Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo comparativo de desempenho de infraestrutura de arquiteturas de rede intra datacenter

Elvio de Sousa

Dissertação apresentada como requisito para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Eduardo A. P. Alchieri

Brasília
2020

de Sousa, Elvio
dES82e Estudo comparativo de desempenho de infraestrutura de arquiteturas de rede intra datacenter / Elvio de Sousa; orientador Eduardo A. P. Alchieri. -- Brasília, 2020.
59 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2020.

1. Estudo comparativo de desempenho de infraestrutura de arquiteturas de rede intra datacenter. I. A. P. Alchieri, Eduardo , orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo comparativo de desempenho de infraestrutura de arquiteturas de rede intra datacenter

Elvio de Sousa

Dissertação apresentada como requisito para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Eduardo A. P. Alchieri (Orientador)
CIC/UnB

Prof. Dr. Luiz Antonio Rodrigues
Unioeste

Prof.a Dr.a Priscila Solis Barreto
UnB

Prof.a Dr.a Aletéia Patrícia Favacho de Araújo
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 20 de janeiro de 2020

Dedicatória

Primeiramente a Deus, por todas as conquistas e pela fortaleza que tem sido em minha vida. A toda a minha família, especialmente à minha querida esposa, Cleideane Santos; ao meu pai, Geraldo Sousa; à minha mãe, Vani Sousa; e aos meus filhos, Kevin e Enzo, por todo o amor, compreensão e apoio ao longo dessa trajetória. A todos eles dedico este estudo por compartilharem do meu esforço e por compreenderem as (muitas) horas de afastamento para me dedicar à pesquisa.

Agradecimentos

Agradeço ao meu orientador e amigo, Prof. Dr. Eduardo Adilo Pelinson Alchieri, pela confiança, pela paciência e pela serenidade. Também ao Prof. Dr. Marcelo Marotta, que, com o Prof. Dr. Marcos Fagundes, sugeriu e modificou a abordagem do trabalho.

Agradeço ao Brian Lebiednik, um dos autores do artigo de pesquisa *Evaluation of Data Center Network Topologies* [1], por ter me ajudado a evoluir com os códigos e **a** entender o emulador *Mininet*. De igual forma, agradeço aos meus parceiros, alunos do mestrado e amigos do trabalho, por todo o apoio prestado.

Agradeço especialmente ao PPCA (Programa de Pós-Graduação em Computação Aplicada) e à Universidade de Brasília - UnB pelo conhecimento e pelo apoio ao longo de toda a trajetória acadêmica.

Resumo

Os *datacenters* de grande escala formam o principal suporte de infraestrutura para os serviços em constante expansão, portanto, as características de desempenho e custo têm um impacto significativo nas instituições que necessitam estes serviços de tecnologia da informação. Os *datacenters* estão se tornando cada vez mais populares com a sua flexibilidade e capacidade de processamento no moderno ambiente de computação. Eles são gerenciados e monitorados para permitir o provisionamento dinâmico de recursos, otimização de desempenho, bem como para permitir a utilização eficiente de recursos disponíveis. Cada *datacenter* consiste em computação massiva, recursos de rede e armazenamento conectados por cabos. A natureza em larga escala dos *datacenters* requer cuidadoso planejamento de computação, armazenamento, nós de rede, interconexão, assim como a intercomunicação para a operação eficaz e eficiente entre os ativos da infraestrutura.

Nesse contexto, foi realizado um significativo trabalho de pesquisa por Reyes e Bauschert em 2018 [2] sobre os custos de arquiteturas de rede em *datacenters* , a fim de elencar as *Data Center Networks (DCNs)* de menor custo financeiro. Tomando esse trabalho de custo das arquiteturas *DCN Leaf-Spine, Fat-tree, Hybrid Fat-tree, Facebook 4-post e Facebook New Fabric* como base, nossa pesquisa visa medir o desempenho dessas DCNs em diferentes cenários. O desempenho destas cinco arquiteturas foi analisado em relação ao atraso e à taxa de transferência, já que no estudo feito por Reyes e Bauschert, apenas o custo monetário foi considerado. Inicialmente, apresentamos a pesquisa em várias topologias DCNs representativas, com comparações e várias propriedades para destacar suas vantagens e desvantagens. A variedade de topologias DCNs leva a uma forte necessidade de um método padronizado para suas avaliações e comparações. Desta forma, usamos o *Mininet*, um simulador de uso geral que suporta as arquiteturas DCNs mais conhecidas e atuais. Este simulador modular e flexível permite fácil extensão para suportar as topologias e as métricas propostas. Com o apoio do *Mininet*, apresentamos o desempenho das avaliações e comparações objetivas das métricas de latência (atraso) e taxa de transferência das cinco arquiteturas DCNs acima mencionadas. Ao final desse trabalho, obtivemos como resultado que as arquiteturas baseadas em *Leaf-Spine* tiveram melhor desempenho, portando nosso estudo sugere que as arquiteturas *Leaf-Spine* tem o melhor custo benefício.

Palavras-chave: Avaliação de Desempenho, Arquitetura de *Datacenter*, Topologia de Rede

Abstract

Large-scale datacenters form the primary infrastructure support for ever-expanding services, so performance and cost characteristics have a significant impact on those IT services-requiring institutions. Datacenters are becoming increasingly popular with their flexibility and processing power in the modern computing environment. They are managed and monitored to enable dynamic resource provisioning, performance optimization as well as efficient utilization of available resources. Each *datacenter* consists of massive computing, network resources and wired storage. The large-scale nature of *datacenters* requires careful planning of computing, storage, network nodes, interconnection as well as intercommunication for their effective and efficient operation among infrastructure assets.

In this context, significant research was done by Reyes and Bauschert in 2018 [2] regarding the cost of *datacenter* network architectures in order to define the *Data Center Network* (DCN) with lower financial costs. Based on this work, that considers five DCNs architectures (Leaf-Spine, Fat-tree, Hybrid Fat-tree, Facebook 4-post, and FacebookNew Fabric) our research aims to measure their performance. We will verify the performance of the five architectures in relation to delay and throughput, since in the study by Reyes and Bauschert only the monetary cost is considered. First, we present the research of several representative DCN topologies, along with comparisons and various properties to highlight their advantages and disadvantages. The variety of DCN topologies leads to a strong need for a standardized method for evaluating and comparing various DCN architectures. Thus, we used Mininet, a general purpose simulator that supports the most well-known and current DCN architectures. This simulator is modular and flexible, allowing easy extension to support the proposed computation topologies and metrics. With Mininet's support, we present the performance of objective evaluations and comparisons of the delay and bandwidth metrics of the five DCN architectures aforementioned. At the end of this work, we obtained the result that the architectures based on Leaf-Spinet had better performance, therefore our study leads us to believe that the Leaf-Spinet architectures have the best cost-benefit trade-off.

Keywords: *Performance Evaluation*, *Datacenter Architecture*, *Network Topology*

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ACL *Access Control List.*

ARP *Address Resolution Protocol.*

CCN *Network Centric Networking.*

CCNA *Cisco Certified Network Associate.*

CCNP *Cisco Certified Network Professional.*

CPU *Central Processing Unit.*

DCN *Data Center Network.*

DCNs *Data Center Networks.*

ECMP *Equal Cost Multipath Load Balance.*

EIGRP *Enhanced Interior Gateway Routing Protocol.*

IEEE Instituto de Engenheiros Eletricistas e Eletrônicos.

IETF *Internet Engineering Task Force.*

LAN *Local Area Network.*

MEMS Sistemas Microeletromecânicos.

OCS *Optical Circuit Switching.*

ONF *Open Networking Foundation.*

OSPF *Open Shortest Pathe First.*

OVS *Open vSwitch.*

PoD *Proof of delivery.*

POE *Power over Ethernet.*

QOS *Quality of Service.*

RFC *Request for Comments.*

S.O. Sistema Operacional.

SDN *Software Defined Networ*k.

STF Superior Tribunal Federal.

STP *Spanning Tree Protocol.*

TCP *Transmission Control Protocol.*

TI Tecnologia da Informação.

VLAN *Virtual LAN.*

VLT *Virtual Link Trunking.*

VLTi *Virtual Link Trunking interconnect.*

VOIP *Voice over Internet Protocol.*

WAAS *Wide Area Application Services.*

WAN *Wide Área Network.*

Capítulo 1

Introdução

1.1 Considerações Iniciais

Um *datacenter* ou centro de dados é uma instalação utilizada para armazenar sistemas e componentes associados, tais como telecomunicações e sistemas de armazenamento [21]. Eles são facilitadores-chave para a computação em nuvem, para fornecer *software* como serviço (SaaS) e infraestrutura como serviço (IaaS), para serviços on-line da Web, para computação de *big data*, para grandes simulações, dentre outros. Atualmente, uma *Data Center Network* (DCN) contém milhares de nós de computação com requisitos significativos de largura de banda de rede. Empresas como Amazon, Google e Microsoft estão construindo grandes *datacenter* para computação em nuvem [22] com o objetivo de acompanhar demandas de seus aplicativos. Tendências recentes mostram empresas como Dropbox e Apple abraçando o movimento para construir suas próprias nuvens privadas, a fim de obter melhor controle, segurança e maior eficiência [23] [24]. A popularidade da nuvem escala os centros de dados com o desejo de alcançar o mais alto nível de aplicação com desempenho, que requer planejamento cuidadoso de computação, armazenamento e rede com uma arquitetura de interconexão. O cenário de consumo de recursos de TI por parte da administração pública, assim como ocorre no restante do mercado, encontra constante oportunidade de modernização de tecnologias.

No entanto, apesar da crescente aplicabilidade dos *data centers* em uma ampla variedade de cenários, há muito poucos estudos sistemáticos de medição de desempenho de *data centers* para orientar questões de atraso e largura de banda. No trabalho de Reyes em 2018 [2], foi realizada uma comparação de custos de 5 arquiteturas de DCNs (*Leaf-Spine, Fat-tree, Hybrid Fat-tree, Facebook 4-post e Facebook New Fabric*), considerando diferentes tamanhos destes DCNs: *mini (*100 *servers), small (*1.536 *servers), medium (*64.000 *servers) e* megasize *(*100.000 *servers) datacenter* . Embora vários aspectos dos *datacenters* ainda precisem de análise empírica substancial, o foco específico desse trabalho está

em questões relativas à operação de uma rede de *datacenters* . Para isso, examinamos o desempenho em relação ao atraso, à largura de banda e falha de link, escalando estas 5 arquiteturas de *Data Center Network (DCN)* até *megasize* com 100.000 servidores usando o simulador *Mininet*. Ao final desse trabalho, esperamos apresentar uma visão geral de várias topologias DCNs, bem como os resultados experimentais para corroborar a análise depreendida em desempenho e custos nas arquiteturas a partir da presente pesquisa.

1.2 Motivação

Para implantar um *datacenter* de uma empresa ou órgão público, um dos primeiros passos é planejar a arquitetura e decidir o modo de conexão dos *switches*, assim sendo viabilizar que todos os usuários consigam acessar suas atividades de forma eficiente. Apos ser definido a arquitetura o próximo passo é desenhar a melhor topologia de rede para o órgão ou empresa. A definição sobre a qual topologia aderir varia de acordo com o tamanho da companhia e, principalmente, com o que se espera dela para o futuro. Existem diversas maneiras de se fazer isso e a grande motivação para o estudo comparativo apresentado neste trabalho é mostrar dados que ajudem a elucidar as diferenças entre as 5 arquiteturas atuais (*Leaf-Spine, Fat-tree, Hybrid Fat-tree, Facebook 4-post e Facebook New Fabric*) consideradas em [2] para estudos sobre seus custos financeiros.

Baseando-se neste trabalho sobre os custos das topologias, estudamos o desempenho dessas arquiteturas com o o objetivo de podermos definir o melhor custo-benefício para a implantação ou para a atualização das redes DCN. Outra motivação, não menos importante, foi a migração da rede do Superior Tribunal Federal (STF), de uma arquitetura tradicional (baseada em árvore) de 3 camadas, para *Leaf-Spine* .

O modelo tradicional hierárquico divide a LAN em três camadas: acesso – como o nome já diz, é pelo qual os equipamentos finais (interface com os usuários) têm acesso à rede; distribuição – concentra os vários acessos, prevendo segurança e segmentação das redes ou VLANs; e núcleo – concentra as distribuições e as conexões externas da rede como a internet, a extranet e a VPN.

Sendo assim, as redes *Leaf-Spine* se difere deste modelo tradicional para um modelo em que os *switches spine* funcionam essencialmente como o núcleo da estrutura e os *switches leaf* são como os acessos, fornecendo conectividade aos servidores e *uplink* aos *switches spines*. Com isso, permite-se maior densidade de portas na rede, com abordagens para todos os *switches* responsáveis pelo alto *throughtput* e menos latência de rede. Uma característica importante desse modelo é que, geralmente, cada *switch leaf* deve se conectar a cada *spine* com *oversubscription*. Esse ponto é o que resolve o problema do modelo tradicional hierárquico, já que agora não interessa em qual *switch* um servidor está co-

nectado, ele sempre precisará percorrer a mesma quantia de dispositivos até alcançar o outro servidor. Isso permite que a latência se mantenha em um grau presumível, visto que o tráfego só precisa atravessar um *switch spine* e um *leaf* para alcançar seu destino.

Uma série de desafios precisam ser tratados para a correta escolha durante a implantação ou migração entre essas duas topologias de rede. Com o objetivo de auxiliar na definição de qual dessas arquiteturas apresenta o melhor custo-benefício, de acordo com as necessidades da organização, esse trabalho analisa o desempenho das 5 arquiteturas acima citadas.

1.3 Objetivo Geral

Apresentar um estudo comparativo de desempenho de infraestrutura entre as seguintes arquiteturas de *DataCenters: leaf-spine, fat-tree, hybrid fat-tree, Facebook 4-Post* e a *new Facebook fabric*, utilizando o simulador *Mininet*. Este estudo tem como intuito orientar as futuras atualizações, compras e contratações de equipamentos e serviços de modernização de DCN, para que adotem a arquitetura mais vantajosas.

1.3.1 Objetivos específicos

Para se alcançar o objetivo geral, os seguintes objetivos específicos foram definidos:

- Estudar a evolução das arquiteturas de rede e definir as mais atuais para efeito do estudo comparativo.
- Definir e estudar as ferramentas que serão utilizadas, bem como implementar as topologias a serem comparadas.
- Definir, executar e analisar experimentos a fim de se comparar as arquiteturas adotadas, considerando-se as métricas de latência (atraso) e taxa de transferência (vazão). Além disso, estas métricas também serão analisadas considerando falhas em *links* de comunicação.

1.4 Organização do Texto

O restante desta dissertação está organizado da seguinte maneira. O Capítulo 2 apresenta a fundamentação teórica e os trabalhos relacionados sobre as arquiteturas DCNs. Em sequência, no Capitulo 3, uma proposta de comparação entre as redes DCNs é apresentada. O Capítulo 4 relata e analisa a avaliação das arquiteturas no simulador *Mininet*. Ao final, conclui-se o trabalho e propõem-se trabalhos futuros no Capítulo 5.

Capítulo 2

Fundamentação Teórica e Trabalhos Relacionados

Este capítulo apresenta os conceitos fundamentais para o trabalho em estudo, os quais incluem os tipos de arquiteturas utilizados em *datacenters* com uma breve revisão sobre o assunto. Em seguida, são apresentadas as arquiteturas do trabalho atual com base na comparação envolvendo essas arquiteturas. Ao final dessa seção, veremos os trabalhos relacionados a esse estudo.

2.1 Fundamentação Teórica

2.1.1 História das Redes de *Datacenters* (DCNs)

Nesta subseção, descrevemos algumas das principais arquiteturas das redes de centro de dados (DCN), topologias que foram propostas ao longo do tempo. Organizamos e apresentamos essas topologias em uma linha do tempo crescente com base em quando o artigo correspondente foi publicado, com a figura da sua respectiva topologia para enfatizar as diferenças estudadas e evoluídas no passar dos anos.

- **1953** - Clos Topologies for Telephony Networks [3]

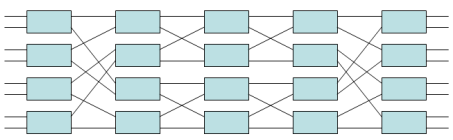

Figura 2.1: Clos Topologies for Telephony Networks [3]

Em 1953, foi uma das primeiras publicações no IEEE já com a proposta de uma arquitetura de redes sem bloqueio. O próprio autor cita que, naquela época, o uso de redes telefônicas com *switches* ou comutadores sem bloqueio era raro.

Vários tipos e tamanhos de combinações e links entre os *switches* estavam sendo testados com o desejo de se encontrar a arquitetura mais econômica para um determinado nível de bloqueio, conforme mostrado na Figura 2.1.

Com os métodos estudados, já se buscava um valor mínimo e equalizado de uma topologia de melhor custo-benefício e sem bloqueio. Ao projetar o sistema sem bloqueio com um número razoável de estágios de comutação, e depois omitir alguns dos caminhos, o projetista pode chegar a uma rede com um determinado nível de bloqueio e estar muito próximo do mínimo em pontos de cruzamento.

- **1985** - Fat-Tree for NoCs [4]

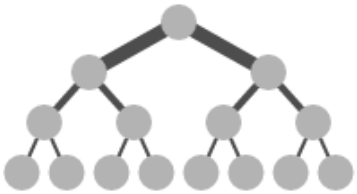

Figura 2.2: Fat-Tree for NoCs [4]

As redes de árvores são propostas por Charles E. Lesierson, em 1985, conforme a Figura 2.2. Vários nós estão se conectando de cima para baixo na árvore NoC e seguem a estratégia baseada em relação pai-filho. Pode ser uma comunicação em camadas de um nó-pai para o próximo nó-filho. O número de links conectados na rede em árvore a partir do topo para baixo é igual ao link de baixo para cima, à medida que os irmãos são conectados da mesma maneira. Na parte superior da rede em árvore, os links ficam mais gordos e os nós-raízes na árvore possuem a maioria dos links em comparação com outras topologias.

- **1994** - Hierarchical Interconnection Networks [5]

As HINs atraíram considerável atenção na comunidade de pesquisa em meados de 1994 como meio de comunicação. HINs aproveitam da localidade de referência no padrão de comunicação de muitos algoritmos paralelos, agrupando comutadores que se comunicam intensamente em *clusters*, conectando-se entre si por uma rede, conforme a Figura 2.3.

Nessa época, o conceito hierárquico foi estendido à interconexão dinâmica domínio de rede. Sob comunicação altamente local, os HINs superam redes não hierárquicas

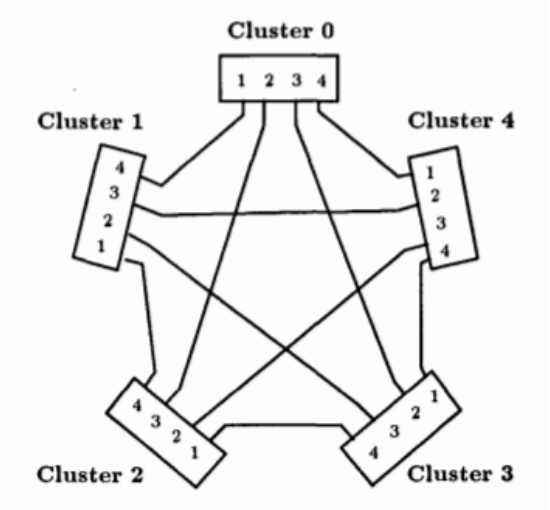

Figura 2.3: Hierarchical Interconnection Networks [5]

de tamanho semelhante. No entanto, os HINs compartilham algumas desvantagens: são frequentemente irregulares, com alto grau de uso e alto diâmetro, ou seja, representam redes não escaláveis. A distribuição desigual da carga de comunicação pela rede leva ao congestionamento em nós de interface, que, por sua vez, resulta em latência aumentada, *buffer* caro e baixa tolerância a falhas. Além disso, o desempenho diminui rapidamente na medida em que a rede é pequena, sendo que a principal causa desse problema é o acesso restrito aos *clusters*. Normalmente, com muito poucos nós em um *cluster*, cada nó serve como interface para distribuir a carga de comunicação uniformemente na rede.

- **1999** - Random Networks [6]

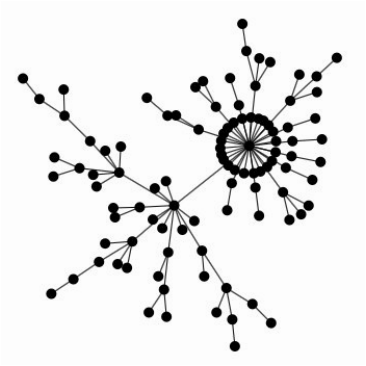

Figura 2.4: Random Networks [6]

Tradicionalmente, redes de topologia complexa têm sido descritas com a teoria de grafos aleatórios de Barabási e Albert [6], publicada em meados de 1999. Mas, na ausência de dados em grandes redes, as previsões dessa teoria raramente eram testadas no mundo real. No entanto, impulsionada pela informatização da aquisição de dados, essas informações topológicas estavam cada vez mais disponíveis, aumentando a possibilidade de entender a estabilidade dinâmica e topológica de grandes redes aleatórias, conforme mostrado na Figura 2.4.

Sistemas tão diversos quanto redes genéticas ou a World Wide Web (www) são melhor descritos como redes com topologia complexa. Uma propriedade comum de muitas redes grandes é que as conectividades de vértices seguem uma distribuição de lei de energia sem escala. Verificou-se no trabalho de Barabási e Albert [6] que esse recurso é uma consequência de dois mecanismos genéricos: (i) as redes se expandem continuamente pela adição de novos vértices e (ii) novos vértices se ligam preferencialmente a sites que já estão bem conectados. Um modelo baseado nesses dois ingredientes reproduz as distribuições estacionárias livres de escalas observadas, indicando que o desenvolvimento de grandes redes é governado por fenômenos robustos de auto-organização que vão além das particularidades dos sistemas individuais. Conforme mostrado na Figura 2.4, são conexões aleatórias que vão se expandindo a uma grande rede, comparada a grande rede de computadores (Internet).

