FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Tecnologia em Segurança da Informação

Tiago Viana

Redes Definidas por Software - OpenSDN

Americana, SP
2016

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Tecnologia em Segurança da Informação

Tiago Viana

Redes Definidas por Software - OpenSDN

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Tecnologia em Segurança da Informação, sob a orientação do Prof. Me. Rossano Pablo Pinto

Área de concentração: Tecnologia da Informação.

Americana, SP

2016

Tiago Viana

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

V668r VIANA, Tiago
Redes definidas por software – OpenSDN. / Tiago Viana. – Americana: 2016.
70f.

Monografia (Curso de Tecnologia em Segurança da Informação). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Ms. Rossano Pablo Pinto

1. Redes de computadores 2. Segurança em sistemas da Informação I. PINTO, Rossano Pablo II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.519

Tiago Viana

REDES DEFINIDAS POR SOFTWARE - OPENSDN

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Segurança de Redes.

Americana, 09 de Dezembro de 2016.

Banca Examinadora:

Rossano Pablo Pinto (Presidente)
Mestre
Fatec Americana

Clerivaldo José Roccia (Membro)
Mestre
Fatec Americana

Ricardo Kyioshi Batori (Membro)
Mestre
Fatec Americana

Agradecimentos

A todos os docentes da Fatec Americana, pela dedicação em compartilhar os conhecimentos e experiências que adquiriam ao longo de suas vidas. Ao Professor Rossano Pablo Pinto, pelo esmero empregado na orientação, fundamental para o desenvolvimento e conclusão deste trabalho. Aos colegas de classe, a muito tempo não os vejo, mas foi olhando o sucesso de muitos deles que consegui a inspiração para concluir essa jornada. Aos amigos do trabalho, especialmente Marcos Medina, Thiago Santos e Marco Aurelio, pela compreensão e apoio ao me conceder um tempo que não tinha. À minha família, por me ensinarem os valores que estão acima de qualquer conhecimento acadêmico. Finalmente, a minha noiva Patricia Felix por me incentivar nos momentos de maior cansaço, por ser a companheira compreensiva e carinhosa que se abdicou junto comigo de muitos finais de semana para me apoiar nessa tarefa.

Resumo

Este Trabalho de Conclusão de Curso tem como objetivo expor o contexto tecnológico dentro dos *data centers* de grandes corporações, que originou a demanda pelo desenvolvimento das redes definidas por *software*. Descreve-se como era o *design* das redes, os dispositivos utilizados até então e as limitações dos mesmos, bem como os desafios futuros que deverão ser enfrentados no âmbito das redes de *data centers* e as mudanças e avanços tecnológicos que tiveram grande impacto sobre elas. São apresentados os conceitos que definem as redes definidas por *software* e como essa tecnologia pode ajudar a enfrentar os desafios propostos. Dando-se uma rápida explicação de algumas diferentes abordagens de redes definidas por software, que com o passar do tempo de tornou um termo genérico que abriga várias tecnologias e aprofundando o assunto na apresentação do *OpenSDN*, que é a abordagem para redes definidas por *software* desenvolvida pela *Open Network Fundation*. Sendo esse padrão apenas teórico, será exposto o projeto *Opendaylight* e uma implementação prática desse modelo *OpenSDN*.

Palavras Chave: Redes; Redes Definidas por *Software*; Automação de Rede; SDN; Segurança da Informação; *Openflow*;

Abstract

This monograph aims to expose the technological context within the data centers of large corporations, which originated the demand for the development of the software defined networks. It describes how the design of the networks used to be, the devices used and their limitations, as well as the future challenges that must be faced in the area of data center networks, the changes and technological advances that have had a great impact on them. The concepts that define software-defined networks are presented and how this technology can help to meet the challenges posed. Giving a quick explanation of some different approaches to SDN, which over time has become a generic term that harbors various technologies and deepening the subject in the presentation of OpenSDN, which is the approach to software-defined networks developed by the Open Network Fundation. Finally, we will expose the Opendaylight project and a practical implementation of this SDN implementation.

Key Words: Networks; Software defined networks; Network Automation; SDN; Data Security; Openflow;

SUMÁRIO

INTRODUÇÃO.

Recordando a história vê-se que na época do surgimento do computador eletrônico em 1946 com o ENIAC nos Estados Unidos (COMPUTER HISTORY MUSEUM, 2016) as redes de transmissão de dados já eram amplamente utilizadas, como é o caso da *Telex Messaging Network* (TMN), desenvolvida na Alemanha TMN era um sistema evoluído dos telégrafos quê em 1930, conectava teleimpressoras para transmissão de mensagens de texto. Utilizada durante a segunda guerra mundial essa tecnologia esteve presente em alguns países até os anos 2000 (COMPUTER HISTORY MUSEUM, 2016).

No entanto, desde que computadores de redes de transmissão de dados foram colocados para trabalhar juntos em 1958, no sistema de militar de vigilância SAGE (COMPUTER HISTORY MUSEUM, 2016), criaram entre si uma relação simbiótica. Logo nos primórdios da computação, as redes foram essenciais, pois na época computadores ocupavam prédios inteiros e custavam muito dinheiro. Então para justificar o investimento aplicado, eles eram utilizados dia e noite por pessoas espalhadas por todo o país conectadas através de consoles ligados a linhas telefônicas, essas pessoas remotamente compartilhavam os recursos de processamento, ainda assim sem a troca de dados entre computadores.

A partir da criação da Arpanet nos Estados Unidos e outras redes de computadores na Europa, tal como a Cyclades Francesa, o volume da transmissão de dados entre os computadores aumentou exponencialmente, bem como a necessidade de melhores métodos de transmissão que proporcionassem melhores taxas de transmissão, padronização, etc. Isso foi a contrapartida motivacional dada pela computação às telecomunicações, que por sua vez tiveram que se desenvolver rapidamente para dar respostas a essas demandas o quê tem ocorrido desde então. Pode-se evidenciar alguns marcos dessa evolução, como a criação do protocolo TCP/IP, descrito pela primeira vez por Vint Cerf and Bob Kahn em 1973 (COMPUTER HISTORY MUSEUM, 2016); O desenvolvimento do protocolo Ethernet em 1974 nos laboratórios da Xerox (COMPUTER HISTORY MUSEUM,

2016); ou até mesmo no final da década de 70, o desenvolvimento do modelo de referência OSI (COMPUTER HISTORY MUSEUM, 2016).

Sendo assim, seguindo essa tradição a computação continua exigindo das redes cada vez mais dinamismo para que as tecnologias de transmissão de dados possam continuar atendendo suas necessidades. Nesse sentido as redes definidas por *software* (SDN) são a mais recente resposta a essas demandas. Seu desenvolvimento é muito recente e se inicia dentro da Universidade de Stanford com as pesquisas do Grupo Mckeown (MCKEOWN, NICK. et al, 2008) e chegando logo depois aos *data centers* de gigantes da internet tais como Google e Facebook, quando esses percebem que o modelo de redes existente, baseado em dispositivos com *software* e *hardware* proprietários já não era capaz de proporcionar o ganho em escala e a velocidade necessária para acompanhar o crescimento de suas infraestruturas de servidores e *storages*.

> "Dez anos atrás, nós descobrimos que não poderíamos comprar a preço nenhum uma rede para *data centers* que pudesse atender nossa combinação de requisitos de escala e velocidade. Então partimos para construir nosso próprio *hardware* e *software* para infraestrutura de redes de *data centers*" (VAHDAT, 2015).

Explicando um pouco melhor o que Amin Vahdat (VAHDAT, 2015) menciona, pode-se dizer que naquele presente momento os melhores equipamentos de rede que o dinheiro podia comprar não eram capazes de atender a suas demandas pois eram dispositivos pensados para propósitos gerais, ou seja, eles poderiam ser utilizados na rede local de um datacenter ou na interligação com redes WAN por exemplo e isso fazia com que eles fossem recheados de funções, que na maioria das vezes desnecessárias. Esses dispositivos eram autônomos e precisavam atenção de individual sempre que uma atividade gerencial era requerida. Além disso eles ficavam a mercê dos interesses dos fabricantes desses dispositivos, o que travava o avanço tecnológico da área, pois sempre que uma funcionalidade nova era desenvolvida nos centros de pesquisa era necessário aguardar a reação dos fabricantes para a mesma estar disponível nos dispositivos.

No intuito de contornar esses e muitos outros problemas o SDN foi desenvolvido, introduzindo no mundo das redes conceitos consagrados na área de infraestrutura de servidores, tais como a virtualização de equipamentos ou a

utilização de sistemas operacionais base, com APIs programáveis que possibilitam a criação dos mais variados programas trazendo para as redes a velocidade de mutação necessária para acompanhar o desenvolvimento das demais áreas. Além disso foram criados novos conceitos como: a importantíssima separação das funções de controle e encaminhamento de pacotes, explicada completamente nos capítulos seguintes.

O objetivo desse trabalho é expor as falhas do modelo convencional de redes e os novos desafios que a computação propõe para a área e relacionar quais são as propostas SDN para contorná-los, explicando detalhadamente os conceitos envolvidos

No primeiro capítulo é elucidado quais foram os fatores que proporcionaram o aumento da complexidade das redes e são relacionados com maiores detalhes o que é e quais são as limitações do modelo atual de redes.

O capítulo 2 irá introduz os conceitos de SDN de forma relativamente superficial, descrevendo as várias abordagens de SDN, que, com o passar do tempo se tornou um termo “Guarda-Chuva” que abriga visões diferentes de alguns desenvolvedores sobre o que é SDN. Aprofunda-se também os conceitos expostos pela abordagem da *Open Network Fundation,* que é a idealizadora da arquitetura OpenSDN.

O terceiro capítulo apresenta o *OpenDaylight,* que é uma implementação da arquitetura *OpenSDN*. Materializando todos os conceitos vistos até então, mostrando conceitos específicos dessa plataforma necessários para melhor entender o modelo de testes criado no capítulo 4.

Finalmente, as conclusões obtidas durante a implementação de modelo de testes são utilizadas para fazer um paralelo comparando as dificuldades para implementação de uma rede hipotética utilizando-se dispositivos tradicionais em contraposição a implementação utilizando SDN.

1. REDES DE COMPUTADORES

Antes que se coloque os computadores em contexto, para se tratar sobre redes de computadores, deve-se falar um pouco sobre o que é comunicação de maneira geral. O dicionário define o termo como sendo o processo de emissão, transmissão e recepção de mensagens por meio de métodos e/ou sistemas convencionados (EDITORA NOVA FRONTEIRA, 2000). Sendo assim, colocando-se foco em como a ação é realizada, pode-se entender esses métodos convencionados como sendo qualquer mecanismo que se use para transmitir uma mensagem ou informação. A linguagem escrita, falada ou sinais gestuais produzidos com a mão podem ser exemplos disso. Exemplos mais sutis também podem ser apresentados, como o ato de piscar os olhos que também pode ser usado para comunicação, para isso basta apenas que todos os indivíduos envolvidos convencionem qual é o significado desse gesto, e o mesmo vale para qualquer outro método mencionado anteriormente.

Tanenbaum (2003) define redes de computadores como sendo um conjunto de sistemas autônomos interconectados por uma única tecnologia no intuito da troca de informação entre si. Considerando o exposto acima identifica-se nessa definição de redes os mesmos elementos presentes na definição do termo comunicação, onde os indivíduos que querem trocar informações são os sistemas autônomos e o método convencionado entre eles para a troca de mensagens é a tecnologia única, acessível a todos os envolvidos. Praticamente desde sua criação essa tecnologia é o conjunto de protocolos TCP/IP que interconecta máquinas ao redor de todo o mundo, como descreve Frances Cairncross.

> A Internet está implantada de acordo com um único padrão de funcionamento porque o Departamento de Defesa dos Estados Unidos, o maior comprador de computadores no final dos anos 60, foi obrigado por lei a ser imparcial com os fabricantes concorrentes. Havia diferentes tipos de máquinas de computação, cada qual com diferentes padrões de funcionamento. Superar essas incompatibilidades era uma exigência fundamental para conectar computadores. O Primeiro padrão verdadeiro ou "protocolo", só foi completado em 1971. E o TCP/IP (*Transmission Control Protocol/Internet Protocol* - Protocolo de controle de transmissão/Protocolo da Internet), que hoje estabelece o formato no qual todos os dados enviados pela internet são "empacotados", foi projetado, principalmente, por Vinton Cerf e Jon Postel, em 1974 e formalmente lançado em 1983, data normalmente aceita como sendo inicio da Internet. (CAIRNCROSS, 2000)

Dentro da definição de redes existe a categorização das mesmas de acordo com a sua cobertura geográfica e as tecnologias por elas empregadas para o melhor desempenho em diferentes distancias e aplicações, são elas: (TANENBAUM, 1996)

- LAN – *Local Area Networks* – Redes privadas, com a cobertura restrita a prédios ou campus não maiores que alguns quilômetros, elas são amplamente usadas para a interconexão de computadores pessoais em empresas e residências. A principal tecnologia de transmissão utilizada nessas redes são as variações dos protocolos Ethernet.
- MAN – *Metropolitan Area Networks* – Basicamente são versões maiores de LANs, utilizadas geralmente na interligação de escritórios corporativos localizados em uma mesma região, geralmente essas redes utilizam as mesmas tecnologias empregadas nas redes locais, entretanto em uma ligação ponto a ponto sem a utilização de switches.
- WAN – *Wide Area Neworks* – Redes de ampla cobertura geográfica, interligando países ou até continentes inteiros, essas rede são operadas por empresas prestadoras de serviços de telecomunicações e empregam as mais diversas tecnologias, ATM, Frame Relay, MPLS, MetroEthernet, etc.

Em uma rede local, a tecnologia mais difundida atualmente é o protocolo Ethernet, em suas mais diversas variações. Ele é o responsável pela efetiva modulação e controle da transmissão de sinais, que percorrem variados meios físicos (Cabos elétricos, fibras óticas e rádio frequência) carregando consigo as informações. Desenvolvido no final da década de setenta por Robert Metcalfe nos laboratórios de pesquisa da Xerox em Palo Alto nos Estados Unidos, seu sucesso deve-se inicialmente pela iniciativa de Metcalfe de torná-lo um padrão aberto, diferentemente do que faziam seus concorrentes, como por exemplo o protocolo Token Ring proprietário da IBM. (URS, 2002).

Portando, no contexto desse trabalho, quando é referenciado o termo “redes de computadores” fala-se de computadores interconectados entre si formando uma LAN, utilizando-se alguma variação do protocolo Ethernet e o conjunto de protocolos TCP/IP.

