Rafael Martins da Cruz

Emulação de uma Solução SD-WAN para Integração dos Campi da UEMA

São Luís

2021

Rafael Martins da Cruz

Emulação de uma Solução SD-WAN para Integração dos Campi da UEMA

Projeto apresentado ao curso de Mestrado Profissional em Engenharia da Computação e Sistemas na Universidade Estadual do Maranhão como pré-requisito para a obtenção do título de mestre sob orientação do Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva e coorientação do Prof. Dr. Luis Carlos Costa Fonseca.

Universidade Estadual do Maranhão

Centro de Ciências Tecnológicas

Programa de Pós-graduação em Engenharia de Computação e Sistemas

Orientador: Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva

Coorientador: Prof. Dr. Luis Carlos Costa Fonseca

São Luís

2021

Cruz, Rafael Martins da.
Emulação de uma solução SD-WAN para integração dos campi da UEMA / Rafael Martins da Cruz. – São Luís, 2021.

71 f

Dissertação (Mestrado) – Curso de Engenharia de Computação e Sistemas, Universidade Estadual do Maranhão, 2021.

Orientador: Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva.
Coorientador: Prof. Dr. Luís Carlos Costa Fonseca.

1.Redes definidas por software. 2.SDN. 3.OpenFlow. 4.Redes definidas por software de áreas de longa distância. 5.SD-WAN. I.Título.

CDU: 004.735.378.4(812.1)

Elaborado por Giselle Frazão Tavares - CRB 13/665

Rafael Martins da Cruz

Emulação de uma Solução SD-WAN para Integração dos Campi da UEMA

Projeto apresentado ao curso de Mestrado Profissional em Engenharia da Computação e Sistemas na Universidade Estadual do Maranhão como pré-requisito para a obtenção do título de mestre sob orientação do Prof. Dr, Leonardo Henrique Gonsioroski Furtado da Silva e coorientação do Prof. Dr. Luis Carlos Costa Fonseca.

Trabalho aprovado. São Luís, ____ de março de 2021:

Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva, UEMA
Orientador

Prof. Dr. Luis Carlos Costa Fonseca, UEMA
Coorientador

Dr.Vicente Angelo de Sousa Junior, UFRN
Examinador Externo à Instituição

Dr. Rogerio Moreira Lima da Silva, UEMA
Examinador Interno

São Luís
2021

Este trabalho é dedicado à minha família, em especial
ao meu filho Lucas Rafael Cruz

Agradecimentos

Ao nosso Senhor Jesus Cristo, pela vida e a possibilidade de investir nesta caminhada evolutiva, por propiciar tantas oportunidades de estudo e por colocar em meu caminho pessoas amigas e preciosas.

A MINHA MÃE Maria do Socorro Martins da Cruz pelo apoio incondicional.

Ao MEU PAI, Antônio Fernandes Lopes da Cruz por todos os ensinamentos e sua referência como pai e amigo (In memorian).

Ao meu ORIENTADOR e COORIENTADOR, agradeço pela confiança, pelo incentivo, paciência, por sua amizade e pela excelente orientação.

Aos meus amigos de trabalho, em especial ao Christiann Dênys da Fonseca Vieira, Denner Araújo Costa, Philipe Manoel Ramos Pinheiro e Richardson da Silva Sousa Lima pelo apoio e ajuda ao longo dessa caminhada, principalmente na reta final.

A minha esposa, pelo apoio e compreensão constante durante todo período do Mestrado.

Aos professores que tive ao longo do Mestrado Profissionalizante da UEMA, por dedicação, comprometimento e compreensão.

Ao pessoal da Secretaria Acadêmica, em especial a Karoline Silva Meireles, pela eficiência, dedicação e simpatia. Meu muito obrigado.

"Se o dinheiro for a sua esperança de independência, você jamais a terá. A única segurança verdadeira consiste numa reserva de sabedoria, de experiência e de competência."(Henry Ford)

Resumo

A internet foi concebida inicialmente em um contexto totalmente diferente do atual. Com o passar do tempo houve muitas mudanças e as redes começaram a ser utilizadas em várias áreas e atividades inesperadas, essas mudanças forçaram o desenvolvimento de novas aplicações as quais as redes foram submetidas e colocadas a prova de novos requisitos e um melhor desempenho. Algumas dessas aplicações não tiveram os seus requisitos atendidos devido a limitação da arquitetura das redes tradicionais. A crescente necessidade de expansão das redes de computadores em todos os âmbitos, seja ela comercial, governamental ou domiciliar, trouxe a possibilidade de ocorrências de erros, que podem comprometer toda a estrutura de comunicação mundial. Com a ideia de aumentar a disponibilidade, recursos de programação e gerência dos ativos de rede, surgiu a proposta do protocolo *Openflow* (protocolo que impulsionou as Redes Definidas por *Software* ou *Software Defined Networking-SDN*), que tem como premissa separar os planos de controle, de administração e de dados. De acordo com o conceito de SDN, os ativos de rede passariam a ser simples caixas que encaminham os dados e um novo elemento é responsável por conhecer e controlar toda a rede para orquestrar a comunicação entre os outros elementos da rede e o controlador. O conceito de Virtualização das Funções de Rede ou *Network Functions Virtualization (NFV)*, também ajudou no crescimento das Redes Definidas por Software. O foco das NFV são a virtualização de serviços e funções como *firewall* ou *load balancers*, isso juntamente com o controlador SDN. O uso dessas tecnologias alinhadas, além de aumentar velocidade no provisionamento de novos elementos de rede, também diminuem os custos operacionais das empresas. Após passar por todos esses conceitos, o objetivo desse trabalho é propor uma arquitetura *Software Defined - Wide Area Network* para as conexões entre o Campus Paulo VI da UEMA e seus Campi localizados em dezenove municípios do Estado do Maranhão, por meio de uma simulação realizada com as ferramentas de virtualização de rede SDN e um controlador SDN.

Palavras-chave: Redes definidas por software; SDN; OpenFlow; redes definidas por software de áreas de longa distância ; SD-WAN.

Abstract

The internet was initially conceived in a context totally different from the current one. Over time there were many changes and networks began to be used in various areas and unexpected activities, these changes forced the development of new applications which networks were submitted and put to the test of new requirements and better performance. Some of these applications did not have their requirements met due to the limitations of the traditional networks architecture. The growing need for expansion of computer networks in all areas, whether commercial, governmental or household, has brought the possibility of occurrences of errors, which can compromise the entire world communication structure. With the idea of increasing availability, programming resources and management of network assets, the Openflow protocol (protocol that boosted Software Defined Networks or Software Defined Networking-SDN) was proposed, whose premise is to separate the control plans , administration and data. According to the SDN concept, network assets would become simple boxes that forward data and a new element is responsible for knowing and controlling the entire network to orchestrate communication between the other elements of the network and the controller. The concept of Network Functions Virtualization or Network Functions Virtualization (NFV), also helped in the growth of Software Defined Networks. The focus of NFV is the virtualization of services and functions such as firewall or load balancers, together with the SDN controller. The use of these aligned technologies, in addition to speeding up the provisioning of new network elements, also lowers companies' operating costs. After going through all these concepts, the objective of this work is to propose a Software Defined - Wide Area Network architecture for the connections between UEMA's Paulo VI Campus and its campuses located in nineteen municipalities in the State of Maranhão, through a simulation carried out with SDN network virtualization tools and an SDN controller.

Keywords: Software defined networking; SDN;OpenFlow; software-defined wide area network; SD-WAN.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AP	Access Point
IaaS	Infrastructure as a Service
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Networks
LTE	Long Term Evolution
MAN	Metropolian Area Network
MPLS	Multiprotocol Label Switching
OSPF	Open Shortest Path First
QoE	Quality of Experience
QoS	Quality of Service
SaaS	Software as a Service
SASE	Secure Access Service Edge
SDN	Software Defined Networking
SD-WAN	Software-Defined Wide Area Network
SSID	Service Set Identifier
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TE	Traffic Enginnering
TLS	Transport Layer Security
UEMA	Universidade Estatual do Maranhão
VLAN	Virtual Local Area Network
WAN	Wide Area Network

Sumário

1 INTRODUÇÃO

Nos últimos anos houve um aumento da utilização das redes de computadores, principalmente no que diz respeito às redes de áreas amplas *(WAN – Wide Area Network)* e os aplicativos de transmissão de áudio e vídeo, interconexão entre data center, conexão de data center a serviço em nuvens, redes empresariais e redes das operadoras *(ISP - Internet Service Provider)*. Com essa crescente expansão e o surgimento de novos aplicativos e cenários operacionais aumentaram os requisitos para a transmissão de dados a longa distância. Assim, as operadoras e administradores de redes foram levados a projetar redes de longas distância a partir de uma outra perspectiva.

Com todas essas demandas de serviços em nuvem e interconexão de data center, surgiu a necessidade de utilizar uma rede com maior qualidade de serviço *(QoS - Quality of Service)*, com maior largura de banda e segurança no tráfego dessas informações, tornado-se uma forma dos administradores de rede gerenciar, de maneira centralizada, os equipamentos das filiais que normalmente nas instituições são de diferentes fabricantes e modelos, possuindo assim configurações especificas de fornecedor. Em uma arquitetura de rede tradicional, a complexidade e o custo para se implementar todos esses serviços é elevado e sem a garantia de entregar uma boa experiência de utilização da rede para os usuários finais.

Diante deste panorama, a rede definida por software *(SDN - Software defined networking)* é considerada uma arquitetura próspera para esse tipo de cenário, oferecendo aos ISP e administradores de redes uma nova perspectiva para construir as redes de longo alcance. De acordo com esse diapasão, a rede de área ampla definida por *software* simplifica a criação e o gerenciamento de conexões entre sites diferentes, por exemplo, entre data centers, data centers e nuvens públicas, em rede entre matriz e filiais; oferecendo flexibilidade necessária, controle centralizado e monitoramento com custo mais baixo.

Comparada com a WAN tradicional, a *SD-WAN (Software-Defined Wide Area Network)*, possui duas características que são cruciais para no desenvolvimento do cenário atual:

- Estrutura programática existente para hospedar aplicativos de controle desenvolvidos de maneira centralizada, levando em consideração a garantia a qualidade da experiência do usuário *(QoE - Quality of experience)*;

- A capacidade de definir as políticas de rede e gerenciar o tráfego sem exigir intervenção manual em cada dispositivo localizado na extensão da rede.

Conforme foi supramencionado, a rede definida por software de área ampla possui

vantagens que permitem fornecer uma garantia de serviço (QoS) para aplicativos, locais e para usuários específicos, além de simplificar as atividades do administrador da rede no que diz respeito às configurações dos equipamentos e a gerência da rede, acelerando as atualizações da rede.

Por consequência do que já foi exposto, pretende-se trabalhar com uma proposta de avaliar a viabilidade da solução SD-WAN, realizando testes em um protótipo criado em um ambiente de virtualização por um simulador de rede SDN, servidor no qual será criada a topologia de rede utilizada para o estudo de caso e um outro servidor em que será instalado o controlador SDN, emulando uma parte da topologia rede da Universidade Estatual do Maranhão -UEMA por meio deste cenário.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo o entendimento no que se refere às tecnologias envolvidas no desenvolvimento do projeto e no funcionamento da solução proposta para melhoria de conectividade do Campus Paulo IV da Universidade Estadual do Maranhão – UEMA, com os Campi dos continentes e uma possível interligação entre o data center e um site *backup*. Os problemas enfrentados por várias instituições, e a gestão dos ativos de rede de diferente fabricantes, controle de tráfego de dados, gerenciar de forma centralizada as redes da matriz e filiais e suas interconexões. Para solucionar, é proposta uma pesquisa aprofundada sobre *WAN – Wide Area Network* tradicionais, *SDN - Software defined networking*, *SD-WAN Software-Defined Wide Area Network*, e por fim, um estudo de caso da solução SD-WAN.

1.1.2 Objetivos Específicos

- Apresentar uma análise dos principais controladores *OpenFlow* da atualidade;
- Apresentar uma análise comparativa das principais soluções de *SD-WAN Software-Defined Wide Area Network* do mercado;
- Instalar e configurar o servidor de virtualização de rede *SDN - Software defined networking* e relatar problemas encontrados;
- Instalar e configurar o servidor de controle da rede *SDN - Software defined networking* e relatar problemas encontrados;
- Propor uma solução baseada em *SD-WAN Software-Defined Wide Area Network* para as conexões do Campi da UEMA com os seus dezenove Campi.

1.2 Metodologia

Neste ensaio apresentou-se uma problemática vivida por várias instituições utilizadoras de Tecnologias da Informação para obter *quality of service - QoS, quality of experience - QoE*, gerência e monitoramento de vários enlaces, gerência e configuração de forma centralizada dos ativos de diferentes fabricantes espalhados ao longo da rede, redução dos custos e segurança na troca de informações. Com intuito de facilitar e melhorar a gestão dessas entidades e dar continuidade a outras pesquisas já iniciadas nesta mesma linha.

Foi realizada uma pesquisa bibliográfica no campo da ciência da computação para melhor entendimento teórico dos conceitos de protocolos de comunicação, virtualização, nuvens públicas, privadas e híbridas, *WAN – Wide Area Network* tradicionais, *SDN - Software defined networking, SD-WAN Software-Defined Wide Area Network* e por fim, as soluções existentes no mercado de SD-WAN.

Em seguida foi feita coleta dos requisitos do cenário proposto para o estudo de caso, desenvolvimento do ambiente de teste, fase de validação, análise de erro e, finalmente, a realização dos testes em cima da solução SD-WAN. O cenário foi todo virtualizado em software e os resultados obtidos nos testes serão ilustrados partir da rede emulada.

1.3 Justificativa

Com o aumento da utilização de aplicativos emergentes, interconexões entre data centers, interconexões entre data centers e nuvens públicas e as empresas em crescimento exponencial, o desafio da rede de área ampla de longa distância é muito grande nos quesitos de qualidade de serviço (Qos), facilidade no gerenciamento da rede, eficiência, custo e o que será aplicado na rede de próxima geração.

A WAN é a rede mais importante do mundo da internet, ela transfere dados por longas distâncias, são projetadas para fornecer a melhor entrega possível, mas sem garantia de qualidade de serviço (QoS) para os requisitos de aplicação. Na WAN o tráfego de dados é entregue por meio de *links* vulneráveis e ativos de redes em condições adversas, deixando a rede de área ampla com vários pontos de falhas. As falhas mais comuns são nos *links* e dispositivos, afetando de forma severa o desempenho da rede e a qualidade de serviço percebida pelo usuário - QoE. Um dos protocolos mais populares desse tipo de rede é o *OSPF - Open Shortest Path First*, ou seja, abrir o caminho mais curto primeiro, atualizando o peso dos *links* com a granularidade de dezenas de segundos, mas ele não é sensível o suficiente nas falhas inesperadas das redes de longas distâncias, o que torna a transmissão mais deteriorada e compromete o desempenho da rede.

Diante deste cenário, em que o aumento dos requisitos e desempenho de transmissão

de dados, a exigência de uma baixa latência no contexto de jogos, de uso da nuvem (pública e privada), de aplicativos de web conferências que são necessários na Educação a Distância - EAD, na telemedicina o objetivo é garantir uma boa experiência para o usuário e qualidade de serviço (Qos) nessas aplicações.

1.4 Aplicabilidade

Com os avanços da tecnologia, preço elevado dos links *MPLS* e os links dedicados com uma inflexibilidade de gerenciamento na operação da rede *WAN – Wide Area Networks* as *Software Defined – Wide Area Network*, tornam mais simples esse gerenciamento, melhorando o desempenho das redes tradicionais e possibilitando maior flexibilidade, agilidade, custos mais baixos e de fácil configuração.

Esse projeto oferece uma proposta de solução para melhoraia de desempenho das redes de longa distâncias e interconexões entre data centers aliado ao monitoramento e gerenciamento centralizado dos ativos de rede, utilizando um estudo de caso para levantar informações sobre as *Software Defined - Wide Area Network*.