- **2008** - Google Fat Tree [7]

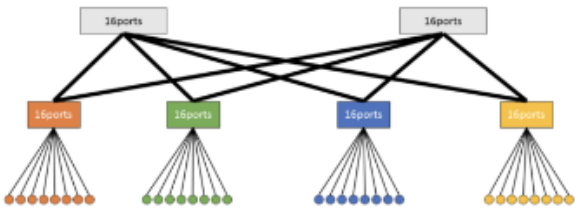

Figura 2.5: Google Fat Tree [7]

Em 2008, a arquitetura Google Fat Tree, conforme Figura 2.5, surge apresentando como maiores preocupações a largura de banda e o preço dos equipamentos para fazer uma rede com alta velocidade e redundância. No trabalho de Mohammad e Amin [7], a grande questão era como aproveitar os *switches Ethernet* mais baratos ou de *commodities* para suportar a largura de banda agregada completa dos *clusters*, consistindo em dezenas de milhares de elementos. Argumentaram que *switches* de *commodities* adequadamente arquitetados e interconectados podem oferecer mais desempenho com menos custo do que os disponíveis nas melhores soluções daquela época.

Nessa época a largura de banda era cada vez mais o gargalo de escalabilidade em redes de larga escala em DCN. As soluções existentes nesse periodo para lidar com esse problema de gargalo em torno de hierarquias de redes baseadas em *switches*, com produtos caros era o grande problema a ser resolvidos pelos grandes fabricantes de DCN. A ideia era trocar vários *switches* caros por muitos *switches* de menor valor. Em um determinado momento, a densidade da porta dos *switches high-end* limita o tamanho geral do *cluster*, enquanto ao mesmo tempo incorre em alto custo. Trata-se de arquitetura de comunicação do *datacenter* que aproveita os *switches* Ethernet para fornecer largura de banda escalável para aplicações em larga escala, com isso resolver o problema de largura de banda com custo acessível.

- **2008** - DCell [8]

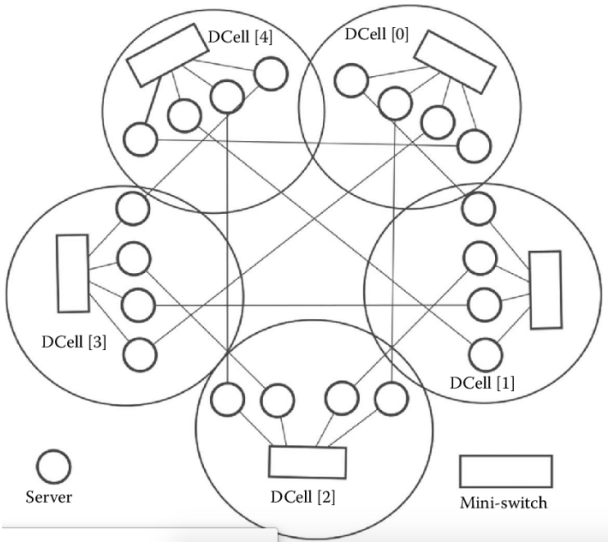

Figura 2.6: DCell [8]

A arquitetura DCell, apresentada na Figura 2.6 e no trabalho de Guo *et al.* [8], é uma estrutura definida recursivamente, a qual um DCell de alto nível é construído a partir de muitos DCells, e DCells no mesmo nível estão totalmente conectadas um ao outro. O DCell é escalonado dupla e exponencialmente à medida que o grau do nó aumenta. O DCell é tolerante a falhas por não ter ponto único de falha e a tolerância a elas distribuída por protocolo de roteamento. Este realiza o roteamento de caminho mais curto, mesmo na presença de falhas graves de link ou nó. Segundo os autores, a arquitetura DCell também fornece maior capacidade de rede do que a estrutura tradicional baseada em árvores para vários tipos de serviços. Além disso, O DCell pode ser expandido de forma incremental.

A prática daquela época de arquitetura DCN era conectar todos os servidores usando uma hierarquia em árvore de *switches*. Essa arquitetura propunha uma nova estru-

tura de rede denominada DCell. O DCell usa uma estrutura definida recursivamente para interconexão de servidores. Cada servidor se conecta a diferentes níveis de DCells por meio de vários links. Uma potencial desvantagem da solução DCell seria o uso de *switches* principais caros com maior custo de cabeamento, pois usa mais e mais links de comunicação em comparação com as estruturas baseadas em árvores.

- **2009** - BCube [9]

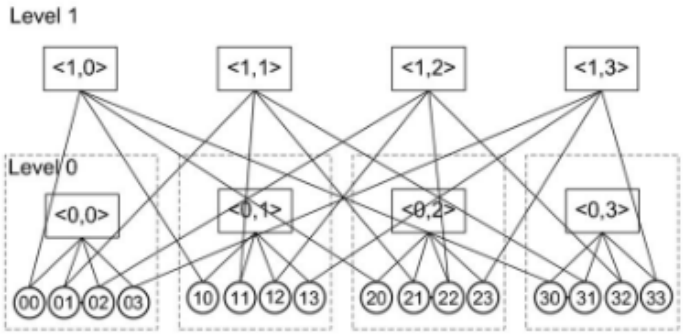

Figura 2.7: BCube [9]

O artigo apresentado em 2009 por Guo *et al.* [9] mostra uma nova arquitetura de rede projetado especificamente para DCNs modulares, conforme mostra a Figura 2.7. No centro da arquitetura do BCube está a sua estrutura de rede centrada no servidor, em que servidores com várias portas de rede se conectam a várias camadas de *switches* menores e mais baratos. Servidores agem não apenas como *hosts* finais, mas também retransmitindo um para o outro.

O BCube suporta vários aplicativos com uso intenso de largura de banda, acelerando os padrões de tráfego um para um, um para vários e um para todos, e fornecendo alta capacidade de rede para tráfego de todos para todos. O BCube exibe degradação de desempenho da rede com a performance dos servidores usados como concentradores e a taxa de falha do *switch* aumenta com a degradação do servidor(concentradores). Essa propriedade é de especial importância para os *datacenters* de contêineres. Desde que o contêiner esteja operacional, torna-se muito difícil reparar ou substituir seus componentes. A arquitetura de rede BCube assume o foco no servidor com várias interfaces em vez da prática orientada para o *switch*. Propõe inteligência em servidores e trabalha com *switches* baratos de pequeno porte. O BCube traz inovações em sua estrutura de interconexão, centrada no servidor, protocolo de roteamento e implementação eficiente.

- **2009** - MDCube [10]

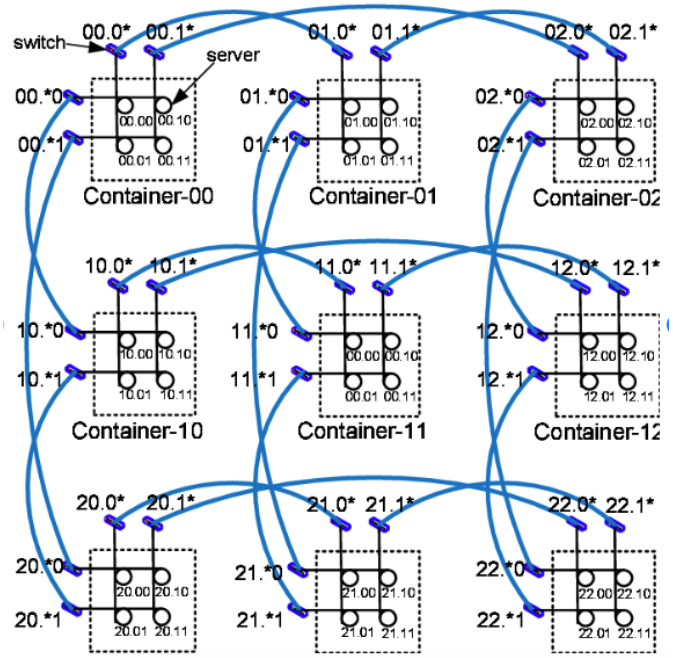

Figura 2.8: MDCube [10]

A arquitetura MDCube, apresentada no trabalho de Wu *et al.* em 2009 [10], propõe uma evolução na arquitetura BCUbe. A MDCube propõe uma estrutura de interconexão de alto desempenho para dimensionar contêineres baseados em BCube para *mega datacenters*. A arquitetura usa as interfaces de ligação rápida de alta velocidade dos *switches* mais baratos nos contêineres BCube para construir uma estrutura entre eles, reduzindo bastante a complexidade do cabeamento. A MDCube coloca suas inteligências de roteamento entre contêineres - externos e internos - apenas nos servidores para lidar com o balanceamento de carga e a tolerância a falhas, aproveitando, assim, diretamente, os *switches* de menor custo em vez de *switches* de última geração. Por meio do estudo, os autores relatam que a MDCube tem baixo diâmetro e alta capacidade.

Os *datacenters* baseados em contêineres foram introduzidos como blocos para a construção de *mega datacenters*. No entanto, é um desafio interconectar esses contêineres com custo razoável e baixa complexidade de cabeamento devido ao fato de que um *mega datacenter* pode ter centenas ou até milhares de contêineres, e a largura de banda agregada entre contêineres pode facilmente atingir terabit por segundo. Como uma nova arquitetura de rede centrada em servidor de contêiner interno, a MDCube interconecta milhares de servidores dentro de um contêiner e fornece alto suporte de largura de banda para padrões de tráfego típicos.

- **2010** - Scafida [11]

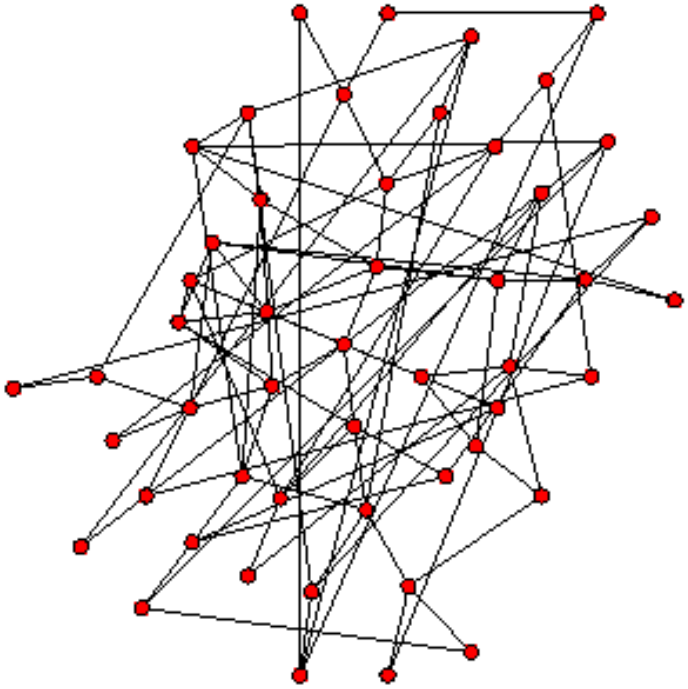

Figura 2.9: Scafida [11]

No estudo de Gyarmati e Anh Trinh [11], em 2010, foi proposta a arquitetura Scafida, um novo algoritmo de *datacenter* que gera uma estrutura inspirada em redes sem escala. O método restringe os níveis de nós para atender às capacidades físicas dos *switches*. Com base nos resultados da simulação feita no trabalho, é apresentado o impacto da limitação dos níveis de nós. O método fornece quase as mesmas propriedades das redes sem escala, isto é, curtas distâncias entre os nós e alta tolerância a falhas, favoráveis no contexto de *datacenter*. Segundo os autores, apesar de sua estrutura assimétrica, como mostra a Figura 2.9, as propriedades do Scafida são semelhantes às dos outros *datacenters*.

Ainda segundo os autores, as redes também podem ter estruturas assimétricas ou apenas simétricas. A existência de redes biológicas, cujas estruturas são principalmente assimétricas, prova que essas estruturas têm propriedades preferíveis, pois sobreviveram a competição evolutiva. Numerosas redes biológicas compartilham uma característica comum, isto é, a distribuição de seus níveis de nós seguem a distribuição da lei de potência - conhecidas como redes sem escala. Redes sem escala têm dois aspectos importantes, a saber: diâmetro ultra-pequeno e alta tolerância a falhas, o que seria favorável em caso de redes de *datacenters* também. Portanto, no

artigo, o autor propõe uma rede sem escala com algoritmo de geração de arquitetura de *datacenter* chamada Scafida.

- **2011** - BCN - Bidimensional Compound Networks [12]

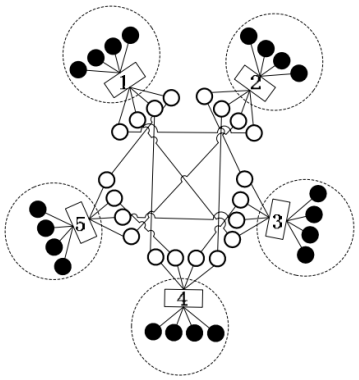

Figura 2.10: BCN - Bidimensional Compound Networks [12]

O estudo da arquitetura BCN - *Bidimensional Compound Networks*, publicada no artigo de Guo *et al.* em 2011 [12], evidencia que existem e são propostas várias estruturas centradas no servidor, conforme Figura 2.10. Ocorre que essas estruturas não são verdadeiramente expansíveis e sofrem baixo grau de regularidade e simetria, e, para resolver esse problema, os autores propõem a estrutura BCN com gráficos compostos hierarquicamente para interconectar grande população de servidores, cada um com apenas duas portas.

Para os servidores com apenas duas portas, os autores acreditam que haja vantagens de expansibilidade e do mesmo grau hierárquico. Além disso, a BCN oferece alto grau de regularidade, escalabilidade e simetria, muito em conformidade com o projeto modular em implementação de *datacenters*. Um BCN de nível um em cada dimensão é o maior DCN conhecido com servidor grau 2 e diâmetro 7. O trabalho mostra várias análises e simulações em que a BCN seria viável em estruturas para *datacenters*.

- **2012** - Jellyfish [13]

A arquitetura Jellyfish [13], apresentada em 2012, defende uma interconexão de rede de alta capacidade, que, adotando uma topologia aleatória de gráfico, rende-se naturalmente à expansão incremental, conforme mostra a Figura 2.11. Os autores acreditam que a Jellyfish é mais econômica do que uma arquitetura *Fat-tree*, suportando até 25% mais servidores em capacidade total usando o mesmo equipamento na escala de alguns milhares de nós, e essa vantagem melhora com a escala. Jellyfish também permite grande flexibilidade na construção de redes com diferentes graus de

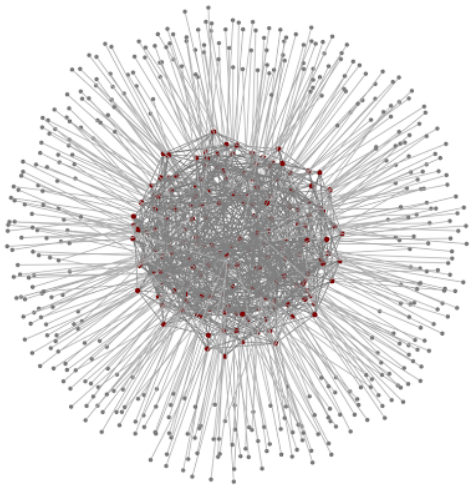

Figura 2.11: Jellyfish [13]

excesso de largura de banda. No entanto, o design não estruturado da Jellyfish traz novos desafios em roteamento, *layout* físico e fiação. Os autores descrevem abordagens para resolver esses desafios, e as avaliações sugerem que o Jellyfish poderia ser implantado nos *datacenters* atuais.

- **2013** - F10 – Fault Tolerant Engineered Network [14]

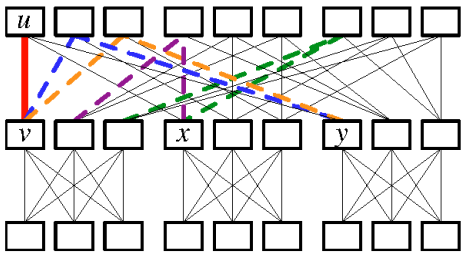

Figura 2.12: F10 Fault Tolerant Engineered Network [14]

A arquitetura denominada F10 (rede projetada para tolerar falhas) [14] propõe uma topologia de rede e um conjunto de protocolos que podem se recuperar rapidamente diante de falhas de rede. Esta rede apresenta uma nova topologia que a torna mais fácil de realizar reparos localizados, reequilibrando-a depois de ocorrências de falhas. Essa nova topologia proposta em 2013 é aplicável ao *Fat-tree* e a outras redes com várias árvores. Em seguida, os autores redesenham os protocolos de roteamento para obter proveito da topologia modificada. Para satisfazer o *failover* extremamente rápido, usa-se um mecanismo de recuperação local que reage quase instantaneamente à conectividade e ao equilíbrio de carga, mesmo na presença de várias falhas.

Os resultados do estudo mostram que, depois das falhas no link de rede e no *switch*, o F10 tem menos de 1/7 da perda de pacotes dos esquemas atuais da época. A arquitetura F10 é baseada em um novo algoritmo de topologia e roteamento de várias árvores. Com isso, obtém-se a restauração quase instantânea da conectividade e do equilíbrio de carga depois de uma falha de *switch* ou link. A abordagem opera inteiramente na rede sem modificações finais no *host*, e os experimentos mostram que as rotas geralmente podem ser restabelecidas com desvios de dois saltos adicionais e sem coordenação global, mesmo durante várias falhas. A avaliação do trabalho mostra uma redução significativa na perda de pacotes e um desempenho aprimorado no nível das aplicações.

- **2013** - Leaf-Spine [15]

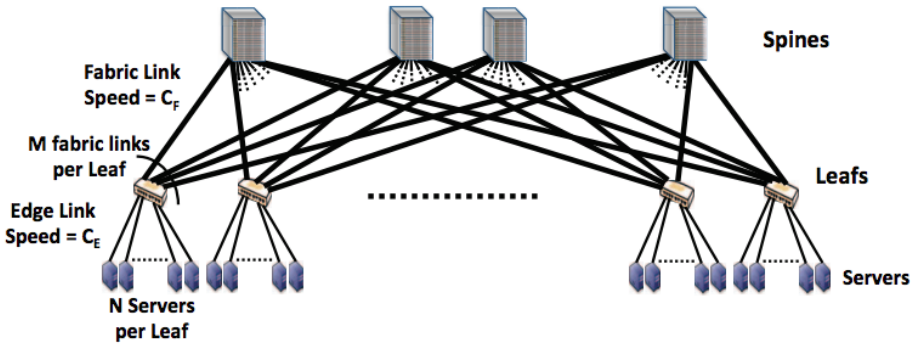

Figura 2.13: Leaf-Spine [15]

Os autores do artigo que propõe a arquitetura *Leaf-Spine* [15] fornecem orientações sobre o impacto das opções de desempenho da comutação de dados em redes *Leaf-Spine* e analisam as topologias Leaf-Spine sob cargas de trabalho de tráfego realistas por meio de simulações de alta fidelidade, identificando o que é e o que não é importante para o desempenho. Assim, as redes de *data centers* não empregam mais a arquitetura tradicional de acesso, agregação e núcleo da grande empresa. Em vez disso, *switches* comercialmente disponíveis são usados para construir a rede em topologias *Fat-Tree.* Geralmente, os arquitetos de rede estão escolhendo *switches* menores, densos e de topo de *rack* para a conectividade do servidor conectado a *switches* modulares maiores, criando um ambiente muito plano e densamente conectado a redes.

O uso liberal de caminhos múltiplos é usado para dimensionar a largura de banda. Essa arquitetura *Leaf-Spine* (mostrada na Figura 2.13) foi projetada para fornecer uma taxa de transferência muito escalável de maneira uniforme e previsível entre milhares e centenas de milhares de portas. Com esse efeito, a rede se aproxima

de uma rede ideal - um *switch* muito grande e sem bloqueio - em que todos os servidores estão conectados diretamente. Neste cenário, o arquiteto de rede está tentando chegar o mais pragmaticamente possível à rede ideal e, ao mesmo tempo, controlando custos.

No presente estudo é investigado o desempenho do caminho de dados da arquitetura *Leaf-Spine* construída, usando links Ethernet de 10, 40 e 100 gigabits. O objetivo é fornecer informações ao arquiteto de rede sobre as implicações das diferentes decisões de projeto que eles estão tomando para o desempenho do caminho de dados de suas redes. A pergunta a ser respondida é o quão perto eles podem chegar da rede ideal enquanto satisfazem suas restrições comerciais de custo e disponibilidade de equipamentos. A abordagem é considerar de forma pragmática a rede do *datacenter (Leaf-Spine)* e estudar as compensações, considerando apenas os parâmetros que o arquiteto da rede pode controlar facilmente, como velocidade do link e *buffer*, ao tentar alcançar a solução mais próxima possível da ideal. O desempenho da arquitetura *Leaf-Spine* é medido via simulações em nível de pacote de alta fidelidade no simulador OMNET.

Os autores concluem afirmando que o artigo investigou o desempenho do caminho de dados dessas redes usando simulações de alta fidelidade - com cargas de trabalho de tráfego realistas e pilhas TCP reais - para esclarecer as diferentes escolhas de projeto, como velocidade de link e *buffer*. Descobriram que o uso de links de maior velocidade na malha (entre as camadas *spine* e *leaf*) em relação à borda (nos servidores) melhora muito a eficiência do balanceamento de carga ECMP, e que ter tamanhos de *buffer* por porta significativamente maiores no *spine* ou *leaf* em comparação a o outro é um desperdício. Além disso, é melhor aplicar *buffer* adicional na camada *leaf* do que no *spine* para controlar os problemas de *Incast*.

