proc. of the ACM SOCC '11*, pages 8:1–8:13, New York, NY, USA, 2011. ACM.

[9] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Understanding data center traffic characteristics. *SIGCOMM Comput. Commun. Rev.*, 40(1):92–99, January 2010.

[10] Jorge Carapinha and Javier Jiménez. Network virtualization: A view from the bottom. In *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, pages 73–80, New York, NY, USA, 2009. ACM.

[11] Martin Casado, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Rethinking packet forwarding hardware. In *Proc. of the ACM HotNets '2008*, pages 22:1–22:6, New York, NY, USA, 2008. ACM.

[12] Martin Casado, Teemu Koponen, Rajiv Ramanathan, and Scott Shenker. Virtualizing the network forwarding plane. In *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, pages 8:1–8:6, New York, NY, USA, 2010. ACM.

[13] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, April 2010.

[14] Cisco Analysis. Cisco Global Cloud Index: Forecast and Methodology, 2012–2017. 2013.

[15] Paolo Costa, Matteo Migliavacca, Peter Pietzuch, and Alexander L. Wolf. Naas: Network-as-a-service in the cloud. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, pages 1–1, Berkeley, CA, USA, 2012. USENIX Association.

[16] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. Devoflow: Scaling flow management for high-performance networks. In *Proceedings of the ACM SIGCOMM 2011 Conference on Data Communication*, pages 254–265, New York, NY, USA, 2011. ACM.

[17] A. Dixit, P. Prakash, Y.C. Hu, and R.R. Kompella. On the impact of packet spraying in data center networks. In *Proceedings of the IEEE INFOCOM*, pages 2130–2138, Abril 2013.

[18] D. Dudkowski, B. Tauhid, G. Nunzi, and M. Brunner. A prototype for in-network management in naas-enabled networks. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 81–88, Maio 2011.

[19] N. Farrington and A. Andreyev. Facebook's data center network architecture. In *IEEE Optical Interconnects Conference*, pages 49–50, Maio 2013.

[20] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn. *Queue*, 11(12):20–40, December 2013.

[21] A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, 15(4):1888–1906, Fourth 2013.

[22] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: A scalable and flexible data center network. *Communications of the ACM*, 54(3):95–104, March 2011.

[23] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54, October 2005.

[24] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, pages 63–74, New York, NY, USA, 2009. ACM.

[25] Chuanxiong Guo, Guohan Lu, Helen J. Wang, Chao Yang, Shuang e Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. Secondnet: A data center network virtualization architecture with bandwidth guarantees. In *Proceedings of Co-NEXT '10*, pages 15:1–15:12, New York, NY, USA, 2010. ACM.

[26] Sangjin Han, Norbert Egi, Aurojit Panda, Sylvia Ratnasamy, Guangyu Shi, and Scott Shenker. Network support for resource disaggregation in next-generation datacenters. In *Proc. of the HotNets '2013*, pages 10:1–10:7, New York, NY, USA, 2013. ACM.

[27] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. Reproducible network experiments using container-based emulation. In *Proc. of the CoNEXT '12*, pages 253–264, New York, NY, USA, 2012. ACM.

[28] R. Jain and S. Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, 51(11):24–31, November 2013.

[29] Eric Keller and Jennifer Rexford. The "platform as a service"model for networking. In *Proceedings of INM/WREN'10*, pages 4–4, Berkeley, CA, USA, 2010. USENIX Association.

[30] J. Kempf, Ying Zhang, R. Mishra, and N. Beheshti. Zeppelin - a third generation data center network virtualization technology based on sdn and mpls. In *IEEE 2nd International Conference on Cloud Networking (CloudNet)*, pages 1–9, Novembro 2013.

[31] Vinh The Lam, Sivasankar Radhakrishnan, Rong Pan, Amin Vahdat, and George Varghese. Netshare and stochastic netshare: Predictable bandwidth allocation for data centers. *SIGCOMM Comput. Commun. Rev.*, 42(3):5–11, June 2012.

[32] P. Lapukhov and E. Nkposong. Centralized routing control in bgp networks using link-state abstraction. Internet-Draft draft-lapukhov-bgp-sdn-00.txt, IETF Secretariat, 2013.

[33] P. Lapukhov, A. Premji, and Ed. J. Mitchell. Use of bgp for routing in large-scale data centers. Internet-Draft draft-lapukhov-bgp-routing-large-dc-06.txt, IETF Secretariat, August 2013.

[34] A. Lara, A. Kolasani, and B. Ramamurthy. Network innovation using openflow: A survey. *IEEE Communications Surveys Tutorials*, 16(1):493–512, First 2014.

[35] J. Lee, M. Lee, L. Popa, Y. Turner, P. Sharma, and B. Stephenson. Cloudmirror: Application-aware bandwidth reservations in the cloud. In *HotCloud'13*, 2013.

[36] Jayaram Mudigonda, Praveen Yalagandula, Mohammad Al-Fares, and Jeffrey C. Mogul. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of the USENIX NSDI'10*, pages 18–18, Berkeley, CA, USA, 2010. USENIX Association.