1.5 Estrutura do Documento

Esta dissertação está estruturada da seguinte maneira: o Capítulo 2 expõe os conceitos de Rede de Computadores, protocolo TCP/IP, os tipos de rede e trata sobre o protocolo *MPLS - Multiprotocol Label Switching* (um dos protocolos mais utilizados nas redes de área ampla). O Capítulo 3 discorre sobre o *Software Defined - Wide Area Network*, *Software Defined Networking* sua arquitetura, sobre o principal protocolo para sua implementação o protocolo *OpenFlow* e sobre a dissociação dos planos de dados e controle e seus elementos. No Capítulo 4 serão apresentadas algumas soluções proprietárias existentes no mercado, juntamente com um comparativo dessas soluções. No Capítulo 5 é dedicado a escolha das ferramentas que são utilizadas para a implementação do estudo de caso e introduz um comparativo sobre os controladores de SDN. O Capítulo 6 apresenta o estudo de caso voltado a um recorte da topologia de rede da Universidade Estadual do Maranhão - UEMA. O Capítulo 7 traz a solução proposta no estudo de caso. No Capítulo 8 traz os resultados obtidos com a solução proposta e as conclusões desta dissertação. O capítulo 9 apresenta os trabalhos futuros.

2 Rede de Computadores

A modernização dos computadores pessoais e o crescimento das aplicações estão ligados ao surgimento e a evolução das redes de computadores. A rede de computadores proporciona a comunicação entre dois ou mais terminais se utilizando do cabo par trançado, fibra óptica ou até mesmo através do ar, pela rede *Wi-Fi - wireless fidelity*.

A fusão dos computadores e das comunicações teve uma profunda influência na forma como os sistemas computacionais eram organizados. O conceito de "centro de computação" como uma sala com grande computador ao qual os usuários levam seu trabalho para processamento agora está completamente obsoleto. O velho modelo de um único computador atendendo a todas as necessidades computacionais da organização foi substituído pelas chamadas redes de computadores, nas quais os trabalhos são realizados por um grande número de computadores separados, mas interconectados (TANENBAUM, 2003).

Os terminais ou *hosts* enviam e recebem as mensagens pela rede. No momento em que um *host* se conecta à rede, ele recebe um endereço que é chamado de *IP - Internet Protocol*. Este endereço IP é uma identificação única para cada *host* da rede e é por meio dele que os outros dispositivos conseguem se comunicar.

Para que essa comunicação fosse eficiente foi criado o protocolo *TCP/IP - Transmission Control Protocol*, no intuito de atender a necessidade de endereçamento e de interconexão de redes heterogêneas locais e remotas. Pode-se considerar o TCP/IP uma arquitetura formada por um conjunto de protocolos de comunicação utilizados em rede locais (*LANs - Local Area Networks*) ou externas (*WANs - Wide Area Network*)(SOUSA, 2000).

2.1 Protocolo TCP/IP

Protocolos são regras ou uma espécie de linguagem, que permitem a comunicação entre dois computadores conectados na Internet. Devido à sua arquitetura e forma de endereçamento, o TCP/IP consegue realizar o roteamento de informações entre redes locais e externas, transferências de arquivos, emulação remota de terminais, e-mail, gerenciamento e outras funções, permitindo a interoperabilidade de diferentes tipos de rede. Vários ambientes e sistemas operacionais suportam o TCP/IP, como o UNIX, DOS, *Windows*, *Linux*, entre outros, permitindo a integração de diferentes plataformas (FILIPPETTI, 2018).

O protocolo IP permite que o bloco de dados (ou datagramas) sejam transmitidos

entre remetente e destinatário. Os usuários são identificados pelo endereço IP. No caso do IPv4 são 32 bits para endereçamento, cerca de 4 bilhões de endereços, já a nova versão do protocolo, o IPv6, utiliza 128 bits.

2.2 Tipo de Rede

As redes são definidas pelo seu tamanho, são elas: a rede de área local *LAN – Local Area Network*; a rede de área metropolitana *MAN - Metropolitan Area Network* e a rede de área ampla *WAN Wide Area Network*. Cada uma dessas redes possuem conceito e características distintas, como pode ser observado na Figura 1. Tais características vão desde a infraestrutura física até a lógica: como a largura de banda e a sensibilidade de falha, tamanho da rede etc..

Figura 1 – Tipos de Rede

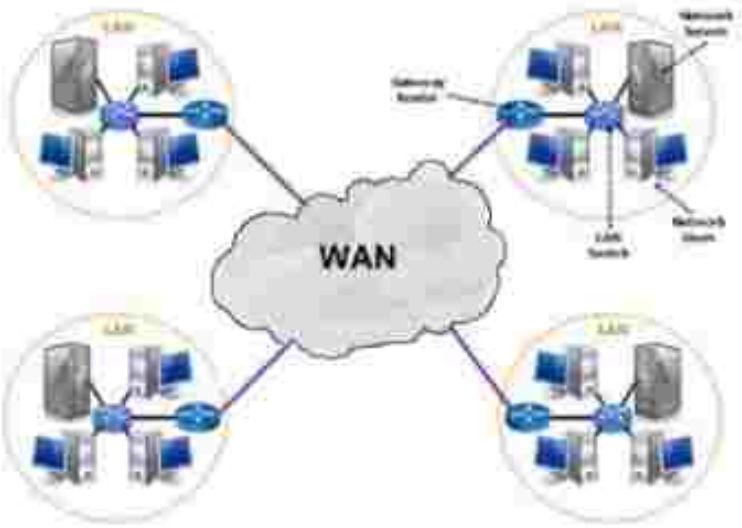
Fonte: (FATEC, 2020).

2.2.1 Wide Area Network - WAN

"WAN" é o acrônimo para *Wide Area Network*, e define uma rede de dados que se espalha por uma grande área geográfica, sendo a internet a maior e mais conhecida WAN de todas (a internet, na verdade, é o resultado de muitas WANs conectadas entre si). Redes WAN são muito usadas para interconectar redes *LANs - Local Area Networks* por meio das grandes distâncias. Uma rede WAN distingue-se de uma LAN pelo seu porte, tipo de equipamentos utilizados, protocolos de comunicação adotado, serviços disponibilizados

e complexidade desses serviços e custos da infraestrutura envolvida (FILIPPETTI, 2018).

Figura 2 – *Wide Area Network*



Fonte: (REIS, 2020).

Existem diferentes tipos conexão que podem ser utilizados na interconexão de redes *WAN* assim como vários tipos de rede *WAN* como: linhas dedicadas (*Leased Lines*), comutação de circuitos e comutação de pacotes. Além disso, são utilizados protocolos e serviços que garantam a qualidade de experiência do usuário (QoE), a qualidade de serviço (QoS) e mecanismos confiáveis de engenharia de tráfego.

O rápido crescimento da internet colocou uma enorme carga nas redes de área ampla aumento o desafio dos provedores de serviços, operadoras e administradores de rede. O problema é que não houve apenas um crescimento desmedido no número de usuários, mas os serviços utilizados como: web conferência, interconexão entre data centers e nuvens públicas e privadas, dentre outros serviços, também passaram a demandar uma quantidade de recursos e qualidade de serviço cada vez maior.

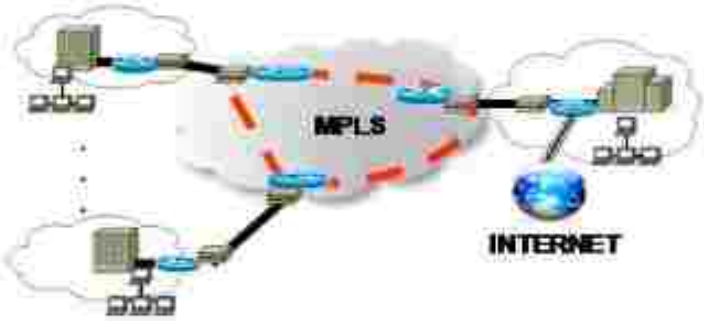
Com intuito de honrar os requisitos que os serviços demandam, já não é suficiente aumentar a quantidade de banda disponível em suas redes, foi preciso, também, identificar novas arquiteturas que pudessem prover qualidade de serviço (QoS), qualidade de experiência do usuário (QoE) e mecanismos confiáveis de engenharia de tráfego (TE), além de manter o custo o mais baixo possível. Então, como resposta a problemática supramencionada, surgiu a tecnologia *MPLS - Multiprotocol Label Switching* (ODOM, 2016).

2.2.2 MPLS - Multiprotocol Label Switching

MPLS - Multiprotocol Label Switching é uma tecnologia de comutação de pacotes definidas nas camadas 2 e 3 do modelo de referência *OSI - Open Systems Interconnection*, possibilitando melhorias na qualidade de serviço (QoS). A marcação *DiffServ - Differeciated Services* permite que o fluxo de dados específicos recebam tratamentos, leia-se prioridade - diferenciados, por meio de identificação *DSCP - DiffServ Code Point* específicas. Isso foi uma nova forma de organizar o tráfego de dados, fazendo com que os roteadores não decorem mais caminhos e números de IP, direcionando o tráfego por meio de outros números de identificação chamados de *labels*.

O MPLS tem muito mais flexibilidade que alguns outros serviços WAN como efeito colateral do encaminhamento de pacotes IP. O MPLS é compatível com qualquer tipo de *link* de acesso que lida com pacotes de IP, como: *Frame-Realy* e *Ethernet* WAN. Na verdade, apesar de o *Frame-Realy* ter se tornado menos popular neste século, ainda é possível encontrá-lo em muitas redes em uso, como *link* de acesso a um serviço MPLS (ODOM, 2016).

Figura 3 – MPLS - *Multiprotocol Label Switching*



Fonte: (WIKIPEDIA, 2020).

A *Traffic Engineering* tem como premissa reordenação do tráfego em uma rede de maneira uniforme, ela direciona parte do fluxo de caminhos sobrecarregados para caminhos que estão sendo menos utilizados e, dessa forma, evita o congestionamento e melhora a utilização dos recursos da rede. Também possibilita uma rápida recuperação em caso de falha. O MPLS, em seu funcionamento, permite o estabelecimento de caminhos determinísticos e independentes baseados na decisão dos administradores de rede.

O MPLS também possibilita a criação de várias classes de serviço, para que a rede atenda a cada uma com qualidade serviço específicos. Os fluxos de dados podem ser diferenciados de acordo com o serviço utilizado. Serviços multimídias que têm uma grande sensibilidade e atrasos de transmissão são diferentes de um serviço de e-mail que não exige

tanto da rede, requisitos estes que são levados em consideração para a escolha de um melhor caminho. Em uma transmissão MPLS, o endereço IP não é levado em consideração, o que faz com que o MPLS possa ser aplicado também sobre outros protocolos da camada de rede.

O protocolo MPLS trouxe diversas melhorias e solucionou algumas demandas de utilização e tráfego, mas com o aumento da infraestrutura de rede e a exigência por automatização e maiores velocidades de transmissão das informações, requer o uso de novas tecnologias para o transporte seguro de dados corporativos e melhor gerenciamento remoto dos equipamentos. Isso corrobora com o surgimento de um novo paradigma chamado de *SDN - Software Defined Networking*, ou seja, Redes Definidas por *Software*.

3 Software Defined - Wide Area Network

As SD-WAN, *Software Defined - Wide Area Network* fornece as vantagens tipicamente associadas as redes SDN - *Software Defined Networking* em data center, mas voltada a uma solução para as redes de longa distância e para nuvens públicas e privadas. Ambos SDN e SD-WAN virtualizam recursos para fornecer melhorias na entrega dos serviços, aumentar o desempenho, disponibilidade, automatizar a implantação e gerenciamento da rede (WILEY; SONS, 2018).

SD-WAN é uma abordagem de arquitetura de rede que habilita a comunicação entre redes de longa distância. A tecnologia é projetada para gerenciar e monitorar vários *links* de diversos tipos (MPLS, 3G, 4G, ADSL, *cable*, fibra, entre outros) e utilizar *software* para determinar parâmetros de qualidade de serviço para escolher a forma mais eficiente de conectar diversos locais, melhorando a experiência do usuário (BLOCKBIT, 2020).

Figura 4 – Arquitetura SD-WAN

Fonte: (RTM, 2020).

A rede definida por *software* em área ampla possui várias vantagens com relação as rede tradicionais: a) acompanhamento automatizado das condições de desempenho dos recursos da rede; b) gerenciamento do congestionamento da rede para obter um melhor desempenho e menor custo, realizando a escolha do *link* com melhor tráfego de rede, pois possui criptografia fim a fim entre os dispositivos interconectados; c) automatização operacional e simplificada na implantação de uma nova rede e possibilidade de carga de trabalho através de conexões.

3.1 Software Defined Networking

O termo *Software Defined Network* foi originalmente criado com a finalidade de representar o desenvolvimento do trabalho em torno do protocolo *OpenFlow* na Universidade de *Stanford* (SHENKER S., 2011). Referente a uma arquitetura de rede que separa o plano de dados do plano de controle e entregado o gerenciamento para um controlador remoto denominado de *Controller*. Esse paradigma vem chamando a atenção da comunidade acadêmica, dos provedores de serviços de internet e administradores de rede, pois surgem cada vez mais estudos e novas soluções para melhorar a sua eficiência.

A *Software Defined Network* é uma arquitetura emergente, dinâmica e adaptável que favorece arquiteturas físicas de rede em que a largura de banda é uma exigência constante pela natureza dinâmica das aplicações atuais *Software Defined Network* (ONF, 2020). Como caraterísticas fundamentais desta arquitetura a *Open Networking Foundation* elenca quatro características principais para *Software Defined Network*.

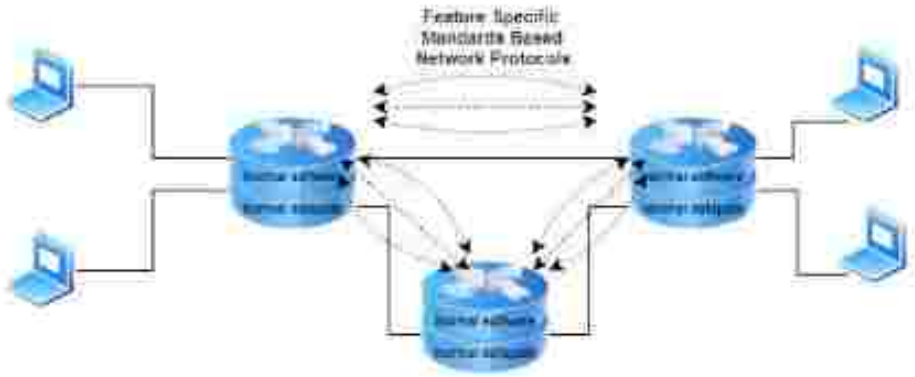
- **Diretamente programável**: o controle de rede é diretamente programável pelos administradores. Permite configurar, gerir, proteger e otimizar os recursos da rede por meio de programas e normas SDN, sem a necessidade de acrescentar nenhuma licença do proprietário dos ativos de rede;
- **Ágil**: porque permite aos administradores ajustar o tráfego da rede de forma a adaptá-lo às necessidades da mudança;
- **Centralmente gerida**: a inteligência lógica da rede é centralizada nos controladores baseados em *software*, que mantêm uma visão global da rede;
- ***Open standards* independentes do fabricante**: – como é implementada seguindo de padrões abertos de *software*, a *Software Defined Network* pode vir a simplificar em muito a administração e operação da rede. As instruções são fornecidas pelos controladores de *Software Defined Network* em vez de dispositivos ou protocolos específicos de fornecedores.

Ao contrário das redes tracionais, que o comutador (*switch*) é composto de dados e planos de controle, em que o plano de dados e controle dependem de implementação do fabricante, as funções de *hardware* e os recursos de *software* dos *switches* permanecem sob controle dos fabricantes dificultando as implementações e customizações externas, testes e desenvolvimento de novos protocolos de rede.