1.1. Contextualizando os *data centers* modernos.

Novos serviços *online* surgem frequentemente: Redes sociais, ferramentas de colaboração, comunicação, publicidade, jogos *online*, serviços de compartilhamento de arquivos, transmissão de filmes e músicas por demanda, comercio digital. Segundo estatísticas do ano de 2015 da *International Telecomunication Union* – ITU, agencia das nações unidas para informação e comunicação (ITU, 2015) estima-se que 43% da população mundial formada de usuários dessas aplicações. Para atender tamanha demanda, cada um desses serviços necessita de uma infraestrutura igualmente gigante e é para esse fim que essas companhias mantém *data centers* ao redor do mundo todo (prédios inteiros dedicados a abrigar toda a infraestrutura necessária para o funcionamento das centenas ou até milhares de servidores, unidades de armazenamento de dados e demais dispositivos responsáveis por manter o máximo possível a disponibilidade dos serviços).

No passado esses locais eram destinados a abrigar supercomputadores ou *mainframes*. Do ponto de vista das redes esses gigantes não representavam grande desavio, devido ao pequeno número de *hosts* a complexidade existente se limitava em garantir a disponibilidade da conexão e a banda necessária (HELD, 2000).

No entanto, esse modelo começou a ser questionado após a publicação dos professores da universidade de Berkeley, David Patterson e Randy Katz e o estudante Garth Gibson, sobre os estudos para desenvolvimento dos mecanismos de armazenamento de dados em múltiplos discos, o RAID (*Redundant Array of Inexpensive Disks* – Conjunto Redundante de Discos de Baixo Custo), que provou ser muito mais eficiente e confiável. Com o avanço da arquitetura PC foi observado que assim como no RAID era possível criar sistemas muito confiáveis, potentes e disponíveis agregando-se inúmeras máquinas com *hardware* de baixo custo atuando juntos para oferecer mais desempenho do que os *mainframes* (HAMILTON, 2009).

Entretanto isso tem um reflexo importantíssimo no ambiente de redes dos *data centers*. Essa mudança de paradigma causou um aumento muito significativo da

quantidade *hosts* na rede e consequentemente o número de dispositivos de rede também cresceu na mesma proporção, elevando o nível de complexidade das mesmas.

1.1.1. A convergência de redes.

Não é de hoje que convergência entre tecnologias acontece na área de redes. Na década de setenta a invenção dos microprocessadores possibilitou criação de centrais telefônicas digitais e a codificação digital da fala. Com isso foi possível o desenvolvimento das primeiras redes de propósito geral ou "redes digitais de serviços integrados" (ISDN), que por sua vez causaram a convergência das tecnologias de transmissão de voz e dados, até então distintas, em uma única plataforma. (HANRAHAN, 2007).

Hanrahan (HANRAHAN, 2007), compara a evolução de novas tecnologias com a evolução biológica das espécies, onde a tecnologia mais adaptada sobrevive e outras que não conseguem se adaptar com o passar do tempo caem em desuso e desaparecem. Na área de redes o protocolo Ethernet mostrou grande capacidade evolutiva, sua taxa de transmissão inicial era de apenas 2.94Mbps. Hoje a taxa de transmissão é de 1Gbps. (a adoção em larga escala do padrão de 10Gbps segue em pleno vapor juntamente com a utilização do padrão de 40Gbps nos *uptlinks* entre *switches* e já estão sendo desenvolvidos os padrões de 100Gbps e 400Gbps previstos para o ano de 2017).

Toda essa evolução tornou viável a inserção de outras funções às redes de dados, como por exemplo as tecnologias VoIP e o iSCSI, que trouxeram para as redes as funções de transmissão de chamadas telefônicas e comandos SCSI.

Do ponto de vista econômico e operacional a convergência de tecnologias é extremamente relevante, a possibilidade de entregar vários serviços utilizando-se uma única infraestrutura, gera economia de recursos em todo o ciclo de vida da atividade proporcionando ganhos durante a implantação, manutenção e operação.

1.1.2. Computação em nuvem.

Conforme mencionado no início do capítulo, existem na internet muitos serviços *online* que podem ser acessados por qualquer usuário da internet. No entanto esses acessos não ocorrem de maneira ordenada e nem com um fluxo contínuo de demanda, eles ocorrem em picos de utilização. Pode-se citar como exemplo, um site que publica a classificação de um vestibular: No momento imediato da publicação o número de pessoas que acessam o site é enorme e após isso a demanda pelo serviço cai consideravelmente.

Isso ocorre com a maiorias desses serviços e a infraestrutura que dá suporte a eles tem que estar preparada para manter os sistemas no ar durante esses picos de utilização. Entretanto manter toda essa infraestrutura ociosa esperando por picos de utilização esporádicos custa muito. Contornar esse problema é o objetivo da tecnologia de computação em nuvem. Sendo assim, estando o site hospedado em uma infraestrutura de nuvem, ele será capaz de alocar temporariamente da nuvem, os recursos necessários para atender aos picos demanda e quando esses recursos não forem mais necessários o site os desocupa e o mantenedor do site terá que pagar ao serviço de hospedagem somente pelo tempo que ficou usando os recursos.

> Quando plugado um equipamento elétrico em uma tomada, não se preocupa como a energia elétrica é gerada ou como ela chega até a tomada. Isto é possível por que a energia elétrica é virtualizada; isto é, ela está prontamente disponível em um plugue na parede que "esconde" as estações de geração e as imensas linhas de transmissão. Quando estendemos para tecnologia da informação, esse conceito significa entregar funções uteis, enquanto se esconde como elas funcionam internamente. Computação por si mesma, para ser considerada inteiramente virtualizada, necessita permitir que computadores sejam construídos a partir de componentes distribuídos, assim como processamento, armazenamento de dados, dados e recursos de *software*.
> Tecnologias como clusters, grids e nuvens computacionais, têm todo o foco em permitir o acesso a uma grande quantidade de poder computacional de uma maneira totalmente virtualizada, agregando recursos e oferecendo a visão de um único sistema. Em adição, um importante objetivo dessas tecnologias tem sido entregar a computação como uma utilidade. Computação como utilidade, descreve um modelo de negócios para entrega de poder computacional sob demanda, consumidores pagam provedores baseado na utilização ("pague enquanto usar"), similar ao modo como se paga atualmente serviços tradicionais de utilidade pública, como

> agua, eletricidade, gás ou telefone. (BUYYA, RAJKUMAR; BROBERG, JAMES; GOSCINSKI, ANDRZEJ M. ed, 2011)

Internamente, essa infraestrutura é composta basicamente de alguns elementos, conforme a figura 1. São eles: (JONES, 2012).

- *Storages* – Dispositivos especializados no armazenamento de dados, funcionam como NAS (*Network Attached Storages*) e ficam conectados aos servidores através de uma rede SAN (*Storage Area Networks*) utilizando o protocolo iSCSI.
- Redes – Todos os equipamentos que proporcionam a conexão entre os servidores e *storages* nas SANs e a conexão dos servidores com as demais redes nas LANs
- Servidores – Computadores físicos responsáveis pela execução das máquinas virtuais armazenadas nos *storages*. Geralmente essas máquinas são ligadas nas redes SAN para conexão com os *storages* e nas redes LAN por onde são acessadas as máquinas virtuais que realmente executam os serviços.
- Sistema de Virtualização – Conhecidos também por *Hipervisors*, *softwares* responsáveis pela abstração do *hardware* físico dos servidores, criando as máquinas virtuais onde serão instalados os serviços providos pela nuvem.
- Sistema de gerência da nuvem – *Software* de funciona em conjunto com o *Hipervisor* para o controle da alocação de recursos pelas máquinas virtuais. Esses *softwares* geralmente oferecem uma API para interação com os serviços hospedados na nuvem.
- Serviços – Assim como o nome sugere, são os variados serviços hospedados na nuvem.

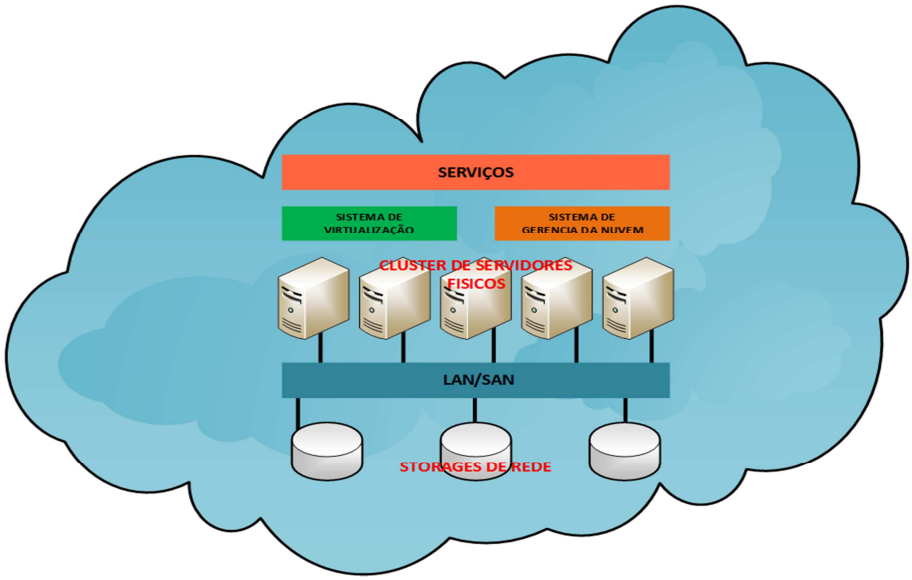


Figura 1. Estrutura interna de uma nuvem computacional.

Fonte: O próprio autor.

1.2. Desafios e o modelo convencional de redes.

.

O foco desse trabalho são as redes locais, no entanto é válida a exposição do crescimento da internet como forma de ilustrar a necessidade do crescimento da infraestrutura dos *data centers* em si, afinal são eles que hospedam os serviços disponíveis na rede mundial.

Nessa exposição, três medidas são importantes, uma delas é a medida do crescimento do número de usuários da Internet no mundo, que mostra o crescimento horizontal da rede, que se espalha através do globo atingindo um número cada vez maior de indivíduos. A segunda é a medida do crescimento da quantidade de dados trafegada pela rede, que revela o crescimento vertical da Internet, que possui cada vez mais dispositivos conectados a ela e cada vez mais serviços disponíveis, o que a torna em alguns lugares utilidade pública de primeira necessidade. A última e mais importante medida para o contexto desse trabalho é a medida do crescimento do tráfego de rede dentro dos *data centers*, que será útil para se avaliar o impacto do crescimento da Internet sobre a infraestrutura local dos mesmos.

Os dados a seguir apresentados, são relativos ao período compreendido entre os anos de 2008 a 2015:

- Crescimento do número de usuários da Internet no mundo – Segundo dados da ITU (agencia da ONU para informação de Comunicação) o número de usuários da *internet* saltou de iniciais vinte e três por cento da população mundial para quarenta e três por cento. Um crescimento da ordem de duas vezes o número inicial dentro do período especificado (ITU, 2015).
- Crescimento da quantidade de dados trafegados pela Internet – Um levantamento publicado pela Cisco, mostra que em 2008 o volume de dados trafegados era da ordem de 6 Exabytes por mês, no ano de 2015 esse número atingiu 42 Exabytes por mês. Um crescimento da ordem de sete vezes o número inicial dentro do período especificado (SUMITS, 2015).
- Crescimento da quantidade de dados trafegados nas redes dos *data centers* – Durante a pesquisa, não foi possível encontrar de fontes confiáveis os valores absolutos do volume de tráfego nas redes dos *data centers* de grandes companhias para referência. Entretanto, em uma publicação no seu blog, o time de infraestrutura do Google afirma um crescimento do tráfego em suas redes locais da ordem de cinquenta vezes o valor inicial dentro do período de 2008 a 2015, (GOOGLE, 2015).

Analisando os números apresentados observa-se que no período o número de usuário da internet duplicou, alcançando metade da população mundial. Seguindo o senso comum, pode-se dizer que com essa tendência de crescimento, futuramente num mesmo período de tempo todo o planeta estará coberto pela internet e então esse crescimento se estabilizará.

Entretanto, vendo os números do crescimento do volume de tráfego que passa pela *internet* observa-se que o crescimento horizontal da rede não é o maior problema. O crescimento vertical da rede é que tem um potencial muito grande de aumento, como pode ser visto na figura 2. Anteriormente existiam nas residências apenas um único computador ligado a internet e que atendia a toda uma família. A

realidade atual é um pouco diferente, cada indivíduo possui um computador e um *smartphone* e é cada vez maior o número de residências que possuem *videogames* e *smart tvs*, todos eles também conectados. Isso é só o começo, novas tecnologias já estão surgindo sob o manto da internet das coisas (IoT – *Internet of Things*) e farão esse número de dispositivos crescer de forma imprevisível e com eles toda uma gama de novos serviços *online*.

Como visto anteriormente o efeito de todo esse crescimento, implica diretamente no aumento da infraestrutura dos *data centers*, na ordem de um para sete, comparando-se com o crescimento do tráfego da *internet*, ou seja, se essa tendência se manter, as redes dos *data centers* deverão suportar um tráfego de rede sete vezes maior para manter o que é trafegado na internet e isso é um grande desafio.

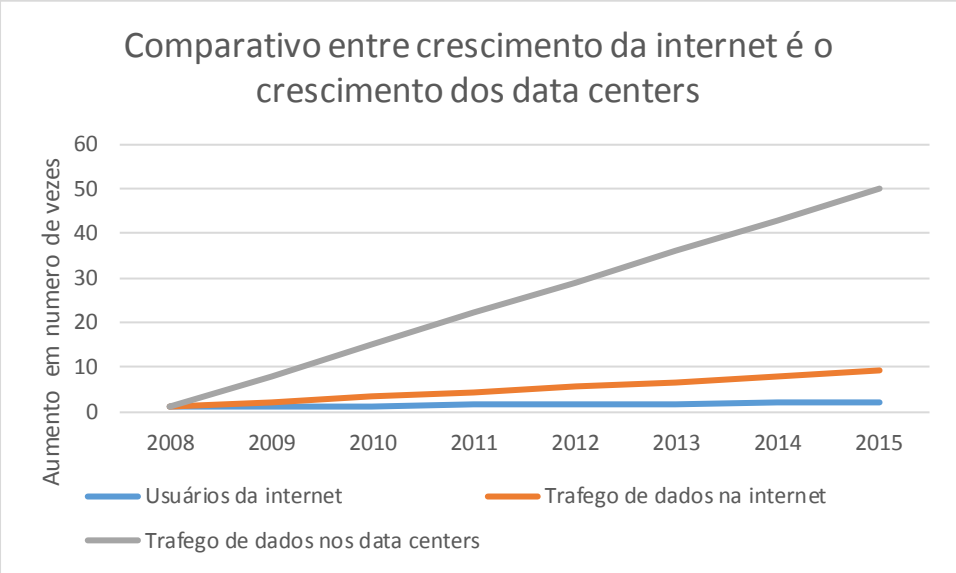


Figura 2. Gráfico comparativo do crescimento de internet e do crescimento das redes dos *data centers*.