- **2013** - Facebook 4-Post [16]

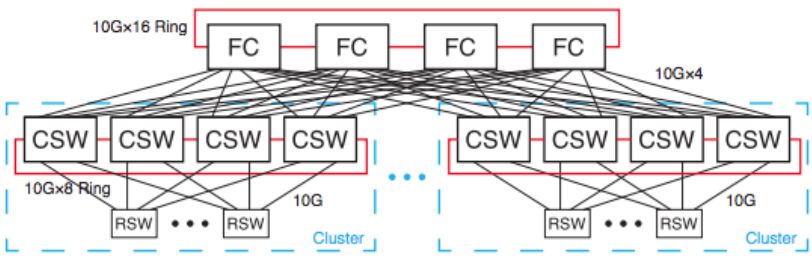

Figura 2.14: Facebook 4-Post [16]

A arquitetura Facebook 4-Post [16] é uma vertente da topologia *Leaf-Spine* e resolveu uma série de problemas que o Facebook havia encontrado no passado. Por exemplo,

as falhas de rede costumavam ser uma das principais causas de interrupção no serviço. A redundância adicional em 4-post tornou essas interrupções raras. Em outro exemplo, o tráfego que precisava cruzar os *clusters* costumava atravessar links caros do roteador. A adição do nível FC(camada mais alta mostrada na Figura 2.14) reduziu bastante o tráfego por meio desses links.

A Figura 2.14 mostra a rede do *datacenter* do Facebook da época. Cada *rack* contém um *switch* de *rack* (RSW) com até quarenta e quatro *downlinks* de 10Gb e quatro ou oito *uplinks* de 10Gb (normalmente, sobrecarga de 10:1), um para cada *switch* do cluster (CSW). Um cluster é um grupo de quatro CSWs e *racks* e RSWs do servidor correspondente. Cada CSW possui quatro *uplinks* de 40G (10G x 4), um para cada um dos quatro *switches* de agregação *FatCat* (normalmente, sobrecarga de 4:1). Os quatro CSWs em cada cluster são conectados em um anel de proteção 80G (10G x 8) e os quatro *switches* FC são conectados em um anel de proteção 160G (10G x 16). Os cabos intrarrack são de cobre com conexão direta SFP +; caso contrário, o MMF é usado (10GBASE-SR). A arquitetura de 4-post evoluiu ao longo do tempo para solucionar os desafios específicos de rede do Facebook.

As principais desvantagens do 4-post são resultado direto do uso de *switches* CSW e FC modulares muito grandes. Primeiro, uma falha no CSW reduz a capacidade intracluster para 75%; ou uma falha de FC reduz a capacidade entre *clusters* para 75%. Segundo, o tamanho do *cluster* é determinado pelo tamanho do CSW. Essa arquitetura resulta em um número menor de *clusters* muito grandes, dificultando a alocação de recursos entre os grupos de produtos do Facebook. Terceiro, grandes *switches* são produzidos em volumes menores de menos fabricantes e possuem CAPEX e OPEX por largura de banda mais alta. Em quarto lugar, os *switches* grandes geralmente são sobrecarregados internamente, significando que nem todas as portas podem ser usadas simultaneamente sem resultar em excesso de sobrecarga. Em quinto e último lugar, *switches* grandes geralmente são proprietários. Isso pode levar meses e anos nas correções de *bugs*, impossibilitando a implantação de protocolos personalizados e complicando seriamente a tarefa de gerenciar, monitorar e medir uma grande rede de *datacenter* .

- **2014** - Facebook New Fabric [17]

  Para o projeto de rede de *datacenter* Facebook New Fabric [17], apresentada na Figura 2.15, o desafio era construir todo *datacenter* como uma rede de alto desempenho em vez de um sistema de *clusters* hierarquicamente sobrecarregado. Também, objetivava-se um caminho claro e fácil para rápida implantação de rede e escalabilidade de desempenho, sem extrair ou personalizar grandes infraestruturas anteriores

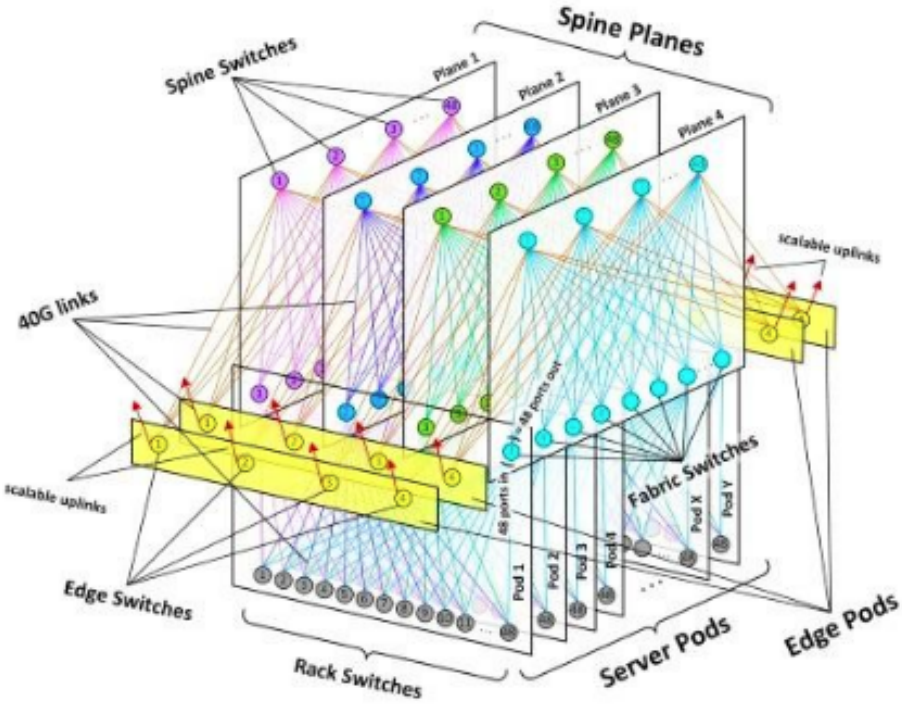

Figura 2.15: Facebook New Fabric [17]

toda vez que fosse preciso criar mais capacidade. Para conseguir isso, adotaram uma abordagem desagregada: em vez de grandes dispositivos e *clusters*, dividiram a rede em pequenas unidades idênticas - *pods* de servidor - e criaram conectividade de alto desempenho uniforme entre todos os *pods* no *datacenter*. Não há nada de especial em um *pod* - é como um *microcluster* de camada3. O *pod* não é definido por nenhuma propriedade física rígida; é simplesmente uma "unidade de rede" padrão na nova estrutura.

Cada *pod* é servido por um conjunto de quatro dispositivos chamado comutadores de malha, mantendo as vantagens da arquitetura atual de quatro colunas 3 + 1 para *uplinks* de topo do *rack* do servidor e escalável, se necessário. Atualmente, cada topo de *rack* possui *uplinks* de 4 x 40G, fornecendo capacidade total de largura de banda de 160G para um *rack* de servidores conectados a 10G, conforme Figura 2.15. A diferença é o tamanho muito menor da nova unidade - cada *pod* possui apenas 48 *racks* para servidores e esse fator é sempre o mesmo para todos os *pods*. É um bloco de construção eficiente que se encaixa perfeitamente em várias plantas de *datacenter* e requer apenas comutadores básicos de tamanho médio para agregar os topos de *rack*. A menor densidade de portas dos comutadores de malha torna sua arquitetura interna muito simples, modular e robusta, e existem várias opções fáceis de serem encontradas, disponíveis em várias fontes.

Outra diferença notável é como os *pods* são conectados juntos para formar uma rede de *datacenter* . Para cada porta de *downlink* para um topo de *rack*, reserva-se uma quantidade igual de capacidade de ligação ascendente nos comutadores de malha do *pod*, o que permite escalar o desempenho da rede até estatisticamente sem bloqueio.

- **2015** - Update on Google Fat Tree [18]

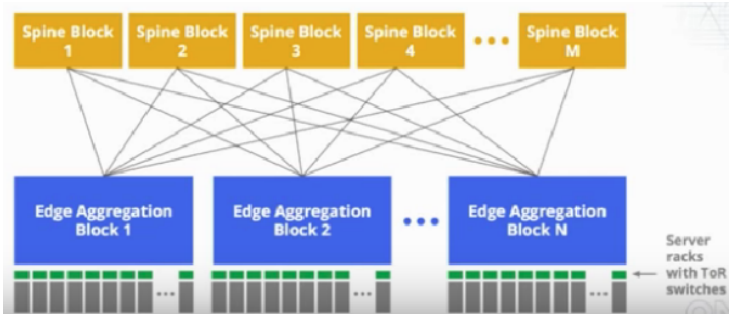

Figura 2.16: Update on Google Fat Tree [18]

O Google fez uma atualização na arquitetura *Fat-tree* publicada em 2015 [18] (Figura 2.16), criando um mecanismo de controle centralizado com base em uma configuração global enviada a todos os *switches* de *datacenter*. A arquitetura *Fat-tree*, como design de hardware modular, associada a um software simples e robusto, permitiu que o *design* também suportasse redes intercluster e de área ampla.

Redes de *datacenters* são executadas em dezenas de sites em todo o planeta, aumentando em 100x a capacidade em dez anos para mais de 1Pbps de largura de banda de bissecção pela Google com essa arquitetura. No geral, a arquitetura de software ou SDN se assemelha mais ao controle em plataformas de computação e armazenamento em larga escala do que aos protocolos de rede tradicionais. Os protocolos de rede geralmente usam o estado flexível com base na troca de mensagens em pares, enfatizando a autonomia local.

Os arquitetos de rede conseguiram usar as características e necessidades distintas nas implantações de *datacenter* para simplificar os protocolos de controle e gerenciamento, antecipando muitos dos princípios das modernas implantações de rede definidas por software.

As redes de *datacenter* descritas neste estudo [18] representam algumas das maiores do segmento, estão em implantação em dezenas de sites em todo o planeta e suportam milhares de serviços internos e externos, incluindo o uso externo pelo *Google Cloud Platform*. A arquitetura de rede de *cluster* encontrou reutilização substancial para redes entre *clusters* no mesmo campus e até implantações de WAN no Google. O artigo apresenta uma retrospectiva em dez anos e cinco gerações de redes de

datacenters de produção. Com técnicas complementares para fornecer mais largura de banda a *clusters* maiores do que seria possível a qualquer custo. Partiram da sabedoria convencional para construir um controlador de rota centralizado que alavancasse a configuração global de um plano de *cluster* predefinido e enviado a todos os comutadores de *datacenter*.

Esse controle centralizado estendeu-se à infraestrutura de gerenciamento, permitindo evitar protocolos complexos em favor das melhores práticas de gerenciamento dos servidores. A abordagem permitiu fornecer largura de banda de bissecção substancial para malhas de escala de construção, todas com benefícios significativos de aplicação.

Vimos nessa linha de evolução - ou linha histórica - das Redes de Datacenter (DCN) que as arquiteturas evoluíram com o passar dos anos de acordo com a necessidade de tráfego de dados nos mega *datacenters* , conforme as arquiteturas organizadas nos itens anteriores em uma linha do tempo crescente com base correspondente à publicação. Esses estudos nos levam a refletir que mega datacenters surgiram com infraestruturas para criação de aplicativos on-line, como pesquisa na web, e-mail, jogos on-line, *big data*, bem como serviços de infraestrutura. Dentro de um *datacenter*, um grande número de servidores estão interconectados e usam um servidor específico para controlar os vários *switches* em uma estrutura de rede *datacenter* (DCN). Nas estruturas baseadas em árvores, é cada vez mais difícil encontrar os objetivos de design dos *datacenters* em razão do crescimento exponencial dos *datacenters* atuais. Consequentemente, várias novas estruturas DCN são propostas com divisão em duas categorias.

Uma é centralizada em *switches* organizados em estruturas como árvore com a inteligência de interconexão nos *switches*. A outra é centrada no servidor ou no controlador, com a inteligência de interconexão nos servidores e usa *switches* apenas como conexões em alta disponibildade. Entre outras, as estruturas centradas no servidor têm as seguintes vantagens: os atuais servidores são mais programáveis que os *switches*, portanto, a implantação da nova estrutura DCN é mais viável; e foram criadas muito boas propriedades topológicas e algoritmos eficientes. Mas, uma desvantagem importante a considerar a respeito dessas topologias é que elas não são verdadeiramente expansíveis. Uma rede é expansível se não houver alterações com relação à configuração do nó e apenas conexões de links são necessárias quando expandidas. Isso pode causar influência negativa nos aplicativos em execução em todos os servidores existentes durante o processo de expansão da topologia.

Por fim, verificamos na evolução das arquiteturas de rede DCN que além das arquiteturas baseadas em servidores, podemos perceber uma divisão sendo: as baseadas em árvore, também chamadas por várias literaturas como topologia tradicionais hierárquicas,

e a *Leaf-Spine*. Grandes fabricantes estão utilizando essas topologias físicas, colocando controladores com algoritmos e criando DCN com *Software Defined Networ*k (SDN). Nas próximas seções, abordaremos as redes tradicionais e *Leaf-Spine* em maiores detalhes.

2.1.2 Arquiteturas Tradicionais

O modelo tradicional, oriundo da arquitetura *Fat-tree*, é utilizado por muitas empresas e está em constante migração para as novas arquiteturas[1]. Um projeto hierárquico [19] envolve a divisão da rede em camadas distintas e cada camada da hierarquia fornece funções específicas que definem sua função na rede geral. Dessa forma, o projetista e o arquiteto da rede podem otimizar e selecionar o hardware, o software e os recursos de rede ideais para executar funções específicas para cada camada de rede. Os modelos hierárquicos se aplicam ao projeto de LAN e de WAN. Um projeto típico de LAN hierárquica em um *campus* [25] corporativo inclui as três camadas a seguir:

- Camada de acesso - proporciona ao grupo de trabalho/usuário acesso à rede.
- Camada de distribuição - oferece conectividade baseada em políticas e controla o limite entre as camadas de acesso e núcleo.
- Camada de núcleo - proporciona o rápido transporte entre os *switches* de distribuição no *campus* corporativo.

A vantagem de dividir uma rede linear em blocos menores e mais gerenciáveis é o tráfego que permanece local. Somente o tráfego destinado a outras redes é movido para uma camada superior. Os dispositivos de camada 2 de uma rede linear oferecem pouca oportunidade para controlar as transmissões ou para filtrar o tráfego não desejado. À medida que dispositivos e aplicativos são adicionados a uma rede linear, os tempos de resposta são reduzidos até que a rede se torne inutilizável.

Não existe absolutamente nenhuma regra que determine a construção de uma rede DCN. Embora seja verdadeiro que muitas redes sejam construídas com o uso de três camadas físicas de *switches*, conforme a Figura 2.17, esta não é uma exigência. Em uma rede menor, é possível ter duas camadas de *switches* em que os elementos de núcleo e distribuição são combinados em um *switch* físico. É o que se chama de projeto de núcleo colapsado.

A Camada de Acesso

Em um ambiente de LAN, a camada de acesso [26] concede aos dispositivos finais o acesso à rede. No ambiente de WAN, ela pode oferecer aos trabalhadores locais ou remotos acesso

[1]Como o adotado para a migração no STF, com equipamentos da Extreme (um dos motivos para o estudo desse trabalho), e, segundo a CISCO no seu curso de certificação para CCNA e CCNP, em redes [19]

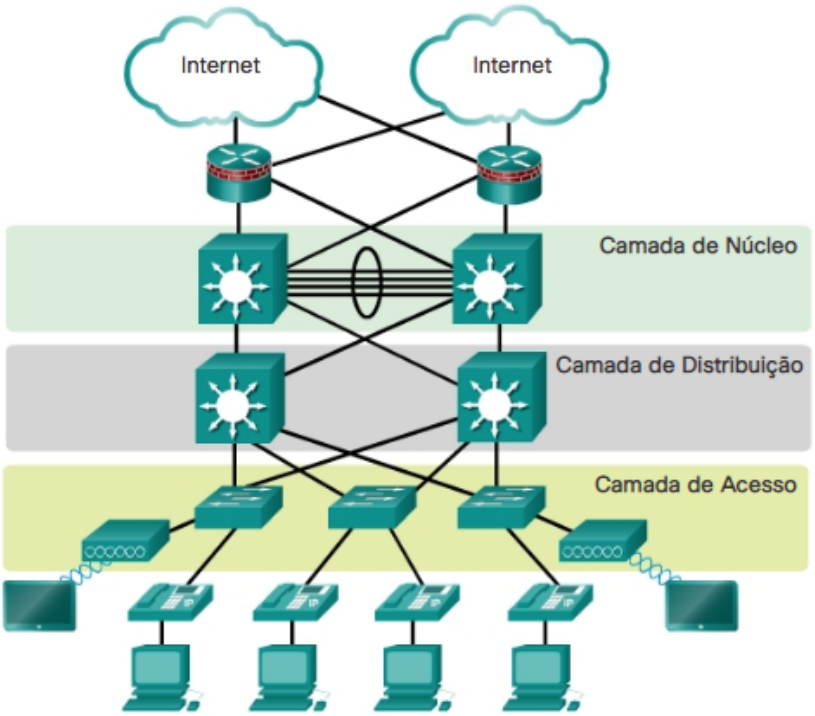

Figura 2.17: Modelo de projeto hierárquico de redes [19]

à rede corporativa por meio das conexões da WAN. Como mostrado na Figura 2.17, a camada de acesso a uma rede corporativa pequena geralmente incorpora *switches* e pontos de acesso de camada 2, proporcionando conectividade entre estações de trabalho e servidores. A camada de acesso atende a diversas funções, incluindo:

- Comutação de Camada 2;
- Alta disponibilidade;
- Segurança de porta;
- Limites de classificação, marcação e confiança de QOS;
- Inspeção de protocolo de resolução de endereços ARP;
- Listas de controle de acesso ACL;
- *Spanning Tree* STP; e
- *Power over Ethernet* POE e VLAN auxiliar para VOIP.

A Camada de Distribuição

A camada de distribuição [27] reúne os dados recebidos dos *switches* de camada de acesso antes de sua propagação para a camada de núcleo para roteamento até a transmissão final.

A camada de distribuição é o limite entre os domínios de camada 2 e a rede roteada de camada 3. O dispositivo de camada de distribuição é o ponto focal nos *datacenters*. Um roteador ou um *switch* multicamada é usado para segmentar grupos de trabalho e isolar problemas de rede em um ambiente de rede *campus*. Um *switch* de camada de distribuição é capaz de ofertar serviços *upstream* para muitos *switches* de camada de acesso. A camada de distribuição pode fornecer:

- Agregação de links da LAN ou WAN;
- Segurança baseada em políticas na forma de listas de controle de acesso (ACL) e filtragem de pacotes;
- Serviços de roteamento entre LANs e VLANs e entre domínios de roteamento com protocolos de roteamento, como, por exemplos, EIGRP e OSPF;
- Equilíbrio de redundância e de carga;
- Um limite para agregação e sumarização de rota, configurado nas interfaces diante de camada de núcleo; e
- Transmissão de controle do domínio, pois os roteadores ou *switches* multicamada não encaminham *broadcasts*. O dispositivo atua como ponto de demarcação entre os domínios de *broadcasts*.

A Camada de Núcleo

A camada de núcleo com frequência é chamada de *backbone* da rede [28]. Essa camada consiste em dispositivos de rede de alta velocidade, os quais foram projetados para comutar pacotes o mais rapidamente possível e interconectar vários componentes do *campus*, como módulos de distribuição, módulos de serviço, *datacenter* e borda da WAN. Como mostrado na Figura 2.17, a camada de núcleo é essencial para a interconectividade dos dispositivos de camada de distribuição. Por exemplo, a interconexão do bloco de distribuição com a WAN e com a borda da Internet. O núcleo deve ser altamente disponível e redundante e agregar o tráfego de todos os dispositivos da camada de distribuição. Portanto, deve ser capaz de encaminhar grandes quantidades de dados rapidamente. As funções da camada de núcleo incluem:

- Proporcionar *switching* de alta velocidade, isto é, transporte rápido;
- Proporcionar confiabilidade e tolerabilidade a falhas;
- Dimensionar o uso de equipamentos mais rápidos e não mais numerosos; e
- Evitar a manipulação de pacotes que fazem uso intensivo da CPU, devido à segurança, à inspeção, à classificação de *Quality of Service (QOS)* ou a outros processos.

Modelo com Núcleo Recolhido em Duas Camadas

O projeto hierárquico de três camadas maximiza o desempenho, a disponibilidade da rede e a capacidade de dimensionar o projeto de rede. Entretanto, muitas redes de empresas pequenas não crescem significativamente com o passar do tempo. Sendo assim, um projeto hierárquico de duas camadas em que as camadas de núcleo e distribuição são recolhidas em uma camada é, com frequência, mais prático [29]. Um "núcleo recolhido" ocorre quando as funções de camada de distribuição e de camada de núcleo são implementadas por um único dispositivo. A principal motivação para o projeto do núcleo recolhido é reduzir o custo da rede, mantendo a maioria dos benefícios do modelo hierárquico de três camadas.