Na arquitetura tradicional de rede, existem várias limitações no âmbito da pesquisa e indústria, os *software* proprietários aumentam a complexidade da rede, em especial nas instituições governamentais que necessitam realizar processos licitatórios para realizar suas aquisições de equipamentos, tornando a padronização muito difícil e assim dificultando

a manutenção, o gerenciamento, a escalabilidade e implementação de novos serviços na rede. Na pesquisa, a arquitetura tradicional limita os testes de novos protocolos. Essas situações mencionadas são alguns problemas vivenciados pela comunidade acadêmica da Universidade Estadual do Maranhão. Na Figura 5 pode-se visualizar um esquema de rede tradicional de forma simplificada.

Figura 5 – Esquema de rede tradicional



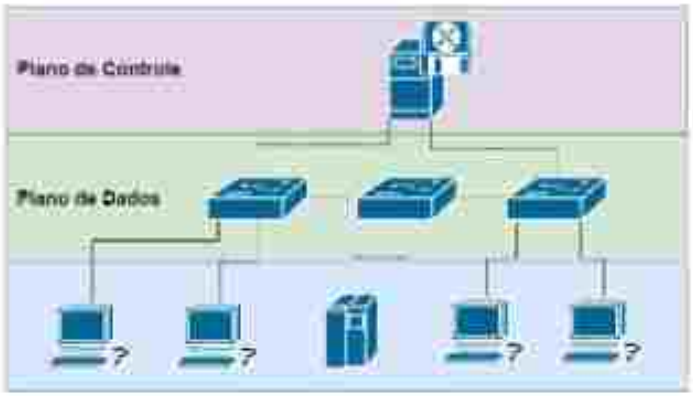
Fonte: O autor.

As limitações elevam a complexidade da rede especialmente quando se lida com produtos de diferentes fabricantes, que é o caso da maioria das instituições e provedores de serviços de internet. Isso aumenta a dificuldade com o gerenciamento da rede, manutenção, lançamento de novos serviços e escalabilidade, o que torna essas empresas cada vez mais vinculadas a um determinado fabricante no intuito de obter uma infraestrutura homogênea.

3.2 Arquitetura

A arquitetura *Software Defined Network* traz grandes vantagens para o estudo de redes. A dissociação dos planos de controle e dados permitem que eles evoluam de forma independente e com ritmo diferente. Além disso, a programação lógica da rede de controle permite experimentar novas ideias e protocolos em ambientes realistas, tornando a rede mais ágil e adaptável às necessidades operacionais, com um ou mais controladores centralizados. Eles mantém uma visão global da rede, facilitando o gerenciamento das políticas, a correção de possíveis problemas e com a implementação de código aberto, promovendo a independência do fabricante, como mostra a Figura 6.

Figura 6 – Esboço de uma *Software Defined Network*



Fonte: O autor.

O padrão *Software Defined Network* foi realmente definido como um modelo de rede após a padronização do protocolo *OpenFlow*, desenvolvido para criar uma interface entre o plano de controle e o plano de dados. Isso possibilitou, por meio dessa interface, uma comunicação segura entre os diversos equipamentos da rede e o controlador. O protocolo *OpenFlow* não é o único protocolo desenvolvido até hoje, mas sim aquele que possui maior estudo e investimento, além de estar em um estado mais avançado, proporcionando assim maior confiança na comunicação por meio de uma interface aberta e padronizada (MCKEOWN T.ANDERSON, 2008).

A tecnologia SDN usa o protocolo *OpenFlow* para poder manipular a largura de banda, por meio de aplicações, adaptando as redes às diferentes necessidades do negócio que vão surgindo, reduzindo a complexidade da gestão e da administração da rede. O protocolo *OpenFlow* é a ferramenta central de interação entre os níveis mais baixos da arquitetura, permitindo visualizar o funcionamento global.

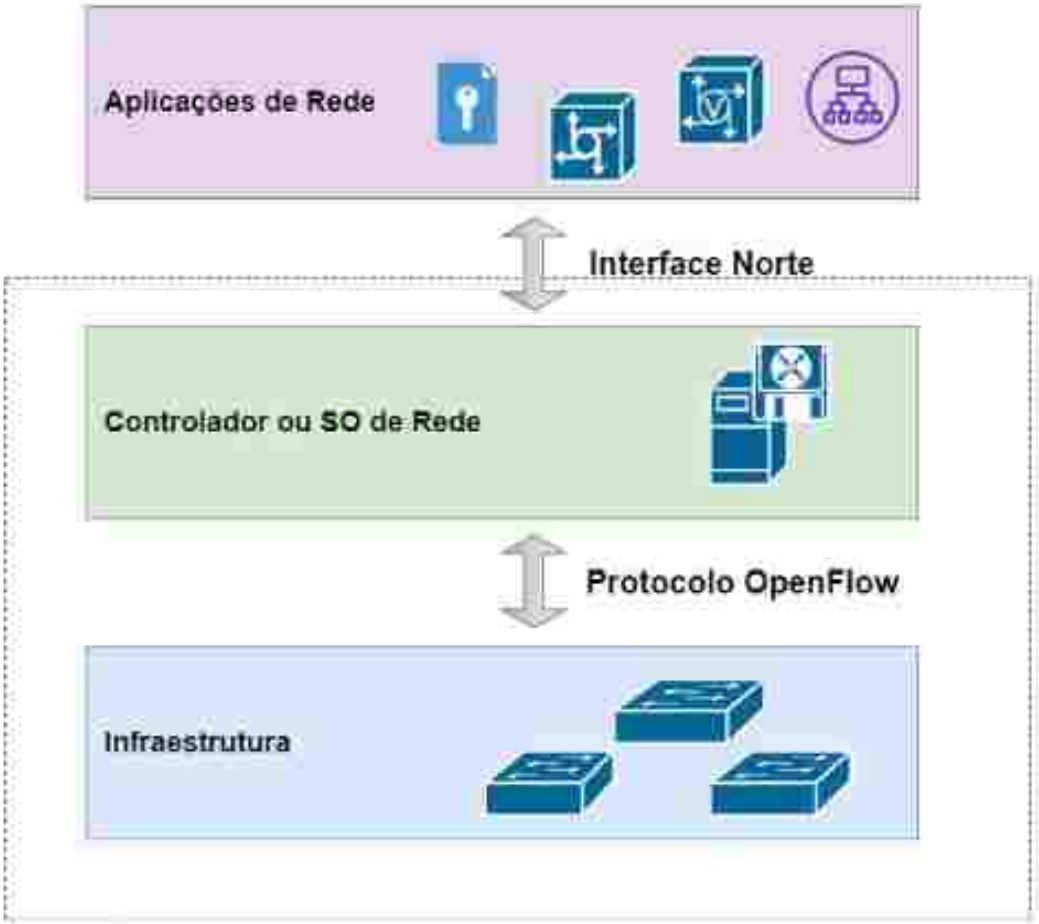
3.3 Protocolo OpenFlow

Em redes definidas por software, o protocolo *OpenFlow* é considerado o mecanismo que permite a programação ao longo da rede. O controlador de rede na arquitetura SDN usa esse protocolo para gerenciar e manipular as tabelas de encaminhamento dos *switches*. O padrão *OpenFlow* define as mensagens de controle de configurações da tabela de fluxo e o comportamento dos *switches OpenFlow*, separando sua infraestrutura em um plano de dados, composta pelos *switch* dentro das especificações do *OpenFlow* e no plano de controle, em que estão um ou mais controladores se comunicando por um canal seguro que viabiliza a comunicação entre os controladores e *switches* ao longo da rede.

Na infraestrutura atual da Universidade Estadual do Maranhão há vários modelos

de *switches* dos mais variados fabricantes, como: DATACOM, INTELBRAS, 3COM e ARUBA.Um *switche OpenFlow* é aquele que atende as especificações de separação do plano de dados do plano de controle. Desta forma, encontram-se diferentes modelos dentro das especificações estabelecidas e não há dependência de um só fabricante. Então, um padrão é implementado na interface de comunicação entre o controlador e a infraestrutura de forma a garantir que essa interface administre diferentes modelos de *switches*. Na Figura 7, tem-se o esboço simplificado da topologia do *OpenFlow*.

Figura 7 – Simplificação da Topologia do *OpenFlow*



Fonte: O autor.

Além da separação do plano de controle e do plano de dados, existem ainda três componentes importantes para a composição do *OpenFlow*. O primeiro é o canal de comunicação seguro com o controlador. O segundo é a interface do protocolo *OpenFlow* que permite ao controlador controlar o *switch*. O terceiro é sua tabela de fluxos, na qual consta uma lista de entradas de fluxo que especificam como o *switch* irá tratar os pacotes recebidos. Uma entrada da tabela de fluxo normalmente contêm características de fluxo que são usadas para corresponder ao pacote recebido, uma ação associada aos critérios

de correspondência que define como o pacote correspondente dever ser processado e um campo de estatísticas que rastreia o número de pacotes correspondentes e o tempo do último pacote encontrado.

3.4 Tabela de Fluxo

O fluxo representa o pacote que pode chegar nas entradas dos *switches OpenFlow*. As regras são definidas de acordo com os pacotes que serão recebidos. A configuração é baseada nos metadados e nos campos de cabeçalho que estarão juntos ao pacote.

A tabela de fluxo dos *switches OpenFlow* possui uma lista de entradas da tabelas de fluxo que especificam como *switch* vai lidar com os pacotes recebidos. Uma entrada da tabela de fluxo normalmente contém informações como a porta de entrada do fluxo e IP de origem, dados necessários para o encaminhamento do pacote. Dependendo do fluxo, diferentes ações podem ser tomadas e quem as define é o controlador. Essas ações são definidas nos cabeçalhos das tabelas de fluxos. Na tabela a seguir é ilustrado, genericamente, um conjunto de campos de cabeçalhos de pacote que podem ser usados para encaminhar um pacote a uma entrada de fluxo na tabela, de acordo com o *switch OpenFlow*(MCKEOWN T.ANDERSON, 2008).

Tabela 1 – Campos de composição das entradas de uma tabela de fluxo genérica

Packet Match Field
Ingress Port
Ethernet Destination Address
Ethernet Type
VLAN Id
VLAN Priority
IP Source Address
IP Transport Protocol ID
IP ToS bits
TCP/UDP Source Port or ICMP type
TCP/UDP Destination Port or ICMP code

Fonte: Adaptado de (MCKEOWN T.ANDERSON, 2008).

Quando o *switch OpenFlow* recebe um pacote, primeiramente o cabeçalho é varrido e enviado para o controlador, esse faz o processamento e retorna com as regras das tabelas. O processamento consiste numa comparação do cabeçalho do pacote com o cabeçalho da

tabela, caso sejam iguais, o pacote pertence ao fluxo e será processado.

O pacote é processado pelo *flow* de maior prioridade que combina com suas características e instruções preestabelecidas podendo ser aplicadas ao pacote, como: inserção e alteração de metadados, mudanças nos cabeçalhos do pacote, encaminhamento do pacote, entre outros. O encaminhamento pode ser para a tabela posterior, para uma interface de saída a partir de onde o pacote será transmitido ou para a tabela de grupos, por exemplo. Esses fluxos possuem campos *macth*, determinando os metadados que serão comparados com aqueles dos pacotes para indicar se estes serão processado ou não.

Conforme foi supracitado no parágrafo anterior, os fluxos podem ser instalados de dois modos para definir as regras das tabelas de fluxo:

- **Reativo**: São os que dependem da chegada do pacote com uma marcação em específico que servem como gatilho para a configuração do fluxo na tabela;
- **Proativo**: São previamente configurados nos *switches* antes mesmo que comece o tráfego de pacotes, configurando cada tabela com regras predeterminadas.

Caso o pacote não corresponda a nenhuma entrada na tabela de fluxo, ou seja, após realizada a varredura no cabeçalho do pacote e não for identificado nenhuma semelhança com a tabela de fluxo, o pacote será encaminhado para o controlador por meio do canal seguro.

3.5 Canal Seguro

O canal seguro é o meio de comunicação pelo qual o controlador distribui as regras de encaminhamento, ou seja, *link* lógico entre os *switches* e controlador de rede. Essa conexão é do tipo *TCP - Transmission Control Protocol*, e as mensagens podem ser criptografadas por meio do *TLS - Transport Layer Security* ou *SSL - Secure Socket Layer*, garantindo a confiabilidade entre a troca dessas mensagens e impedindo que sofra ataques de elementos mal-intencionados.

3.6 Controlador

É o responsável pelas regras e ações que gerenciam o encaminhamento dos pacotes, abstraindo, assim, o que seria a configuração dos *switches*. Ele se comunica com a aplicação e com o plano de dados permitindo a comunicação entre eles, garantindo flexibilidade em múltiplas aplicações. As mensagens trocadas entre o *switch* e o controlador são:

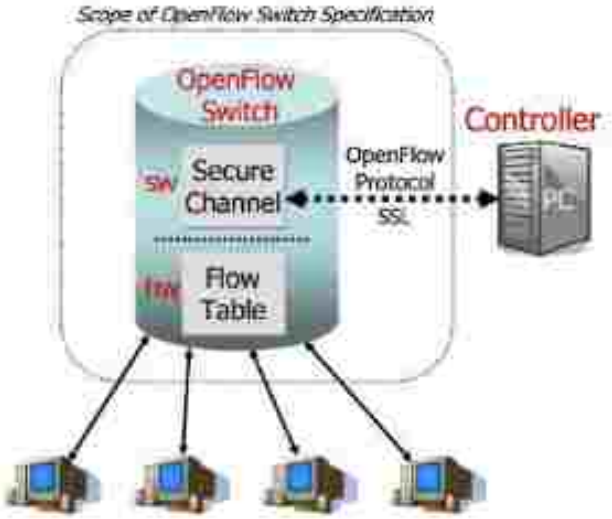
- **Controlador para *switch***: Mensagens enviadas do controlador para o *switch* quando solicitado por aplicações ou pelo controlador. Fazem parte desse grupo de

mensagens as estruturas para configuração do *switch*, requisição de informações e estatísticas, pacotes para serem enviados e mudanças de estado, como: instalação de *flow*, modificação de características das portas, entre outras;

- **Assíncrona**: Mensagens enviadas do *switch* para o controlador de forma assíncrona com o objetivo de informar a chegada de um novo pacote ao *switch*, retirada de um *flow* ou mudanças de estado de algum elemento;

- **Síncrona**: São mensagens que podem ser enviadas tanto pelo controlador quanto pelo *switch* sem aviso sobre a comunicação, com resposta predefinida. Fazem parte desse grupo de mensagens de *hello* e de *eco*, usadas no estabelecimento e na manutenção da conexão, muitas vezes para verificar se o canal seguro está disponível. Aqui também são definidas as estruturaras de mensagens de erro que são enviadas do *switch* para o controlador.

Figura 8 – Escopo da especificação do *switch OpenFlow*



Fonte: (GREG, 2020).

Após a chegada de um pacote no *switch*, o seu cabeçalho é comparado com os cabeçalhos das entradas de várias tabelas de fluxos que são canalizadas, então o pacote se moverá de uma tabela para outra até corresponder. Se um pacote corresponder a uma das regras de cabeçalho definidas em uma das entradas da tabela de fluxo, a ação associada será aplicada (MCKEOWN T.ANDERSON, 2008). A partir disso, as principais ações a serem tomadas, são:

• Encaminhamento do pacote para uma determinada porta ou para todas as portas (isso permite que um pacote seja encaminhado de acordo com uma rota por meio da rede);

• Descarte do pacote;

• Redirecionamento do pacote para o controlador, (isso permite que mais processamento seja aplicado no controlador);

• Modificação dos campos de cabeçalho do pacote, (isso permite aplicar funcionalidades de engenharia de tráfego, como por exemplo, MPLS).