Fonte: O próprio autor.

1.2.1. A arquitetura das redes convencionais.

A arquitetura convencional de redes, é baseada em dispositivos denominados ASICs, sigla em inglês para "*application-specific integrated circuit*" ou em uma tradução livre "circuitos integrados de aplicação específica" dispositivos com

hardware e *software* proprietários com funções predefinidas e que não podem ser modificados ou até mesmo programados pelos usuários. Contudo dentro de uma rede existem diferentes necessidades, servidores por exemplo necessitam de mais banda de rede do que uma estação de trabalho. Desse modo são necessários diferentes dispositivos de acordo com o design da rede. Para atender essa necessidade a Cisco elaborou um modelo de design padrão e desenvolveu seus produtos em cima desse design. Esse padrão se tornou amplamente utilizado e outros fabricantes de dispositivos seguiram essa mesma abordagem para seus dispositivos. (CISCO NETWORK ACADEMY, 2014)

O modelo de redes hierárquicas da Cisco é composto de três camadas conforme figura 3, cada uma delas com uma função distinta. São elas:

- Camada do núcleo da rede – Composta por equipamentos com grande capacidade de encaminhamento de pacotes e poucas funcionalidades. É o ponto central da infraestrutura.
- Camada de distribuição – Composta por equipamento ainda com grande capacidade de encaminhamento, no entanto com maiores funcionalidades como por exemplo controles de acessos e funções de QoS.
- Camada de Acesso – Composta por dispositivos de menor capacidade de encaminhamento de pacotes designados para a conexão dos computadores a rede. Devido a isso eles possuem muitas funcionalidades, como controles de acesso, QoS ou PoE.

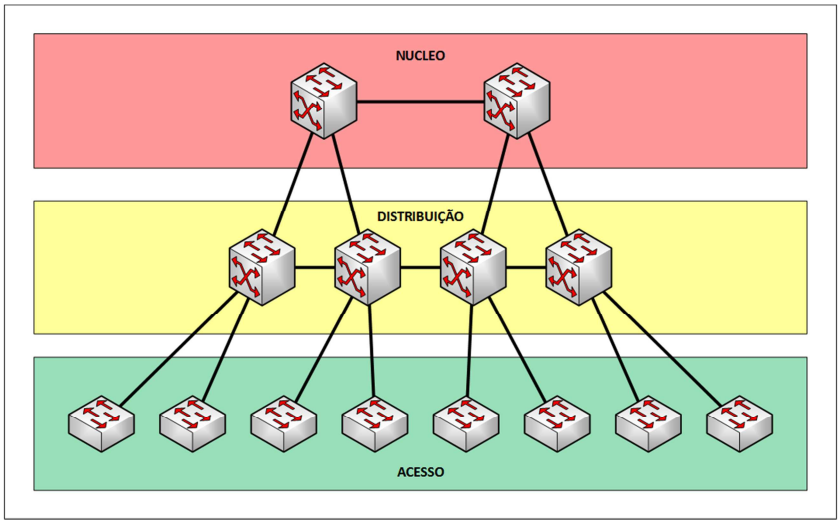


Figura 3. Design de redes hierárquicas da Cisco.

Fonte: O próprio Autor.

Exemplos de dispositivos utilizados em cada uma das camadas:

- Camada do núcleo: Cisco WS-C6504-E. (Figura 4)

Figura 4. Switch Cisco modelo WS-C6504-E.

Fonte: CISCO. Disponível em <http://www.cisco.com/c/en/us/support/switches/catalyst-6504-e-switch/model.html>. Acesso em 12/11/2016.

- Camada de distribuição: Cisco 4500-X. (Figura 5)

Figura 5. Switch Cisco modelo 4500-X.

Fonte: CISCO. Disponível em <http://www.cisco.com/c/en/us/support/switches/catalyst-4500x-32-sfp-switch/model.html>.

Acesso em 12/11/2016

- Camada de acesso: Cisco 2690-X. (Figura 6)

Figura 6. Switch Cisco modelo 2690-X

Fonte: CISCO. Disponível em < http://www.cisco.com/c/en/us/support/switches/catalyst-2960-24pc-l-switch/model.html>

Acesso em 12/11/2016.

1.2.2 Encaminhamento de pacotes.

No que tange o contexto desse trabalho, que são as redes locais que utilizam o protocolo Ethernet, o encaminhamento de pacotes ocorre através da utilização do protocolo ARP sigla para o termo em inglês "*Address Resolution Protocol"*. Desenvolvido por David C. Plummer e publicado como a RFC 826 em 1982, esse protocolo tem a função de mapear endereços IP e endereços "físicos" (Endereços da camada MAC – Subcamada da camada de enlace). Ele funciona da seguinte maneira. No protocolo Ethernet a cada dispositivo ligado à rede é atribuído um endereço, que se consiste de uma sequência de 12 caracteres hexadecimais, esse endereço é atrelado à interface de rede no momento de sua fabricação, ele é único no mundo e esses endereços são referenciados no frame Ethernet como origem e destino no momento da troca de dados na rede.

Isso posto, imagina-se um ambiente hipotético onde três máquinas denominadas aleatoriamente como "A", "B" e "C" estão ligadas em um único *switch* e que nesse ambiente a máquina "A" necessita enviar dados para a máquina "B", para isso ela cria um pacote de dados IP mencionando como destino o IP da máquina "B" que é conhecido previamente. Então logo após isso a máquina "A" encapsula esse pacote IP em um frame Ethernet mencionando como endereço de origem seu próprio endereço Ethernet e como destino o endereço "FF:FF:FF:FF:FF:FF", esse é o maior endereço possível com algarismos da base hexadecimal e foi escolhido como sendo um endereço de broadcast em redes ethernet. Sendo esse pacote de *broadcast*, ele é enviado a todas as máquinas conectadas a esse *switch,* nesse exemplo máquinas "B" e "C". Isso ocorre dessa forma pois no início da comunicação a máquina "A" não conhece o endereço ethernet da máquina "B" e vice e versa. Portanto como o pacote é enviado para todos na rede, o mesmo é analisado por cada máquina que o recebeu, que por sua vez removem o encapsulamento ethernet e verificam se o IP de destino no pacote lhes pertence, caso não pertença o pacote é descartado. Entretanto, se o IP de destino for o da máquina em questão, essa dá continuidade no processamento do pacote, respondendo a máquina de origem, nesse caso a máquina "A" através de um pacote contendo os endereços ethernet de ambos e então a partir desse momento a comunicação é direcionada somente entre

as duas máquinas em questão sem qualquer envolvimento de outros computadores, esse processo é descrito na figura 7. (PLUMMER, 1982)

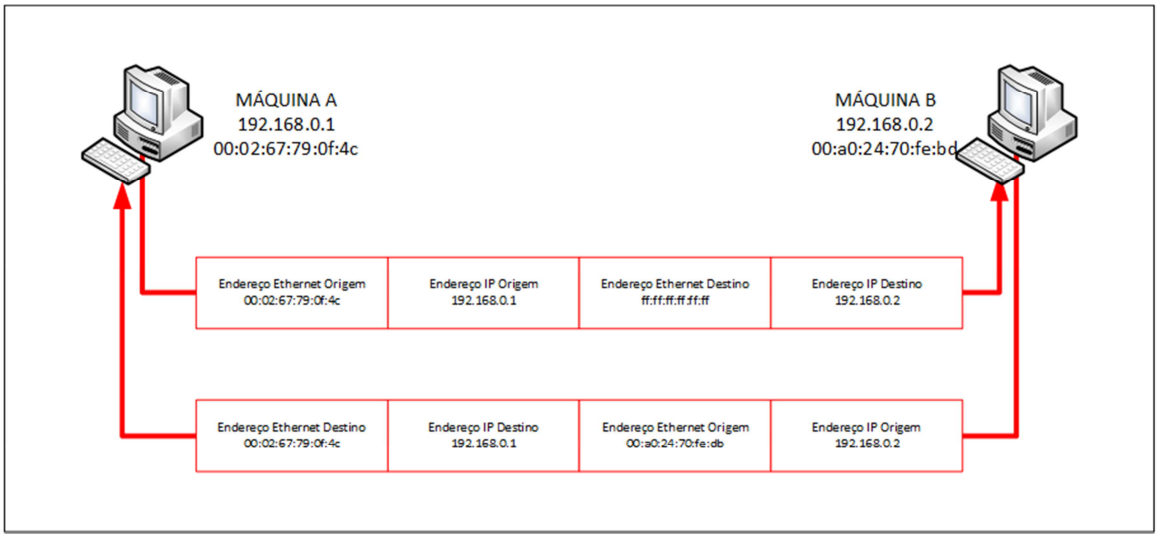


Figura 7. Diagrama de representação do encaminhamento de pacotes em uma rede ethernet.

Fonte: O próprio autor

2. REDES DEFINIDAS POR *SOFTWARE*

Observando o panorama apresentando no capítulo 1, observa-se que o modelo de redes tradicional encontrou obstáculos intransponíveis utilizando-se as tecnologias até então existentes, fazendo-se necessária uma nova abordagem para resolução dos problemas, um novo paradigma de desenvolvimento de sistemas de redes.

Perseguindo seus objetivos, o Grupo *Mckeown* desenvolveu o projeto *Openflow*, que mais tarde fomentou o desenvolvimento do padrão *OpenSDN* objeto de estudo desse trabalho e que por sua vez estabeleceu uma nova interpretação para o conceito de Redes Definidas por *Software* (SDN), propondo contornar os problemas do padrão convencional de rede através da criação de uma infraestrutura de rede que fosse altamente programável e que ainda possibilitasse a implementação em dispositivos tradicionais de rede. (MCKEOWN, NICK. et al. 2008)

Entretanto o conceito de redes programáveis não é novo, em meados da década de 90 já existiam propostas como a do pesquisador K.L. Calvert da Universidade do Kentucky nos estados unidos, que em 1998 propôs a tecnologia *"Active Networks"* que tinha como objetivo estabelecer uma grande evolução no encaminhamento de pacotes nas redes de dados, através da implementação de mecanismos de controle dinâmico e modificação do comportamento das redes. (FEAMSTER, NICK; REXFORD, JENNIFER; ZEGURA ELLEN)

De forma contemporânea aos esforços do grupo Mckeown muitas outras iniciativas vêm buscando implementar programabilidade e/ou recursos de gerenciamento centralizado as redes de dados e para todas essas iniciativas tem se atribuído a denominação "SDN". Sendo assim SDN se tornou um termo "guarda-chuva" que abriga todas essas tecnologias que podem diferir muito da abordagem OpenSDN que tem como suas principais características a separação dos planos de dados e controle e a utilização do protocolo Openflow como interface de comunicação entre o controlador e os dispositivos da rede. A fim de ilustrar quais

são as diferenças entre essas visões de redes definidas por *software* e o OpenSDN, algumas dessas tecnologias são apresentadas nesse capítulo.

2.1. SDN via APIs

No modelo tradicional de redes, os dispositivos são configurados individualmente através de uma CLI (*Command Line Interface)* ou de uma GUI (*Graphical User Interface)*, onde são passados comandos e parâmetros para a configuração apropriada do dispositivo. Sendo esse modelo extremamente desfavorável para a escalabilidade do ambiente, propõe-se para contornar esse problema, a construção de dispositivos de rede que tenham os mesmos mecanismos do encaminhamento tradicional de pacote como descrito na seção 1.2.2 e sem qualquer diferenciação entre planos de dados e controle, mas que possam ser configurados através de uma poderosa API (*Application Program interface*). Assim administradores de redes não necessitam mais conectar individualmente em cada dispositivo para configurá-lo apropriadamente, ao invés disso, é possível programar um script ou até um sistema inteiro de gestão da rede e eliminar todas as tarefas repetitivas, executadas antes através das CLIs e GUIs. (GÖRANSSON, BLACK, CULVER. p 149, 2014)

Exemplo de implementação dessa tecnologia são os switches da linha Nexus da Cisco e seu sistema operacional NX-OS, que podem ser configurados através de sua rica API que pode ser acessada utilizando-se Python.

2.2. SDN via Virtualização.

A virtualização de servidores não é uma novidade. Criar uma camada de *software* abstraindo o *hardware* é uma solução antiga na área de computação (fomentou até mesmo a criação dos sistemas operacionais) e a utilização de um *hypervisor* para instanciar máquinas virtuais já se faz presente na infraestrutura de grandes empresas bem como em pequenos escritórios há alguns anos. Tornar os servidores independentes do *hardware* que os suporta adiciona grandes recursos à

infraestrutura, aumentando muito seu dinamismo, resiliência e disponibilidade. (GÖRANSSON, BLACK, CULVER. p 155, 2014)

Em um ambiente virtualizado, quando ocorre a falha em um servidor físico, basta mover as máquinas virtuais que ele suporta para um outro servidor. Atualizações de sistemas se tornam muito mais simples com a utilização da função de “snapshots” que cria uma imagem dos sistemas da forma como eles eram antes das atualizações, facilitando imensamente procedimentos de *rollback* no caso de alguma falha com as novas versões dos sistemas.

Sendo assim, se a virtualização traz tantos benefícios para a infraestrutura de servidores, porque não trazer esses mesmos benefícios para a infraestrutura de redes? Essa é a abordagem de tecnologias como o NSX da VMWare (descrita na figura 8): criar toda uma infraestrutura rede virtual, composta de switches, roteadores, firewalls, balanceadores de carga e etc, todos instanciados em um hypervisor e sobrepostos sobre o *hardware* dos dispositivos de rede, que a partir de então atuam somente como um modo de conexão entre os diversos dispositivos de rede. Ficando a segmentação, aplicação de ACLs, políticas de QoS e etc a cargo dos dispositivos virtuais, ou seja, toda a inteligência do processo fica fora dos dispositivos físicos, assim como na virtualização de servidores.