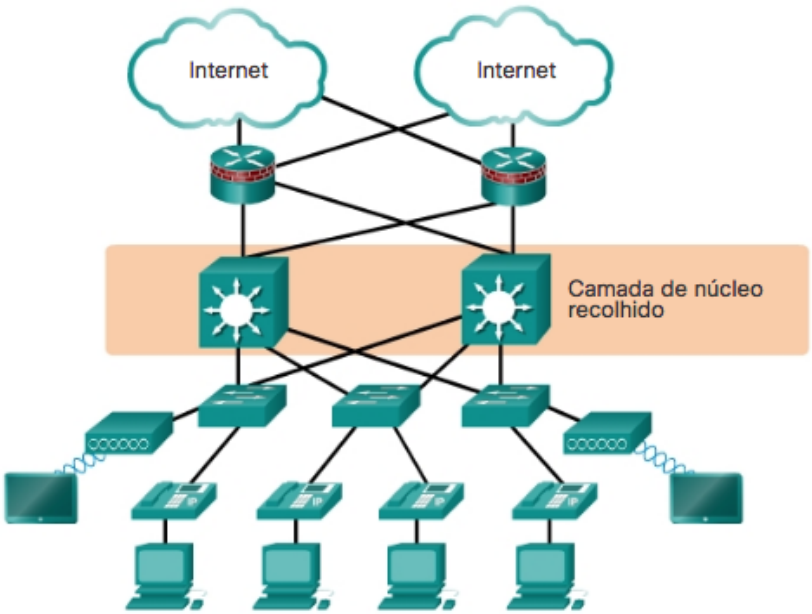

Figura 2.18: Modelo com Núcleo Recolhido

O exemplo na Figura 2.18 colapsou ou uniu as funcionalidades das camadas de distribuição e de núcleo em dispositivos de *switch* de várias camadas. O modelo de rede hierárquico oferece uma estrutura modular que proporciona flexibilidade no projeto de rede e facilita a implementação e a identificação e solução de problemas. A camada de distribuição oferece conectividade aos serviços baseados na rede, na WAN e na borda da Internet. Estes serviços baseados na rede podem incluir e não estão limitados à *Wide Area Application Services (WAAS)* e a controladores de LAN sem fio [26]. Dependendo do tamanho da LAN, esses serviços e a interconexão à WAN e à borda da Internet podem residir em um *switch* da camada de distribuição, que também agrega a conectividade da camada de acesso da LAN. Isso também é conhecido como um projeto de núcleo colapsado, pois a distribuição serve como a camada 3 de agregação para todos os dispositivos.

2.1.3 Arquitetura convergente *Leaf-Spine*

Esta arquitetura de alto desempenho possui várias derivações. Segundo o fabricante DELL [30], nessa arquitetura de dois níveis, cada *switch* de camada inferior (camada de *leaf*) é conectado a cada um dos *switches* de camada superior (camada de *Spine*) em uma topologia de malha completa (Figura 2.19). A camada de *leaf* consiste em *switches* de acesso que se conectam a dispositivos como servidores. A camada de *Spine* é o *backbone* da rede e é responsável por interconectar todos os *switches* da *leaf*. Cada *switch leaf* se conecta a cada *switch spine* na rede.

O caminho é escolhido aleatoriamente para que a carga de tráfego seja distribuída uniformemente entre os *switches* da camada superior. Se um dos *switches* de nível superior falhar, ocorre apenas uma ligeira degradação no desempenho em todo o *datacenter*. Se houver *oversubscription* de um link (ou seja, se mais tráfego for gerado do que pode ser agregado no link ativo de uma só vez), o processo de expansão da capacidade será direto. Um *spine* adicional pode ser usado e os *uplinks* podem ser estendidos para cada *switch leaf*, resultando na adição de largura de banda entre as camadas e redução do *oversubscription*. Se a capacidade da porta do dispositivo se tornar uma preocupação, um novo *switch leaf* pode ser adicionado, conectando-o a cada *switch spine* e adicionando a configuração de rede ao *switch*. A facilidade de expansão otimiza o processo do departamento de TI em escalonar a rede. Se não houver *oversubscription* entre os *switches* da camada inferior e seus *uplinks*, uma arquitetura sem bloqueios poderá ser obtida.

Com uma arquitetura *Leaf-Spine* [31], não importa em qual *switch leaf* um servidor esteja conectado, seu tráfego sempre precisará cruzar o mesmo número de dispositivos para chegar a outro servidor (a menos que o outro servidor esteja localizado no mesma *leaf*). Essa abordagem mantém a latência em um nível previsível, pois uma carga útil só precisa alternar entre *switch spine* e outro para acessar o seu destino.

Figura 2.19: Topologia Típica *Spine and leaf*

As conexões entre os *spine* e os *leafs*s podem ser camada 2 (comutada) ou camada 3 (roteada). Em ambas as topologias, conexões *downstream* para servidores, armazena-

mento e outros dispositivos *endpoint* dentro dos *racks* são camada 2, e conexões para redes externas são camada 3. Os conceitos a seguir aplicam-se às topologias da camada 2 e da camada 3 [32]:

- Cada *leaf* se conecta a cada *spine* na topologia; e
- Servidores, matrizes de armazenamento, roteadores de borda e dispositivos similares sempre se conectam ao *leaf*, nunca aos *spines*.

As topologias de camada 2 e camada 3 utilizam, cada uma, dois *leaf* no topo de cada *rack* [15]. O número total de conexões de *leaf* é igual ao número de conexões do *leaf* multiplicados pelo número de conexões dos *spines*. A largura de banda da rede pode ser aumentada adicionando-se conexões entre os *leaf*s e os *Spines*, desde que a camada da *spine* tenha capacidade para as conexões adicionais.

Topologia de camada 3 *Leaf-Spine*

Em uma rede de camada 3, o tráfego entre *spine*s e *leaf*s é roteado. O limite entre a camada 3 e a camada 2 está nos *leafs*. Os *switches spines* são conectados entre si em uma topologia de camada 3. O *Equal Cost Multipath Load Balance (ECMP)* é usado para balancear a carga de tráfego na rede da camada 3. Conexões dentro de *racks* de *hosts* para *switches leaf* são camada 2. As conexões para redes externas são feitas de um par de bordas, como mostrado na Figura 2.20.

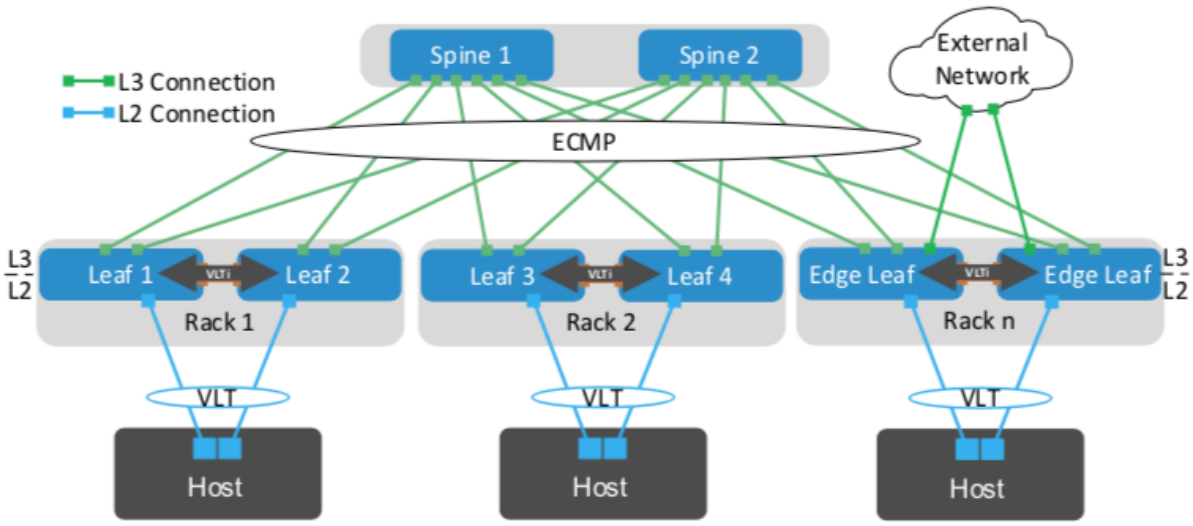

Figura 2.20: Topologia da arquitetura DCN *Spine and leaf* de camada 3

Topologia de camada 2 *Leaf-Spine*

Em uma rede camada 2, o tráfego entre *leaf* e *spine* é trocado (exceto por um par de *leaf* de borda) como mostrado na Figura 2.21. *Virtual Link Trunking* (VLT), em conjunto com *Virtual Link Trunking interconnect* (VLTi), é usado para múltiplos caminhos e tráfego de balanceamento de carga por meio da rede da camada 2. As conexões dos *hosts* para os *switches leafs* também são em camada 2. Para conexões com redes externas, os links da camada 3 são adicionados entre os *spines* e um par de *leaf* de borda usando o ECMP.

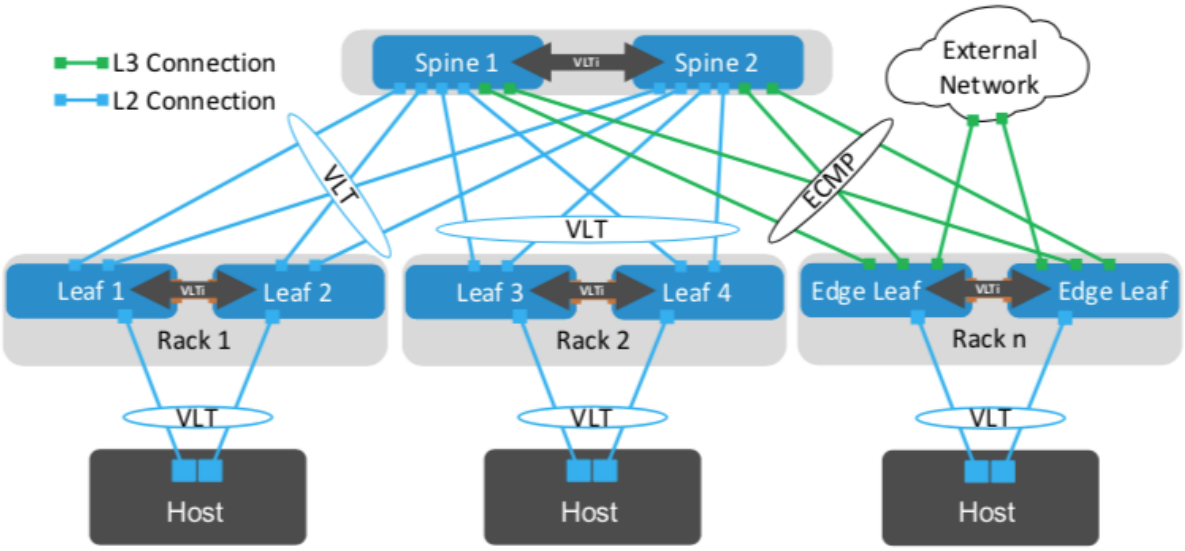

Figura 2.21: Topologia da arquitetura DCN *Spine and leaf* de camada 2

Como veremos posteriormente no Capitulo 3 desse estudo, as arquiteturas que serão simuladas nesse trabalho são derivações da *Leaf-Spine*. Depois dessa breve explicação sobre a história das hierarquias de DCN e seus conceitos, nas próximas seções exploraremos os trabalhos já realizados e relacionados a essa pesquisa.

2.2 Trabalhos Relacionados

Requisitos rigorosos sobre o desempenho, como escalabilidade, segurança, tolerância a falhas, baixas latências e capacidade de lidar com padrões de tráfego dominados por fluxos de tráfego leste-oeste [33, 34], são características das arquiteturas DCN. Uma variedade de arquiteturas de rede tem sido proposta na literatura para abordar esses requisitos [35, 1, 36]. Embora os custos de 5 arquiteturas DCN sejam conhecidos pelo trabalho publicado em 2018, de Ronald Reyes e Thomas Bauschert [2] (veja Figura 2.22), nada é abordado acerca de desempenho. Sendo assim, ainda é necessário continuar trabalhando para entender a eficiência da largura de banda e do atraso dessas arquiteturas de rede

de *datacenter*. A análise destas duas informações em conjunto (custos e desempenho) auxiliará os projetistas na escolha da arquitetura que apresente o melhor custo-benefício para sua organização.

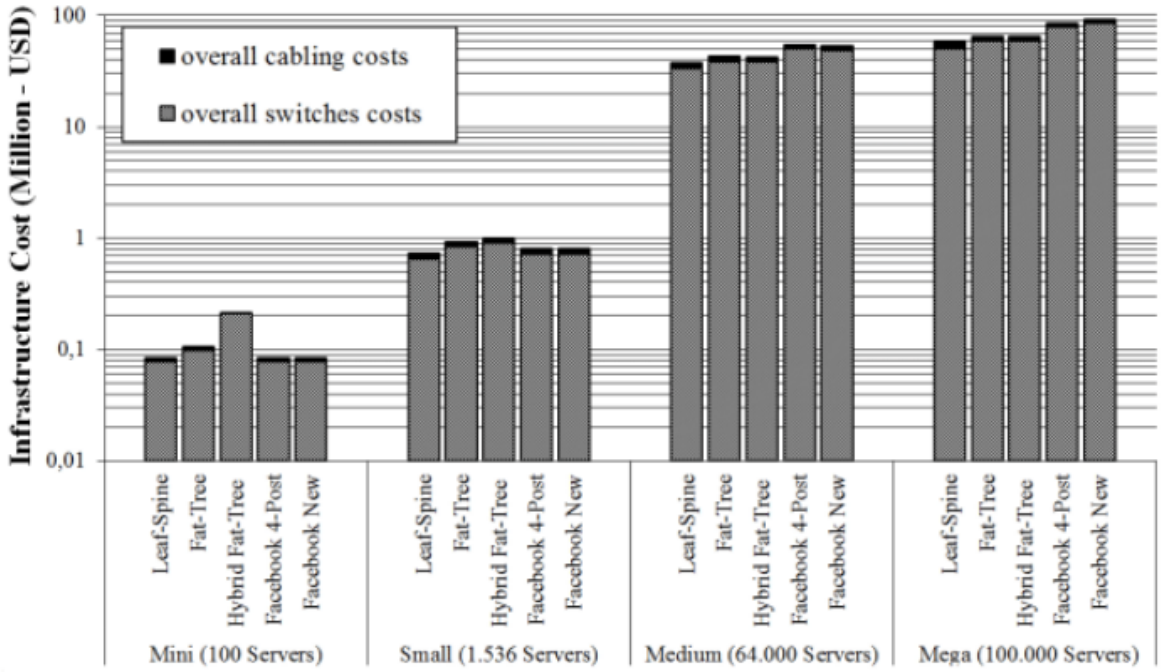

Figura 2.22: Custo da Infraestrutura DCN [2].

Apesar de alguns trabalhos [37] compararem a arquitetura *Fat Tree* [7, 4] e *DCell* [8], sob um simulador de plataforma de código aberto chamado ns-3 [38] ou *Mininet* [39], há poucos trabalhos na literatura sobre estudos de desempenho de diferentes arquiteturas DCN para considerá-los como referência. A seguir são discutidos alguns trabalhos encontramos sobre desempenho de arquiteturas DCNs.

2.2.1 Comparativo entre *Fat-Tree* e *DCell*

Um dos primeiros trabalhos de comparativo de arquiteturas de rede de *datacenter* , em 2012, apresentou um estudo comparando as arquiteturas *Fat-Tree* e *DCell* [37]. No artigo, são simulados no ns3 estes dois modelos de arquitetura DCN, comparando sua eficácia monitorando as taxas de transferência de rede e as latências médias de pacotes. A análise apresentada pode ser um pano de fundo para os estudos adicionais sobre a simulação e a implementação das topologias personalizadas do DCN e sobre os protocolos de endereçamento personalizados nos *datacenters* de grande escala. Os autores acreditam que esse é o primeiro estudo comparativo de arquiteturas de rede de *data center* usando implementação e simulação. Uma análise de simulação simples apresentada no artigo permite

comparar o comportamento e o desempenho das arquiteturas propostas em diferentes cargas de trabalho e condições de rede. As arquiteturas DCN utilizadas na análise foram implementadas em sistemas de pequena escala, com 20 servidores no caso do modelo DCell.

A análise de simulação pode ser considerada como um teste geral para redes realistas com grande número de hosts e vários padrões de comunicação e tráfego. A análise também pode ser usada para os "datacenters verdes" ao projetar protocolos de comunicação com eficiência energética em arquiteturas DCN. Nas simulações, o padrão de comunicação e a geração de tráfego são alcançados usando a distribuição aleatória linear.

Os resultados da simulação mostram que a topologia *Fat-Tree* apresenta taxa de transferência praticamente constante, sendo que uma leve degradação nessa taxa é observada quando o número de nós é aumentado. Mais de 1 milhão de pacotes é trocado na simulação da topologia *Fat-Tree* com 72 *pods*. A taxa de transferência média da rede, de 256 a 93.000 nós, foi observada na faixa de 169Mbps a 165Mbps, respectivamente. O atraso médio de pacotes na arquitetura baseada em árvore *Fat-Tree* também é observado como quase constante. O atraso médio observado dos pacotes varia de 0,043ms a 0,049ms para a simulação de 4 a 72 *pods*, respectivamente. Os resultados observados mostram que o desempenho da arquitetura baseada em árvore *Fat-Tree* é independente do número de nós.

No caso da arquitetura DCell, os resultados observados mostram um declínio na curva quando o número de nós é aumentado. O DCell supera a arquitetura baseada em árvore *Fat-Tree* para um pequeno número de nós, mas diminui gradualmente em termos de taxa de transferência quando o número de nós e os níveis do DCell aumentam. Um comportamento semelhante é observado no atraso médio de pacotes. Os resultados mostram que a taxa de transferência diminui bastante à medida que o número de nós aumenta de 20 para 500. No entanto, os resultados mostram uma pequena declinação da curva depois do número de nós atingir 500. Os resultados mostram que a arquitetura básica *Fat-Tree* supera a DCell em termos de taxa de transferência média da rede e latência de pacotes.

2.2.2 Comparativo entre arquiteturas *Fat-Tree e Bcube*

Wong *et al.* [40] apresentam um estudo que compara o desempenho entre as arquiteturas *Fat Tree* e *Bcube*, também avaliando largura de banda e atraso. Com o estudo, os autores investigaram até que ponto o desempenho (medido em termos de taxa de transferência e atraso) é consistente em relação aos resultados relatados ou reivindicados na literatura.

Neste estudo, simularam os modelos de desempenho da árvore *Fat* e a arquitetura *BCube* no ns-3. Para uma comparação justa, o tamanho das duas arquiteturas, as configurações dos dispositivos, os protocolos de rede e a geração do fluxo de tráfego foram

consistentes. A única variável que deve ser alterada é a topologia de rede. O tamanho das duas arquiteturas variou de 16 a 3.456 servidores. Para gerar fluxo de tráfego, foram selecionados aleatoriamente pares de comunicação em todos os nós, com cada par consistindo em um remetente e um receptor. Cada par enviava simultaneamente um fluxo de dados de 1Mbps de seu remetente para o seu receptor e a simulação para ambas as arquiteturas é executada em 100 segundos.

O padrão de fluxo de tráfego para ambas as arquiteturas segue um comportamento de "ligado e desligado" com distribuição aleatória linear, e demonstrou modelar razoavelmente fluxos de tráfego em *datacenters* do mundo real. Com as configurações de simulação decididas, os autores continuaram a construir a árvore *Fat* e a topologia de rede *BCube*. As estatísticas de desempenho são então reunidas e analisadas usando o módulo *Flow Monitor* no ns-3. Com base nos resultados médios da taxa de transferência, é possível observar que a arquitetura *BCube* oferece consistentemente muito mais desempenho da taxa de transferência do que a arquitetura da árvore *Fat*, mesmo quando o número de nós aumenta de 16 para 3.456. Há uma ligeira degradação na taxa de transferência de rede, com a arquitetura do *BCube* variando de 64 para 3.375 servidores, passando de 237Mbps para 174Mbps. A taxa de transferência de rede da arquitetura em árvore *Fat* permanece constante no intervalo de 117 Mbps a 126 Mbps.

Para os resultados médios do atraso de pacotes, observa-se que a arquitetura *BCube* tem um atraso ligeiramente menor que a arquitetura em árvore *Fat*, com a arquitetura *BCube* tendo um intervalo de 0,036ms a 0,048ms e a arquitetura em árvore *Fat* tendo um intervalo entre 0,066ms a 0,072ms. Ambas as arquiteturas apresentaram um aumento bastante gradual no atraso de pacotes, à medida que o número de servidores aumentou de 64 para 3.456. Uma das razões pelas quais o *BCube* obteve um desempenho melhor em ambas as métricas é devido à arquitetura de rede ter vários caminhos possíveis para enviar um fluxo de tráfego do ponto A ao ponto B. Isso trouxe menor congestionamento de tráfego e mais largura de banda disponível. Além disso, os servidores na arquitetura do *BCube* atuam como servidores de retransmissão e ajudam uns aos outros a acelerar o roteamento de pacotes e o fluxo de tráfego, resultando em melhoria tanto na taxa de transferência quanto no desempenho do atraso de pacotes na arquitetura da árvore *Fat*.

Quanto à arquitetura de árvores *Fat*, a razão do desempenho geral consistente está nas propriedades fundamentais das redes de árvores *Fat*, que, teórica e comprovadamente, são muito eficientes na interconexão de redes. Também se pode argumentar que, em grande parte, o desempenho do atraso de pacotes de ambas as arquiteturas é independente do tamanho da rede. Os resultados do estudo mostraram que a arquitetura do *BCube* apresenta um desempenho ligeiramente melhor que a arquitetura da árvore *Fat* em termos de taxa de transferência média e atraso de pacotes. Nesse caso, a *BCube* teve melhor desempenho

com simulador ns-3, escalando exponencialmente até 3200 *hosts* ou servidores.

2.2.3 Comparativo entre as arquiteturas *Google Fat-Tree* , *Facebook Fat-Tree* e *DCell*

Lebiednik *et al.* [1] apresentaram um comparativo de desempenho entre as arquiteturas *Google Fat-Tree* , *Facebook Fat-Tree* e *DCell*. O foco principal do trabalho também foi analisar a latência e largura de banda. Além disso, implementaram as aquiteturas *Google Fat-Tree* [7], *Facebook Fat-Tree* [17] e *DCell* [8] em dois simuladores de redes diferentes - gem5 [41] e *Mininet* [42].