A tabela de encaminhamento de um *switch OpenFlow* é composta por um cabeçalho (que contém informações sobre o fluxo) por uma ação (que define o que fazer com o pacote) e pelas estatísticas do fluxo (que informam o número de pacotes e o tamanho da transmissão). Por isso, essas tabelas permitem ao desenvolvedor criar diversas aplicações, por exemplo, balanceamento de carga e entrega de conteúdo. No Capítulo 5 são citados exemplos e características de controladores SDN, assim como o emulador de rede e as ferramentas que são utilizadas no estudo de caso.

4 Soluções existentes no mercado

A busca por melhoria no desempenho das redes de longa distância e redução de custos vêm se tornando uma necessidade, o mercado desse tipo de soluções está bastante aquecido e existem vários fabricantes que estão na disputa por uma melhor solução para essa problemática.

De acordo com a pesquisa realizada pela *Global Market Insights*, o mercado mundial de SD-WAN ultrapassou 1 bilhão de dólares em 2019 e deve ter um crescimento de 60% entre 2020 e 2026, chegando a uma projeção de valor de mercado superior a 30 bilhões de dólares (NSB, 2021).

O mercado de roteadores está evoluindo, geralmente chamados de roteadores de borda implementando o protocolo *MPLS-Multiprotocol Label Switching*, usados para conectar as filiais aos data centers para uma arquitetura descentralizada com cargas de trabalho em nuvem, para uma tecnologia SD-WAN repleta de recursos e vantagens (GARTNER, 2021).

A SD-WAN está substituindo os roteadores tradicionais pela seleção de caminhos com reconhecimento de aplicativos entre vários *links*, orquestração centralizada e segurança nativa, bem como funções de otimização de desempenho de aplicativos. Consequentemente, inclui vários fornecedores e emergentes de vários mercados, como: segurança, otimização de WAN e SD-WAN. Cada um traz seus próprios diferenciais e limitações.

As redes tradicionais de longa distância são utilizadas para realizar conexões entre usuários e filiais e seus servidores instalados em data centers sejam eles privados ou públicos. O mais comum era a utilização de circuitos MPLS nas redes WAN como já foi dito no Capítulo 2 deste trabalho, porém, com o aumento da utilização de aplicações *SaaS - Software as a Service* e *IaaS - Infrastructure as a Service*, o desempenho dos circuitos MPLS diminuíram, visto que as redes WANs não foram projetadas para aguentar esse aumento de tráfego, principalmente depois do surgimento da nuvem.

Existem alguns atributos comuns das soluções SD-WAN, como descrito na próxima seção com intuito de realizar algumas comparações sobre as soluções que serão exemplificadas neste trabalho.

4.1 Características dos Fabricantes

Os atributos citados a seguir são encontrados na maioria das soluções SD-WAN. Os fabricantes têm uma série de características comuns, mas a *Gartner* indica uma lista de atributos opcionais presentes apenas nos fabricantes líderes de mercado (GARTNER,

2021).

- **Funcionalidade Básicas**
  - **Software Licenciado**
    - Encaminhamento;
    - Reconhecimento de aplicativo: opção utilizada para determinar o tráfego de aplicação com roteamento dinâmico, ou seja, que ao reconhecer as aplicações que estão sendo utilizadas para oferecer melhor qualidade de serviço e melhor experiência do usuário, o direcionando para um *link* apropriado;
    - *Virtual Private Network - VPN* e *firewall* de camada 4.
  - **Fatores de forma**
    - Virtual e/ou físico;
    - *Edge* e *headend*.
  - **Orquestrador (local ou na nuvem)**
    - Configuração;
    - Gestão;
    - Visibilidade.
  - **Funcionalidade Opcional**
    - Segurança avançada nativa: são recursos de segurança como *Firewall* integrado a solução, promovendo controle mais eficiente de tráfego da rede, ao mesmo tempo que permite flexibilidade de configurar acesso locais de internet e de maneira centralizada;
    - *Gateways* de nuvem;
    - Otimização do desempenho do aplicativo.

De acordo com as características e critérios mencionados: a fácil utilização da ferramenta, integração da parte de segurança *Secure Access Service Edge - SASE* na rede, custo do equipamento, orquestração e integração com soluções de terceiros, dentre outras características, a *Gartner* levou em consideração e separou esses fabricantes por categorias, criando o quadrante mágico para o seguimento SD-WAN.

O quadrante mágico, é uma série de relatórios de pesquisa de mercado publicados pela empresa de consultoria de Tecnologia da Informação *Gartner* que contam com métodos de análise qualitativas e dados proprietários para demostrar tendências do mercado. Na próxima seção deste trabalho será discutido sobre essas categorias.

4.2 Principais Fabricantes

Nesta seção serão apresentados os principais fabricantes de solução SD-WAN, de acordo com o quadrante mágico, além das características exemplificadas na seção anterior. Foram observados mais alguns requisitos: a capacidade do produto, desempenho empresarial e financeiro, habilidade para executar bem os serviços, execução de *marketing*, experiência do usuário, otimização do desempenho do aplicativo, suporte para tecnologia 5G, *Long Term Evolution (LTE)* e outros casos de uso de celular sem fio. Após levantar todas essas informações a *Gartner* criou na Figura 9 o *Magic Quadrant* para *WAN Edge Infrastructure*.

Figura 9 – *Magic Quadrant* para *WAN Edge Infrastructure*

Fonte: (GARTNER, 2021).

Tem no quadro as categorias levando em consideração o quadrante mágico, são elas: *leaders, visionaries, challengers e niche players*.

- ***Leaders* ou Líderes**: é o grupo mais desejado por qualquer empresa de *Information Tecnology(TI)*. São classificadas nesta categoria as organizações que demonstram uma grande compreensão das necessidades daquele mercado em específico, com força

para direcionar os rumos do desenvolvimento e suas soluções para a satisfação do cliente, servindo de base para as outras organizações. A seguir, apresentam-se as empresas classificadas nesta categoria:

  - VMWare: com a tecnologia *VeloCloud* que inclui o dispositivo *SD-WAN edge* (VCE), *gateways* (VCG);
  - Fortinet: com *Secure SD-WAN* incluindo o *hardware FortiGate* e dispositivos virtuais com *software* de rede e segurança;
  - Versa Networks: oferta duas soluções a VOS anteriormente chamada de *FlexVNF* e o *Versa Titan*;
  - Cisco: com duas soluções a *Meraki* (que possui dispositivos MX e *software* com orquestração possibilitando a implantação do *Cisco Umbrella* como recurso de segurança) e a solução *Viptela* (que inclui o *Viptela OS* ou *software* IOS XE com orquestração);
  - Silver Peak: sua solução *Unity EdgeConnect SD-WAN*;
  - Palo Alto Networks: sua solução é o CloudGenix SD-WAN com dispositivos *ION edge* e *Prisma Access* opcional para segurança avançada integrada.

- ***Visionaries* ou Visionários**: neste grupo aparecem as organizações que têm aspectos fortes de inovação e visão abrangente de mercado, mas ainda não conseguem entregar uma solução robusta. Fazem parte desta categoria:

  - Juniper Networks: Sua oferta é o Juniper SD-WAN, que inclui seus *appliances* de borda SRX com *Contrail Service Orchestration*, *Mist WAN Assurance*, *Marvis Virtual Network Assitant (VNA)*, entre outras ferramentas;
  - HPE (Aruba): O nome da sua solução é Aruba SD-Branch, que inclui Aruba Central *Cloud Platform*, Aruba *Branch/Headend Gateways*, Aruba Virtual *Gateways*, *ClearPass Policy Manager* e *ClearPass Device Insight*.

- ***Challengers* ou Desafiadoras**: são as organizações com uma boa participação de mercado e com capacidade de executar suas estratégias de forma similar ao grupo dos *Leaders*, mas ainda não contam com um planejamento capaz de manter uma proposta de valor para os seus clientes e pecam na inovação. Estão nessa categoria:

  - Citrix: Sua solução é chamada de Citrix SD-WAN e inclui dispositivos físicos, virtuais e serviços *Cloud Direct* gerenciado por meio do orquestrador;
  - Huawei: O CloudWan é o roteador da série *NETEngine* e *gateways* da série *USG HISecEngine*.

- ***Niche players* ou Jogadores de Nicho**: as organizações deste grupo focam com sucesso em segmento menor no mercado com sucesso, ou não possuem uma direção

bem definida. Mostram também limitações para atuar com uma abrangência maior e com muitas limitações com as questões de inovação. Fazem parte desta categoria:

- Nuage Networks: com *Virtualized Network Services (VNS)*;
- Peplink: Mushroom;
- Barracuda: *CloudGen WAN*;
- FatPipe Networks: MPVPN;
- Cradlepoint: *NetCloud*;
- Teldat: M8Smart;
- Riverbed: SteelConnect EX.

Todas as soluções apresentadas são proprietárias e com valor aquisitivo elevado para ser adquirido no intuito de realizar o estudo de caso proposto nesse trabalho. Então, foi realizado uma pesquisa de ferramentas gratuitas que podem ser utilizadas para emular esse tipo de solução.

5 Escolha das Ferramentas

Levando em consideração as informações coletadas no Capítulo 4 sobre as Soluções Existentes no Mercado, todas elas são proprietárias com um custo elevado de aquisição. Para a escolha das ferramentas a serem usadas neste trabalho, são consideradas as *Open Source*, não somente por serem soluções gratuitas, mas por conta do suporte oferecido por suas comunidades que mantêm os projetos, ainda assim não deixam de ter o seu lado negativo: o tempo investido para aprender sobre cada inicintiva.

No decorrer deste capítulo, são descritas as principais iniciativas *Open Source* do mercado SDN e ferramentas que são utilizadas no desenvolvimento do estudo de caso.

5.1 Open Network Operating Systema

O controlador SDN, *Open Network Operanting System - ONOS* de código aberto e usa a licença do *Apache* 2.0. Essa plataforma é baseada em JAVA e utiliza o protocolo *OpenFlow*. O ONOS foi projetado para trabalhar com replicação virtualmente ilimitada para dimensionar a capacidade do plano de controle, com alto desempenho, resiliência, suporte a dispositivos legados e suporte a dispositivos de última geração.

5.2 Floodlight open SDN controller

É um controlador *OpenFlow* de código aberto e que utilizou o controlador *Beacon* como modelo, baseado em *Java* com a licença *Apache*. Ele é projetado para trabalhar com um grande número de dispositivos de rede, virtuais ou físicos que suportem o protocolo *Openflow*. A versão de código aberto do FloodLight não oferece resiliência ou escalabilidade como na versão comercial, chamado *Big Network Controller*, que é baseado em servidores em *cluster* para *High Availability* e utiliza o núcleo do FloodLight (THENEWSTACK, 2021).

O controlador *Floodlight* tem uma *Wiki* que abrange a instalação da aplicação, os *switches* compatíveis, um guia para usuários e outro para desenvolvedores.

5.3 Ryu OpenFlow controller

Ryu é uma estrutura de rede definida por *software* baseada em componentes. A plataforma oferece uma API bem definida que torna mais fácil para os desenvolvedores criarem novos aplicativos de gerenciamento e controle de rede. Suporta vários protocolos

para gerência dos dispositivos de rede, tais como: *OpenFlow, Netconf, OF-config* e etc. O *Ryu OpenFlow controller* também tem licença pela *Apache* e foi escrito em *Phyton* (RYU, 2021).

5.4 NOX e POX controller

O NOX foi o primeiro controlador SDN escrito em C++ e também fornece uma API para *Python*, desenvolvido inicialmente por Nicira *Networks* junto com o *Openflow*. Foi doado para a comunidade SDN *opensource*, virando base para as próximas soluções de controladores.

Segundo (OREILLY, 2021), o NOX é dividido em três frentes de desenvolvimento:

- **NOX clássico**: conhecida como linha de desenvolvimento, possui licença pública geral e contém suporte para *Python* e C++ junto com vários aplicativos de rede. Em contrapartida, esta linha encontra-se obsoleta e não há planos para desenvolvimentos adicionais;
- **O novo NOX**: apresenta suporte somente para C++, possuindo uma quantidade inferior de aplicativos de rede em comparação com o NOX clássico. Ele é mais rápido e tem uma base de código mais limpa;
- **POX**: Fornece suporte para *Python* e é conhecido como irmão do NOX, pois permite um desenvolvimento mais rápido e prototipagem, está sendo mais utilizado que o NOX.

5.5 OpenDayLight SDN controller

O controlador *OpenDayLight SDN controller*, é uma plataforma aberta modular para personalizar e automatizar redes de qualquer tamanho e escala. O projeto ODL, como também é chamado, surgiu do movimento SDN, com um foco claro na programação de rede (OPENDAYLIGHT, 2021a). O item descrito será tratado com mais detalhes no Capítulo 6, por ser o controlador escolhido para realizar o estudo de caso.

5.6 Comparativo entre os controladores SDN

Nessa seção são apresentadas duas tabelas com algumas comparações entre os controladores supramencionados. A Tabela 2 contém informações técnicas sobre os controladores, já a Tabela 3 contém informações sobre alguns casos de uso que os controladores suportam.

Tabela 2 – Informações Técnicas dos Controladores

Controller	Language	GUI	Modularity	Support
ONOS	Java	Web	Alta	Linux, MAC, Windows
Floodlight	Java	Web/Java	Baixa	Linux, MAC, Windows
Ryu	Python	Contém	Baixa	Linux
NOX/POX	C++, com suporte a Python	Contém	Média	Linux, MAC, Windows
Open Day Light	Java	Web	Média	Linux, MAC, Windows

Fonte: O autor

Tabela 3 – Informações de Casos de Uso

Casos de uso	ONOS	Floodlight	Ryu	NOX/POX	ODL
Virtualização de Funções de Rede	Não	Parcial	Sim	Não	Sim
Suporte OpenStack	Não	Sim	Sim	Sim	Sim
Interação com Ativos Legados	Parcial	Não	Não	Parcial	Sim
Monitoramento da Rede	Sim	Sim	Sim	Sim	Sim
Aplicação de Políticas	Parcial	Parcial	Não	Não	Sim
Balanceamento de Carga	Não	Não	Não	Não	Sim
Roteamento	Parcial	Não	Não	Não	Parcial

Fonte: O autor

Com base nas tabelas construídas e com todas as informações coletadas, sem dúvidas, foi utilizado o controlador *Open Day Light* em razão de todas as suas características técnicas, especialmente: virtualização de funções de rede, interação com ativos legados e roteamento. Junto ao controlador ODL é necessário utilizar o *Mininet*, que não é um controlador SDN propriamente dito, e sim um emulador de Redes Definidas por *Software*.

5.7 Mininet

É um emulador de rede *Software Defined Network* que cria uma rede virtual realista executando em um *kernel* real. Ele é um simulador do tipo *CBE(Container-Based Emulation)*, que emprega a virtualização a nível de processo, uma forma mais leve de virtualização em que muitos recursos do sistemas são compartilhados e, com isso, alcança maior escalabilidade que os outros que realizam uma virtualização completa para um dispositivo, necessitando usar uma máquina virtual para cada ativo de rede.

O *Mininet* é capaz de simular uma rede virtual com *switches*, *hosts*, *links* e controladoras, tudo dentro de uma única *Virtual Machine - VM*. Por meio de uma API em *Python*, o *Mininet* utiliza seus recursos de customização da topologia de rede, ou até mesmo via interface gráfica usando o aplicativo *"miniedit"* localizado no caminho /home/mininet/mininet/examples/miniedit.py, após a mudança do *login* para usuário sudo.

Além disso, o *Mininet* possui APIs que permitem o desenvolvimento de aplicações em *Python* para a criação de topologias e para simulação de comportamentos de rede. Estas APIs foram extremamente relevantes para a realização dos testes utilizados para simular o comportamento de uma rede de maneira automatizada e criação da topologia proposta.