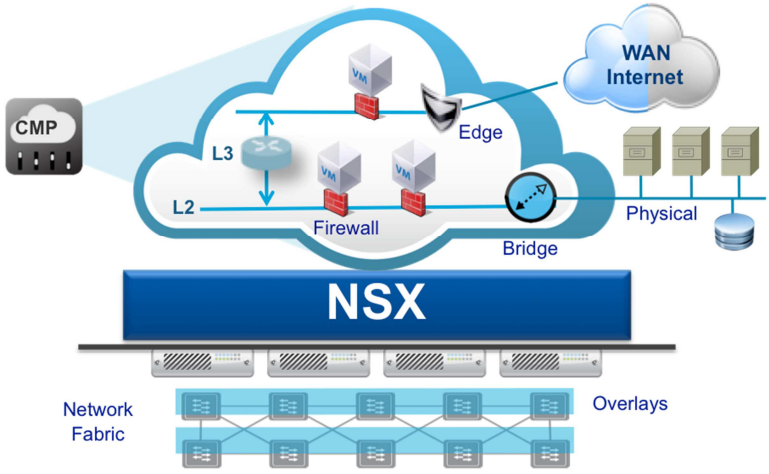


Figura 8. Diagrama de representação da tecnologia NSX da VMWare Fonte: VMWARE. Disponível em <https://cto.vmware.com/introducing-vmware-nsx-the-platform-for-network-virtualization/> Acesso em 12/11/2016.

2.3. White Box Switches

Também conhecidos como “Bare Metal Switches” ou “Brite Box Switches” os White Box Swiches não são exatamente uma tecnologia distinta de SDN, na verdade eles são fruto do movimento dos operadores de datacenters de larga escala para tornar os dispositivos de rede commodities, como parte da iniciativa de criação de uma infraestrutura que superasse os problemas do modelo tradicional de rede.

Um grande exemplo de uma dessas iniciativas são os projetos de switches “Wedge” e “6 Pack” ambos desenvolvidos inicialmente pelo Facebook e abertos como uma plataforma livre disponível através da Open Compute Project. (BACHAR, YUVAL. 2015)

Seu desenvolvimento de baseia na ideia da separação entre *hardware* e sistema operacional do dispositivo, assim como ocorre nos computadores, habilitando esse equipamento para que possa ser utilizado com o sistema operacional que o operador preferir, ao contrário dos dispositivos tradicionais que são uma solução única de *hardware* e *software*.

Devido a essa possibilidade de executar os mais variados sistemas operacionais, os White Box Switches podem ser utilizados para implementação de SDN bem como também para implementação de redes tradicionais. (PICA8, 2014).

Exemplos de sistemas operacionais suportados por esse tipo de switch e que implementam SDN são o PicOS, Cumulos Linux e o Switch Light.

2.4 OpenSDN

O modelo OpenSDN de redes definidas por *software* assim como já abordado no início do capítulo, começou a ser estruturado em 2008 através da divulgação dos trabalhos do Grupo Mckeown sobre a implementação do protocolo OpenFlow, no artigo intitulado “*OpenFlow: Enabling Innovation in Campus Networks*” publicado em 2 de abril daquele ano, os pesquisadores apresentam a proposta de se executar protocolos experimentais em redes que se usa no dia-a-dia. A intenção deles naquele momento era incentivar empresas atuantes no mercado de dispositivos de rede a incluir em seus produtos suporte ao protocolo OpenFlow, pois isso, conforme argumentam no artigo poderia conviver harmonicamente com os sistemas

proprietários que já rodavam nesses equipamentos. Dessa forma esses dispositivos poderiam ser instalados no campus de universidades e usados para executar as mais diversas tecnologias frutos de pesquisas na área de redes, enquanto que ao mesmo tempo poderiam ser usados da maneira tradicional por empresas ou outros que não têm interesses em tais pesquisas. (MCKEOWN, NICK. et al)

Observando-se os eventos consequentes a essa publicação pode-se dizer que os objetivos do Grupo Mckeown foram alcançados, as soluções dos problemas propostas por eles vieram ao encontro dos anseios do mercado, que através de seus grandes representantes começou a interagir em torno dessa tecnologia. A interação entre os acadêmicos e agentes da indústria ganhou corpo até que em 2011, um consórcio inicialmente formado pelas empresas Deutsche, Telekom, Facebook, Google, Microsoft, Verizon e Yahoo! fundou a ***Open Network Foundation*** **(ONF)**, uma organização sem fins lucrativos com o objetivo de padronizar e fomentar a adoção de redes definidas por *software* que seguissem os seguintes preceitos. (OPEN NEWORK FUNDATION, 2013)

- A separação das funções de plano dados e plano controle e a centralização dos plano de controle.
- A adição de recursos de tornem as redes diretamente programáveis.
- A utilização somente de padrões abertos no desenvolvimento da tecnologia.

A esse modelo deu-se o nome de OpenSDN. Existem hoje no mercado algumas implementações dessa visão de SDN e algumas delas são:

- Opendaylight – Controlador de código livre desenvolvido de forma colaborativa pela comunidade e gerenciado pela Linux Foundation. Esse controlador é o exemplo de implantação do modelo OpenSDN escolhido para ser apresentado nesse trabalho, então maiores detalhes sobre esse controlador serão elencados no Capítulo 3.
- VAN SDN Controller – Controlador proprietário desenvolvido pela HP, tem como um de seus principais recursos uma loja *online* de aplicativos aos moldes das lojas de aplicativos para dispositivos móveis como a

Google Play, onde programadores podem colocar à venda seus aplicativos criados para a plataforma.

- RYU - Controlador de código aberto escrito em Python, é extremamente flexível e voltado para o uso em ambientes de pesquisa.

A seguir serão apresentadas em detalhes as principais características que definem esse modelo.

2.4.1. Separação do Plano de Controle e Plano de Dados.

De maneira geral roteadores e demais dispositivos de rede tem uma anatomia parecida com a de computadores, ambos são desenvolvidos seguindo conceitos de organização estruturada em níveis, onde o mais baixo nível são os circuitos eletrônicos do dispositivo e os níveis mais altos da arquitetura são as abstrações proporcionadas por sistemas operacionais, que dão suporte a linguagens de programação de alto nível, utilizadas para criação de *software*s aplicativos, como descrito por Andrew S. Tanenbaum através da figura 9 extraída de seu livro "Organização estruturada de computadores".

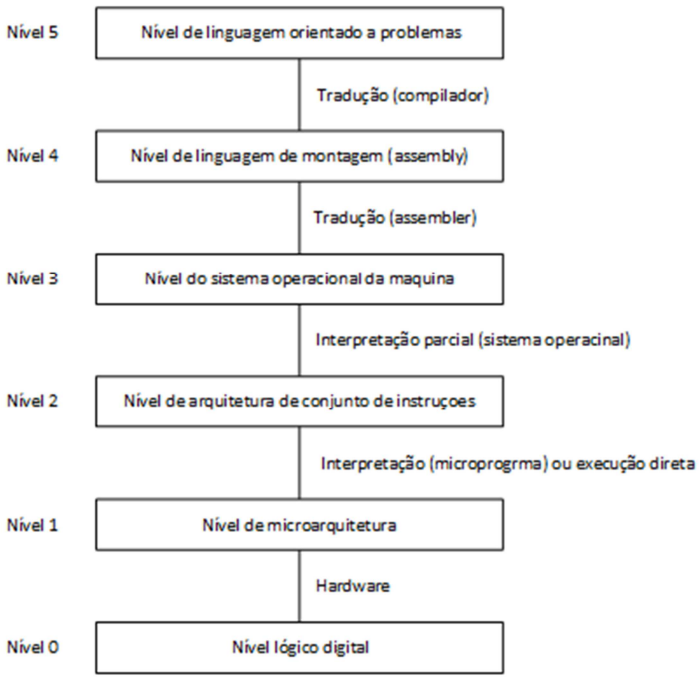


Figura 9. Computador de seis níveis.

Fonte: TANENBAUM, 2007

Levando-se em consideração o que é explicado por Tanenbaum, roteadores e demais dispositivos de rede são computadores otimizados para um propósito específico que é o encaminhamento de pacotes e todo o processamento de dados que isso envolve.

“O desafio de um roteador ou qualquer dispositivo de rede é se manter operacional” (CAMERON, ROB. et al, 2010. 74 p. Adaptado)

Durante muito tempo, interrupções nas redes de dados foram provocadas por travamento de processos executados nos sistemas operacionais de roteadores e switches, esses construídos de maneira monolítica assim como a maioria dos sistemas contemporâneos, possuíam aplicações ou partes do kernel que englobavam muitas funções que poderiam variar do processamento de um protocolo de roteamento até o processamento do encaminhamento de pacotes em sí.

Com o passar do tempo observou-se que as rotinas de encaminhamento de pacotes são processos muito robustos e que raramente apresentam falhas, ao contrário dos demais processos executados por um switch, tais como o processamento de um protocolo de roteamento o uma interface CLI, que podem sofrer mais falhas. Então mantendo juntos esses processos é imposto ao encaminhamento de pacotes a mesma instabilidade dos outros processos. Tendo em vista que o encaminhamento é a função principal desses dispositivos isso deveria ser mudado.

Consequentemente, algumas iniciativas foram tomadas visando essa separação. Uma delas foi a modificação do kernel Linux (sistema base para muitos dispositivos) para a implantação do Netlink que proporcionou a separação dos modulos de encaminhamento de pacotes dos demais processos e criou um sistema de sinalização para o funcionamento coordenado entre eles. Com isso a parte do sistema responsável pelo encaminhamento de pacotes foi denominada “Plano de Dados” e o conjunto as aplicações que controlam o que é encaminhado recebeu a denominação de plano de controle. (J. SALIM. et al, 2003).

Outro exemplo da separação da arquitetura do sistema em plano de dados e plano de controle é o sistema operacional JunOS desenvolvido pela Juniper, que em alguns de seus produtos, implementa a separação funções inclusive utilizando *hardware* distinto para cada uma das partes, como alguns modelos da linha SRX. Dentro dessa plataforma a parte do *hardware* responsável pelo encaminhamento de dados é chamada de "*Packet Forwarding Engine*" (PFE), enquanto a parte responsável pelo controle e chamada de "*Route Engine*" (RE). (CAMERON, ROB. et al, 2010. 75 p.)

Sendo assim, se a separação de funções em planos de dados e controle já está presente em inúmeros sistemas operacionais e plataformas, o que o modelo OpenSDN apresenta de novo quanto a isso ?

A inovação do modelo OpenSDN nessa área tem como características principais a implementação do protocolo OpenFlow como interface entre os planos de dados e controle e a centralização do plano de controle em um único ponto da rede denominado controlador, sendo esse por sua vez um servidor responsável pela execução de todos os processos do plano de controle e que interage com um plano de dados distribuído entre todos os switches da topologia.

Perceba que nos exemplos anteriores a separação existe, mas ambos os componentes permanecem no mesmo dispositivo, ainda que em circuitos eletrônicos diferentes, como no caso da Juniper. Além disso a comunicações entre os elementos da arquitetura ocorre através dos mecanismos de sinalização do seus respectivos kernels, ao contrário do OpenSDN onde essa função é exercida pelo protocolo OpenFlow, apresentado no decorrer desse capítulo com maiores detalhes.

2.4.2 Infraestrutura, Controle e Aplicação

Dentro do modelo OpenSDN a separação dos planos de dados e controle como é uma característica muito importante. Ela possibilita que o design da tecnologia seja estruturado em três diferentes camadas, cada uma delas com uma função distinta. São elas:

- Camada de Infraestrutura: Composta por todos os *switches* e roteadores da topologia. Essa é a camada que abriga o plano de dados da rede, também costuma ser chamado de *Network Fabric.* Sua função é conectar fisicamente todos os elementos da rede e encaminhar os pacotes de um computador a outro uma vez que esse tráfego já tenha sido viabilizado pelo controlador da rede.
- Camada de controle: Composta pelo controlador da rede, como o próprio nome sugere é onde fica centralizado o plano de controle, o ponto central na rede onde são tomadas todas as decisões em relação ao encaminhamento de pacote criando o conjunto de regras de encaminhamento de são enviadas para o plano de dados, além de proporcionar recursos que garantem a programabilidade da rede.
- Camada de Aplicação: Composta por todos programas desenvolvidos por terceiros que interagem com o controlador da rede para prover os mais variados recursos, tais como funções para se adicionar e remover regras de fluxo dos switches da rede, ou a implementação de um novo protocolo de roteamento. A camada de aplicação se apoia na abstração do *hardware* da rede proporcionada pela camada de controle, assim como os aplicativos de um computador se apoiam em seu sistema operacional.

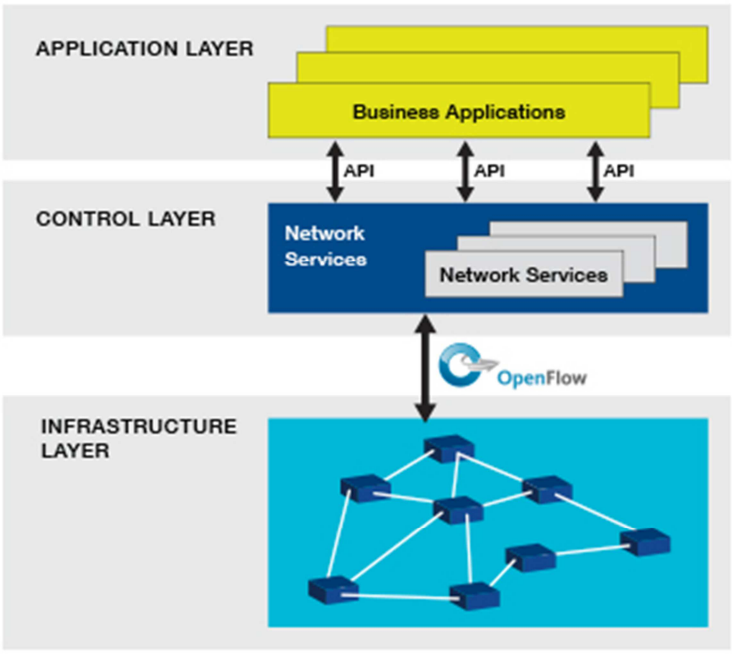


Figura 10. Camadas do modelo OpenSDN.

Fonte: ONF, disponível em <https://www.opennetworking.org/sdn-resources/openflow> Acesso em 12/11/2016.