Os testes em ambos simuladores - *Mininet* e *Gem5* - mostraram que a topologia do *Google Fat-Tree* supera a topologia da *DCell* em taxa de transferência. Além disso, a arquitetura *Facebook Fat-Tree* forneceu a melhor latência geral. Porém, em várias instâncias dos experimentos as outras topologias apresentaram melhores resultados. Nesse trabalho, chegou-se somente a 1200 *hosts*, sendo relevante somente para pequenos *data-centers.*

2.2.4 Análise de desempenho da arquitetura *Fat-tree* usando comunicação centrada em conteúdo

Escrito em 2016 foi publicado um estudo de comparação de desempenho [43], utilizando o simulador NS-3, entre a comunicação baseada em *Network Centric Networking* (CCN) ou *Content Centric Networking* e a comunicação baseada em IP, usando a arquitetura *Fat-tree*. Os resultados da simulação mostram que a CCN tem um desempenho melhor que o IP em termos de atraso médio de pacotes e taxa de transferência média, variando o número dos *hosts*.

Segundo os autores, a CCN representa um novo paradigma de rede criado para acomodar o colossal crescimento da Internet e o surgimento de novos aplicativos. Com a CCN, os *data centers* devem suportar serviços que troquem volumes de dados grandes. De fato, o conteúdo pode ser armazenado em *cache* em locais arbitrários da rede e reutilizado a partir daí, quando solicitado. De forma contrária, esse não é o caso do IP pois depois que uma solicitação é enviada para o endereço IP de destino, o conteúdo não pode ser reutilizado, a menos que sejam retransmitidas a solicitação e a resposta.

O objetivo principal do estudo é investigar a adequação da CCN nas arquiteturas da DCN. Além disso, outro objetivo deste trabalho é investigar o desempenho do mecanismo de *cache* na rede da CCN combinado com a arquitetura DCN. Os autores acreditam que nenhum outro artigo investigou o uso do paradigma CCN no gerenciamento de DCN. Por isso resolveram avaliar o uso de uma das implementações mais conhecidas da CCN,

chamada NDN (*Named Data Networking*). Foi simulada a arquitetura *Fat-Tree* DCN usando o NS-3 para reproduzir os mesmos resultados para comunicação IP e NDN. Depois de várias simulações, os autores afirmam que o atraso médio de pacotes do *Fat-Tree* combinado com a NDN, diminui de 0,026ms para 0,012ms. Por outro lado, pode-se observar que o atraso médio de pacotes do *Fat-Tree* combinado com a comunicação IP, aumenta levemente de 0,064ms para 0,068ms. A comunicação IP demora muito mais que a NDN quando o número de nós aumenta para 3456. Isso ocorre devido ao grande número de solicitações de dados que podem causar um congestionamento na rede usando IP. Considerando a diminuição do número de nós, vários caminhos possíveis são fornecidos para os usuários alcançarem um conteúdo. Isso justifica a degradação do atraso com a comunicação baseada em NDN, caracterizado por um mecanismo de armazenamento em *cache* na rede e assegurado pela estrutura armazenada em cada nó. O conteúdo pode ser obtido a partir de um nó mais próximo que o existente em cache.

A NDN oferece maior rendimento que o IP. Com a NDN, a taxa de transferência atinge 710Mbps, enquanto que com o IP a taxa de transferência da rede não excede a 122Mbps. O rendimento da rede aumenta com o aumento do número de nós usando a NDN. Por outro lado, a taxa de transferência da rede diminui com o aumento do número de nós usando a comunicação IP. A degradação na taxa de transferência com comunicação baseada em IP é explicada pelo grande número de solicitações de dados trocadas para recuperar o conteúdo. De fato, se o número de solicitações aumentar com o número de nós, o servidor não poderá responder rapidamente a todas as solicitações, portanto, a taxa de transferência diminui e o atraso aumenta. Com a NDN, o *cache* na rede explica novamente o aumento da taxa de transferência. Como resultado, a NDN supera o IP em termos de atraso e de taxa de transferência usando o *Fat-Tree*. Com o *Fat Tree*, um grande número de comutadores de rede, colocados em diferentes níveis, é estruturado para fornecer um local de *cache* na rede mais eficiente. Por fim, o objetivo dos autores era provar, usando uma comparação entre a comunicação baseada em NDN e IP, que a arquitetura DCN da *Fat-Tree* pode oferecer um melhor desempenho de comunicação. Os resultados da simulação mostraram que o NDN supera o IP em termos de atraso e de taxa de transferência médios de pacotes. Isso se deve principalmente ao mecanismo de *cache* na rede permitido pela estrutura de armazenamento de conteúdo e mantida em cada nó de conteúdo.

2.3 Considerações Finais

Como observado nas pesquisas reportadas das seções anteriores, todos os artigos em suas abordagens conseguem se aproximar da nossa proposta, comparando o desempenho de

até duas arquiteturas, conquanto seja distinguindo o desenvolvimento do nosso trabalho em proporção e escalabilidade. Portanto, neste estudo, propomos simular 5 arquiteturas de DCN (*leaf-spine, fat-tree, hybrid fat-tree, Facebook 4-Post e Facebook new fabric*) analisando o desempenho da largura de banda e da latência, escalando o tamanho das arquiteturas, variando de: mini (100 *servers*), *small* (1.536 *servers*), *medium* (64.000 *servers*) até megasize (100.000 *servers*), usando o simulador *Mininet* como mostraremos nos próximos capítulos. Além disso, ao final, como já temos o trabalho sobre os custos dessas arquiteturas publicados em 2018 por Reyes e Bauschert [2], espera-se prover informações suficientes para que os projetistas consigam definir a arquitetura de melhor custo-benefício para suas organizações.

Capítulo 3

Proposta de Comparação

Este capítulo descreve a nossa proposta para comparação de arquiteturas DCN que requerem um desenho das topologias a serem estudadas. Serão propostas cinco topologias, e, em seguida, a definição de comparação de desempenho de largura de banda e latência resumida nas métricas ou circunstâncias que afetam cada desempenho das arquiteturas. O *hardware* e o sistema operacional utilizados na simulação também serão descritos. Finalmente, apresentaremos os simuladores *Mininet, EVE-NG e Packet Tracer*, com suas vertentes de testes. No início do nosso estudo usamos o simulador NS-3, mas não conseguimos escalar os testes das arquiteturas propostas até 100000 servidores ou um DCN mega como proposto a seguir.

3.1 Arquiteturas de Rede Datacenter (DCN)

Para descrever corretamente todas as arquiteturas de rede que estudaremos, as seguintes definições são usadas no documento. Um *switch leaf* é um dispositivo que fornece acesso aos pontos finais da rede (por exemplo, servidores). *Switches Spines* são dispositivos de agregação que conectam *switches leaf*. Um ponto de entrega (PoD) é a unidade básica de rede do *datacenter* e é composto por *switches Spine* e *Leaf*. Os *switches super-spine* são dispositivos usados para interconectar os PoDs do *datacenter*.

As arquiteturas de rede estão representadas nas próximas subseções. No caso das redes com K PoDs ou o número de PoDs representado por K, cada PoD consiste em P *switches Leafs* e N *switches Spines*. Além disso, os PoDs são interconectados por M *switches super-spine*.

3.1.1 Arquitetura *Leaf-Spine*

A topologia mais simples *Leaf-Spine* seria com um único PoD configurado como uma rede de duas camadas, a qual, como mostrado na Figura 3.1, cada *switch leaf* se conecta a todos os *spines*. Essa topologia fornece conectividade redundante entre qualquer par de *leafs* [44]. Uma rede de um único PoD é adequada para pequenos *datacenters*. Para *datacenter* de grande porte, a rede *Leaf-Spine* é construída com uma topologia de três camadas que consiste em vários PoDs interconectados por *switches Super-Spines*. Cada *Spine* se conecta a todos os *switches Super-Spines*, fornecendo conectividade redundante entre qualquer par dos PoDs, conforme Figura 3.1.

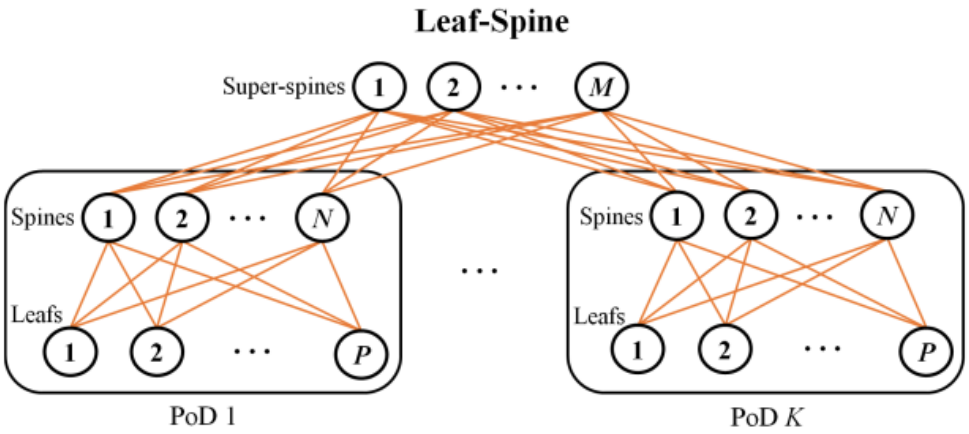

Figura 3.1: Arquitetura *Leaf-Spine*.

3.1.2 Arquitetura *Fat-Tree*

Diferentes topologias baseadas em árvore (*fat-tree*) estão disponíveis na literatura de redes DCN [7, 37]. Em particular, consideramos a arquitetura *Fat-Tree* representada na Figura 3.2 e estudada por Balanici *et al.* [45]. Nessa rede, um PoD é composto de P *leafs* e N=2 *spines*. Cada *leaf* se conecta a todos os *spines* do PoD. Os PoDs são interconectados por meio de M *switches Super-Spines*, cada um dos quais se conecta a um *spine* de cada PoD de rede.

3.1.3 Arquitetura *Hybrid Fat-tree*

Essa arquitetura é semelhante a *Fat-Tree*. Para se tê-la híbrida, adicionam-se *switches* de circuito óptico *Optical Circuit Switching* (OCS) aos PoDs da rede, como mostrado na Figura 3.3. Um OCS substitui *switches spine* para fornecer conectividade de alta velocidade (por exemplo, 40-100Gbps) por meio de conexões ópticas estabelecidas dinamicamente entre pares de *leafs*. Cada *leaf* é conectada aos *spines* no PoD e aos dispositivos OCS.

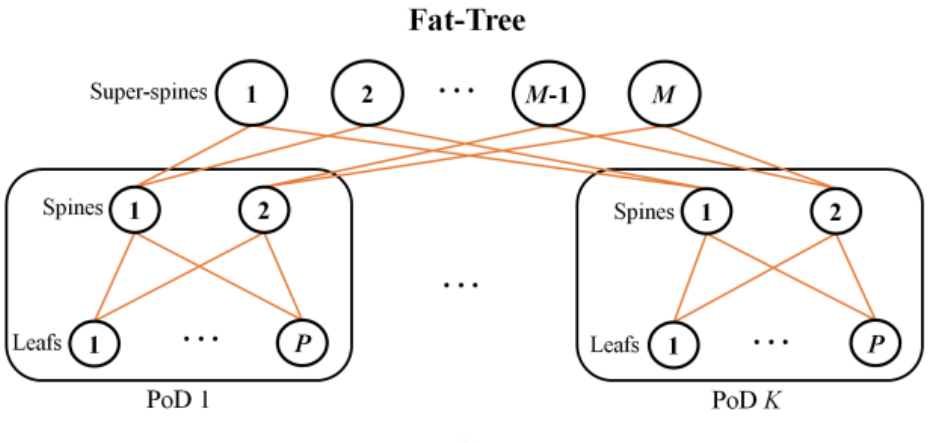

Figura 3.2: Arquitetura *Fat-Tree*.

Na Figura 3.3, o caso é mostrado em que um OCS substitui um *spine* em cada PoD. A solução estudada neste trabalho baseia-se na rede *c-through* [46], que pode ser facilmente implementada em *fabrics* OCS disponíveis no mercado [47]. Estes *fabrics* OCS são construídos com Sistemas Microeletromecânicos (MEMS) [45].

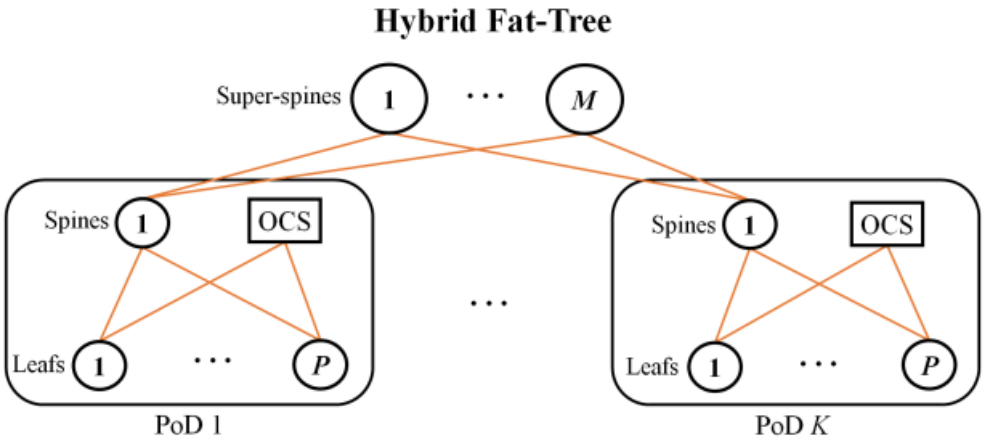

Figura 3.3: Arquitetura *Hybrid Fat-Tree*.

3.1.4 Arquitetura *Facebook 4-Post*

A topologia Facebook 4-Post [16] é uma versão modificada da arquitetura *Leaf-Spine*. Como visto na Figura 3.4, cada PoD tem N = 4 *spines* conectados em um anel de proteção. Por outro lado, a camada *Super-Spines* é construída com um anel de proteção com M = 4 *Super-Spines*, que se conectam a todos *spines* do PoD. Além disso, cada *leaf* em um PoD se conecta a todos os *spines* do mesmo PoD.

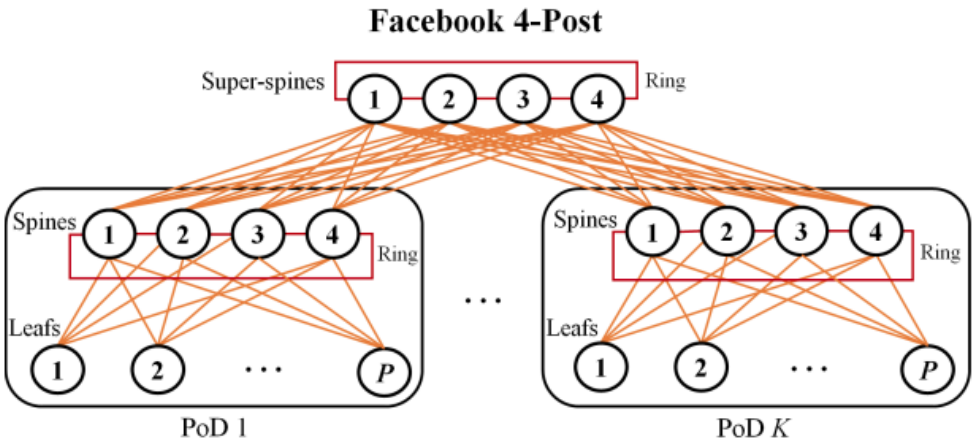

Figura 3.4: Arquitetura *Facebook 4-Post*

3.1.5 Arquitetura *Facebook New Fabric*

Na topologia *Facebook New Fabric* [16], os PoDs têm a mesma configuração de uma rede *leaf-spine*. Em particular, cada PoD contém P = 48 *leafs* e N = 4 *switches spines* (ver Figura 3.5). Essa arquitetura, no entanto, difere da *leaf-spine* no projeto da camada dos *super-spine*.

Como mostrado na Figura 3.5, esta camada é construída por quatro *super-spine* formados por até 48 *switches* de *super-spine* cada. Portanto, o número máximo de *super-spine* é M = 48 X 4 = 192. O papel dos *plane super-spine* é conectar os *switches spines i* aos PoDs da rede (*i*= 1,2,3,4). Por exemplo, os *switches super-spine* no *plane 1* conectam os primeiros *swiches spines* a todos os PoDs. Em comparação com a *leaf-spine*, o *Facebook New Fabric* implementa uma camada *super-spine* que aumenta a largura de banda para comunicação entre os PoDs. Além disso, mostra melhor resiliência porque oferece maior conectividade redundante entre qualquer par de PoDs.

Todas as 5 arquiteturas citadas anteriormente (*leaf-spine, fat-tree, hybrid fat-tree, Facebook 4-Post e Facebook new fabric*) consistem em uma camada de *leaf*, uma de *spine* e outra de *super-spine*. *Super-spines* somente são implementados em redes *multi-PoD*, ou seja, grandes DCN.

3.2 Aspectos comparativos

Para uma comparação justa, o tamanho das arquiteturas, as configurações dos dispositivos e os protocolos de rede, com as gerações de fluxo de tráfego, devem ser consistentes. A única variável que deve ser mudada é a topologia de rede. O quadro representado na Figura 3.6 mostra as configurações de simulação usadas para os modelos das arquiteturas.

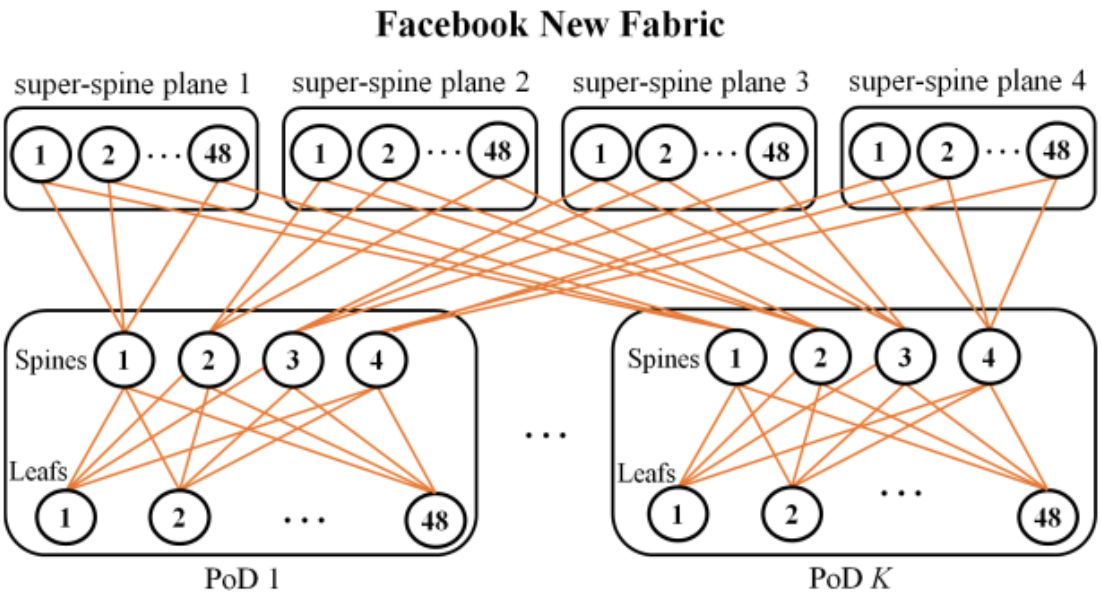

Figura 3.5: Arquitetura *Facebook New Fabric*.

	Leaf-Spine	Fat-tree	Hybrid Fat-tree	Facebook 4-Post	Facebook New Fabric
Número de PODs (k)	K até mega	K até mega	K até mega	K até mega	K até mega
Número de niveis	3	4	4	4	4
Números de Links /Dispositivo	P x N e N x M	P x N=2 e N =2 x M	P x N=2 e N=1 x M	P x N=4 e N=4 x M=4	P=48 x N=4 e N=4 x M=(48 x 4)
Número de servidores	Mini 100-100000 Mega	Mini 100-100000 Mega	Mini 100-100000 Mega	Mini 100-100000 Mega	Mini 100-100000 Mega
Tempo de Execução	100s	100s	100s	100s	100s
Tamanho do pacote	1024 bytes	1024 bytes	1024 bytes	1024 bytes	1024 bytes
Taxa de dados para envio de pacotes	10 Mbps	10 Mbps	10 Mbps	10 Mbps	10 Mbps
Taxa de dados para o link do dispositivo	10.000 Mbps	10.000 Mbps	10.000 Mbps	10.000 Mbps	10.000 Mbps
Seleção de pares de comunicação	Seleção aleatória com probabilidade uniforme	Seleção aleatória com probabilidade uniforme	Seleção aleatória com probabilidade uniforme	Seleção aleatória com probabilidade uniforme	Seleção aleatória com probabilidade uniforme
Padrão de fluxo de tráfego	Tráfego aleatório Linear	Tráfego aleatório Linear	Tráfego aleatório Linear	Tráfego aleatório Linear	Tráfego aleatório Linear

Figura 3.6: Configurações de simulação para as arquiteturas.

O tamanho das arquiteturas varia de 100 até 100.000 hosts, representando *mini (*até 100 *servers), small (* até 1.536 *servers), medium (* até 64.000 *servers) e* megasize *(* até 100.000 *servers) datacenters.* Os *switches ethernet* de 10.000Mbps com endereços IPs realistas usados no simulador *Mininet* [48], como o emulador de rede e POX como controlador *OpenFlow*, foram escolhidos porque proporcionaram a melhor convergência de rede e taxas entre os controladores OpenFlow testados (*RIPLPOX, OpenDaylight e Floodlight*) [49]. Os testes foram realizados com *Iperf* [50] e *ping* (regido pelo IETF

na RFC 2925). Por meio de comandos, o *Iperf* pode calcular a largura de banda disponível entre dois *hosts*, criando um controle de transmissão cliente-servidor com conexão do protocolo TCP. O comando *ping*, bem conhecido por técnicos e administradores de redes, pode não só fornecer resultados de conectividade na rede, mas também ser usado para medidas de latência. O comando também fornece a capacidade para determinar o tamanho do pacote que você está enviando a rede, que, por sua vez, permite enviar pacotes maiores para emular o envio de arquivos maiores. Para gerar fluxo de tráfego neles, selecionamos aleatoriamente a comunicação entre pares em todos os nós, com cada par consistindo de um remetente e um receptor.