5.8 VirtualBox

É um *software* de virtualização similar ao *VMware Workstation*, com a principal diferença na licença de utilização da ferramenta, considerando que o *software* da *VMware* é proprietário. Sua função é a criação de ambientes com sistemas operacionais distintos, permitindo a utilização de um ou mais sistemas operacionais com seus respectivos *softwares*, compartilhando os mesmos recursos físicos (*hardware*). O *VirtualBox* é uma ferramenta gratuita intuitiva, pode ser instalada em vários sistemas operacionais hospedeiros, permitindo a criação e gerenciamento de várias *Virtual Machine (VMs)* executando versões e derivações do *Windows*, *Linux*, *BSD*, *OS/2*, *Solaris*, *Haiku*, *OSx86* e outras, e virtualização limitada de convidados *macOS* no *hardware* da *Apple*.

5.9 Draw.io

Esta ferramenta foi utilizada para realizar a ilustração da infraestrutura de rede da UEMA e do recorte da topologia proposta no estudo de caso. Foi utilizada a versão 13.9.9 do *Draw.io*. O *Draw.io* é um editor gráfico *online* e gratuito no qual é possível desenvolver desenhos, gráficos, fluxogramas e outros. Também está disponível para o

sistema operacional *Windows*. É um *software* de interface simples e intuitiva, torna a sua utilização amigável e foi muito útil para ilustrar as duas topologias mencionadas.

6 Estudo de Caso

Este trabalho tem sua proposta alicerçada na base estrutural do parque computacional da rede de computadores da Universidade Estatual do Maranhão – UEMA, como o apoio da Coordenação de Tecnologia da Informação e Comunicação – CTIC e sua Divisão de Infraestrutura de Tecnologia da Informação e Comunicação – DITIC.

Esse ambiente universitário serviu como inspiração para a identificação de ambientes convergentes à utilização das tecnologias necessárias para garantir uma conexão estável e segura entre os seus dezenove *Campi* com gerenciamento centralizado e monitoramento.

Para a proposta do estudo de caso utilizou-se uma rede definida por *software*. Portanto, fez-se necessário o estudo de ferramentas, *softwares*, conceitos de SDN, sistemas operacionais, protocolos como: *Openflow* e MPLS, gerenciamento e monitoramento de redes de computadores para, então, poder construir o ambiente virtualizado, de modo que pudesse ser realizado os testes necessários sem causar impacto com a rede de produção da Universidade Estatual do Maranhão.

6.1 Diagnóstico da Infraestrutura de Rede

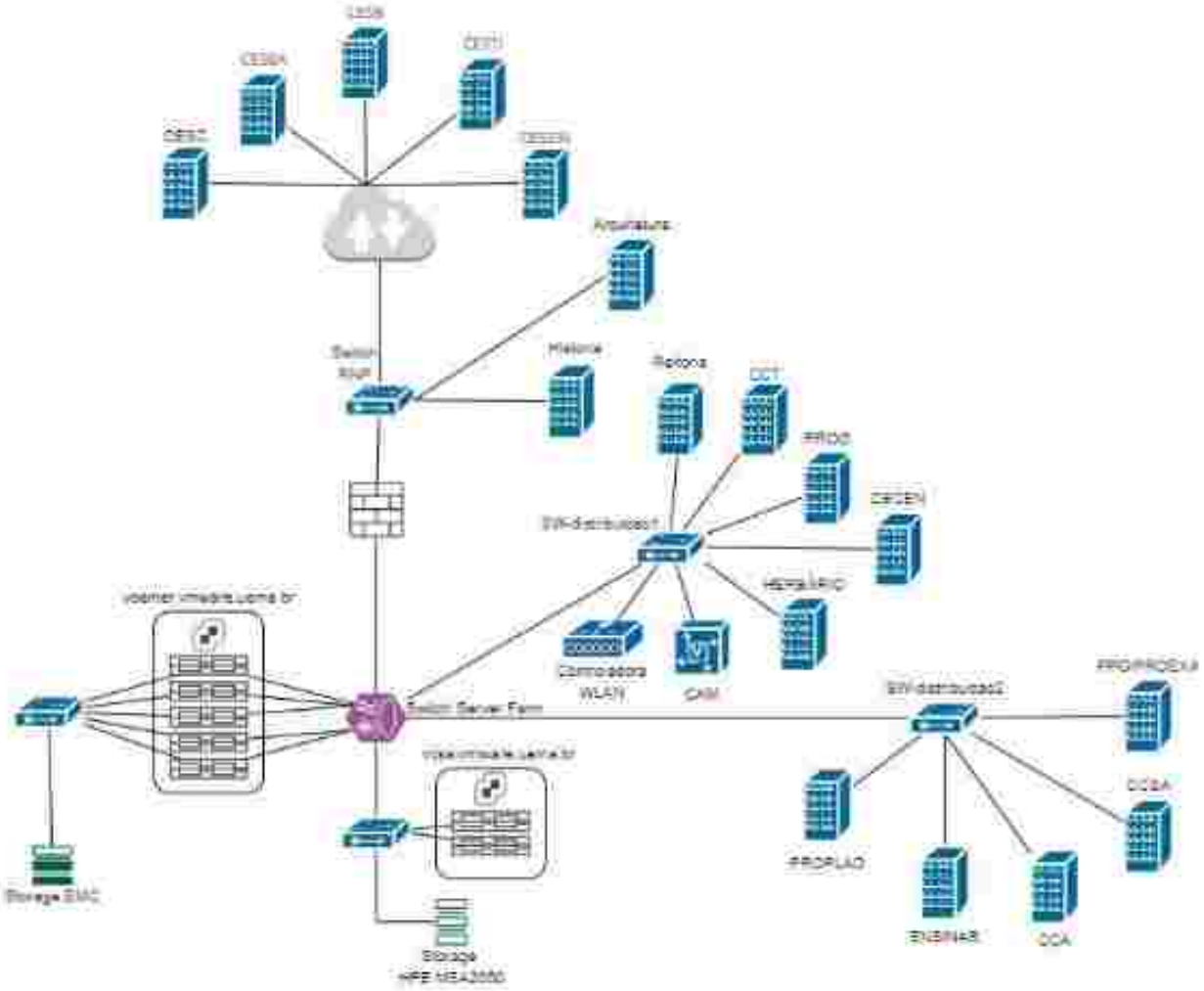
No início da implementação deste estudo de caso, a Universidade Estadual do Maranhão – UEMA possuía, em sua estrutura de rede, sete servidores separados em dois *cluster* de virtualização. Existiam cinco *hosts* físicos no primeiro *cluster*, com cento e trinta e sete máquinas virtualizadas e, no segundo *cluster*, dois *hosts* físicos com apenas seis máquinas virtualizadas.

Estava sendo utilizado o *software* de virtualização chamado *VM Ware ESXI* na versão 6.7.0.16713306 nos *hosts* físicos e duas instâncias do *vCenter*, que é responsável pelo gerenciamento unificado por *cluster* de todos os *hosts ESXI* da estrutura *VMware*.

Os servidores estão interligados a dois *storage* através de *links* de 1000 Mbps, duas controladoras *WLAN 7220* da Aruba nas quais são realizados o gerenciamento das licenças e da rede Wi-fi. Existe também uma central telefônica da CAM, tecnologia utilizada hoje apenas como *slot* para dois chips GSM para saídas de ligações externas.

Por conta da limitação de *hardware* da CAM, foi criado um servidor *Elastix* para gerenciamento dos 362 ramais VoIPs instalados por toda a universidade, um *firewall*, um *router* e sete *switches* para conectar todos os equipamentos. Na parte elétrica, têm-se dois *no-breaks* de 20 kVA cada, dois bancos de baterias (com 48 baterias de 50 Ah) e um gerador com autonomia de 48 horas. Essa infraestrutura pode ser visualizada na Figura 10.

Figura 10 – Infraestrutura de rede da UEMA



Fonte: O autor.

Têm-se apenas três instalações de *Windows Server*, duas máquinas na versão 2008 r2, que são os controladores de domínio *Active Directory Domain Services – AD DS*. Ele fornece os métodos para armazenar dados de diretório e disponibilizar esses dados para usuários e administradores de rede. Já a terceira instalação é uma VM com *Windows Server* 2012 data center em que se encontra o servidor de impressão. As demais máquinas virtuais hospedadas no data center da UEMA totalizam o número de cento e sessenta e oito VMs. Até a data da realização do levantamento das informações para este estudo estão com o sistema operacional *CentOS* na versão 7, que é um *software* livre derivado das fontes gratuitas da *Red Hat Enterprise (RHEL)*, mantido pela comunidade e adequado para utilização em diferentes equipamentos, tais como *laptops*, *desktops* e servidores.

A estrutura física de rede está divida em três tipos: LAN localizada dentro do Campus Paulo VI, possuindo trinta e cinco prédios conectados ao data center, dois pontos

em uma rede MAN que são os prédios de arquitetura e história, localizados no centro histórico da cidade de São Luís. Todas essas conexões são de 1000 Mbps e a rede WAN tem dezenove Campi conectados por *links* variando de 30, 50 e 100 Mbps entregues por uma empresa.

A universidade tem alguns serviços sendo executados dentro dessa infraestrutura mencionada, como: telefonia *VoIP*, rede *Wi-Fi*, projeto de *outsourcing* de impressão, sistema de videomonitoramento (todos estes serviços estão separados por *Virtual Local Area Network-VLANs*), sistemas que utilizam VPN, acesso a servidores de homologação destinados para realização de teste de aplicação, que são desenvolvidas dentro de ambientes de desenvolvimento pelos colaboradores da Divisão de Desenvolvimento de Sistemas - DDS da CTIC.

6.2 Levantamento das demandas de Rede

Esta seção relata o levantamento das demandas para justificar o projeto e uma possível migração da rede tradicional para uma rede baseada em *software*. Buscou-se identificar situações que tenham uma demanda da rede e que possam aumentar o tráfego, fluxo e verificação do monitoramento da rede, podendo causar lentidão. Foram levantados dados dos serviços que atuam na rede, enfatizando os serviços disponibilizados que já foram citados na seção anterior e sistemas, como: sistema acadêmico (*siguema*: https://sis.sig.uema.br/), os ambientes virtuais de aprendizagem – AVA com sete instancias em utilização, sistema de folha de pagamento (*renovarh*: http://renovarh.segep.ma.gov.br/), sistema de planejamento de gestão fiscal (*sigef*: http://sigef.seplan.ma.gov.br/) e de inscrição do vestibular (*sigconcursos*: https://www.seletivos.uema.br/).

No final do ano de 2014 a universidade começou a implantação dos seus sistemas para atender as demandas organizacionais dos cursos e de suas pró-reitorias, principalmente a de graduação, fazendo a migração dos dados do sistema legado. O sistema implantado pela ESIG faz conexão com o banco de dados que está hospedado no mesmo *host* físico da aplicação. Atualmente, a estrutura de acesso aos sistemas é um balanceador com duas instâncias de aplicação do sistema chamado *siguema* e dois bancos de dados.

Além do sistema acadêmico, foram implantados os seguintes módulos: protocolo, comunicação interna, ouvidoria, turma virtual, módulo de avaliação da graduação, *sigadmin*, módulo de extensão (curso e evento), módulo de bolsa, módulo assistência ao estudante, programa ensinar, módulo de graduação, módulo ensino a distância, módulo diploma, módulo *latu sensu*, módulo *stricto sensu*.

Com base no levantamento das informações, realizadas de forma empírica, por meio de reuniões com os colaboradores das Divisões de Desenvolvimento e Infraestrutura, sobre os diversos sistemas utilizados e serviços que são executados na rede da Universidade, bem

como o uso da rede pelos os mesmos. Foi possível constatar que o sistema acadêmico é um dos que tem a maior demanda de rede em seguida os Ambientes Virtuais de Aprendizagens, devido ao seu uso constante em diversos setores e pela quantidade elevada de usuários. O sistema siguema pode ser acessado dentro e fora da Universidade via *web* da mesma forma que os AVAs.

Outro ponto importante a se destacar são os serviços disponibilizados ao longo de sua rede, como: rede Wi-Fi, projeto de *outsourcing* de impressão, sistema de videomonitoramento e telefonia VoIP. São serviço utilizados diariamente de forma maciça pela comunidade acadêmica e administrativa da Universidade.

Na rede Wi-Fi da universidade, os usuários são obrigados a se identificarem por um portal de autenticação chamado de *Captive Portal*, responsável por controlar e gerenciar o acesso a internet. Mas para que isso possa acontecer, os administradores da rede tem que passar alguns parâmetros ao longo da rede até chegar no determinado AP, transformando o *Instant Access Point(IAP)* em *Campus Access Point (CAP)* ou em *Remote Access Point (RAP)*, que são modos de configuração do fabricante *Aruba Network*.

Os rádios em modo *IAP* não necessitam de uma controladora física e conseguem construir o seu próprio *cluster* deste que todos os rádios estejam na mesma sub-rede, configuração que os administradores da rede não têm controle de acesso, tráfego e uso das aplicações. Esses aspectos mudam nos outros modos de configuração, mas temos um ponto importante a ser comentando como relação ao *gateway* que seria a saída padrão para internet.

No modo *CAP* os rádios fecham comunicação com a *Mobility Controller* com criptografia *CPsec*, passando seu tráfego pela *Mobility Controller* e tendo como *gateway* padrão para internet em seu *link* principal, já no modo *RAP* é fechada uma comunicação com a *Mobility Controller*, e por meio de um túnel *VPN L2TP/IPsec*, tem o seu *gateway* padrão para internet no *link* principal do Campus Paulo VI em São Luís, gerando um grande fluxo de dados e lentidão em alguns momentos.

Outros fatores relevantes a serem considerados envolvem: 1) a comunicação das impressoras com o servidor de impressão, realizada através de uma VPN para cada *Campi* da universidade e com dependência de configuração pela terceirizada; 2) Departamento de pessoal que, entre as datas 15 à 30 de cada mês, faz a geração da folha de pagamento por meio do Sistema renovarh e sigef; 3) a grande dificuldade de gerenciar e configurar a rede LAN e WAN, devido a uma grande variedade de modelos e fabricantes, e equipamentos não gerenciáveis.

6.3 Tecnologias e Ferramentas Empregadas

Após o levantamento dos requisitos físicos e lógicos da rede de computadores da Universidade Estatual do Maranhão foi constatado que seria necessário realizar os testes em um ambiente controlado, com intuito de verificar os pontos positivos e negativos da solução proposta e se é necessário realizar teste com soluções proprietárias.

Em primeiro momento realizou-se a criação do ambiente de virtualização utilizando a ferramenta *VirtualBox* na versão 6.1, (um *software* de virtualização de computadores). Foi criado um ambiente com três máquinas virtuais, a primeira máquina com o sistema operacional *Ubuntu Server* na versão 20.04.2 LTS, sendo instalado nessa *Virtual Machine(VM)* o controlador SDN *Open Day Light* na versão *Carbon* 6.4.

A segunda máquina, com o sistema operacional *Ubuntu Server* na versão 20.04.1 LTS, foi instalado o emulador de rede *Mininet* na versão 2.3.0. Na terceira máquina, um *desktop* com o sistema operacional *Windows* 10 Pro na versão 20H2, foi utilizado para estabelecer conexões seguras entre a VMs do controlador e do emulador de rede, através do protocolo *SSH (Secure Socket Shell)*, protocolo que permite o acesso remoto e a gerência de maneira segura dos servidores.

Após a criação do ambiente de virtualização, foram realizadas as configurações de rede na ferramenta *VirtualBox* acrescentando duas interfaces de rede para cada VM, sendo uma interface para acesso à rede interna, estabelecendo a comunicação entre os servidores criados, e a outra para acesso à rede mundial (internet). Logo em seguida foram feitas as instalações do controlador *SDN OpendayLight (ODL)*, do emulador de rede *SDN Mininet* e da máquina *Windows* utilizada para realizar os acessos remotos aos dois servidores citados.