Como pode ser visto a na figura 10, dentro modelo OpenSDN a camada de controle é o ponto central. O controlador da rede e o elemento que atua entre as camadas de infraestrutura e aplicação, servindo de interface para ambas, de um lado trabalhando de forma intima com o *hardware* da rede e de outro lado garantindo a abstração necessária para o funcionamento das aplicações assim como os sistemas operacionais atuam nos computadores.

Ainda da mesma forma como os sistemas operacionais dos computadores, interagir com as aplicações os controladores disponibilizam APIs e como elas estão presentes para atender a camada superior que é a camada de aplicação, elas são demonindas APIs de lado norte ou ***northbound API***. Assim como não existem especificações definidas para o desenvolvimento dos controladores, não existem especificações para essas APIs, são os dos desenvolvedores dos controladores que as definem. O Opendaylight por exemplo implementa uma API Rest e uma API Java, enquanto que o RYU implementa um módulo Python para ser instanciado dentro da aplicação. (RYU).

Assim como as APIs que atendem a camada superior são chamadas de *northbound* APIs, as que atendem a camada inferior à de controle são chamadas de APIs de lado sul ou *soulthbound* APIs. Em contrapartida as APIs de lado norte para lidar com a camada de infraestrutura os controladores devem implementar protocolos com especificações muito bem definidas como OVSDB, Netconf ou SNMP, que cumprem várias funções relacionadas a configuração e monitoramento dos switches, dentre outras funções que não estão relacionadas ao encaminhamento de pacotes em sí, pois para isso existe o OpenFlow que é o protocolo que proporciona toda a interação entre o controlador e os equipamentos da rede para definição de como os pacotes serão encaminhados.

2.4.3 Openflow

Muitas vezes mencionado como sendo o próprio modelo OpenSDN, o OpenFlow é o protocolo que viabiliza essa visão de SDN. É ele que proporciona o meio pelo qual o plano de controle (camada de controle) irá interagir com o plano de dados (camada de infraestrutura) e vice-versa.

Como mencionado no início do capítulo foi sua publicação que provocou grande movimentação da indústria, que estabeleceu o OpenFlow como um padrão de mercado. Mas voltando um pouco mais no tempo observa-se que a especificação do protocolo OpenFlow foi fruto de outro projeto da Universidade de Stanford, o projeto Ethane, desenvolvido como tese de doutorado por Martin Casado teve como objetivo a implementação de um modelo de redes nos moldes do que veio a se tornar o OpenSDN, ou seja uma rede onde os planos de dados e de controle fossem separados e que o plano de controle fosse centralizado, utilizando o switch programável netfpga como *hardware* base para o desenvolvimento, surgiram como fruto desse projeto o controlador NOX e a especificação do protocolo OpenFlow versão 1.0. (CASADO, MARTIN. Et al, 2007).

Regido pela ONF a especificação do protocolo OpenFlow se encontra atualmente na versão 1.5, entretanto a versão mais difundida do protocolo é a versão 1.3. Nesse trabalho serão apresentadas as características da versão 1.3.

2.4.3.1 Componentes do switch

Desde sua concepção, o protocolo OpenFlow foi pensado para sua utilização em dispositivos de rede Ethernet.

De acordo com a especificação do protocolo, um switch OpenFlow possui os seguintes componentes básicos, descritos na figura 11. São Eles: (OPEN NETWORKINK FOUNDATION, 2012)

- Uma ou mais tabelas de fluxo, que são basicamente listas de regras permitindo ou bloqueando determinados tráfegos de acordo com suas características.
- Um canal de comunicação seguro com o controlador, que se consiste de uma conexão TLS criptografada do switch com o controlador da rede, por onde são transportadas as instruções do protocolo OpenFlow.
- Tabelas de Grupos, contendo que é chamado de entradas de grupos, que por sua vez se consistem de listas de ações a serem executadas com os pacotes de dados e que podem ser especificadas nas regras de fluxo.

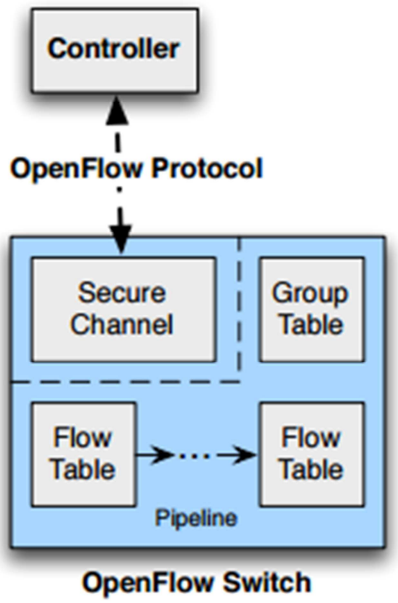


Figura 11. Diagrama de representação da anatomia de um switch Openflow.

Fonte: ONF. Disponível em <https://www.opennetworking.org/ images/stories/downloads/ sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf>. Acesso em 12/11/2016

2.4.3.2 Regras de fluxo e tabelas de fluxo

Uma regra de fluxo explicada de forma simples é uma lista de características capazes de distinguir um determinado tráfego de rede e uma lista de ações que deverão ser aplicadas a esse tráfego sempre que ele ocorra. Por exemplo: sempre que a máquina com ip 192.168.0.1/24 enviar um pacote de dados para um servidor na internet (lista de características), descarte o pacote e informe o controlador sobre esse descarte (lista de ações).

Uma tabela de fluxo é uma lista onde ficam relacionadas todas as regras de fluxo do dispositivo de rede, ou seja, é a lista de regras que determinam como os pacotes serão encaminhados. Ela é gerada pelo controlador e exportada para os dispositivos da rede que não são capazes de altera-la. Somente o controlador é capaz de fazer isso, o que ocorre com frequência. O dispositivo de rede por sua vez à utiliza, lendo as regras de acordo com sua prioridade, que é um número inteiro atribuído à regra, sendo a de menor prioridade a regra em que esse campo é igual a zero, ou seja, quanto maior o número no campo prioridade, maior vai ser a precedência dessa regra em relação às demais. Sendo assim, o dispositivo de rede compara o tráfego que passa por ele com todas as regras da tabela procurando por uma que descreva características que combinem com o tráfego em questão e quando encontra aplica à esse tráfego as ações especificadas. No decorrer desse capítulo é explicado com maiores detalhes como ocorre o encaminhamento de pacotes em uma rede com o protocolo OpenFlow, então esses conceitos ficarão mais claros.

Segue a descrição completa de todos os campos que fazem parte da tabela de fluxo: (OPEN NETWORKINK FOUNDATION, 2012)

- Campos de combinação: onde são especificadas as características do tráfego para o qual as ações serão aplicadas.

- Prioridade: número inteiro positivo que define qual é a prioridade da rede dentro da tabela, sendo zero o número com menor prioridade.
- Contadores: Contador número de pacotes para os quais determinada regra foi aplicada.
- Timeout: Tempo máxima até a expiração da regra e sua exclusão da tabela.
- Cookie: número atribuído à regra pelo controlador, é usado para levantamento de estatísticas sobre o encaminhamento de pacotes, mas não tem relação nenhuma com o encaminhamento de pacotes em si.

2.4.3.3 Regras de fluxo reativas e proativas

As regras de fluxo presentes na tabela de fluxo dos dispositivos de rede OpenFlow podem ser classificadas de duas formas: ou elas são proativas ou reativas. As regras proativas contemplam situações onde se sabe que um determinado tipo de tráfego poderá ocorrer e então cria-se a regras específicas para tratativa desses pacotes. Como no exemplo acima em que se sabendo que uma máquina com determinado endereço IP irá tentar se comunicar com a internet, cria-se a regra para bloquear esse tráfego.

Para contemplar as situações onde não existem regras proativas, são inseridas regras reativas que geralmente repassam para o controlador da rede a decisão sobre o que fazer com o pacote. Geralmente essas são regras com menor prioridade e que não especificam nenhuma característica para distinção de tráfego, ou seja, são regras posicionadas geralmente no fim da tabela de fluxos, combinam com qualquer tipo de tráfego e geralmente encaminham a decisão de como o pacote será encaminhado para o controlador da rede.

2.4.3.4 Nova forma de encaminhar pacotes

Um equipamento de rede OpenFlow é um equipamento Ethernet. Como descrito na seção 1.2.2 que trata sobre como os pacotes são encaminhados no modelo tradicional, a todos os computadores de uma rede são atribuídos endereços MAC que são utilizados para orientar a direção do tráfego. Entretanto dentro de uma rede OpenFlow o tráfego não é encaminhado da maneira tradicional, com os switches realizando transmissões de broadcast para descobrir os endereços MAC dos computadores e populando suas tabelas ARP com eles. Ao contrário disso, em uma rede OpenFlow, os switches não têm autonomia para descobrir onde se encontra um determinado endereço. Quando isso é necessário ele conta com o controlador da rede para essa função.

Sendo assim, em uma rede OpenFlow o encaminhamento de pacotes ocorre da seguinte maneira.

Imaginando-se novamente uma rede hipotética, dessa vez com dois switches OpenFlow nomeados “A” e “B”, interconectados entre si por um *uplink*. Conectado ao switch A encontra-se o computador C e conectado ao switch B encontra-se o computador D e o Controlador da rede, todos endereçados conforme a figura 12.

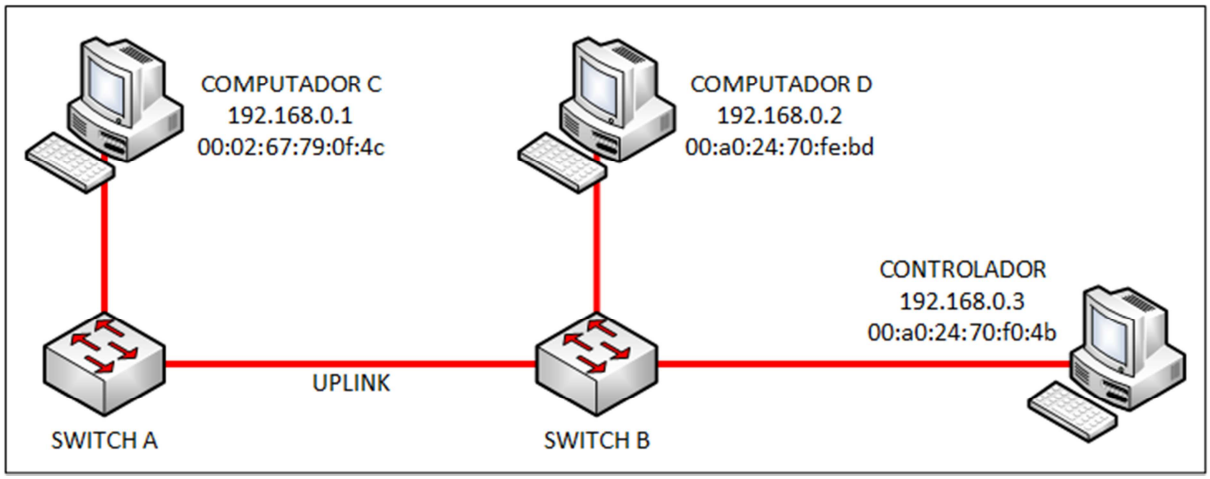


Figura 12. Diagrama de representação do encaminhamento de pacotes em uma rede que utiliza o protocolo Openflow.

Fonte: O próprio autor

Dessa forma quando o computador C envia um pacote de dados para o computador D, ele cria um pacote IP encapsulado em um frame ethernet e o envia para switch A da mesma forma como ocorreria em uma rede tradicional. No entanto, quando esse frame é recebido pelo switch A, ele é analisado e comparado com as regras de fluxo presentes na tabela de fluxo desse switch, até que se encontre uma regra que combine com as características desse frame para que ela então indique uma ação a ser tomada que pode ser permitir ou bloquear o tráfego entre outras. Contudo se nenhuma regra combinar com o tráfego, entra em ação uma regra geralmente posicionada no final da tabela com prioridade igual a zero e que combina com qualquer tráfego que não foi tratado até então. Sua ação é enviar para o controlador através do canal seguro uma mensagem chamada "*packet in*", perguntando ele o que fazer com o frame. Então o controlador irá verificar em suas próprias tabelas ou aplicações conectadas a ele o que fazer com esse novo tráfego. Caso ele não possua nenhuma restrição quanto a essa comunicação, ele responde para o switch através de uma mensagem "*packet out*" dizendo ao switch que ele pode executar um broadcast na rede para identificar onde está o computador destino nessa comunicação, que nesse caso é o computador D.

No momento da resposta do computador D, o mesmo processo ocorre no sentido contrário com o switch B enviando mensagens "*packet in*" para o controlador perguntando o que fazer e recebendo respostas através de mensagens "*packet out*", pois as regras de fluxo são unidirecionais. Então para permitir um determinado tráfego é necessário criar regras que contemplem o envio e o recebimento dos pacotes de dados.

Uma vez que uma primeira troca de pacotes entre as máquinas ocorreu, as tabelas MAC dos switches possuem os endereços de ambas as máquinas e não existem regras bloqueando esse tráfego, o mesmo passa a ser enviado de uma máquina a outra sem a participação do controlador da rede.

2.5. Superando limites, proporcionando segurança da informação.

Por parte dos operadores de grandes redes, atender aos requisitos do mercado é virtualmente impossível com as tradicionais arquiteturas de redes.

Limitadas com orçamentos reduzidos, os departamentos de TI das empresas estão tentando apertar para aproveitar ao máximo os recursos de suas redes. Usando ferramentas de gerenciamento individual de dispositivos e processos manuais, provedores de serviço enfrentam muitos desafios devido à explosão da demanda por banda e mobilidade, impulsionada pela democratização do acesso a dispositivos móveis e notam seus lucros sendo corroídos pelo custo crescente dos equipamentos de rede e quedas de receitas devido à concorrência de mercado. As arquiteturas de rede existentes não foram desenvolvidas para atender a enorme demanda atual dos usuários. Empresas, provedores de serviço e inclusive projetistas de redes estão amarrados pelas limitações do modelo atual de rede que são:

- Complexidade que leva à estagnação: Hoje em dia a tecnologia de redes se consiste de conjuntos de protocolos designados a conectar *hosts* de forma confiável nas mais diferentes distâncias, velocidades e topologias. Para conciliar necessidades comerciais e técnicas, nas últimas décadas, a indústria desenvolveu protocolos de rede para entregar performance, confiabilidade, ampla conectividade e melhor segurança.