Cada par envia simultaneamente um fluxo de dados de 10Mbps do seu remetente para o seu receptor, e a simulação para as arquiteturas é executada por 100 segundos. O padrão de fluxo de tráfego para as arquiteturas segue um comportamento com distribuição aleatória linear, o qual modela razoavelmente os fluxos de tráfego no mundo real dos DCNs [34]. Esses parâmetros de testes foram baseados no artigo de Benson *et al.* [34] que se baseia no entendimento das características de trafego dos *datacenters* também usado pela Microsoft. Nesse artigo os autores apresentaram um estudo empírico preliminar de padrões de tráfego de ponta a ponta em redes de *datacenter* que podem informar e ajudar a avaliar pesquisas e operações. Os autores analisaram *logs* de gerência SNMP coletados em 19 *datacenters* para examinar variações temporais e espaciais nas cargas de links e perdas. Também verificaram os rastreamentos de pacotes coletados em switches de *datacenter* encontrando evidências de comportamento de tráfego mostrando que a estrutura pode ser usada para avaliar abordagens de engenharia de tráfego. Como esse estudo foi usado para os trabalhos relacionados descrito nessa dissertação, usamos as evidências desses valores para simular a realidade mais próxima de um *datacenter* nas nossas simulações. Além disso, foi verificado o trafego do *datacenter* do STF e as evidências colaboraram para usarmos os valores descritos na figura Figura 3.6.

Para o estudo de desempenho, concentramo-nos em duas importantes métricas: atraso médio no pacote (latência) e taxa de transferência média (vazão). O atraso médio do pacote pode ser calculado da seguinte forma:

$$D_{avg} = \frac{1}{n} \sum_{i=1}^{n} d_i \tag{3.1}$$

Em que D_{avg} se refere ao atraso médio do pacote, n é o total de número de pacotes recebidos na rede e di é o atraso de cada pacote i.

A taxa de transferência média da rede pode ser calculada do seguinte modo:

$$\tau = \frac{\sum_{i=1}^{n} (p_i \times \delta_i)}{\sum_{i=1}^{n} d_i} \tag{3.2}$$

Em que τ se refere à taxa de transferência média na rede, pi ϵ [0,1] com $pi = 0$ representando a perda do pacote i e $pi = 1$ representando a recepção do pacote i; δ_i como o tamanho do pacote i em bits, di como o atraso do pacote i e n é o número total de pacotes recebidos na rede.

As ferramentas utilizadas e o ambiente de testes foram emulados por meio do simulador *Mininet*, enquanto a ferramenta *Iperf* foi utilizada para geração de tráfegos, mensurando o desempenho da largura de banda, e o *ping*, para verificar a latência. Estas ferramentas são detalhadas a seguir.

3.3 Ambiente Experimental e Ferramentas Utilizadas

O cenário proposto na Seção 3.2 foi implantado em um servidor Dell Poweredge 16 CPU, 72GB de RAM, 6TB de HD e virtualizador da Citrix XenServer 7.5. Foi virtualizado o S.O. linux Ubuntu 14.04 LTS - 64 bit com *Mininet* 2.2.2. Para o Ubuntu, foram disponibilizados 16 CPUs com 64Gb de memória RAM e 500Gb de HD.

3.3.1 Simulador *Mininet*

O *Mininet* [51] simulador usado para projetar ou desenvolver novos recursos (de hardware ou software) para DCN. Alguns exemplos podem envolver tarefas, como testar configurações de rede, projetar topologias de redes que atendam a casos específicos (como *datacenters*), desenvolver novos algoritmos de rede (roteamento, balanceamento de carga, etc.), além de muitas outras possibilidades em simulações que nem sempre temos um ambiente real a nossa disposição. Esse simulador de redes de código aberto (escrito em Python) foi desenvolvido inicialmente por alunos e professores da Universidade de Stanford e atualmente é mantido por uma comunidade de desenvolvedores. É possível emular componentes da rede que podem executar aplicações reais e até se comunicar com redes reais. Por exemplo, pode-se executar *scripts*, códigos (programas) próprios ou mesmo aplicações prontas (servidor Web, DNS, Firewall, etc.) nos *hosts* da rede emulada.

O *Mininet* é um *software* que permite agilizar a prototipação dentro de um computador, tanto de redes pequenas quanto de grandes redes (*datacenters*) [42]. O *Mininet* consegue gerar redes DCNs por meio de técnicas de virtualização utilizando processos e *namespaces* em rede. Além disso, o *Mininet* tem a capacidade de emular diversos elementos de rede (ex.: *hosts*, *switches* de camada 2 *Open vSwitch (OVS)*, roteadores de camada 3 e links) [48] e pode funcionar em um computador por meio do *kernel Linux*,

tendo o propósito de emular uma rede completa por intermédio do comando *mn* e seus parâmetros.

O *Mininet* tem a função de criar o ambiente necessário para a estrutura da rede por meio da instanciação de *switches*. Por sua vez, um controlador (como o *POX*) assume o papel do plano de controle da rede, tomando as decisões sobre o fluxo dos dados. Já o protocolo *openflow*, usado para comunicação entre os switches e o controlador, foi originado na Universidade de Stanford em 2008. Em dezembro de 2009, a versão 1.0 da especificação do *switch OpenFlow* foi lançada. Desde a sua criação, o *OpenFlow* foi gerenciado pela *Open Networking Foundation (ONF)*. Desde o seu lançamento, várias empresas e projetos de código aberto, como o *OpenDaylight Project*, suportam o *OpenFlow* e até mesmo fornecem controladores, como o *OpenDaylight*. Outras empresas, como Cisco e Brocade, também oferecem controladores habilitados para *openflow*, com Cisco XNC e Brocade Vyatta Controller.

As principais limitações das redes emuladas no *Mininet* referem-se à capacidade de banda disponível e CPU, que não podem exceder à capacidade e à banda disponível no servidor em que o *Mininet* está instalado. E ao fato também de o *Mininet* não executar aplicações que não sejam compatíveis com Linux.

O *Mininet* permite a criação de topologias complexas com vários *switches* e servidores, em árvore ou linear. O *Mininet* permite a construção de topologias padronizadas por meio de APIs em Python, base de construção do *Mininet*. As APIs em *Python* são usadas para orquestração, no entanto a emulação é em C compilado. A Tabela 3.1 ilustra algumas das principais APIs do Mininet.

O *Mininet* pode criar elementos de rede, personalizá-los, compartilhá-los com outras redes e realizar interações. Esses elementos incluem *hosts*, *switches*, controladores e links. Um *host* no *Mininet* é um processo simples com o seu próprio ambiente de rede em execução no sistema operacional. Cada um fornece aos processos interface de rede virtual de propriedade exclusiva, portas, endereços e tabelas de roteamento (como ARP e IP). Os *switches* criados pelo *Mininet* fornecem a mesma semântica de entrega de pacotes que seria fornecida por um *hardware* real de um *switch*. Todos os comandos e literatura sobre esse emulador está disponível em `http://mininet.org/`. Os códigos por nós desenvolvidos para esse trabalho estão disponíveis em `https://github.com/elvioktm/unb`.

3.3.2 Ferramentas *Iperf* e *ping*

O *Iperf* [52] é uma ferramenta para medições ativas da largura de banda máxima alcançável em redes IP [50]. Suporta o ajuste de vários parâmetros relacionados à temporização, a protocolos e a *buffers*. Para cada teste, ele informa a largura de banda. Esse tipo de fer-

Tabela 3.1: Principais APIs do Mininet [20]

API do Mininet	**Propósito**
net = Mininet()	net é um objeto de Mininet( ).
h1 = net.addHost('h1')	h1 é um objeto de Servidor( ).
s1 = net.addSwitch('s1')	s1 é um objeto de Switch( ).
c0 = net.addController('c0')	c0 é um controlador.
net.addLink(h1, s1)	cria um objeto Link( ).
net.start( )	Aloca os recursos computacionais e inicia a rede net.
h2.cmd ('python –m SimpleHTTPServer 80 &'	Na linha de comando de h2 roda a aplicação python de HTTP Server.
net.stop( )	Desaloca os recursos computacionais para a rede net.
Net = Mininet(link, TCLink, host=CPULimitedHost)	Habilita na rede net modelo de desempenho dos enlaces e classes de "hosts".
net.addLink(h2, s1, bw=10, delay='50ms')	Limita a banda e adiciona um atraso em um determinado enlace.
net.addHost('h1', cpu=.2)	Limita o processamento da CPU.
CLI (net)	Abre os terminais com a CLI em cada nó da rede net.
h2.cmd ('kill %python')	Envia um comando, aguarda uma saída e retorna.

ramenta de medição tem como objetivo injetar pacotes de teste na rede, para, a partir de então, medir o desempenho da rede avaliando como esse tráfego de teste se comporta. Ele suporta o ajuste de vários parâmetros relacionados a tempo, buffers e protocolos (TCP, UDP, SCTP com IPv4 e IPv6). Para cada teste, ele relata a largura de banda, a perda e outros parâmetros. O iPerf foi desenvolvido originalmente pela NLANR / DAST. O iPerf3 é desenvolvido principalmente pelo ESnet / Lawrence Berkeley National Laboratory. É liberado sob uma licença BSD de três cláusulas.

O *Iperf* é considerado um gerador de trafego permitindo que seja gerado fluxos de dados com características específicas para simular o acesso a uma aplicação, como o tráfego de voz ou vídeo, por exemplo. A maneira típica como o iperf é usado é primeiro iniciar um processo do iperf em execução no modo de servidor como receptor de tráfego e, em seguida, iniciar outro processo do iperf em execução no modo de cliente em outro host como remetente de tráfego.

A outra ferramenta que iremos utilizar para medir latência é o *ping*, um utilitário que trabalha em conjunto ou dentro do *Mininet* e usa o protocolo ICMP para testar a conectividade entre equipamentos. O ping é relatado quantitativamente como um tempo médio em milissegundos (ms). Quanto menor o ping, menor a latência e menor o atraso. ICMP (*Internet Control Message Protocol*) é um protocolo integrante do Protocolo IP, definido pelo RFC 792, utilizado para fornecer relatórios de erros ao emissor. O teste do ping é realizado na camada-3 do modelo OSI (ou Internet do TCP/IP) e não interessa quais

os dispositivos (roteadores e *switches*) que estão ao longo do caminho, dando o resultado do tempo normalmente em milissegundos. No entanto, nesse trabalho tivemos que medir em microssegundos pela velocidade das arquiteturas e pela capacidade do servidor escolhido.

3.3.3 Simulador EVE-NG

O EVE-NG (Ambiente Virtual Emulado - Próxima Geração) foi o simulador escolhido para encenar e comprovar os testes de performance derrubando um dos links redundantes [53], com isso testando falhas nos links. É um simulador visual e intuitivo, diferentemente do *Mininet* que é mais robusto para escalar redes, conforme pode ser observado na Figura 3.7.

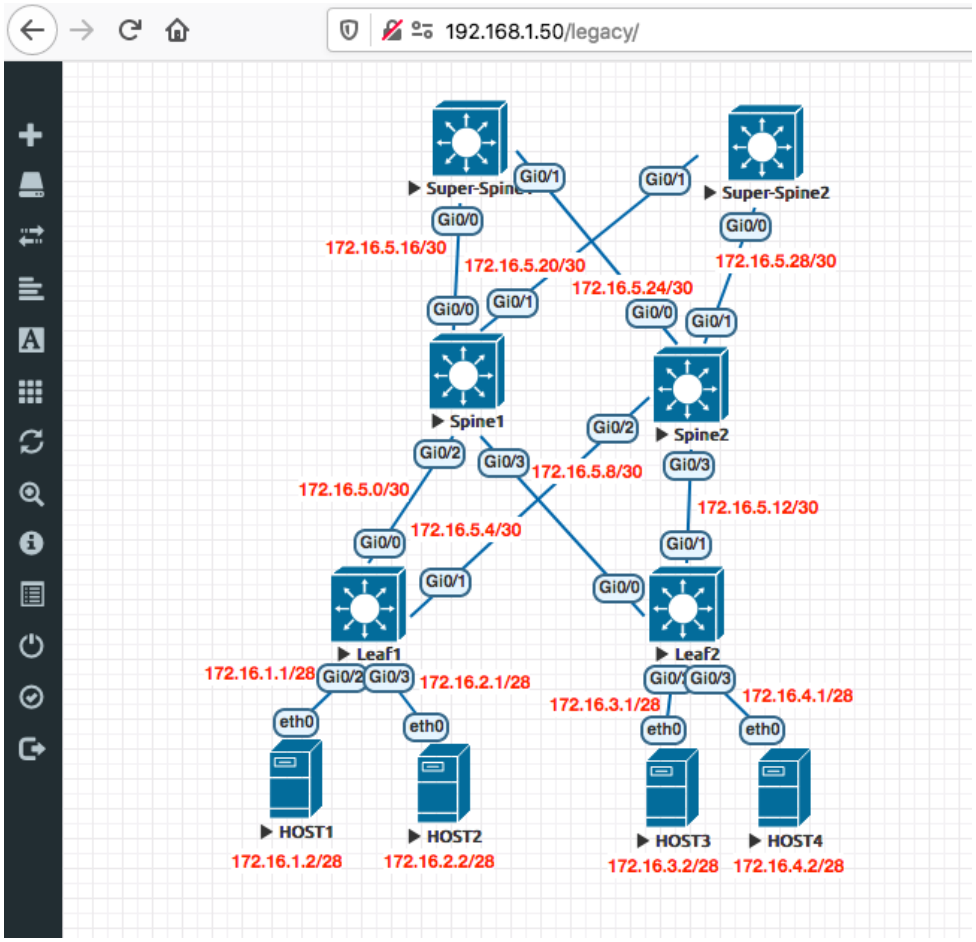

Figura 3.7: Arquitetura Leaf-Spine no EVE-NG.

O EVE-NG é um ambiente virtual de próxima geração, emulado, que permite criar um laboratório virtual completo com hardware e software de rede dos principais fabricantes mundiais. Trata-se de uma máquina virtual VMware completa e estável, baseada no Linux Ubuntu 16.04 x64. Todo o ambiente emulado é executado dentro dessa máquina

e acessado por meio de um navegador da *web* comum. Usando o navegador, é possível criar visualmente topologias de rede que incluem vários fabricantes, executar dispositivos emulados, conectar-se a eles com o console e também monitorar visualmente os recursos de computação consumidos do EVE-NG. Para a conexão com esses dispositivos, pode-se usar um cliente pré-instalado especial ou simplesmente a interface HTML5 usual. Dependendo do tipo de dispositivo, o console pode se conectar a eles via protocolos telnet, RDP ou VNC. E os dispositivos já configurados corretamente podem ser acessados via SSH, telnet, HTTP, HTTS, todos pela rede.

3.3.4 Simulador *Cisco Packet Tracer*

O Packet Tracer é uma ferramenta de simulador de rede muito popular, principalmente para estudantes acadêmicos. Embora o Packet Tracer seja uma ferramenta robusta de simulador que complementa sua experiência de aprendizado em rede de auto-estudo, ela não constitui uma solução completa e escalável [54]. No Packet Tracer existem vários dispositivos que podem ser usados na construção lógica da rede. Nesse sentido, é possível simular roteadores, *switches*, *access points*, *firewalls*, telefones, computadores e servidores, em que é possível habilitar e configurar serviços como https, DHCP, DNS e FTP. Fornecendo recursos de simulação, visualização, autoria, avaliação e colaboração para facilitar o ensino e o aprendizado de conceitos tecnológicos complexos.

Esse simulador permite criar uma rede com um número quase ilimitado de dispositivos, incentivando a prática, a descoberta e a solução de problemas. O ambiente de estudo baseado em simulação ajuda o pesquisador a desenvolver habilidades, como tomada de decisão e resolução de problemas.

Neste estudo, exploramos os recursos do Cisco Packet Tracer na simulação da arquitetura *Leaf-Spine*, conforme Figura 3.8, para provar os resultados dos testes de performance quando se derruba um dos links redundantes da topologia [55].

3.4 Considerações Finais

A proposta de comparação foi apresentada nesse capítulo, relatando as 5 arquiteturas de rede DCN que serão simuladas nesse trabalho. Para uma comparação justa entre as arquiteturas (leaf-spine, fat-tree, hybrid fat-tree, Facebook 4-Post e Facebook new fabric), o tamanho delas, as configurações dos dispositivos e os protocolos de rede, com as gerações de fluxo de tráfego, devem ser consistentes. A única variável que deve ser mudada é a topologia de rede. Apos apresentar as topologias e suas variáveis foi apresentado os aspectos comparativos levando em consideração os trabalhos relacionados anteriormente. Os simuladores utilizados Mininet, EVE-NG e Packet-Tracer também são apresentados

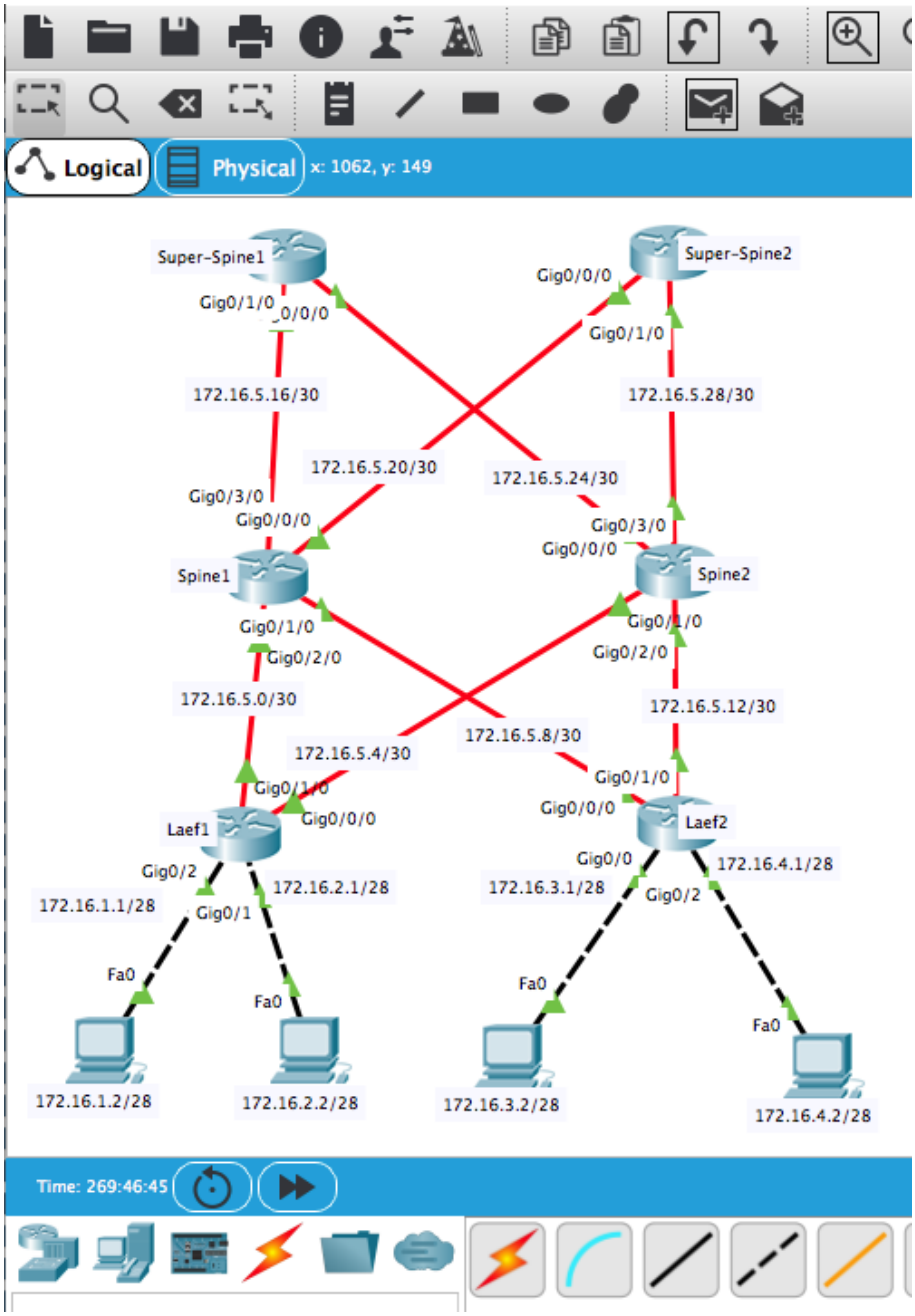

Figura 3.8: Arquitetura *Leaf-Spine* no Packet Tracer.

nessa seção. Já No próximo capítulo, os resultados experimentais coletados, considerando o cenário modelado e as arquiteturas descritas, serão apresentados e discutidos.

Capítulo 4

Avaliação

Neste capítulo avaliamos e comparamos as cinco arquiteturas DCN de acordo com diferentes aspectos comparativos. Para este propósito, cada avaliação é colocada em uma seção ordenada conforme os aspectos discutido na Seção 3.2. Aspectos esses, baseados no tamanho das arquiteturas que varia de 100 até 100.000 hosts, representando *mini (*até 100 *servers), small (*até 1.536 *servers), medium (*até 64.000 *servers) e* megasize *(*até 100.000 *servers) datacenters.* Utilizamos *switches ethernet* de 10.000Mbps com endereços IPs realistas. Para gerar fluxo de tráfego neles com o IPERF, selecionamos aleatoriamente a comunicação entre pares em todos os nós, com cada par consistindo de um remetente e um receptor.