A ferramenta *OpenDayLight* é uma plataforma *Open Source*, desenvolvida na linguagem *java*, criada em 2013 voltada para Rede Definida por *Software* (*SDN*) e mantida pela *Linux Foundation*. O ODL oferece uma plataforma aberta e modular para customizar e automatizar redes de pequeno, médio e grande porte. A ferramenta utiliza protocolos abertos para fornecer controle centralizado e programático dos dispositivos físicos e virtuais além de realizar o monitoramento de dispositivos de rede através de uma interface gráfica *Web* (GUI). Como muitos outros controladores *SDN*, o *OpenDayLight* suporta o protocolo *OpenFlow* e faz uso das seguintes ferramentas:

- *Maven*: É uma ferramenta de automação de compilação utilizada primariamente em projetos *JAVA*. O *OpenDaylight* usa *Maven* para facilitar a automação da construção. O *Maven* usa o arquivo "pom.xml" (modelo de objeto do projeto) para fazer o *script* das dependências entre o pacote e para descrever quais pacotes devem ser carregados e iniciados;

- *OSGi* (*Open Services Gateway Initiative*): esta estrutura é o *back-end* do *OpenDaylight*, pois permite o carregamento dinâmico de *bundles* e pacotes de arquivos *JAR* (*Java ARchive*) e vinculação de *bundles* para troca de informações;
- Interfaces JAVA: as interfaces Java são usadas para escuta de eventos, especificações e padrões de formação. Esta é a principal maneira pela qual pacotes específicos implementam funções de retorno de chamada para eventos e para indicar a consciência de um estado específico;
- *APIs REST*: são APIs para o norte, como gerenciador de topologia, rastreador de host, programador de fluxo, roteamento estático e assim por diante (OPENDAYLIGHT, 2021b).

O controlador dispõe de APIs abertas para o norte que são usadas por aplicativos. A estrutura *OSGi* e *REST* bidirecional são suportados para as *APIs* em direção ao norte. A estrutura *OSGi* é usada para aplicativos executados no mesmo espaço de endereço que o controlador, enquanto a *API REST* (baseada na *web*) é usada para aplicativos que não são executados no mesmo espaço de endereço que o controlador.

Sua lógica de negócios e os algoritmos residem nos aplicativos. Esses aplicativos usam o controlador para reunir inteligência de rede, executar seu algoritmo para fazer análises e, em seguida, orquestrar as novas regras em toda a rede. No sentido sul, vários protocolos são suportados como *plug-ins*, por exemplo, *OpenFlow* 1.0, *OpenFlow* 1.3, *BGP-LS* e assim por diante.

O controlador *OpenDaylight* começa com um *plugin Southbound OpenFlow* 1.0, usado para a comunicação entre o controlador *SDN* e os *switches* o roteadores, facilitando o controle sobre a rede e permitindo que o *controller* faça alterações dinâmicas de acordo com as demandas e necessidades em tempo real. Esses módulos são vinculados dinamicamente em uma camada de abstração de serviço ou *Service Abstraction Layer (SAL)*. Essa camada fornece acesso de serviço do plano de controle, gerenciamento e aplicação para serviços e aplicações do plano de aplicações.

Na instalação do controlador *OpenDayLight*, foi necessário instalar o pacote *Java Runtime Environment (JRE)* na versão 8, utilizado para executar aplicações da plataforma *JAVA*. Ao concluir a instalação do *JRE* deve ser acrescentado as seguintes linhas no final do arquivo "*bashrc*":

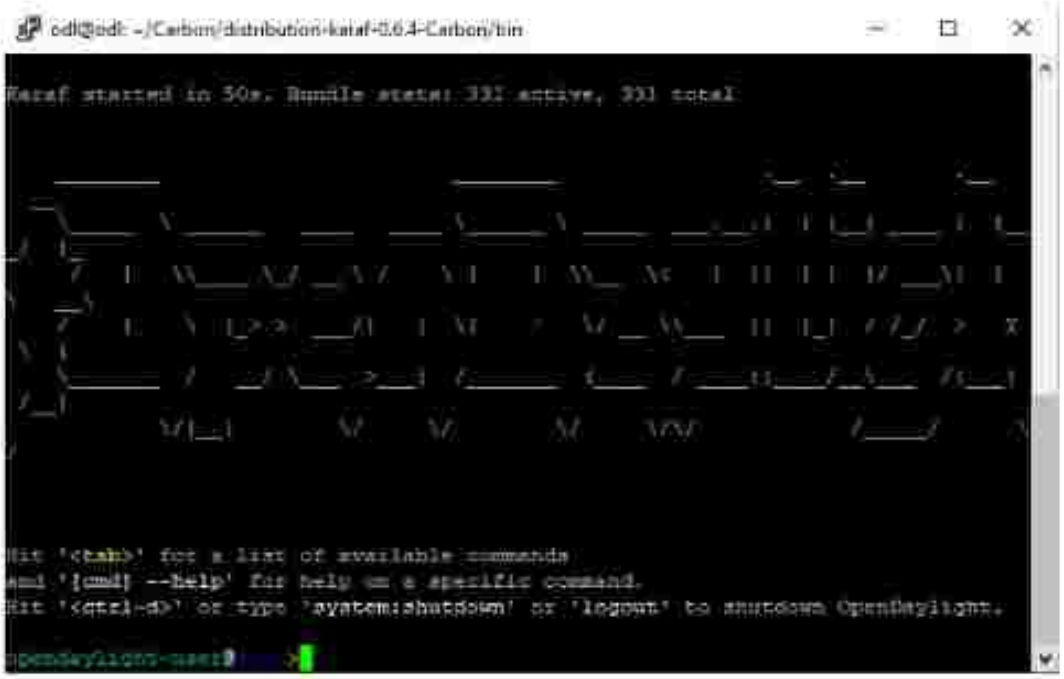
- export MAVEN_OPTS="-Xms256m -Xmx512m"
- export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64

O *controller OpenDayLight* pode ser executado a partir de qualquer sistema operacional e *hardware* desde que suporte *Java*. Ao concluir a instalação do pacote *Java* e a

realizada a alteração no arquivo "*bashrc*", o servidor está pronto para receber o *OpendayLight*. Os procedimentos citados são necessários para assegurar um bom funcionamento do *controller*.

O *controller* instalado foi *OpendayLight* na versão *Carbon* 6.4 disponibilizada no *site* oficial da ferramenta (https://www.opendaylight.org/). Ao descompactar o arquivo 'distribution-karaf-0.6.4-Carbon'e executar o "*karaf*", localizado no diretório distribution-karaf-0.6.4-Carbon/bin/, é aberto a interface *Command-Line Interface (CLI)* do *controller*. Mostrado na Figura 11.

Figura 11 – *Controller OpendayLight - Command-Line Interface (CLI)*



Fonte: O autor.

Com sua arquitetura modular, o *OpenDayLight* permite a instalação de suas *feature* de acordo com a necessidade do projeto. Na *Command-Line Interface* do *controller* foi instalado a *feature* de acesso a interface gráfica de gerenciamento pelo comando: "feature:install odl-dluxapps-applications". O acesso à interface *web* é realizada por meio do endereço: http://192.168.60.30:8181/index.html#/login, sendo que o IP: 192.168.60.30 é o endereço do servidor e a porta 8181 é padrão do ODL. Foi utilizado usuário e senha admin no painel de *login*. Para interação entre aplicações de alto nível e o controlador é necessário utilizar *REST-API*. Dessa forma, finalizando a instalação do *controller ODL*.

Para emular a rede SDN foi utilizado o *Mininet* na versão 2.3.0. Essa ferramenta pode ser utilizada em sistemas operacionais *UNIX/LINUX*, sendo bastante versátil. Como foi mencionado no Capítulo 5, o *mininet* é capaz de simular *links*, *hosts*, *switches* e

controladores. Além disso possui APIs que permitem o desenvolvimento de aplicações em *Python* para a criação de topologia e para simulação de comportamentos de rede.

Para a instalação do emulador de rede SDN, primeiramente foi realizado um *download* da imagem pré-fabricada da *VM* com o sistema operacional *Ubuntu LTS* 20.04.1 no *site* oficial da ferramenta (http://mininet.org/) e logo em seguida um *import appliance* dentro ferramenta *VirtualBox*.

Após finalizar o *import* foi necessário acrescentar duas variáveis de ambiente: "mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 d9071c7a49fa98e73397d6ee92e85ab8" e "localhost:10.0", essas varbíveis de ambiente sofrem alterações todas as vezes que é aberta uma seção SSH na ferramenta *PUTTY*, ferramenta instalada na máquina *Windows* para fazer acesso remoto as VMs. A variáveis foram acrescentadas para corrigir erros ao abrir a interface gráfica do *mininet*. Também foram realizadas alterações no arquivo "*miniedit.py*" para salvar corretamente e carregar os arquivos gerados com extensão ".mn". A sua interface gráfica pode ser observada na Figura 12.

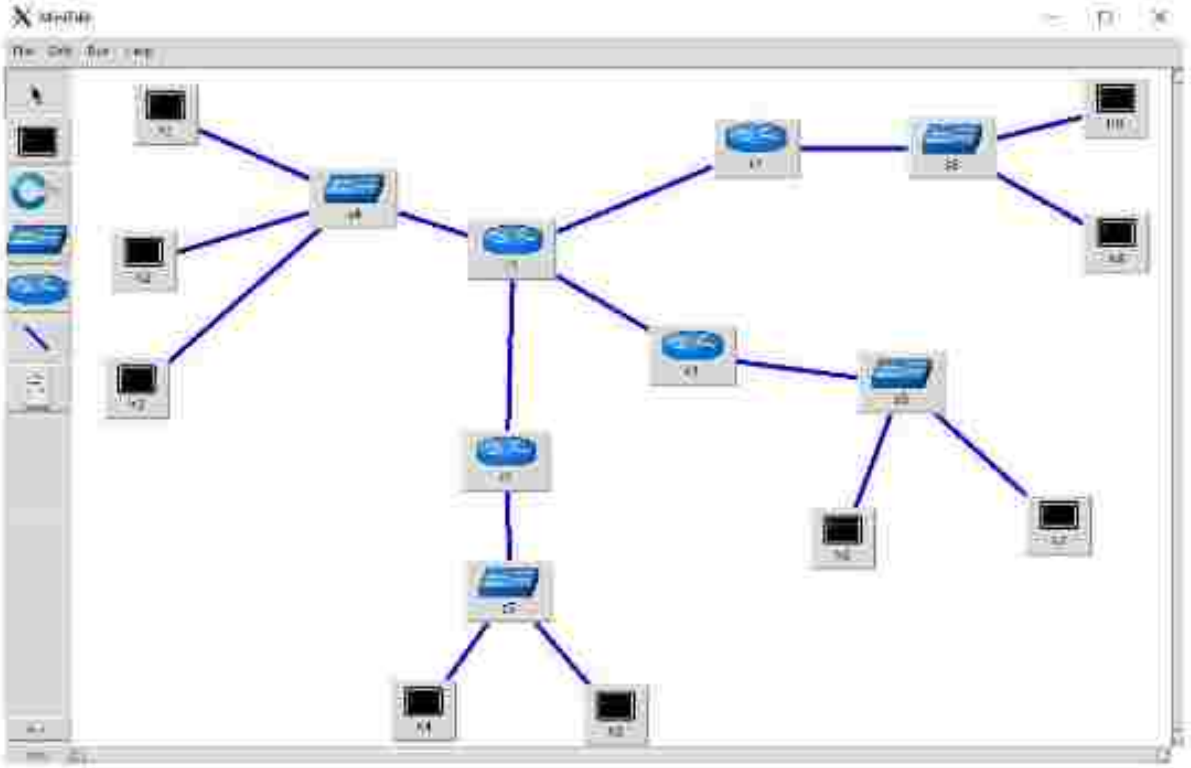
Figura 12 – Interface Grafica do *Mininet (Miniedit)*

Fonte: O autor.

Por meio das APIs que permitem o desenvolvimento de aplicações em *Python*, foi desenvolvido um *script* com as topologias de rede tradicional e SDN propostas para serem utilizados nesta demonstração. Tais redes foram baseadas em um recorte feito da infraestrutura de rede principal da universidade. Na topologia de rede tradicional criada foram adicionados 9 *hosts*, 4 *Legacy switches*, 4 *Legacy router*, 4 *switches* e 1 *controller* e

os *links* entre eles. A Figura 13 ilustra a topologia proposta.

Figura 13 – Topologia de Rede Tradicional criada para demonstração

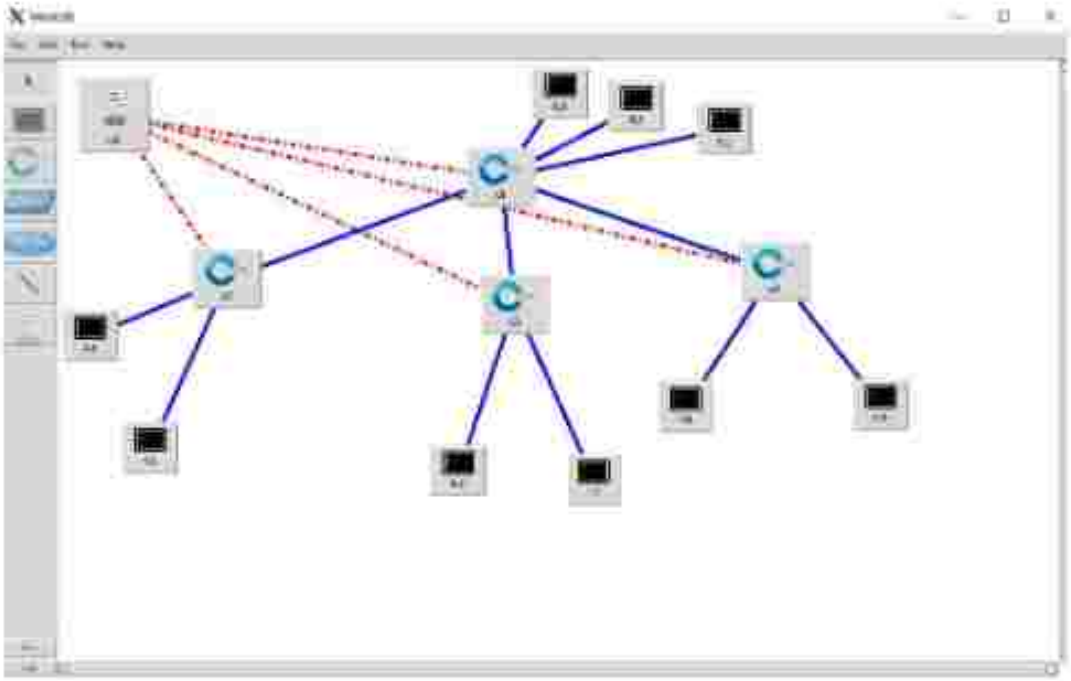


Fonte: O autor.

Os ativos foram separados em quatro redes: a primeira rede é considerada como a rede *LAN* da Universidade (Campus Paulo VI) com endereço de rede 192.168.0.0/24, composta por 3 servidores (h1, h2 e h3), 1 *Legacy switch* de distribuição (s4) e 1 *router* (r1). A segunda rede é o Centro de Caxias com endereço de rede 192.168.20.0/24, composta de 2 *hosts* (h4 e h5), 1 *Legacy switch* (s5) e 1 *router* (r2). A terceira rede é considerada o Centro de Bacabal com endereço 192.168.30.0/24, composta de 2 *hosts* (h6 e h7), 1 *Legacy switch* (s6) e 1 *router* (r3). E na quarta rede se modelou o Centro de Timon com endereço 192.168.40.0/24, composta com 2 *hosts* (h8 e h9), 1 *Legacy switch* (s8) e 1 *router* (r7). O *script* que gerou a topologia está no Apêndice A.