  Sendo esses protocolos desenvolvidos de forma isolada, cada um deles resolvendo um problema especifico criou-se grande complexidade na operação das redes atuais. Por exemplo, para adicionar ou remover qualquer dispositivo da rede, a equipe de TI necessita mexer em uma variedade de outros dispositivos tais como roteadores, *firewalls*, *switches* e etc, atualizando configurações de VLANs, ACLs e políticas de QoS, e muitas outras coisas, tudo isso de forma manual com o gerenciamento individual de cada dispositivo. Isso tudo leva a uma estagnação da rede pois devido a essa complexidade os times de infraestrutura procuram mexer o mínimo possível em seus ambientes de rede a fim de não ocasionar interrupções nos serviços comprometendo a disponibilidade da informação. (OPEN NETWORKING FOUNDATION, 2012)
- Inconsistência de políticas de segurança: Para implementar políticas através de toda a rede, o departamento de TI necessita configurar centenas de equipamentos e mecanismos. Por exemplo, sempre que uma nova máquina virtual é instalada, leva-se horas ou até dias para que a equipe de suporte consiga aplicar as ACLs necessárias para seu funcionamento em toda a rede.

Hoje em dia a complexidade das redes torna muito difícil para os times de infraestrutura manter consistentes políticas de segurança em cada canto da rede, e com isso brechas e vulnerabilidades pode surgir a qualquer momento, trazendo para as empresas inúmeras consequências negativas.
(OPEN NETWORKING FOUNDATION, 2012)

- Impossibilidade de escalar: Dentro de um *data center* as demandas crescem rapidamente, então para atende-las a rede deve crescer também. Entretanto como mencionado anteriormente, a rede se torna muito complexa com a adição de centenas ou até milhares de dispositivos de rede, que por sua vez necessitarão de manutenção e gerência. Gigantes da área de tecnologia enfrentam sérios problemas com isso. Esses provedores de serviço empregam algoritmos de processamento de larga escala em seus grids computacionais e conforme o escopo de aplicações de usuários finais aumenta, como por exemplo quando se faz a indexação de toda a internet, o volume de dados processados por cada um dos nós nesses grids aumenta exponencialmente. Então essas empresas necessitam do que vem sendo chamado de redes hyper escaláveis que são capazes de prover alta performance e conectividade em ambiente compostos por centenas de milhares e potencialmente milhões de servidores físicos. Redes assim não são gerenciadas dispositivo a dispositivo manualmente. (OPEN NETWORKING FOUNDATION, 2012)
- Dependência de fabricantes: Empresas e provedores de serviços buscam sempre implementar em suas redes novas funcionalidades e serviços em respostas rápidas a mudanças nos ambientes de negócios e mudanças nos perfis de uso de seus usuários. Entretanto sua rapidez em reagir a isso é limitada pelos fabricantes de dispositivos de rede, os quais tem sempre um ciclo de vida para suas linhas de produtos que em média dura três anos ou mais. Sendo assim novas funcionalidades desenvolvidas somente são colocadas no mercado muito tempo depois pelos fabricantes, e ainda assim caso haja interesse dos mesmos.

2.5.1. Redes sem limitações com OpenSDN.

No primeiro capítulo desse trabalho foram mostradas as características que definem o modelo convencional de redes e os desafios ainda a serem enfrentados

No segundo capítulo foram apresentados os conceitos principais que definem o que é SDN e mais especificamente o que é e como funciona o OpenSDN. Então agora fazendo um contraponto com os problemas relacionados na primeira parte desse capítulo, são apresentadas as soluções propostas no modelo OpenSDN para cada uma das limitações apresentadas. São eles:

- Complexidade que leva a estagnação: No modelo OpenSDN, a diminuição da complexidade da rede se deve a separação dos planos de dados e controle e a centralização do plano de controle, que promoveram a simplificação do *hardware* de rede que não mais necessita de prover funções que responsabilidade do controlador da rede. O controlador por sua vez, proporciona aos administradores de rede um ponto único de onde é possível gerenciar toda a rede, não sendo mais necessário acessar cada dispositivo para alterar configurações sempre que uma mudança ocorre na topologia. (OPEN NETWORKINK FOUNDATION, 2012)
- Inconsistência de políticas de segurança: Nesse ponto novamente a centralização do plano de controle e a implementação de um controlador para rede, proporciona um controle muito mais preciso e completo das políticas de segurança que são aplicadas na rede. O simples fato da configuração da rede estar centralizada em um único ponto e não mais distribuída entre todos os dispositivos da rede facilita muito o trabalho de gestão de políticas que são implementas com uma única ação de forma consistente em todos os dispositivos da rede. Além disso através das APIs de lado norte é possível criar aplicativos que podem consultar e analisar as tabelas de fluxo, automatizando auditorias de segurança entre outras funções de análise da rede. (OPEN NETWORKINK FOUNDATION, 2012)
- Impossibilidade de escalar: Dentro dos grandes datacenters, as funções que adicionam programabilidade à rede são extremamente importantes, pois como foi explicado no tópico que fala sobre *Cloud Computing*, suas

demandas são variáveis. Ferramentas como o OpenStack, interagem com controladores OpenSDN, e gerenciam de forma automatizada todos os recursos de rede presentes na infraestrutura que dá suporte a seus sistemas finais. (OPEN NETWORKINK FOUNDATION, 2012)

- Dependência de fabricantes: De forma geral toda tecnologia OpenSDN liberta da mão da indústria de dispositivos de rede o monopólio que decide quais novas funções são implementadas nos equipamentos. Através dos recursos de proporcionam a programação da rede e a implementação do protocolo OpenFlow, inúmeros desenvolvedores de algoritmos foram habilitados a testar suas ideias em ambientes reais de tráfego.

3. SDN NA PRÁTICA COM OPENDAYLIGHT

O objetivo desse capítulo é apresentar uma infraestrutura que demonstre o funcionamento de uma rede OpenSDN. Devido a sua grande participação no mercado foi escolhido para isso o controlador OpenDaylight, suas principais características são apresentadas na são 3.1 e então na seção seguinte a 3.2 é apresentada com detalhes a implementação prática desse controlador.

3.1. OpenDaylight

Muito mais que um controlador OpenSDN, o OpenDaylight é uma grande plataforma de automação de redes e implementação de redes definidas por *software*.

O OpenDaylight segue fielmente as determinações da ONF, para sua caracterização como sendo uma implementação da tecnologia OpenSDN. A figura 13 representa a anatomia da plataforma. Pode-se ver plenamente distinguidas as camadas do modelo OpenSDN (Aplicação, Controle e Infraestrutura) e como o OpenDaylight cumprindo seu papel na camada de controle se posiciona em relação as demais camadas. (OPENDAYLIGHT.ORG)

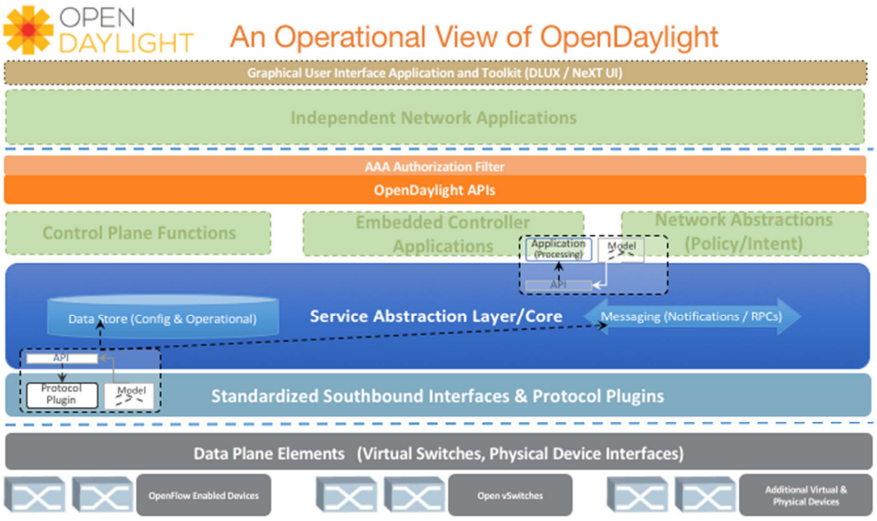


Figura13. Diagrama de representação da anatomia da plataforma OpenDaylight.

Fonte: OPENDAYLIGHT. Disponível em <https://www.opendaylight.org/platform-overview/> Acesso em 12/11/2016

Entretanto o OpenDaylight não se limita a cumprir a especificação da ONF quanto ao modelo OpenSDN, ele vai muito além disso e oferece suporte aos mais variados protocolos de rede e recursos de desenvolvimento e integração da plataforma com outras tecnologias, como por exemplo o OpenStack.

Construído de maneira modular ele é adaptável às mais variadas aplicações, que podem variar desde a simples centralização da gerencia de redes de variados tamanhos, podendo ser utilizado para a entrega automatizada de recursos

computacionais nos ambientes de computação em nuvem até pesquisa acadêmica auxiliando no desenvolvimento de novas tecnologias, dentre muitas outras aplicações.

Para exemplificar sua flexibilidade, pode-se citar o fato de que o OpenDaylight no momento em que é executado, não é capaz de executar nenhuma ação por padrão, sendo necessário antes informar ao sistema uma relação de funções que deverão ser instaladas, para que o usuário possa então ter os recursos necessários para continuar com sua implementação. Essa lista de funções podem ser o carregamento de uma interface gráfica para interação do usuário com o controlador de rede, o carregamento de plugin para interação do controlador com um determinado tipo de switch ou até mesmo o carregamento de uma API que possibilite a interação de uma aplicação com o controlador.

Toda essa modularidade garante ao Opendaylight uma variada gama de funções sem torná-lo um sistema inchado ou que necessite de grandes investimentos para sua implementação.

3.1.1. APIs de lado norte.

Como pode ser visto na figura 13, entre as camadas de aplicação e controle o OpenDaylight posiciona uma camada de APIs responsáveis por garantir um meio de interação entre os programas de terceiros e o controlador.

De modo simples pode-se dizer que o OpenDaylight disponibiliza dois tipos diferentes de APIs, uma delas é uma API Java que utiliza componentes OSGI para interação com o controlador. Utilizando-se essa API é possível criar aplicações capazes de reagir a eventos que ocorrem com a rede, de acordo com mensagens “packet-in” enviadas pelos dispositivos. Aplicações que fazem uso dessa API devem ser executadas no próprio controlador de rede, de maneira semelhante aos programas que são executados nos computadores e que fazem uso das APIs de seus sistemas operacionais. Por essa característica pode-se dizer que o OpenDaylight é um sistema operacional de redes. (OPENDAYLIGHT.ORG)

Um outro tipo de API que o OpenDaylight disponibiliza é uma API do tipo Rest, amplamente utilizada em webservices de grandes sites como o Facebook, Twitter, Google Maps entre outros. Rest é um tipo de API que utiliza o protocolo HTTP para o transporte de requisições descritas utilizando-se os padrões XML ou JSON entre um cliente e um servidor e também as respostas desse servidor de volta ao cliente, descritas da mesma forma.

As aplicações que fazem uso da API Rest podem ser executadas no próprio controlador, assim como as que usam a API java, ou podem também ser executadas em qualquer computador da rede que tenha acesso ao controlador, inclusive através da internet. Entretanto, através da API Rest não é posível ler as mensagens "packet-in" recebidas pelo controlador, o que impossibilita programar *flows* de forma reativa, é possível somente programar flows de forma proativa.

3.1.2. APIs de lado sul.

Quando se fala das APIs de lados sul, trata-se dos protocolos, plug-ins e demais recursos suportados pelo OpenDaylight para interação com os dispositivos de rede que compõem o plano de dados, como por exemplo o protocolo OpenFlow, descrito com maiores detalhes no capítulo 2. (OPENDAYLIGHT.ORG)

Alguns outros protocolos e plug-ins suportados seguem relacionadas a seguir.

- NETCONF – Especificado através da RFC 6241 esse protocolo é utilizado para a configuração geral de dispositivos de rede, mudando parâmetros de gerência, tais como configurações de endereçamento IP de administração do dispositivo, dados de monitoramento via SNMP, configuração de logs centralizados e etc.
- OVSDB – Especificado através da RFC 7047 esse protocolo é responsável por prover ao OpenDaylight os recursos necessários para a interação com os switches virtuais Open vSwitch.
- OPFLEX – Desenvolvido pela Cisco, esse protocolo é quase que um concorrente do protocolo OpenFlow, entretanto sem uma de suas principais características, a centralização do plano de controle.

3.2. Cenário.

Para atingir objetivo de implementar uma infraestrutura de rede SDN é utilizado o *software* de virtualização Virtualbox na versão 5.1.10 e sobre ele instanciadas máquinas virtuais que emulam as camadas do modelo, da maneira como segue:

- Camada de infraestrutura: Para emular a camada de infraestrutura foi instanciada uma máquina virtual chamada MININET. Ela executa o sistema operacional Ubuntu na versão 14.04 e um aplicativo também chamado Mininet na versão 2.2.1. No site do projeto Mininet (mininet.org) é possível fazer o *download* dessa mesma máquina virtual com todos os aplicativos necessários em formato OVA pronta para ser importada no Virtualbox. Esse aplicativo de código aberto, criado inicialmente por Bob Lantz e Brandon Heler , tem a capacidade de criar toda uma infraestrutura de switches e *hosts* virtuais conectados a esses switches, formando um ambiente de testes que pode ser integrada com um controlador SDN interno aos próprios switches que ela cria (OVS) ou externo como é nesse ambiente que é proposto. Com o auxílio do Mininet foi criada uma infraestrutura de testes composta por quatro switches ligados entre si formando uma topologia linear e em cada um desses switches foi conectada uma máquina virtual, conforme ilustra a figura 14. É sobre o tráfego de rede dessas máquinas que o Openflow atua criando políticas que permitem ou bloqueiam determinados serviços ou até todo o tráfego se assim o usuário julgar necessário.

  Um ponto importante a ser explicado sobre o Mininet é que ele divide a infraestrutura de testes criada em dois seguimentos de rede distintos: um deles que conecta as máquinas virtuais que são alvo do teste e outro que conecta os switches da topologia ao controlador SDN através de um NAT configurado na interface de rede da máquina onde o Mininet é executado. Na Figura 14 esses dois seguimentos distintos são representados através das cores diferentes

  nos links que interligam os elementos. Os de cor vermelho indicam o seguimento da rede que interliga as máquinas virtuais de teste e os de cor

azul indicam o seguimento de rede que interligam os switches com o controlador SDN.