Cada par envia simultaneamente um fluxo de dados de 10Mbps do seu remetente para o seu receptor, e a simulação para as arquiteturas é executada por 100 segundos. O padrão de fluxo de tráfego para as arquiteturas segue um comportamento com distribuição aleatória linear, o qual modela razoavelmente os fluxos de tráfego no mundo real dos DCNs [34]. Os valores reportados nos gráficos desta seção representam a média de todas as comunicações entre os pares de cada cenário. Por exemplo, no cenário com 50 servidores, o valor reportado representa a média da latência ou taxa de transferência dos 50 pares definidos para comunicação.

4.1 Resultados Experimentais e Análise

Com base nos resultados experimentais e análise media dos rendimentos, serão apresentadas nas próximas subseções as comparações dos gráficos de latência e taxa de transferência. Por fim, também é apresentada uma análise a respeito de falhas de links.

4.1.1 Latência

Na tentativa de expressar visualmente os dados e valores coletados no simulador *Mininet*, facilitando a compreensão com os gráficos de linha que são compostos por dois eixos - um vertical e outro horizontal - e por uma linha que mostra a evolução ou crescimento dos servidores em um DCN. No eixo vertical, os valores estão em microssegundos ou 10^{-6} de segundo. Já no eixo horizontal estão as quantidades de servidores, correspondendo ao tamanho dos DCNs em *mini* com até *mega datacenters*, como mostrado na Seção 3.3. As linhas correspondem à evolução das cinco arquiteturas estudadas, conforme mostrado na Seção 3.1 e também no trabalho de Reyes e Bauschert [2], que reporta os custos dessas arquiteturas.

Nos gráficos de latência podemos observar que as arquiteturas *Fat-Tree* e *Hybrid Fat-Tree* demonstraram ser mais lentas que as outras três arquiteturas (*Leaf-Spine, Facebook-4-Post e Facebook-New-Fabric*), em média 40 microssegundos, como podemos ver nas figuras 4.1, 4.2, 4.3 e 4.4. As três arquiteturas baseadas em *Leaf-Spine* (*Leaf-Spine, Facebook-4-Post e Facebook-New-Fabric*) se apresentaram com menos latência que as duas baseadas em *Fat Tree* (*Fat-Tree e Hybrid-Fat-Tree*). Esse melhor desempenho se deve ao fato de que as arquiteturas baseadas em *Leaf-Spine* têm um equilíbrio de carga entre vários caminhos com um custo igual, sendo todos os *switches* ligados com todos, chamados de *Equal Cost Multipath Load Balance (ECMP)*. Como já observado, os valores reportados nos gráficos representam a média da latência obtida por cada par de processos, de acordo com o número de servidores. Para efeitos de comprovação estatística, repetimos 100 execuções para o cenário com 50 servidores e obtivemos valores próximos aos apresentados no gráfico, com desvio padrão sempre inferior a 0.4 microssegundos.

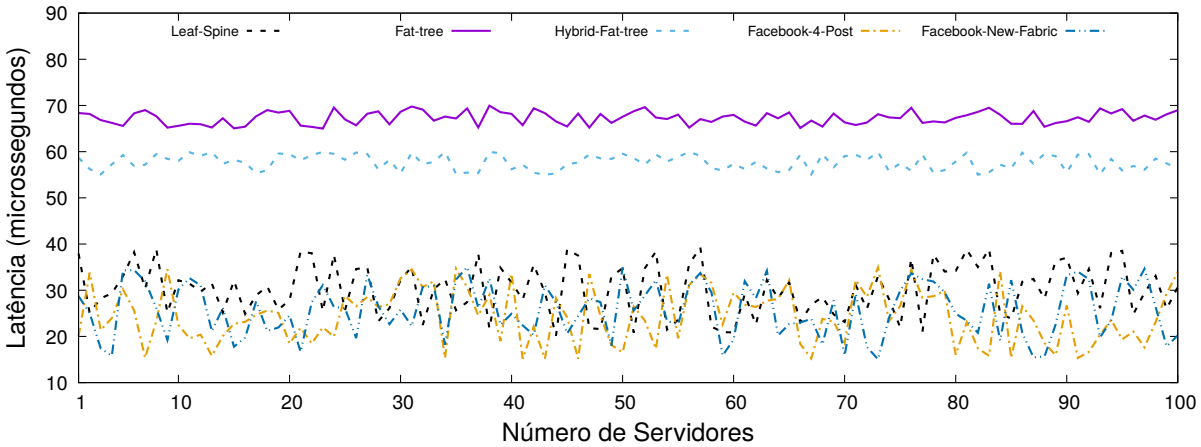

Figura 4.1: Latência em microssegundos para um DCN *mini*.

Podemos observar nos gráficos das figuras 4.1, 4.2, 4.3 e 4.4, que, mesmo com o crescimento linear do tráfego (ver Seção 3.2), não obtivemos alteração significativa no desvio

do comportamento da latência nas topologias. Tal situação pode ser devido à alta largura de banda disponibilizada de 10Gbs e ao simulador que se encontra em um servidor de alta performance, como mencionado na Seção 3.3. As duas arquiteturas (*Fat-Tree e Hybrid-Fat-Tree*) com mais lentidão variaram seus valores entre 55μS e 80μS. As três arquiteturas (*Leaf-Spine, Facebook-4-Post e Facebook-New-Fabric*) com mais velocidade variaram seus valores entre 20μS e 40 μS. Pode-se também argumentar que, em grande medida, o desempenho em relação ao atraso de pacotes das cinco arquiteturas é independente do tamanho da rede. Os resultados aqui mostraram que as arquiteturas baseadas em *Leaf-Spine* com ECMP executam ligeiramente melhor que as arquiteturas baseadas em árvore ou *Fat Tree*, em termos médios de atraso no pacote.

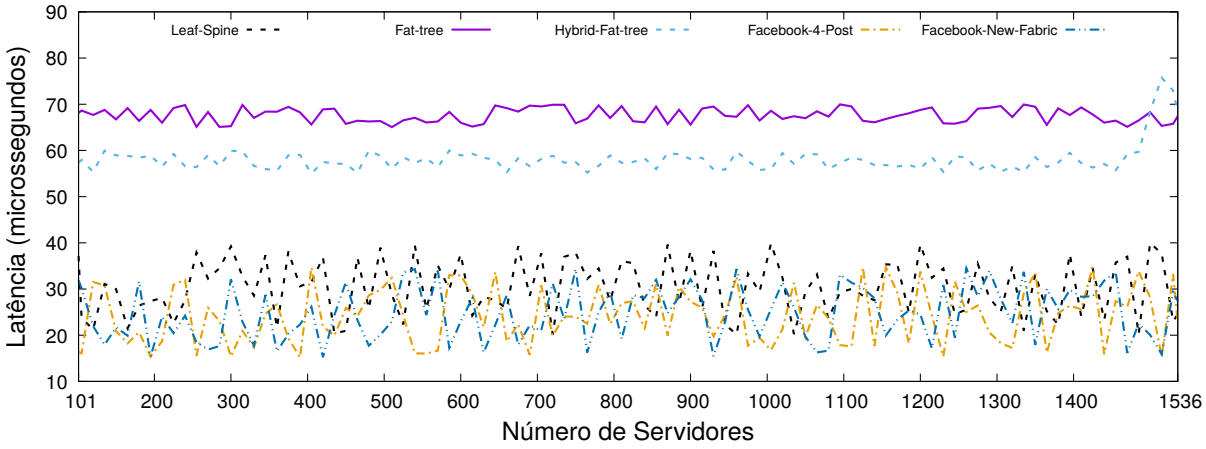

Figura 4.2: Latência em microssegundos para um DCN *small*.

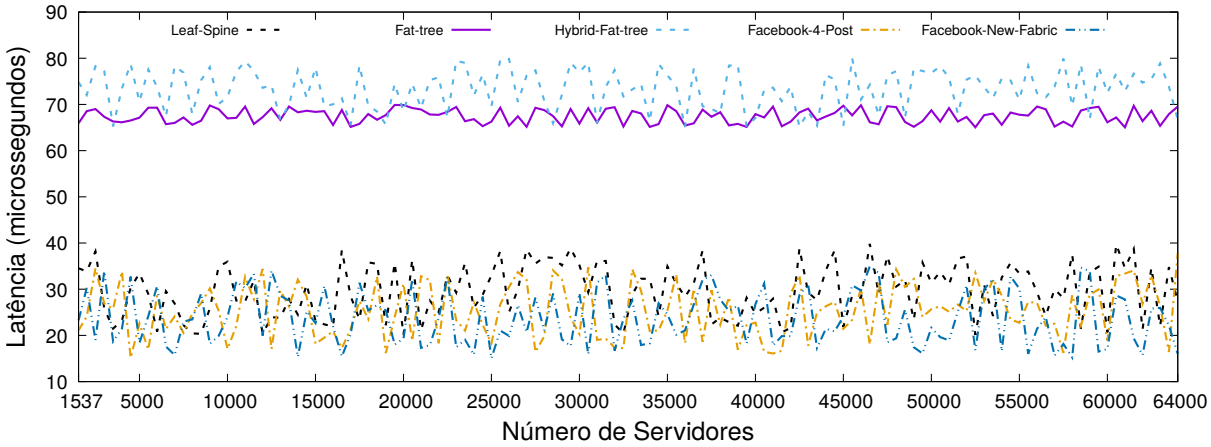

Figura 4.3: Latência em microssegundos para um DCN *medium*.

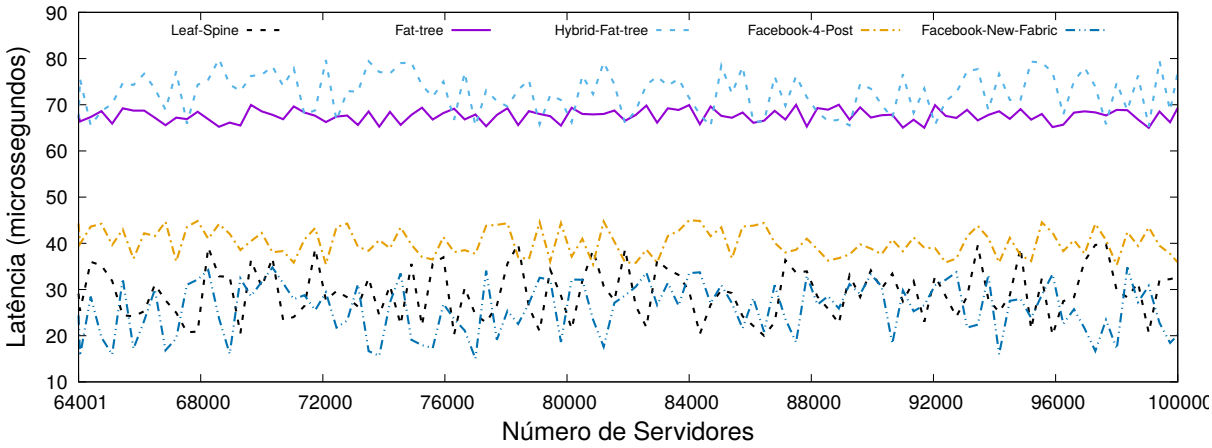

Figura 4.4: Latência em microssegundos para um DCN *mega*.

4.1.2 Taxa de Transferência

A taxa de transferência (vazão ou *throughput*) é apresentada nesta seção por gráficos de linha mostrados compostos por dois eixos - um vertical e outro horizontal **-** e por uma linha que mostra a evolução ou crescimento dos servidores em um DCN. No eixo vertical, os valores estão em *megabytes*, e no eixo horizontal estão as quantidades de servidores correspondentes ao tamanho dos DCNs. As linhas correspondem à evolução das 5 arquiteturas estudadas.

O desempenho da taxa de transferência foi medido levando em consideração as métricas mostradas na Seção 3.2 usando o *Iperf* [50]. Com base nos resultados médios da taxa de transferência mostrados nas figuras 4.5, 4.6, 4.7 e 4.8, podemos observar que as três arquiteturas baseadas em *Leaf-Spine* (*Leaf-Spine, Facebook-4-Post e Facebook-New-Fabric*) oferecem consistentemente mais desempenho da taxa de transferência do que as duas arquiteturas baseadas em árvore (*Fat-Tree e Hybrid-Fat-Tree*), mesmo quando o número de servidores aumenta de um DCN *mini* até um DCN *mega*. Como já observado, os valores reportados nos gráficos representam a média da taxa de transferência obtida por cada par de processos, de acordo com o número de servidores. Para efeitos de comprovação estatística, repetimos 100 execuções para o cenário com 50 servidores e obtivemos valores próximos aos apresentados no gráfico, com desvio padrão inferior a 2 MB.

As três arquiteturas baseadas em *Leaf-Spine* variam a taxa de transferência entre 330MB **e** 460MB, enquanto que as duas topologias baseadas em árvore variam de 190MB até 250MB. As arquiteturas baseadas em *Leaf-Spine* demostraram ter melhor desempenho que a *Fat-Tree e Hybrid-Fat-Tree*, devido à arquitetura de rede ter vários caminhos possíveis para enviar um fluxo de tráfego entre os servidores usados na simulação pelo *Mininet*. Sendo assim, temos menor congestionamento de tráfego e maior largura de banda disponível.

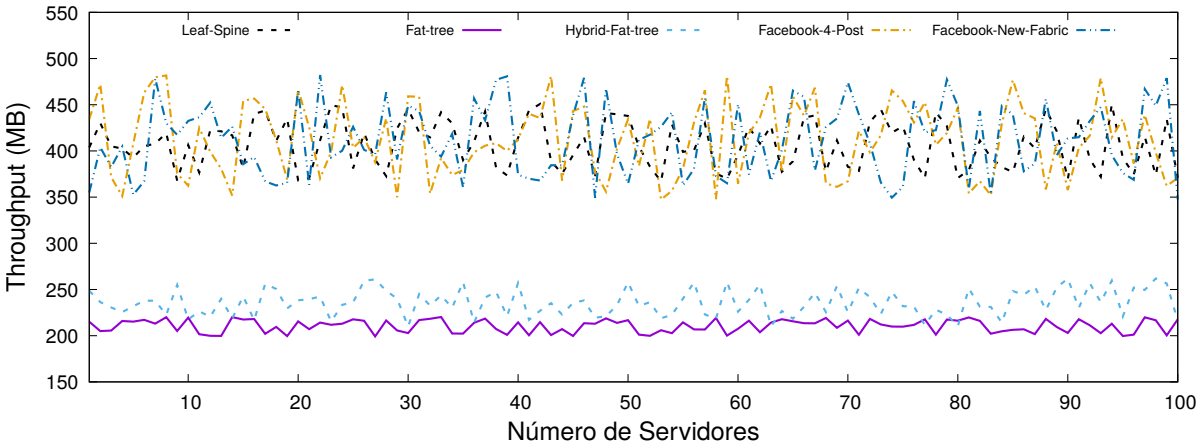

Figura 4.5: Taxa de transferência em mega bytes para um DCN *mini*.

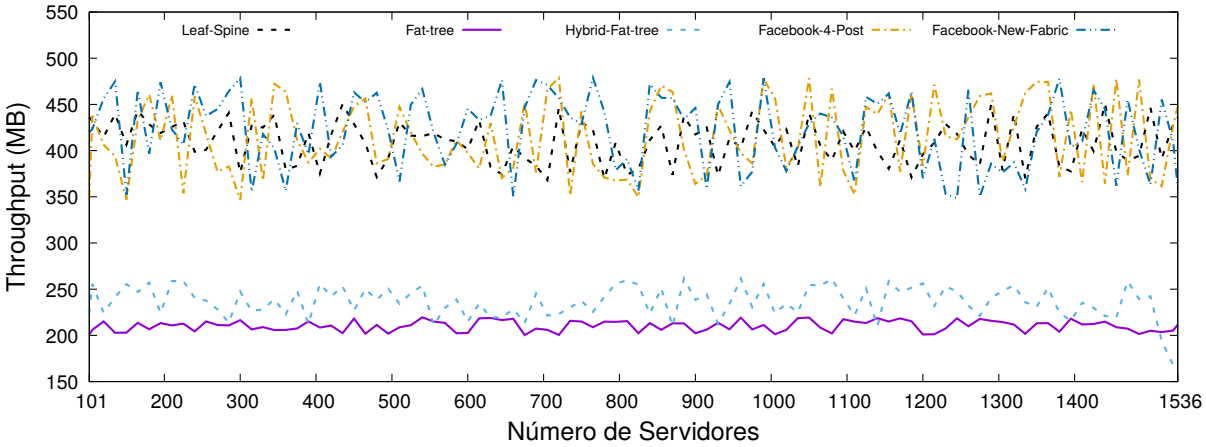

Figura 4.6: Taxa de transferência em mega bytes para um DCN *small*.

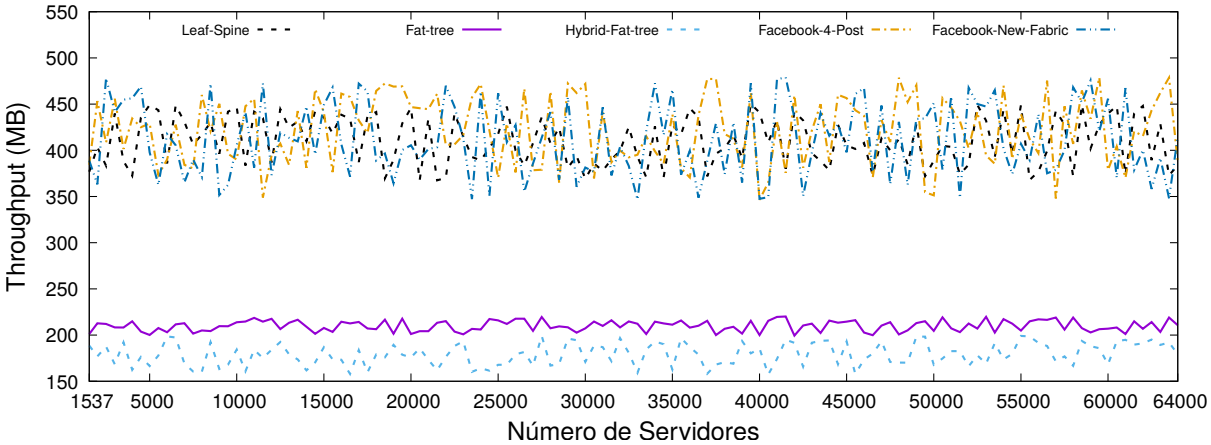

Figura 4.7: Taxa de transferência em mega bytes para um DCN *medium*.

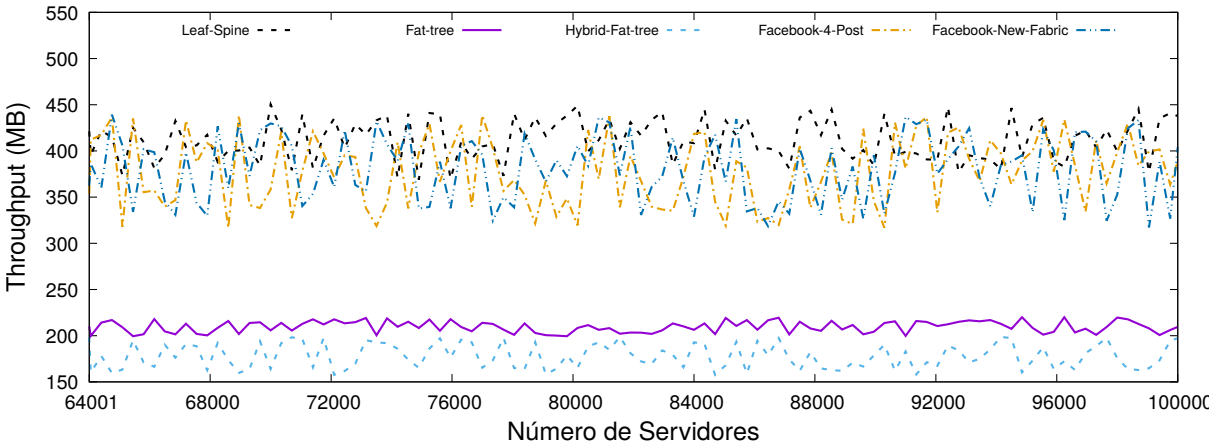

Figura 4.8: Taxa de transferência em mega bytes para um DCN *mega*.

4.1.3 Teste de desempenho com falha de link

Para avaliar o efeito de falhas na topologia de rede, simulamos as topologias e obtivemos os valores das métricas analisadas considerando dois tipos de falha: transitória (falha presente por um período curto de tempo fixo) e permanente (falha que dura todo o tempo de simulação).

Um *design* prático deve ter alguns recursos. Primeiro, ele deve aproveitar os protocolos disponíveis nos *switches* de hoje, significando que *switches* devem sempre encaminhar o tráfego pelos caminhos mais curtos. Quando vários caminhos mais curtos estão disponíveis, a comutação ou o roteamento de caminhos múltiplos de custo igual (ECMP) pode ser utilizado para obter equilíbrio de tráfego na rede [56]. Além disso, o próprio ECMP pode evitar links com falha imediata quando outros caminhos mais curtos estiverem disponíveis. Segundo, a rede deve se recuperar de qualquer falha de link único, imediatamente depois de a falha ser detectada.

O modelo de arquitetura de rede adotado no estudo garante que a rede ainda estará conectada mesmo que ocorra qualquer falha de link único. Essa é a base das DCNs projetadas para os *datacenters* com esquema de recuperação de falhas. A resiliência da rede está se tornando um dos principais aspectos no planejamento e operação de redes IP. No caso de ocorrer uma falha, as redes IP podem se recuperar, desde que seja fornecida conectividade física suficiente. Novos conceitos para acelerar o desempenho de recuperação de redes de dados, como o ECMP, estão sendo usados nas redes, tornando as quedas de links redundantes imperceptíveis para o usuário final. As informações em todos os *switches* da rede para gerar decisões consistentes de comutação devem ser as mesmas. No caso de uma falha no link, o *switch* alterna automaticamente os fluxos para um dos links restantes da arquitetura de vários caminhos. O fato de a reação ser local é a razão pela qual esse esquema é muito rápido e imperceptível para o usuário final.