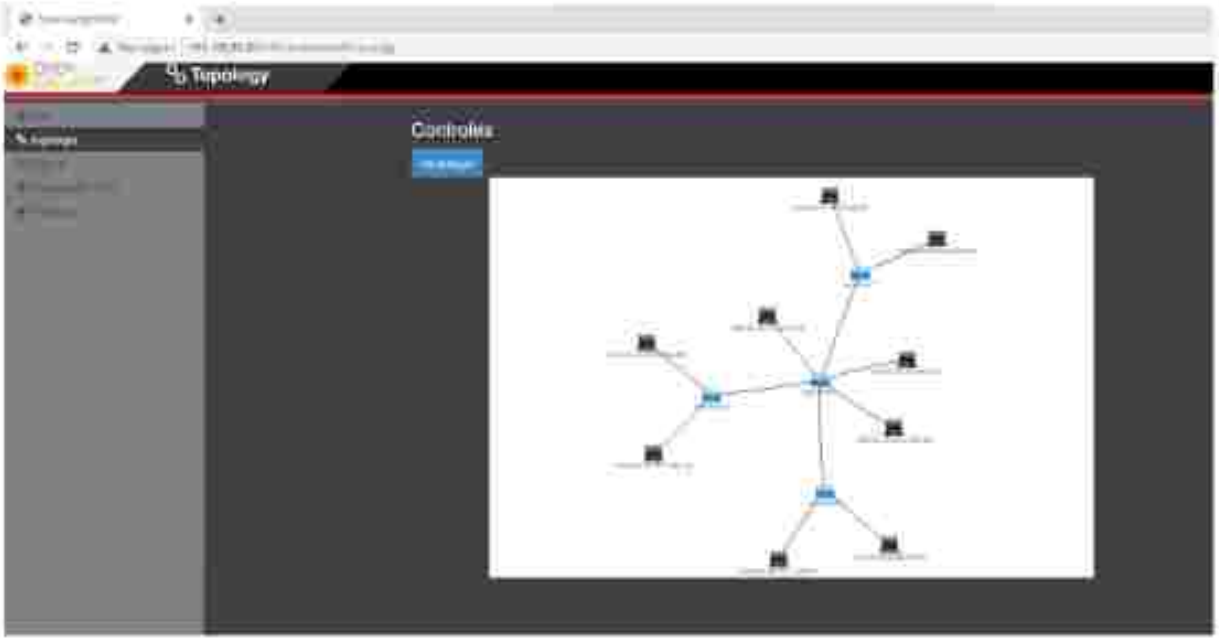
Para realizar a comparação e os teste com a ferramenta *Iperf* foi criado uma infraestrutura com a tecnologia SDN. Foram adicionados 9 *hosts*, 4 *switches* e 1 *controller* e os seus respectivos *links*, como mostrado na Figura 14, na interface gráfica do emulador *Mininet*, na Figura 15 (interface *web* do ODL). A ferramenta *iPerf* é usada para teste de medição ativa de largura de banda máximas em redes IPs, suporta vários ajustes de parâmetros relacionados a temporização, *buffers* e protocolos (TCP, UDP, SCTP com IPV4 e IPV6). Para cada teste o *iPerf* relata a largura e banda, perdas e outros parâmetros (IPERF, 2021).

Figura 14 – Topologia de Rede SDN criada para demonstração



Fonte: O autor.

Figura 15 – Interface *Web* do ODL com a Topologia SDN



Fonte: O autor.

Os ativos da rede SDN foram separados em quatro redes: a primeira rede é considerada como a rede LAN da Universidade (Campus Paulo VI), composta por 3 servidores (h1, h2 e h3), 1 *switch openflow* (s1); a segunda rede é o Centro de Caxias,

composta de 2 *hosts* (h4 e h5), 1 *switch openflow* (s2); a terceira rede é considerada o Centro de Barabal, composta de 2 *hosts* (h6 e h7), 1 *switch openflow* (s3); e a quarta rede é o Centro de Timon, composta com 2 *hosts* (h8 e h9), 1 *switch openflow* (s4). O script que gerou a topologia está no Apêndice B.

Ao finalizar o diagnóstico da infraestrutura em questão juntamente com o levantamento das demandas da rede, a criação do ambiente virtualizado e as configurações necessárias para cada servidor e suas aplicações, foram realizadas as configurações na topologia de rede tradicional e da rede SDN e os testes com a ferramenta *iPerf*, que serão tratados na próxima seção.

7 Resultados

Conforme o ambiente criado, foram realizados os testes em cima das duas infraestruturas: a tradicional e a SDN. Como já foi mencionado no Capítulo 6, têm-se dezenove conexões na topologia real da universidade, mas os testes foram criadas apenas três conexões ao ponto principal (Campus Paulo VI em São Luís). Cada centro conectado possui velocidade diferente de conexão, sendo o centro de Caxias com 100 Mbit/s, Bacabal com 100 Mbit/s e Timon com 50 Mbit/s. Ao realizar os testes com as duas infraestruturas emuladas, foi observado variações nos seguintes parâmetros:

- Taxa de transferência: é a velocidade em que uma quantidade máxima de dados pode ser transportada de uma origem ao seu respectivo destino;
- Largura de banda: é a taxa máxima de transferência de dados de uma rede em um determinado momento de uma conexão específica;
- *Jitter*: é a variação de tempo entre a chegada de pacotes do endereço de origem ao seu destino.

Os teste foram realizados em cima dos protocolos UDP e TCP. O primeiro teste foi com o protocolo *UDP – User Datagram Protocol*. Ele é um protocolo de transporte sem conexão (*conection less*), tecnicamente falando, não é necessário *handshake* para estabelecer uma comunicação.

Já o *TCP – Transmission Control Protocol* também é um protocolo de comunicação da camada de transporte, mas é voltado a conexão e muito utilizado nas aplicações *web*. Para manter a confiabilidade do envio dos dados, o protocolo TCP utiliza um aperto de mãos de três vias, o *three way handshake*, também chamado de SYN, SYN-ACK, ACK.

7.1 Teste com protocolo UDP

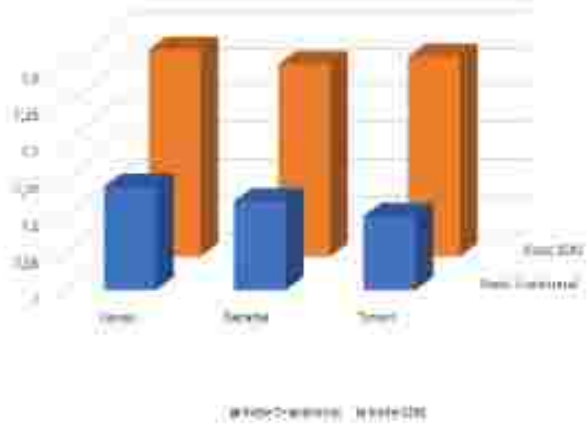
Para dar início aos testes foi verificado a conexão dos servidores do emulador e controlador ODL. Logo após aberto os terminais CLI dos *host* emulados no *Mininet* para as duas infraestruturas, os seguintes pares de *hosts* (h3,h4), (h2, h6) e (h1,h8), os *hosts* (h1, h2 e h3) estão fazendo o papel de servidores na rede do Campus Paulo VI em São Luís, já o *host* h4 é uma possível máquina conectada ao centro de Caxias, o *host* h6 no centro de Bacabal e o *host* h8 está centro de Timon.

Nos *hosts* que representam os servidores foi inserido o seguinte comando "iperf -s -u -p 5566 -i 60", sendo que o parâmetro"-s" executa o *iPerf* no modo servidor, "-u" usa o

protocolo UDP para realizar o teste, "-p" indica a porta para o servidor escultar e o "-i" para definir o tempo de intervalo em segundos entre os relatórios periódicos de largura de banda, *jitter* e taxa de transferência.

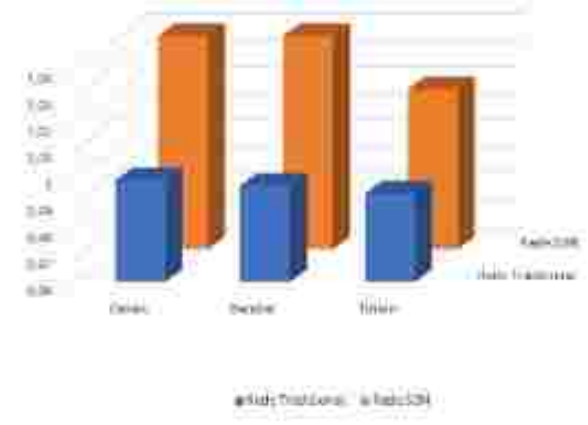
Nos terminas clientes, que são as máquinas dos centros, o comando inserido foi: "iperf -c (ip do servidor) -u -p 5566 -t 60", sendo que "-c" indica que o *iPerf* está em modo cliente, "-u" indica a utilização do protocolo UDP, "-p" indica a porta para o servidor escultar (que deve ser igual ao do servidor) e "-t" que é o tempo em segundos para a transmissão. Os resultados obtidos para as três conexões são ilustrados nas Figura 16, Figura 17 e Figura 18, que mostram os gráficos comparativo das duas infraestruturas.

Figura 16 – Taxa de Transferência - Protocolo UDP



Fonte: O autor.

Figura 17 – Largura de Banda - Protocolo UDP



Fonte: O autor.

Figura 18 – *Jitter* - Protocolo UDP

Fonte: O autor.

Analisando os gráficos das Figura 16, Figura 17 e Figura 18, pode-se observar leves alterações em todos os parâmetros, com a tecnologia SDN tendo um desempenho superior ao da rede tradicional. Como esperando, evidencia-se melhorias em taxa de transferência de dados e na largura de banda, ao ponto que o *Jitter* sofreu um pequeno aumento em relação às redes tradicionais.

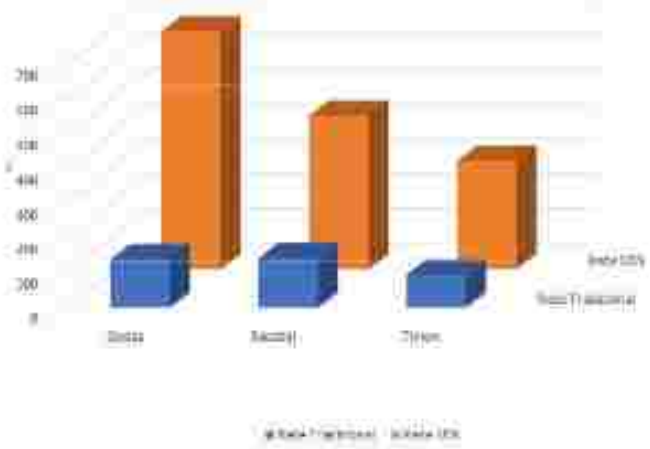
7.2 Teste com protocolo TCP

Iniciando os teste com protocolo TCP, foram abertos os terminais CLI dos *host* emulados no *Mininet* para as duas infraestruturas.

Nos *hosts* que representam os servidores foi inserido o seguinte comando: "iperf -s -p 5566 -i 60", sendo que o parâmetro "-s" executa o *iPerf* no modo servidor, "-p" indica a porta do servidor para o servidor escultar e o "-i" para definir o tempo de intervalo em segundos entre os relatórios periódicos de largura de banda e taxa de transferência.

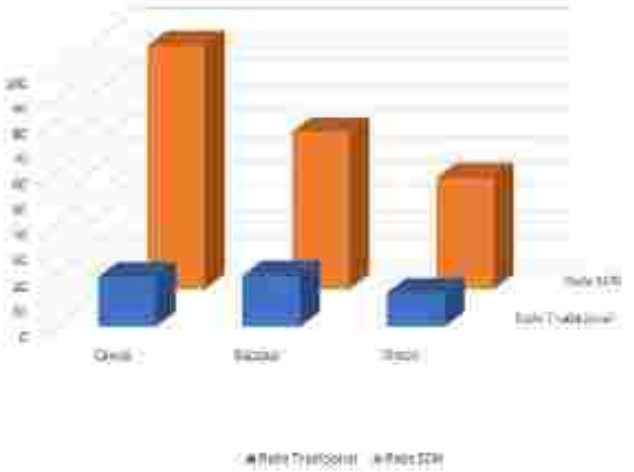
Nos terminas clientes, que são as máquinas dos centros, o comando inserido foi: "iperf -c (ip do servidor localizado em São Luís) -p 5566 -t 60", sendo que "-c" indica que o *iPerf* está em modo cliente, "-p" indica a porta para o servidor escultar, que deve ser igual ao do servidor e "-t" que é o tempo em segundos para a transmissão. Os resultados obtidos da saída dos comandos executados em cada *host*, para as três conexões, são ilustrados nas Figura 19 e Figura 20, que mostram os gráficos comparativos das duas infraestruturas de rede.

Figura 19 – Taxa de Transferência - Protocolo TCP



Fonte: O autor.

Figura 20 – Largura de Banda - Protocolo TCP



Fonte: O autor.

Ao contrário dos testes com o protocolo UDP, os testes realizados com o protocolo TCP tiveram alterações sensíveis nos parâmetros: taxa de transferência e largura de banda. Como pode-se observar nas Figura 19 e Figura 20, os índices nas redes definidas por *software* são bem elevados em relação às redes tradicionais.

A partir dos resultados obtidos nos testes é possível afirmar que as redes definidas por *software* têm um desempenho superior às redes tradicionais, para os parâmetros abordados nos testes realizados em ambiente virtualizado com a ferramenta *iPerf*. Essas

alterações de desempenho nos testes são atribuídas as características que às redes definidas por *software* têm ao separar o plano de controle do plano de dados da infraestrutura de rede tradicional trazendo mais eficiência, flexibilidade e aumento o desempenho.

8 Conclusão

A metodologia de pesquisa utilizada para o desenvolvimento deste trabalho foi *Design Science Research Methodology (DSRM)*, pois durante a realização deste projeto foi possível realizar um diagnóstico de como se encontrava a rede da Universidade Estatual do Maranhão, bem como a demanda de serviços desta rede. Além disso foi possível observar vários problemas com o gerenciamento, padronização dos equipamentos, configuração e monitoramento dos equipamentos e aplicações, custos elevados de *links MPLS* e *links* dedicados, construindo desta forma a primeira fase, de identificação do problema e motivação da pesquisa.

Após esse primeiro momento nota-se que é importante compreender muito bem o funcionamento da instituição e sua infraestrutura para que seja possível obter informações sobre as rotinas de trabalho dos seus diferentes setores e serviços. Isso acaba se tornando uma tarefa complexa, pois existem várias demandas e prioridades para serem observadas.

A partir das informações coletadas pode-se observar melhor o ambiente e, com mais clareza a topologia de rede a ser utilizada para o ensaio proposto, objetivando atingir as expectativas do projeto com a realização dos testes em cima da rede virtualizada e realizar os comparativos entre os dados coletados.

Apesar de não ter sido possível realizar todos os testes desejados neste ensaio, e uma possível migração, pode-se constatar que foi implantado no ambiente virtualizado uma infraestrutura mínima do recorte da topologia inicial demostrada efetuando as configurações nos ativos de rede virtualizados no emulador de rede e no *controller* SDN.

Considerando a complexidade de uso das ferramentas utilizadas para o desenvolvimento deste projeto em razão da documentação escassa, o objetivo foi alcançado com a criação do ambiente de virtualização, instalação dos servidores para emular a rede *SDN* e seu controlador, criação da rede emulada e configurações realizadas nos *hosts*, *switches* e *router* e conexão feita entre o emulador e controlador SDN. Foram conclusos os testes e realizado as comparações planejadas para o trabalho.

9 Trabalhos Futuros

A proposta do trabalho era a criação do ambiente virtualizado da rede da universidade, a fim de realizar teste comparativos e uma possível migração da rede tradicional para uma rede definida por *software* para redes de longa distância. O controlador *OpenDayLight* é uma ferramenta muito complexa e com várias funcionalidades que pódem ser exploradas em outros projetos.

Como trabalhos futuros ainda existem pontos a serem melhorados e explorados para se efetivar uma possível migração, tais como:

- Utilizar o controlador em uma rede real, com dispositivos físicos e virtuais;
- Efetuar testes com dispositivos reais que tenham compatibilidade com o protocolo *OpenFlow*;
- Integrar o *Postman* com o *OpenDayLight* para utilização das APIs;
- Realizar estudo bibliográfico e teste com os protocolos *SFlow* e *NetFlow*.

Referências

BLOCKBIT. 2020. Disponível em: <https://www.blockbit.com/pt/produtos/sd-wan/>. Citado na página 23.