- Camada de controle: Para emular a camada de controle foi instanciada a máquina virtual chamada CONTROLLER, ela executa o sistema operacional Debian GNU/Linux na versão 7.1 e o OpenDaylight na versão Beryllium-SR4. Este, como dito anteriormente, necessita que sejam instaladas funções para que ele tenha os recursos necessários para essa topologia, nesse caso foram instaladas as funções: “odl-restconf”, que é a API Rest, “odl-l2switch-switch”, plugin necessário para interação com os switches Open vSwitch, “odl-mdsal-apidocs”, ferramenta de navegação pela API da camada de abstração, por fim odl-dlux-all que é a interface web padrão para interação com o controlador.
- Camada de aplicação: Para emular a camada de aplicação foi instanciada a máquina virtual chamada MANAGER. Ela também executa o sistema operacional Debian GNU/Linux na versão 7.1 e sobre ela é executada a aplicação desenvolvida durante a elaboração desse trabalho e que interage com o controlador através de sua api Rest. Essa aplicação por sua vez se chama “Flow-manager”. Ela se consiste de um script escrito em GNU Bash Shell Script na versão 4.2 que é capaz de interagir com o controlador da rede e listar fluxos cadastrados em um switch, adicionar novos fluxos e remover fluxos existentes.

  Para realizar a operação de listagem de fluxos essa aplicação faz uso do utilitário cURL na versão 7.26 que realiza uma requisição HTTP ao controlador interagindo com a API Rest e retornando para a aplicação uma lista de dados codificada em um arquivo XML. Esse arquivo então é interpretado utilizando-se o utilitário XMLStarlet na versão 1.3.1 e assim uma listagem contendo os fluxos do switch em questão é impressa na tela como retorno para o usuário.

  Para realizar as operações de inclusão e remoção de fluxo, são realizadas através do utilitário cURL, enviando requisições HTTP contendo tais instruções ao controlador SDN.

Essas três máquinas são conectadas entre si em uma rede local conforme a figura 14.

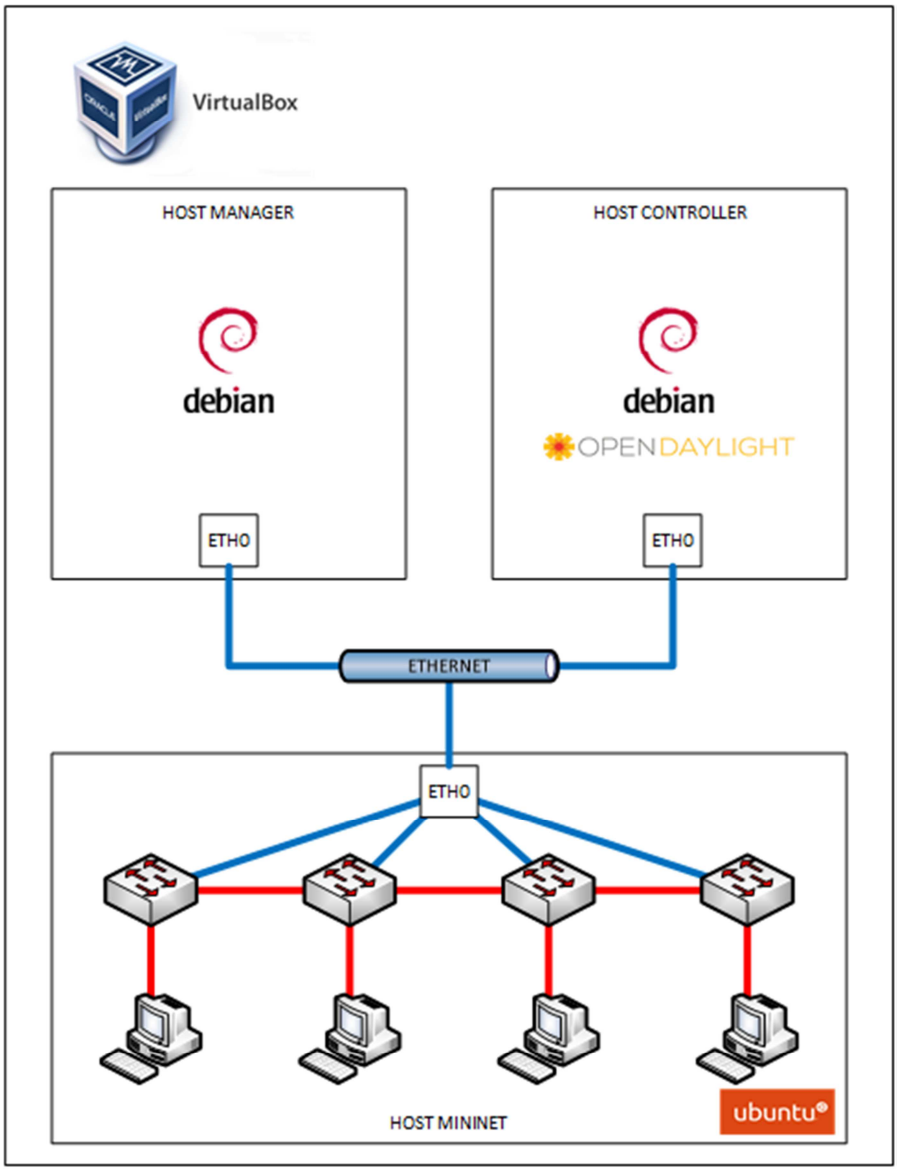


Figura 14. Diagrama de representação da topologia de testes criada para a demonstração pratica.
Fonte: O próprio autor.

3.3. Resultado.

As figuras apresentadas a seguir, são "*printscreens*" das telas da aplicação "Flow-manager" de cada uma das funções do sistema, acompanhadas da parte do código fonte da aplicação responsável pela função.

A Figura 15 apresenta o menu de opções do programa, com as funções de listagem de fluxos, adição de fluxo e remoção de fluxos, além disso uma quarta opção foi adicionada para encerrar a execução do programa.

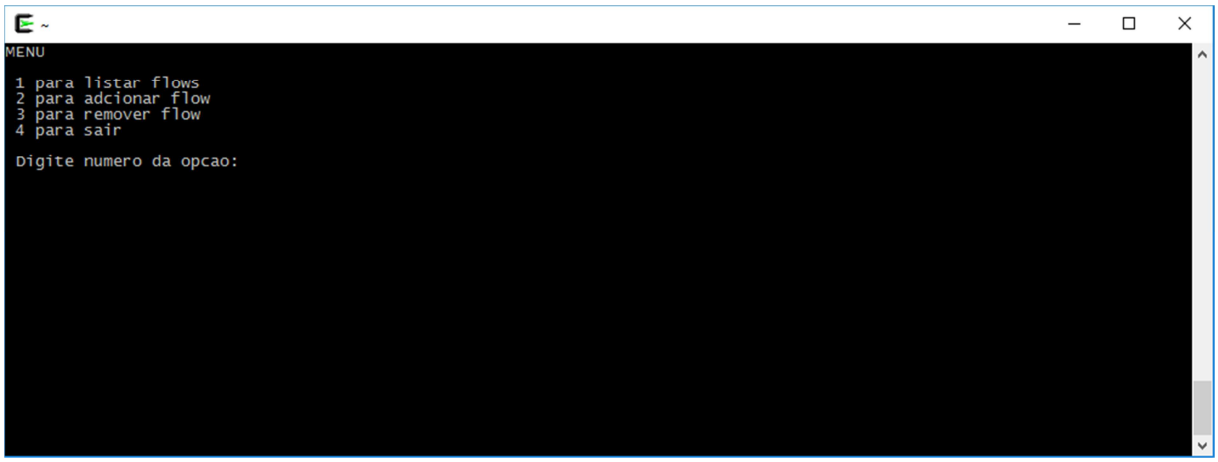


Figura 15. Menu de opções do programa Flow-manager.
Fonte: O próprio autor

O código fonte que implementa o menu principal do sistema é onde reside toda a lógica do programa, ele se consiste de um laço "*while*" que roda de modo perpétuo, repetindo a execução do programa indefinidamente e sempre aguardando que o usuário forneça como entrada para a variável "*OPTION*" um número que corresponda a uma das opções do sistema "1" para a função de listagem de fluxos, "2" para a função de adição de fluxos, "3" para a função de remoção de fluxos e "4" para a função que encera a execução do programa.

Quando a variável é preenchida pelo usuário um condicional do tipo "*case*" é acionado e executa de acordo com a opção do usuário o código de uma das quatro funções do sistema. Esse fragmento de código pode ser visto na figura 16.

```
#Inicio Programa.

        while [ true ]
                do
                clear
                OPTION=""
                printf "MENU\n\n"
                printf " 1 para listar flows\n"
                printf " 2 para adcionar flow\n"
                printf " 3 para remover flow\n"
                printf " 4 para sair\n\n"
                read -p " Digite numero da opcao: " OPTION
                case "$OPTION" in
                        1)
                                list-flow
                                ;;

                        2)
                                add-flow
                                ;;

                        3)
                                del-flow
                                ;;

                        4)
                                quit
                                ;;

                esac
                done

#Fim do Programa.
```

Figura 16. Código fonte do menu de opções do programa Flow-manager.
Fonte: O próprio autor

A Figura 17, apresenta a rotina de listagem de fluxos, onde ela pergunta ao usuário qual o switch do qual se deseja listar os fluxos, então retorna a lista dos fluxos cadastrados no switch em questão.

```
MENU

 1 para listar flows
 2 para adcionar flow
 3 para remover flow
 4 para sair

 Digite numero da opcao: 1
***LISTAGEM DE FLOWS***

Digite o nome do switch que deseja listar os flows e aperte [ENTER]: openflow:1

ID                      PRIORITY   SRC-IP              DST-IP              SRC-PORT   DST-PORT   ACTION
2                       2000                           10.0.10.2/24
3                       2003       17.1.2.3/8          172.168.5.6/16                 8080
40                      4000       10.0.0.0/24         10.0.1.0/24         4500       80
30                      3000       10.0.0.0/24         10.0.1.0/24         45888      80


Press enter to continue...
```

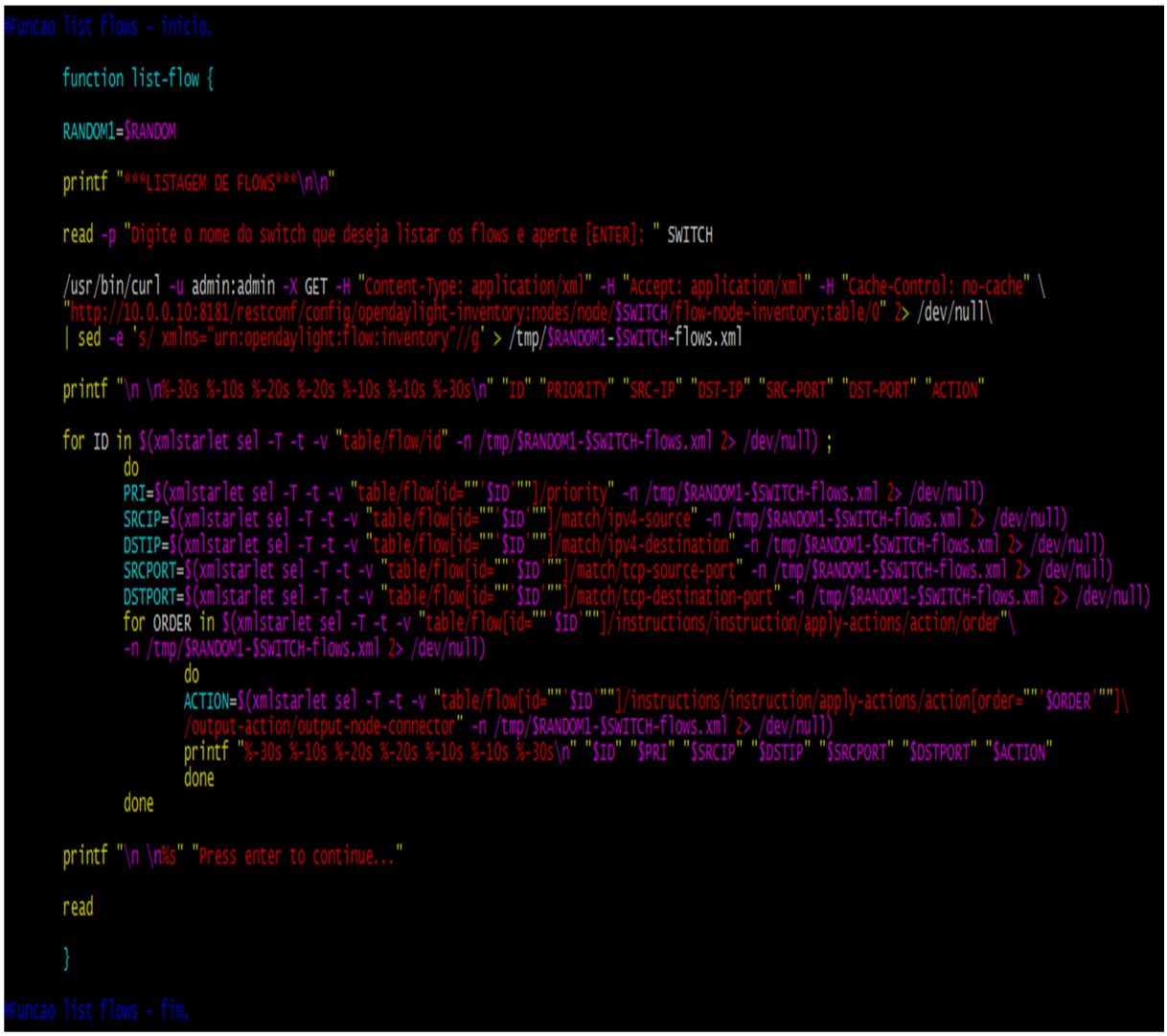
Figura 17. Listagem de fluxo gerada pelo aplicativo Flow-manager.
Fonte: O próprio autor.

Na figura 18 é exposto o código fonte da função de listagem de fluxo, essa é a maior função do sistema e sua lógica se consiste em perguntar ao usuário o nome do

switch do qual serão listados os fluxos, e armazenar esse dado na variável "*SWITCH*".

Após isso com a ajuda do utilitário cURL, o sistema faz uma requisição HTTP ao controlador da rede e a resposta do controlador é armazenada em um arquivo de texto no formato XML dentro do diretório "/tmp" da máquina onde o programa é executado.