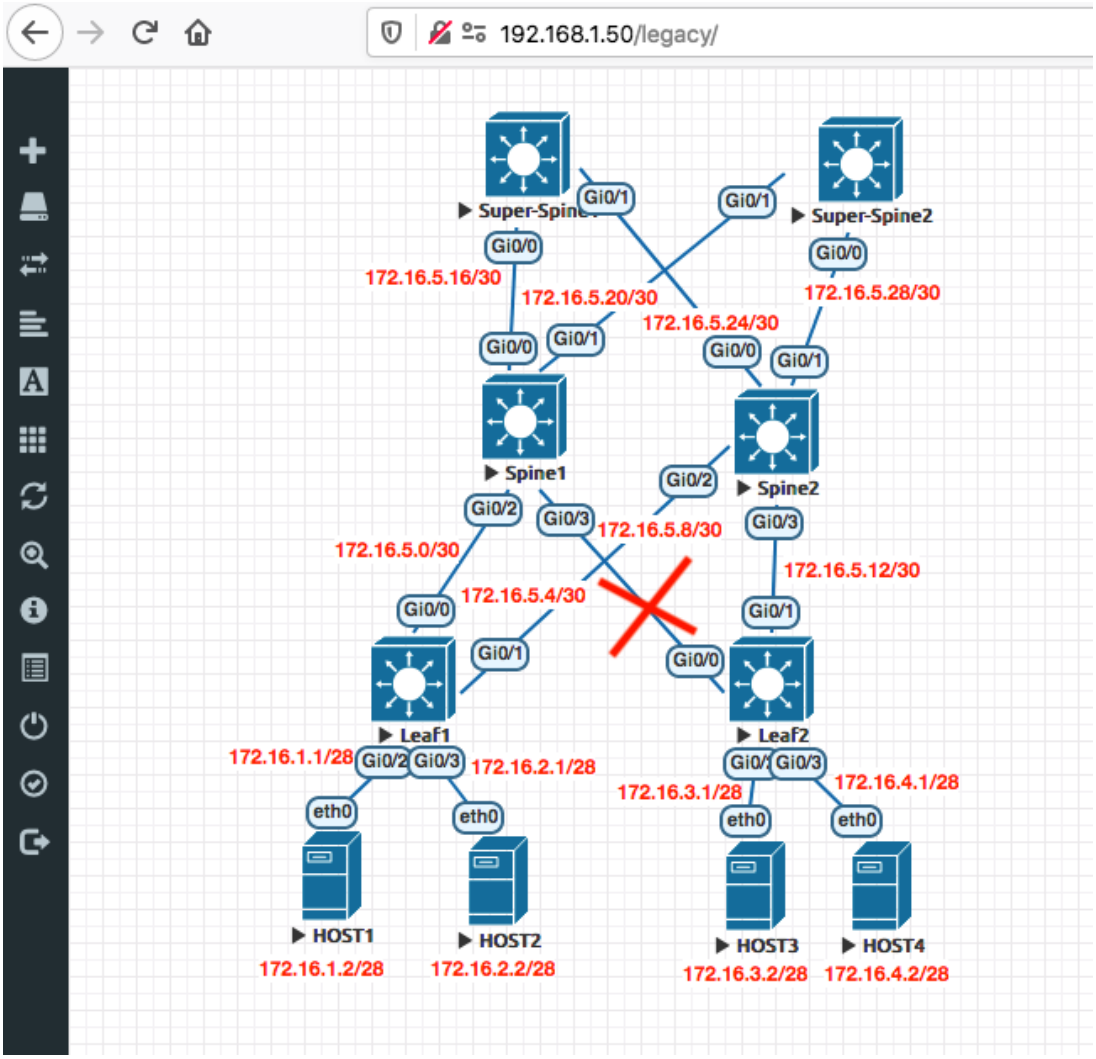

Figura 4.9: Simulador EVE-NG Arquitetura Leaf-Spine com falha no link.

O mecanismo ECMP, usado por fabricantes como Cisco, com *fabric-path*, Dell com VLT ou funcionalidades de cascateamento, permite uma distribuição dos pacotes em vários links de saída em cada *switch* da rede entre a origem e o destino final. Isso, na medida em que existam vários caminhos mais curtos (na métrica de custo selecionado para a comutação) entre o nó de distribuição considerado e o destino final. Assim que um *switch* detectar (principalmente por mecanismos de hardware) a não operacionalidade de um link de saída, ele alternará localmente para encaminhar os pacotes nos links restantes, com o mesmo custo para o destino. A grande vantagem do ECMP é a sua integração em vários protocolos e tecnologias, tornando-o prontamente disponível nos maiores fabricantes de rede. A principal limitação dessa abordagem resulta do fato de que as redes devem ter custo igual para todos os destinos, conforme se observou na comparação das arquiteturas estudadas e simuladas nesse trabalho.

A região de falha escolhida foi um dos links redundantes (veja exemplo na Figura 4.9),

permitindo que o pacote possa encontrar um caminho alternativo a seguir. Os testes da simulação feitas no *Mininet* não tiveram alteração significativa dos resultados discutidos nas seções 4.1.1 e 4.1.2, por isso não serão repetidos nesta seção. Devido ao ECMP e aos links redundantes com métricas iguais nas arquiteturas, a falha de um link é imperceptível nos ambientes testados. Essas simulações estão documentadas em `https://github.com/elvioktm/unb`. Para comprovar a validade destes resultados similares aos anteriores, escolhemos dois simuladores, EVE-NG [57] e Packet Tracer [54], e os resultados comprovaram que a falha de um link nas arquiteturas redundantes estudadas não resulta em parada nos pacotes perceptível ao usuário, pois são comutados rapidamente para o outro link redundante.

O comportamento das topologias com falha transitória e permanente, no caso de tráfego individual entre os dois *hosts* localizados na rede usando métricas iguais para os destinos, não teve influência nem degradação na transferência de dados e na latência. O comportamento foi monitorado na *interface* (porta do *switch*), na topologia e no resultado do *ping* no *host*.

No Packet Tracer, mesmo usando roteadores da Cisco em camada 3 com protocolo OSPF, os resultados foram similares ao Mininet e ao EVE-NG, i.e., quando se tem uma queda é imperceptível para o usuário final. Este comportamento ocorre sempre, a não ser que a falha ocorra nos dois links ou no próprio roteador, evidenciando mais uma vez que a arquitetura Leaf-Spine cumpre o seu papel de redundância mesmo com protocolos padrão de mercado.

Fizemos testes similares também em duas redes *Leaf-Spine* reais em produção : redes do STF (Superior Tribunal de Federal) e da empresa Comando Autopeças. Em ambas, o mesmo comportamento foi demonstrado, sendo, inclusive, imperceptível a queda para o usuário final de um link redundante. Essas duas redes usam *switches* com redundância da DELL com VLT.

4.2 Considerações Finais

Neste capítulo, relatamos o desempenho comparativo de algumas topologias de rede de *datacenters* conhecidas por meio de um estudo de simulação e também analisamos falhas de links. Realizamos um estudo comparativo para avaliar as principais topologias de DCN com os simuladores mais usados academicamente e ambientes de tráfego entre redes. Com base nos resultados experimentais e análise media dos rendimentos, esse capitulo apresenta o resultado das comparações dos gráficos de latência e taxa de transferência. Por fim, também é a presentada uma análise a respeito de falhas de links. O próximo capítulo apresenta as conclusões deste trabalho e apresentamos as sugestões para trabalhos futuros.

Capítulo 5

Conclusões e Trabalhos Futuros

Neste estudo, apresentamos uma comparação de cinco arquiteturas de rede em *datacenters*: *leaf-spine, fat-tree, hybrid fat-tree, Facebook 4-Post e Facebook new fabric*. Essas arquiteturas foram programadas e simuladas no *Mininet* em um ambiente experimental, composto por um servidor Dell Poweredge com 16 CPUs, 72Gb de RAM e 6Tb de HD com virtualizador da Citrix XenServer 7.5. Usando esse ambiente experimental, avaliamos qual arquitetura tem o melhor desempenho em termos de latência e de taxa de transferência, simulando um crescimento linear dos servidores ou *hosts* de um DCN até 100.000. Avaliações envolvendo as cinco arquiteturas em termos de latência e de taxa de transferência permitiram descobrir que as três arquiteturas baseadas em *Leaf-Spine* têm quase o dobro de desempenho com relação às duas arquiteturas baseadas em árvore.

Também simulamos as topologias em alguns cenários de falha de links. Concluise que todas as topologias consideradas têm capacidade de recuperação de falhas, pois vários caminhos estão disponíveis nas topologias. No entanto, o desempenho depende de hardware, softwares e protocolos utilizados.

Através da combinação da análise dos resultados desse trabalho, com o trabalho de Reyes e Bauschert [2] sobre os custos dessas mesmas cinco arquiteturas, leva-nos à conclusão de que as arquiteturas de melhor custo-benefício são as baseadas em *Leaf-Spine*. Neste trabalho verificamos que estas arquiteturas apresentam melhor desempenho, enquanto que no estudo de Reyes e Bauschert foi verificado que o custo monetário são semelhantes entre as arquiteturas, embora em alguns cenários as arquiteturas baseadas em árvore também apresentam um maior custo. Nessa comparação, ficou evidente que as arquiteturas com múltiplos caminhos para um mesmo destino ou *Equal Cost Multipath Load Balance (ECMP)* têm mais desempenho e menos custo financeiro, além de suportar falhas de links.

Como trabalhos futuros, sugere-se explorar diferentes arquiteturas com mudanças de parâmetros, como TCP, UDP janelamento, controle de fluxo e até mesmo a simulação de

alta disponibilidade das redes estudadas. Além de verificar se o cenário seria diferente para padrões de tráfego maiores com falhas durante a operação e esgotamento de banda, i.e., se neste cenário iria ocorrer descartes de pacotes.

Referências

[1] Lebiednik, Brian, Aman Mangal e Niharika Tiwari: *A survey and evaluation of data center network topologies*. arXiv preprint arXiv:1605.01701, 2016. v, 26, 30

[2] Reyes, Ronald Romero e Thomas Bauschert: *Infrastructure cost comparison of intra-data centre network architectures*. Em *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, páginas 1–7. IEEE, 2018. vi, viii, xi, 1, 2, 26, 27, 32, 46, 53

[3] Clos, Charles: *A study of non-blocking switching networks*. Bell System Technical Journal, 32(2):406–424, 1953. xi, 4

[4] Leiserson, Charles E: *Fat-trees: universal networks for hardware-efficient supercomputing*. IEEE transactions on Computers, 100(10):892–901, 1985. xi, 5, 27

[5] Breznay, Peter Thomas e Mario Alberto Lopez: *A class of static and dynamic hierarchical interconnection networks*. Em *1994 International Conference on Parallel Processing Vol. 1*, volume 1, páginas 59–62. IEEE, 1994. xi, 5, 6

[6] Barabási, Albert László e Réka Albert: *Emergence of scaling in random networks*. science, 286(5439):509–512, 1999. xi, 6, 7

[7] Al-Fares, Mohammad, Alexander Loukissas e Amin Vahdat: *A scalable, commodity data center network architecture*. Em *ACM SIGCOMM Computer Communication Review*, volume 38, páginas 63–74. ACM, 2008. xi, 7, 27, 30, 34

[8] Guo, Chuanxiong, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang e Songwu Lu: *Dcell: a scalable and fault-tolerant network structure for data centers*. Em *ACM SIGCOMM Computer Communication Review*, volume 38, páginas 75–86. ACM, 2008. xi, 8, 27, 30

[9] Guo, Chuanxiong, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang e Songwu Lu: *Bcube: a high performance, server-centric network architecture for modular data centers*. ACM SIGCOMM Computer Communication Review, 39(4):63–74, 2009. xi, 9

[10] Wu, Haitao, Guohan Lu, Dan Li, Chuanxiong Guo e Yongguang Zhang: *Mdcube: a high performance network structure for modular data center interconnection*. Em *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, páginas 25–36. ACM, 2009. xi, 10

[11] Gyarmati, László e Tuan Anh Trinh: *Scafida: A scale-free network inspired data center architecture*. ACM SIGCOMM Computer Communication Review, 40(5):4–12, 2010. xi, 11

[12] Guo, Deke, Tao Chen, Dan Li, Yunhao Liu, Xue Liu e Guihai Chen: *Bcn: Expansible network structures for data centers using hierarchical compound graphs*. Em *2011 Proceedings IEEE INFOCOM*, páginas 61–65. IEEE, 2011. xi, 12

[13] Singla, Ankit, Chi Yao Hong, Lucian Popa e P Brighten Godfrey: *Jellyfish: Networking data centers randomly*. Em *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, páginas 225–238, 2012. xi, 12, 13

[14] Liu, Vincent, Daniel Halperin, Arvind Krishnamurthy e Thomas Anderson: *F10: A fault-tolerant engineered network*. Em *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, páginas 399–412, 2013. xi, 13

[15] Alizadeh, Mohammad e Tom Edsall: *On the data path performance of leaf-spine datacenter fabrics*. Em *2013 IEEE 21st Annual Symposium on High-Performance Interconnects*, páginas 71–74. IEEE, 2013. xi, 14, 25

[16] Farrington, Nathan e Alexey Andreyev: *Facebook's data center network architecture*. Em *2013 Optical Interconnects Conference*, páginas 49–50. Citeseer, 2013. xi, 15, 35, 36

[17] Andreyev, Alexey: *Introducing data center fabric, the next-generation facebook data center network*. Facebook, Nov, 14:13, 2014. xi, 16, 17, 30

[18] Singh, Arjun, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano *et al.*: *Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network*. ACM SIGCOMM computer communication review, 45(4):183–197, 2015. xi, 18

[19] CCNP, CISCO Certificação: *Projeto de rede hierárquico*. `https://static-course-assets.s3.amazonaws.com/ScaN503/pt/index.html#1.1.1.3`, 2017. xi, 20, 21

[20] Mininet: *An instant virtual network on your laptop (or other pc)*. `http://http://mininet.org/`, 2018. xiii, 41

[21] Zhang, Qi, Lu Cheng e Raouf Boutaba: *Cloud computing: state-of-the-art and research challenges*. Journal of internet services and applications, 1(1):7–18, 2010. 1

[22] IEEE By Randy H. Katz spectrum: *Tech titans building boom google, microsoft, and other internet giants race to build the mega data centers that will power cloud computing*. `https://spectrum.ieee.org/green-tech/buildings/tech-titans-building-boom`, 2009. 1

[23] VentureBeat, JORDAN NOVET: *Inside project mcqueen, apple's plan to build its own cloud*. https://venturebeat.com/2016/03/17/apple-cloud-project-mcqueen/, 2016. 1

[24] Dropobox, Akhil Gupta by: *Scaling to exabytes and beyond*. https://blogs.dropbox.com/tech/2016/03/magic-pocket-infrastructure/, 2016. 1

[25] COSTA, M: *Gonçalves da*. História do bispado e cidade de Lamego, 3, 2000. 20

[26] CISCO: *Campus resumo do design*. https://www.cisco.com/c/dam/r/pt/br/internet-of-everything-ioe/assets/pdfs/en-05_campus_wireless_wp_cte_pt_br_42333.pdf, 2014. 20, 23

[27] CCNP, CISCO Certificação: *Projeto de rede hierárquico*. https://static-course-assets.s3.amazonaws.com/RSE6/pt/index.html#4.1.1.5, 2017. 21

[28] Costa, Celso José: *Modelos de educação superior a distância e implementação da universidade aberta do brasil*. Brazilian Journal of Computers in Education, 15(2), 2007. 22

[29] CCNP, CISCO Certificação: *Nucelo recolhido*. https://static-course-assets.s3.amazonaws.com/RSE6/pt/index.html#4.1.1.4, 2017. 23

[30] DELL: *Leaf-spine deployment and best practices guide for greenfield deployments*. https://www.dell.com/support/article/br/pt/brbsdt1/sln314312/leaf-spine-deployment-and-best-practices-guide-for-greenfield-deployments, 2017. 24

[31] Li, Xiaolin: *An Energy Aware Green Spine Switch Management System in Spine-Leaf Datacenter Networks*. Tese de Doutoramento, Carleton University, 2015. 24

[32] Roberts, Errol e Loukas Paraschis: *The role of optical interconnections in data-center architecture evolution*. Em *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013*, páginas 1–3. IEEE, 2013. 25

[33] Benson, Theophilus, Aditya Akella e David A Maltz: *Network traffic characteristics of data centers in the wild*. Em *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, páginas 267–280. ACM, 2010. 26

[34] Benson, Theophilus, Ashok Anand, Aditya Akella e Ming Zhang: *Understanding data center traffic characteristics*. Em *Proceedings of the 1st ACM workshop on Research on enterprise networking*, páginas 65–72. ACM, 2009. 26, 38, 45

[35] Wu, Kaishun, Jiang Xiao e Lionel M Ni: *Rethinking the architecture design of data center networks*. Frontiers of Computer Science, 6(5):596–603, 2012. 26

[36] Wu, Caesar e Rajkumar Buyya: *Cloud Data Centers and Cost Modeling: A complete guide to planning, designing and building a cloud data center*. Morgan Kaufmann, 2015. 26

[37] Bilal, Kashif, Samee Ullah Khan, Joanna Kolodziej, Limin Zhang, Khizar Hayat, Sajjad Ahmad Madani, Nasro Min-Allah, Lizhe Wang e Dan Chen: *A comparative study of data center network architectures.* Em *ECMS*, páginas 526–532, 2012. 27, 34

[38] Siraj, Saba, A Gupta e Rinku Badgujar: *Network simulation tools survey.* International Journal of Advanced Research in Computer and Communication Engineering, 1(4):199–206, 2012. 27

[39] Zhang, Hailong, Xiao Guo, Jinyao Yan, Bo Liu e Qianjun Shuai: *Sdn-based ecmp algorithm for data center networks.* Em *2014 IEEE Computers, Communications and IT Applications Conference*, páginas 13–18. IEEE, 2014. 27

[40] Wong, Daji, Kiam Tian Seow, Chuan Heng Foh e Renuga Kanagavelu: *Towards reproducible performance studies of datacenter network architectures using an open-source simulation approach.* Em *2013 IEEE Global Communications Conference (GLOBECOM)*, páginas 1373–1378. IEEE, 2013. 28

[41] Binkert, Nathan, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti *et al.*: *The gem5 simulator.* ACM SIGARCH Computer Architecture News, 39(2):1–7, 2011. 30

[42] De Oliveira, Rogério Leão Santos, Christiane Marie Schweitzer, Ailton Akira Shinoda e Ligia Rodrigues Prete: *Using mininet for emulation and prototyping software-defined networks.* Em *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, páginas 1–6. IEEE, 2014. 30, 39

[43] Dhiab, Imen, Yosra Barouni, Sofiane Khalfallah e Jaleleddine Ben Hadj Slama: *Datacenter network architecture performance analysis using content centric communication.* Em *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, páginas 1–4. IEEE, 2016. 30

[44] Dewangan, A: *Brocade data center fabric architectures: Solution design guide.* Brocade communication systems, 2016. 34

[45] Balanici, Mihail e Stephan Pachnicke: *Traffic-modeling-based design and evaluation of a hybrid electro-optical intra-data center network.* Em *Photonic Networks; 18. ITG-Symposium*, páginas 1–8. VDE, 2017. 34, 35

[46] Kachris, Christoforos e Ioannis Tomkos: *A survey on optical interconnects for data centers.* IEEE Communications Surveys & Tutorials, 14(4):1021–1036, 2012. 35

[47] Wang, Guohui, David G Andersen, Michael Kaminsky, Michael Kozuch, TS Ng, Konstantina Papagiannaki, Madeleine Glick e Lily Mummert: *Your data center is a router: The case for reconfigurable optical circuit switched paths.* 2006. 35

[48] Keti, Faris e Shavan Askar: *Emulation of software defined networks using mininet in different simulation environments.* Em *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*, páginas 205–210. IEEE, 2015. 37, 39

[49] Kaur, Sukhveer, Japinder Singh e Navtej Singh Ghumman: *Network programmability using pox controller*. Em *ICCCS International Conference on Communication, Computing & Systems, IEEE*, volume 138, 2014. 37

[50] Passos, Diego, Douglas Vidal Teixeira, Débora C Muchaluat-Saade, Luiz C Schara Magalhães e Célio Albuquerque: *Mesh network performance measurements*. Em *5th International Information and Telecommunication Technologies Symposium*, volume 2006, 2006. 37, 40, 48

[51] Yan, Lisa e Nick McKeown: *Learning networking by reproducing research results*. ACM SIGCOMM Computer Communication Review, 47(2):19–26, 2017. 39

[52] Diniz, Pedro Henrique e Nilton Alves Junior: *Ferramenta iperf: geração e medição de tráfego tcp e udp*. NOTAS TÉCNICAS, 4(2), 2014. 40

[53] Korniyenko, Bogdan e Liliia Galata: *Implementation of the information resources protection based on the centos operating system*. Em *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, páginas 1007–1011. IEEE, 2019. 42

[54] Trabelsi, Zouheir e Heba Saleous: *Exploring the opportunities of cisco packet tracer for hands-on security courses on firewalls*. Em *2019 IEEE Global Engineering Education Conference (EDUCON)*, páginas 411–418. IEEE, 2019. 43, 52

[55] Chou, Te Shun, S Baker e Miguel Vega-Herrera: *A comparison of network simulation and emulation virtualization tools*. Em *Proc. ASEE Annu. Conf. Expo.*, páginas 1–9, 2016. 43

[56] Iselt, Andreas, A Kirstadter, A Pardigon e Thomas Schwabe: *Resilient routing using mpls and ecmp*. Em *2004 Workshop on High Performance Switching and Routing, 2004. HPSR.*, páginas 345–349. IEEE, 2004. 50

[57] Korniyenko, BY: *Open systems interconnection model investigation from the viewpoint of information security/b. korniyenko, o. yudin, e. novizkij*. The Advanced Science Journal, (8):53–56, 2013. 52