[37] Marcelo R Nascimento, Christian Rothenberg, Rodrigo R Denicol, Marcos R Salvador, and Maurıcio F Magalhaes. Routeflow: Roteamento commodity sobre redes programáveis. *XXIX Simpósio Brasileiro de Redes de Computadores and Sistemas Distribuıdos-SBRC*, 2011.

[38] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. Portland: A scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, pages 39–50, New York, NY, USA, 2009. ACM.

[39] Open Network Foundation. Openflow switch specification version 1.3.0 (wire protocol 0x04). Technical report, ONF, 2012.

[40] Ben Pfaff, Justin Pettit, Keith Amidon, Martin Casado, Teemu Koponen, and Scott Shenker. Extending networking into the virtualization layer. In *HotNets*. ACM SIGCOMM, 2009.

[41] Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: Sharing the network in cloud computing. In *Proc. of the ACM HotNets '2011*, pages 22:1–22:6, New York, NY, USA, 2011. ACM.

[42] Lucian Popa, Sylvia Ratnasamy, Gianluca Iannaccone, Arvind Krishnamurthy, and Ion Stoica. A cost comparison of datacenter network architectures. In *Proceedings of the 6th International COnference on emerging Networking EXperiments and Technologies*, pages 16:1–16:12, New York, NY, USA, 2010. ACM.

[43] Lucian Popa, Praveen Yalagandula, Sujata Banerjee, Jeffrey C. Mogul, Yoshio Turner, and Jose Renato Santos. Elasticswitch: Practical work-conserving bandwidth guarantees for cloud computing. In *Proc. of the ACM SIGCOMM '2013*, pages 351–362, New York, NY, USA, 2013. ACM.

[44] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Integrating microsecond circuit switching into the data center. *SIGCOMM Comput. Commun. Rev.*, 43(4):447–458, August 2013.

[45] M.G. Rabbani, R. Pereira Esteves, M. Podlesny, G. Simon, L. Zambenedetti Granville, and R. Boutaba. On tackling virtual data center embedding problem. In *IFIP/IEEE IM 2013*, pages 177–184, 2013.

[46] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. Improving datacenter performance and robustness with multipath tcp. In *Proceedings of the ACM SIGCOMM 2011 Conference on Data Communication*, pages 266–277, New York, NY, USA, 2011. ACM.

[47] Henrique Rodrigues, Jose Renato Santos, Yoshio Turner, Paolo Soares, and Dorgival Guedes. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *Proceedings of WIOV'11*, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association.

[48] Raphael Rosa, Christian Esteve Rothenberg, and Edmundo Madeira. Redes virtuais de data center mapeadas como serviço em redes definidas por software. In *XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Maio 2014.

[49] Raphael Rosa, Christian Esteve Rothenberg, and Edmundo Madeira. Virtual data center networks embedding through software defined networking. In *IEEE Network Operations and Management Symposium (NOMS) 2014*, Maio 2014.

[50] Christian Esteve Rothenberg, Marcelo Ribeiro Nascimento, Marcos Rogerio Salvador, Carlos Nilton Araujo Corrêa, Sidney Cunha de Lucena, and Robert Raszuk. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In *Proc. of HotSDN '12*, pages 13–18, New York, NY, USA, 2012. ACM.

[51] Siddhartha Sen, David Shue, Sunghwan Ihm, and Michael J. Freedman. Scalable, optimal flow routing in datacenters via local link balancing. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, pages 151–162, New York, NY, USA, 2013. ACM.

[52] M. Sharkh, M.A. e Jammal, A. Shami, and A. Ouda. Resource allocation in a network-based cloud computing environment: design challenges. *IEEE Communications Magazine*, 51(11):46–52, November 2013.

[53] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. Can the production network be the testbed? In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association.

[54] Alan Shieh, Srikanth Kandula, Albert Greenberg, and Changhoon Kim. Seawall: Performance isolation for cloud datacenter networks. In *Proc. of the HotCloud '10*, pages 1–1, Berkeley, CA, USA, 2010. USENIX Association.

[55] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers randomly. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, pages 17–17, Berkeley, CA, USA, 2012. USENIX Association.

[56] Arsalan Tavakoli, Martin Casado, Teemu Koponen, and Scott Shenker. Applying nox to the datacenter. In *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)*, 2009.

[57] Fabio Luciano Verdi, Christian Esteve Rothenberg, Rafael Pasquini, and Mauricio Magalhaes. Novas arquiteturas de data center para cloud computing. In *28th Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) 2010 - Minicursos*, May 2010.

[58] Anjing Wang, M. Iyer, R. Dutta, G.N. Rouskas, and I. Baldine. Network virtualization: Technologies, perspectives, and frontiers. *Journal of Lightwave Technology*, 31(4):523–537, Feb 2013.

[59] Di Xie, Ning Ding, Y. Charlie Hu, and Ramana Kompella. The only constant is change: Incorporating time-varying network reservations in data centers. In *Proceedings of SIGCOMM '12*, pages 199–210, New York, NY, USA, 2012. ACM.

[60] M.F. Zhani, Qi Zhang, G. Simon, and R. Boutaba. Vdc planner: Dynamic migration-aware virtual data center embedding for clouds. In *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 18–25, Maio 2013.