FATEC. 2020. Disponível em: <https://marcciohenrique.wordpress.com/2014/03/22/tipos-de-redes-wan-lan-man/>. Citado na página 19.

FILIPPETTI, M. A. ***CCNA 6.0 guia completo de estudo/Marcos Aurélio Filippetti***. Rio de Janeiro: 2.ed. Alta Books, 2018. Citado 2 vezes nas páginas 18 e 20.

GARTNER. 2021. Disponível em: <https://www.gartner.com/doc/reprints?id=1-1XTMJ3XA&ct=191127&st=sb>. Citado 3 vezes nas páginas 32, 33 e 34.

GREG. 2020. Disponível em: <http://gregsowell.com/?p=4442>. Citado na página 30.

IPERF. 2021. Disponível em: <https://iperf.fr/>. Citado na página 50.

MCKEOWN T.ANDERSON, H. B. G. L. J. S. J. ***"Openflow: enabling innovation in campus networks". ACM SIGCOMM Computer Communication Review.*** [S.l.]: vol.38, no.2, 2008. Citado 3 vezes nas páginas 26, 28 e 30.

NSB. 2021. Disponível em: <https://nsb.com.br/mercado-sd-wan-crescimento-de-60/>. Citado na página 32.

ODOM, W. ***Cisco CCNA: routing e switching: ICND2 200-101: guia oficial de certificação / Wendell Odom.*** Rio de Janeiro, RJ: RJ: Alta Books, 2016. Citado 2 vezes nas páginas 20 e 21.

ONF. 2020. Disponível em: <https://opennetworking.org/sdn-definition/>. Citado na página 24.

OPENDAYLIGHT. 2021. Disponível em: <https://www.opendaylight.org/about/platform-overview>. Citado na página 38.

OPENDAYLIGHT. 2021. Disponível em: <https://nexus.opendaylight.org/content/sites/site/org.opendaylight.docs/master/userguide/manuals/userguide/bk-user-guide/content/_opendaylight_controller_overview.html>. Citado na página 47.

OREILLY. 2021. Disponível em: <https://www.oreilly.com/library/view/software-defined-networking-with/9781783984282/b0f4e4c4-dab7-4361-ad34-28e227ed8f15.xhtml>. Citado na página 38.

REIS, F. 2020. Disponível em: <http://www.bosontreinamentos.com.br/redes-computadores/qual-a-diferenca-entre-lan-man-e-wan-em-redes-de-dados/>. Citado na página 20.

RTM, N. 2020. Disponível em: <https://www.rtm.net.br/conectividade/sd-wan/>. Citado na página 23.

RYU. 2021. Disponível em: <https://ryu-sdn.org/>. Citado na página 38.

SHENKER S., C. M. K. T. M. N. ***The future of networking, and the past of protocols.*** [S.l.]: et al., 2011. Citado na página 24.

SOUSA, L. B. ***Redes de Computadores guia total.*** São Paulo – SP: Érica Ltda, 2009. Citado na página 18.

TANENBAUM, A. S. ***Redes de Computadores.*** Rio de Janeiro: 11ª reimpressão, Elsevier, 2003. Citado na página 18.

THENEWSTACK. 2021. Disponível em: <https://thenewstack.io/sdn-series-part-v-floodlight/>. Citado na página 37.

WIKIPEDIA. 2020. Disponível em: <https://pt.wikipedia.org/wiki/Multi_Protocol_Label_Switching>. Citado na página 21.

WILEY, J.; SONS, I. ***Software-Defined WAN for Dummies, 2nd VMware Special Edition.*** HOBOKEN, NEW JERSEY: 2nd VMWare Special Edition, 2018. Citado na página 23.

Apêndices

APÊNDICE A – Codificação da topologia de Rede Tradicional utilizada

A topologia de rede tradicional utilizada para o estudo de caso no Capítulo 6 foi criada por meio de *scripts* em linguagem *Python*.

Script Python para montagem da topologia no *Mininet*:

```
#!/usr/bin/env python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

net = Mininet( topo=None,
build=False,
ipBase='10.0.0.0/8')

info( '*** Adding controller\n' )
info( '*** Add switches\n')
r1 = net.addHost('r1', cls=Node, ip='0.0.0.0')
r1.cmd('sysctl -w net.ipv4.ip_forward=1')
r2 = net.addHost('r2', cls=Node, ip='0.0.0.0')
r2.cmd('sysctl -w net.ipv4.ip_forward=1')
r3 = net.addHost('r3', cls=Node, ip='0.0.0.0')
r3.cmd('sysctl -w net.ipv4.ip_forward=1')
s4 = net.addSwitch('s4', cls=OVSKernelSwitch, failMode='standalone')
s5 = net.addSwitch('s5', cls=OVSKernelSwitch, failMode='standalone')
s6 = net.addSwitch('s6', cls=OVSKernelSwitch, failMode='standalone')
```

```
r7 = net.addHost('r7', cls=Node, ip='0.0.0.0')
r7.cmd('sysctl -w net.ipv4.ip_forward=1')
s8 = net.addSwitch('s8', cls=OVSKernelSwitch, failMode='standalone')

info( '*** Add hosts\n')
h1 = net.addHost('h1', cls=Host, ip='192.168.0.10',
defaultRoute='via 192.168.0.1')
h2 = net.addHost('h2', cls=Host, ip='192.168.0.11',
defaultRoute='via 192.168.0.1')
h3 = net.addHost('h3', cls=Host, ip='192.168.0.12',
defaultRoute='via 192.168.0.1')
h4 = net.addHost('h4', cls=Host, ip='192.168.20.10',
defaultRoute='via 192.168.20.1')
h5 = net.addHost('h5', cls=Host, ip='192.168.20.11',
defaultRoute='via 192.168.20.1')
h6 = net.addHost('h6', cls=Host, ip='192.168.30.10',
defaultRoute='via 192.168.30.1')
h7 = net.addHost('h7', cls=Host, ip='192.168.30.11',
defaultRoute='via 192.168.30.1')
h8 = net.addHost('h8', cls=Host, ip='192.168.40.10',
defaultRoute='via 192.168.40.1')
h9 = net.addHost('h9', cls=Host, ip='192.168.40.11',
defaultRoute='via 192.168.40.1')

info( '*** Add links\n')
h1s4 = {'bw':1000}
net.addLink(h1, s4, cls=TCLink , **h1s4)
h2s4 = {'bw':1000}
net.addLink(h2, s4, cls=TCLink , **h2s4)
h3s4 = {'bw':1000}
net.addLink(h3, s4, cls=TCLink , **h3s4)
s4r1 = {'bw':1000}
net.addLink(s4, r1, cls=TCLink , **s4r1)
r1r2 = {'bw':100,'delay':'15','loss':5,'jitter':'20'}
net.addLink(r1, r2, cls=TCLink , **r1r2)
r1r3 = {'bw':100,'delay':'15','loss':5,'jitter':'55'}
net.addLink(r1, r3, cls=TCLink , **r1r3)
net.addLink(s5, r2)
h4s5 = {'bw':1000}
```

```
net.addLink(h4, s5, cls=TCLink , **h4s5)
h5s5 = {'bw':1000}
net.addLink(h5, s5, cls=TCLink , **h5s5)
h6s6 = {'bw':1000}
net.addLink(h6, s6, cls=TCLink , **h6s6)
h7s6 = {'bw':1000}
net.addLink(h7, s6, cls=TCLink , **h7s6)
net.addLink(s6, r3)
r1r7 = {'bw':50,'delay':'15','loss':5,'jitter':'55'}
net.addLink(r1, r7, cls=TCLink , **r1r7)
net.addLink(r7, s8)
s8h9 = {'bw':1000}
net.addLink(s8, h9, cls=TCLink , **s8h9)
s8h8 = {'bw':1000}
net.addLink(s8, h8, cls=TCLink , **s8h8)

info( '*** Starting network\n')
net.build()
info( '*** Starting controllers\n')
for controller in net.controllers:
controller.start()

info( '*** Starting switches\n')
net.get('s4').start([])
net.get('s5').start([])
net.get('s6').start([])
net.get('s8').start([])

info( '*** Post configure switches and hosts\n')
h1.cmdPrint('ip route add 10.10.10.0/24 via 192.168.0.1
&& ip route add 10.10.11.0/24 via 192.168.0.1
&& ip route add 10.10.12.0/24 via 192.168.0.1
&& ip route add 192.168.20.0/24 via 192.168.0.1
&& ip route add 192.168.30.0/24 via 192.168.0.1
&& ip route add 192.168.40.0/24 via 192.168.0.1')
h2.cmdPrint('ip route add 10.10.10.0/24 via 192.168.0.1
&& ip route add 10.10.11.0/24 via 192.168.0.1
&& ip route add 10.10.12.0/24 via 192.168.0.1
&& ip route add 192.168.20.0/24 via 192.168.0.1
```

```
&& ip route add 192.168.30.0/24 via 192.168.0.1
&& ip route add 192.168.40.0/24 via 192.168.0.1')
h3.cmdPrint('ip route add 10.10.10.0/24 via 192.168.0.1
&& ip route add 10.10.11.0/24 via 192.168.0.1
&& ip route add 10.10.12.0/24 via 192.168.0.1
&& ip route add 192.168.20.0/24 via 192.168.0.1
&& ip route add 192.168.30.0/24 via 192.168.0.1
&& ip route add 192.168.40.0/24 via 192.168.0.1')
h4.cmdPrint('ip route add 10.10.10.0/24 via 192.168.20.2
&& ip route add 10.10.11.0/24 via 192.168.20.1
&& ip route add 10.10.12.0/24 via 192.168.20.1
&& ip route add 192.168.0.0/24 via 192.168.20.1
&& ip route add 192.168.30.0/24 via 192.168.20.1
&& ip route add 192.168.40.0/24 via 192.168.20.1')
h5.cmdPrint('ip route add 10.10.10.0/24 via 192.168.20.2
&& ip route add 10.10.11.0/24 via 192.168.20.1
&& ip route add 10.10.12.0/24 via 192.168.20.1
&& ip route add 192.168.0.0/24 via 192.168.20.1
&& ip route add 192.168.30.0/24 via 192.168.20.1
&& ip route add 192.168.40.0/24 via 192.168.20.1')
h6.cmdPrint('ip route add 10.10.10.0/24 via 192.168.30.2
&& ip route add 10.10.11.0/24 via 192.168.30.1
&& ip route add 10.10.12.0/24 via 192.168.30.1
&& ip route add 192.168.0.0/24 via 192.168.30.1
&& ip route add 192.168.20.0/24 via 192.168.30.1
&& ip route add 192.168.40.0/24 via 192.168.30.1')
h7.cmdPrint('ip route add 10.10.10.0/24 via 192.168.30.2
&& ip route add 10.10.11.0/24 via 192.168.30.1
&& ip route add 10.10.12.0/24 via 192.168.30.1
&& ip route add 192.168.0.0/24 via 192.168.30.1
&& ip route add 192.168.20.0/24 via 192.168.30.1
&& ip route add 192.168.40.0/24 via 192.168.30.1')
h8.cmdPrint('ip route add 10.10.10.0/24 via 192.168.40.2
&& ip route add 10.10.11.0/24 via 192.168.40.1
&& ip route add 10.10.12.0/24 via 192.168.40.1
&& ip route add 192.168.0.0/24 via 192.168.40.1
&& ip route add 192.168.20.0/24 via 192.168.40.1
&& ip route add 192.168.30.0/24 via 192.168.40.1')
h9.cmdPrint('ip route add 10.10.10.0/24 via 192.168.40.2
```

```
&& ip route add 10.10.11.0/24 via 192.168.40.1
&& ip route add 10.10.12.0/24 via 192.168.40.1
&& ip route add 192.168.0.0/24 via 192.168.40.1
&& ip route add 192.168.20.0/24 via 192.168.40.1
&& ip route add 192.168.30.0/24 via 192.168.40.1')

CLI(net)
net.stop()

if __name__ == '__main__':
setLogLevel( 'info' )
myNetwork()
```

APÊNDICE B – Codificação da topologia de Rede SDN utilizada

A topologia de rede SDN utilizada para o estudo de caso no Capítulo 6 foi criada por meio de *scripts* em linguagem *Python*.

Script Python para montagem da topologia no *Mininet*:

```
#!/usr/bin/env python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

net = Mininet( topo=None,
build=False,
ipBase='10.0.0.0/8')

info( '*** Adding controller\n' )
c0=net.addController(name='c0',
controller=RemoteController,
ip='192.168.60.30',
protocol='tcp',
port=6633)

info( '*** Add switches\n')
s1 = net.addSwitch('s1', cls=OVSKernelSwitch, listenPort=6633)
s2 = net.addSwitch('s2', cls=OVSKernelSwitch, listenPort=6633)
s3 = net.addSwitch('s3', cls=OVSKernelSwitch, listenPort=6633)
s4 = net.addSwitch('s4', cls=OVSKernelSwitch, listenPort=6633)
```

```
info( '*** Add hosts\n')
h4 = net.addHost('h4', cls=Host, ip='10.0.0.1', defaultRoute=None)
h5 = net.addHost('h5', cls=Host, ip='10.0.0.2', defaultRoute=None)
h6 = net.addHost('h6', cls=Host, ip='10.0.0.4', defaultRoute=None)
h7 = net.addHost('h7', cls=Host, ip='10.0.0.5', defaultRoute=None)
h8 = net.addHost('h8', cls=Host, ip='10.0.0.6', defaultRoute=None)
h9 = net.addHost('h9', cls=Host, ip='10.0.0.7', defaultRoute=None)
h3 = net.addHost('h3', cls=Host, ip='10.0.0.8', defaultRoute=None)
h2 = net.addHost('h2', cls=Host, ip='10.0.0.9', defaultRoute=None)
h1 = net.addHost('h1', cls=Host, ip='10.0.0.10', defaultRoute=None)

info( '*** Add links\n')
s2s1 = {'bw':100,'delay':'15','loss':5,'jitter':'20'}
net.addLink(s2, s1, cls=TCLink , **s2s1)
s3s1 = {'bw':100,'delay':'15','loss':5,'jitter':'55'}
net.addLink(s3, s1, cls=TCLink , **s3s1)
s4s1 = {'bw':50,'delay':'15','loss':5,'jitter':'55'}
net.addLink(s4, s1, cls=TCLink , **s4s1)
h4s2 = {'bw':1000}
net.addLink(h4, s2, cls=TCLink , **h4s2)
h5s2 = {'bw':100}
net.addLink(h5, s2, cls=TCLink , **h5s2)
h6s3 = {'bw':1000}
net.addLink(h6, s3, cls=TCLink , **h6s3)
h7s3 = {'bw':1000}
net.addLink(h7, s3, cls=TCLink , **h7s3)
h8s4 = {'bw':1000}
net.addLink(h8, s4, cls=TCLink , **h8s4)
h9s4 = {'bw':1000}
net.addLink(h9, s4, cls=TCLink , **h9s4)
s1h3 = {'bw':1000}
net.addLink(s1, h3, cls=TCLink , **s1h3)
s1h2 = {'bw':1000}
net.addLink(s1, h2, cls=TCLink , **s1h2)
s1h1 = {'bw':1000}
net.addLink(s1, h1, cls=TCLink , **s1h1)

info( '*** Starting network\n')
```

```
net.build()
info( '*** Starting controllers\n')
for controller in net.controllers:
controller.start()

info( '*** Starting switches\n')
net.get('s1').start([c0])
net.get('s2').start([c0])
net.get('s3').start([c0])
net.get('s4').start([c0])

info( '*** Post configure switches and hosts\n')
s1.cmd('ifconfig s1 192.168.60.100')
s2.cmd('ifconfig s2 192.168.60.101')
s3.cmd('ifconfig s3 192.168.60.102')
s4.cmd('ifconfig s4 192.168.60.103')

CLI(net)
net.stop()

if __name__ == '__main__':
setLogLevel( 'info' )
myNetwork()
```