Em um terceiro passo esse arquivo armazenado no diretório "/tmp" e lido com a ajuda do utilitário XMLStarlet, que em um laço do tipo "*for*" percorre todas as entradas contidas no arquivo XML e as lista em forma de uma tabela com a ajuda do comando "printf" para dar a devida formatação.



```
#funcao list flows - inicio.

        function list-flow {

        RANDOM1=$RANDOM

        printf "***LISTAGEM DE FLOWS***\n\n"

        read -p "Digite o nome do switch que deseja listar os flows e aperte [ENTER]: " SWITCH

        /usr/bin/curl -u admin:admin -X GET -H "Content-Type: application/xml" -H "Accept: application/xml" -H "Cache-Control: no-cache" \
        "http://10.0.0.10:8181/restconf/config/opendaylight-inventory:nodes/node/$SWITCH/flow-node-inventory:table/0" 2> /dev/null\
        | sed -e 's/ xmlns="urn:opendaylight:flow:inventory"//g' > /tmp/$RANDOM1-$SWITCH-flows.xml

        printf "\n \n%-30s %-10s %-20s %-20s %-10s %-10s %-30s\n" "ID" "PRIORITY" "SRC-IP" "DST-IP" "SRC-PORT" "DST-PORT" "ACTION"

        for ID in $(xmlstarlet sel -T -t -v "table/flow/id" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null) ;
                do
                PRI=$(xmlstarlet sel -T -t -v "table/flow[id=""$ID""]/priority" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                SRCIP=$(xmlstarlet sel -T -t -v "table/flow[id=""$ID""]/match/ipv4-source" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                DSTIP=$(xmlstarlet sel -T -t -v "table/flow[id=""$ID""]/match/ipv4-destination" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                SRCPORT=$(xmlstarlet sel -T -t -v "table/flow[id=""$ID""]/match/tcp-source-port" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                DSTPORT=$(xmlstarlet sel -T -t -v "table/flow[id=""$ID""]/match/tcp-destination-port" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                for ORDER in $(xmlstarlet sel -T -t -v "table/flow[id=""$ID""]/instructions/instruction/apply-actions/action/order"\
                -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                        do
                        ACTION=$(xmlstarlet sel -T -t -v "table/flow[id=""$ID""]/instructions/instruction/apply-actions/action[order=""$ORDER""]\
                        /output-action/output-node-connector" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                        printf "%-30s %-10s %-20s %-20s %-10s %-10s %-30s\n" "$ID" "$PRI" "$SRCIP" "$DSTIP" "$SRCPORT" "$DSTPORT" "$ACTION"
                        done
                done

        printf "\n \n%s" "Press enter to continue..."

        read

        }

#funcao list flows - fim.
```

Figura 18. Código fonte da função de listagem de fluxos do programa Flow-manager.
Fonte: O próprio autor

A Figura 19, apresenta a rotina de adição de um novo fluxo. Ela pergunta ao usuário qual o switch a ser cadastrado o fluxo, o ID do fluxo, IP de origem e destino e portas TCP de origem e destino. Como se trata de uma aplicação simples somente para demonstração do que é possível fazer com uma rede programável, a única ação possível até o momento é bloquear todo o tráfego identificado no fluxo e isso se faz cadastrando um fluxo sem nenhuma ação, conforme foi explicado no Capítulo 2. Por isso a ação a ser executada não é perguntada ao usuário. Sendo assim a ação dos fluxos cadastrados através do Flow-manager sempre será o bloqueio do tráfego de rede identificado no fluxo.

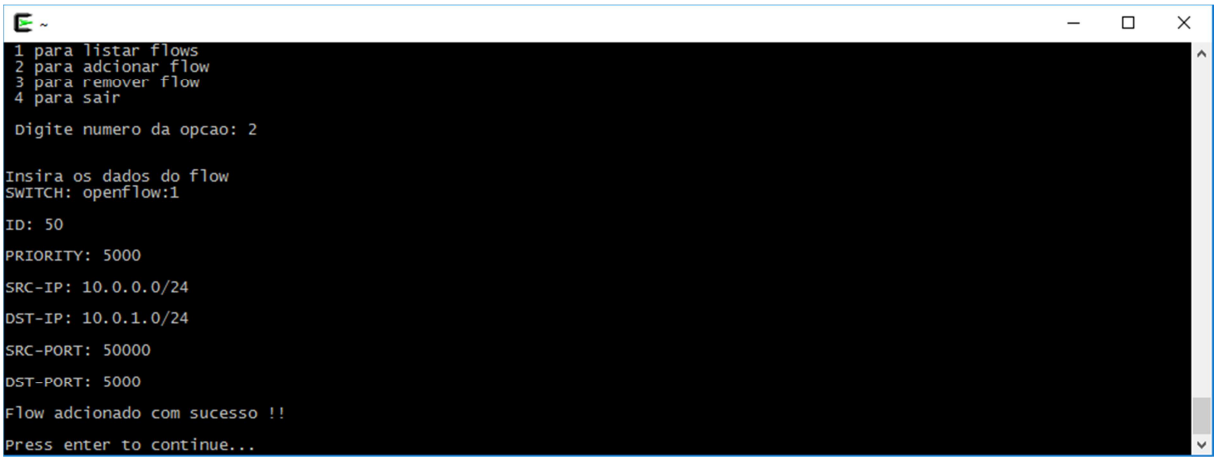


Figura 19. Adição de um fluxo utilizando o aplicativo Flow-manager.
Fonte: O próprio autor.

A função de adição de fluxos é composta de duas etapas, na primeira são feitas perguntas ao usuário sobre as informações necessárias para se definir o fluxo, essas informações são armazenadas em variáveis que serão utilizadas na segunda etapa da função. O código fonte dessa primeira etapa pode ser visto na figura 20.

Figura 20. Primeira parte do código fonte da função de adição de fluxos do programa Flow-manager.
Fonte: O próprio autor

A segunda parte da função de adição de fluxos de dedica a fazer novamente uma requisição HTTP ao controlado utilizando o utilitário cURL, com a instrução PUT do solicitando ao controlado que grave as instruções contidas do corpo da requisição codificas em XML preenchido com o conteúdo das variáveis informados pelo usuário na etapa anterior. O código fonte da segunda parte a função de adição de fluxos pode ser visto na figura 21

Figura 21. Segunda parte do código fonte da função de adição de fluxo do programa Flow-manager.
Fonte: O próprio autor

A Figura 22, apresenta a rotina de remoção de um fluxo. Ao ser executada a rotina pergunta ao usuário o nome do switch do qual o fluxo será removido e o ID do fluxo. Vale lembrar que a remoção do fluxo permite que o tráfego de rede volte a ser encaminhado no switch, pois não existe mais na tabela um fluxo que o identifique,

dessa forma é aplicada a ele a ação padrão cadastrada no switch que é a permissão do encaminhamento.

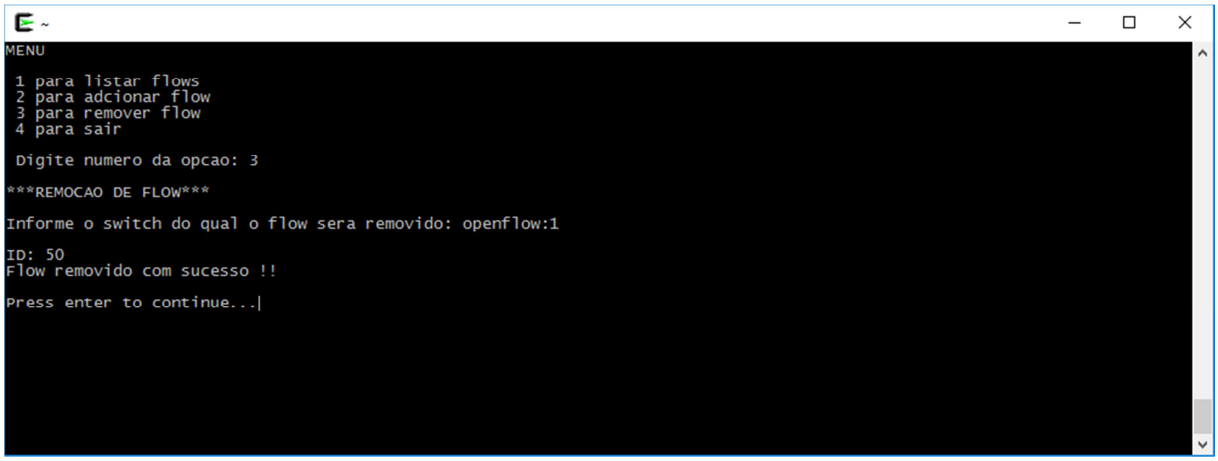


Figura 22. Remoção de um fluxo utilizando o aplicativo Flow-manager.

Fonte: O próprio autor.

O código fonte da função de remoção de fluxos (Figura 23) se assemelha muito ao código da função de adição. Primeiro ele solícita as informações necessárias para identificação do fluxo, armazena essas funções em variáveis e então executa novamente o cURL com uma requisição HTTP ao controlador, formulada utilizando os valores contidos nas variáveis informadas pelo usuário.

```
#Funcao remove flow - inicio

        function del-flow {

        printf "\n***REMOCAO DE FLOW***\n\n"

        read -p "Informe o switch do qual o flow sera removido: " SWITCH

        printf "\n"

        read -p "ID: " ID

        curl -u admin:admin -X DELETE -H "Content-Type: application/xml" -H "Accept: application/xml" -H "Cache-Control: no-cache"\
        "http://10.0.0.10:8181/restconf/config/opendaylight-inventory:nodes/node/"$SWITCH"/table/0/flow/"$ID""

        if [ "$?" -eq "0" ]
                then
                printf "Flow removido com sucesso !!"
                else
                printf "Falha ao remover flow"
        fi

        printf "\n \n%s" "Press enter to continue..."

        read


        }

#Funcao remove flow - fim.
```

Figura 23. Código fonte da função de remoção de fluxo do programa Flow-manager.

Fonte: O próprio autor

4. CONSIDERAÇÕES FINAIS.

Na elaboração desse trabalho, foram utilizados livros de autores renomados, publicações acadêmicas de autoria dos desenvolvedores da tecnologia, *white papers* de grandes empresas do seguimento tais como Cisco, Big Switch networks entre muitos outros. Como resultado da pesquisa e experimentações realizadas de acordo com a proposta do mesmo, algumas considerações finais são apresentadas a seguir.

Durante a implementação desse modelo de testes, foi possível verificar o enorme potencial que o OpenDaylight tem para possibilitar a automação completa de uma rede, seus recursos que habilitam às redes se tornarem programáveis descortinam um novo universo no campo da transmissão de dados, onde são muito claros os ganhos em produtividade na gestão da infraestrutura, principalmente quando a escala é muito grande.

Nesse ambiente de testes composto por apenas quatro switches foram elaboradas rotinas de gestão que são aplicáveis a esse pequeno ambiente e também são aplicáveis em ambientes com centenas ou até milhares de switches. Funções repetitivas podem ser descritas em um algoritmo e executadas N vezes sem qualquer interação com o usuário, modificando a configuração de inúmeros dispositivos com um único clique, poupando inúmeros recursos do operador da rede e melhorando parâmetros de disponibilidade do ambiente, uma vez que erros humanos não são mais um problema.

A simples função de listagem de fluxos serve de exemplo de como pode ser simples a auditoria das políticas de segurança da rede. Com poucos passos é possível obter uma listagem com todas as regras de encaminhamento de um switch, sem a necessidade de se conectar manualmente em cada equipamento da rede e navegar por sua interface que pode variar de modelo para modelo ao acessar a função desejada e obter o mesmo resultado. Essa simplicidade de auditoria confere à rede grande melhora em sua segurança, proporcionando confiabilidade ao dado transmitido.

Observando todas as características do modelo OpenSDN, é nítido seu potencial de para tornar as redes de dados melhor nos quesitos confidencialidade, integridade e disponibilidade, proporcionado por consequência segurança da informação.

REFERÊNCIA BIBLIOGRÁFICA.

BACHAR, YUVAL. **Introducing "6-pack": the first open hardware modular switch**, 2015. Disponível em: <https://code.facebook.com/posts/717010588413497/Introd

ucing-6-pack-the-first-open-hardware-modular-switch/> Acesso em 15/11/2016.

BUYYA, RAJKUMAR; BROBERG, JAMES; GOSCINSKI, ANDRZEJ M. ed. **Cloud Computing: Principles and Paradigms,** Jhon Wiley & Sons, 2011.

CAIRNCROSS, FRANCES. **O fim das distancias: Como a Revolução Nas Comunicações Transformará Nossas Vidas**, 2000.

CAMERON, ROB. et al. **Junos Security.** 2010.

CASADO, MARTIN. Et al. **Ethane: Taking Control of the Enterprise,** 2007. Disponível em: <http://yuba.stanford.edu/~nickm/papers/ethane-sigcomm07.pdf>. Acesso em 15/11/2016.

CISCO NETWORK ACADEMY**. Hierarchical Network Design,** CiscoPress, 2014. Disponível em:<http://www.ciscopress.com/articles/article.asp?p=2202410 &seqNum= 4> . Acesso em 13/06/2016.

COMPUTER HISTORY MUSEUM. **Timeline of computer history,** 2016**.** Disponível em: <http://www.computerhistory.org/timeline/networking-the-web/> . Acesso em 13 jun. 2016.

EDITORA NOVA FRONTEIRA. **Mini Aurélio – O Minidicionário Da Lingua Portuguesa**, 4° Edição, 2000.

FEAMSTER, NICK; REXFORD, JENNIFER; ZEGURA ELLEN. **The Road to SDN: An Intellectual History of Programmable Networks.** Disponível em:

<https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf >. Acesso em 15/11/2016.

GOOGLE, INC. **Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network**, 2015**.** Disponível em: < http://conferences.sigcomm.org/sigcomm/2015/pdf/papers/p183.pdf>. Acesso em 13/06/2016.

GÖRANSSON, PAUL; BLACK, CHUCK; CULVER, TIMOTHY. **Software Defined Networks: A Comprehensive Approach**, 2° Edição, Elsevier, 2014.

HAMILTON, JAMES. **Networking: The Last Bastion of Mainframe Computing**, 2009**.** Disponível em: http://perspectives.mvdirona.com/2009/12/networking-the-last-bastion-of-mainframe-computing/>. Acesso no dia 13/06/2016.

HANRAHAN, BY HU. **Network Convergence: Services, Applications, Transport, and Operations Support**, Wiley, 2007.

HELD, GILBERT. **Server Management,** CRC Press, 2000.

ITU, **Internet Statitics**, Disponível em:<http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2015/stat_page_all_charts_2015.xls>. Acesso em 13/06/2016.

JONES, M. TIM. **Anatomy of an open source cloud,** 2012. Disponível em: < http://www.ibm.com/developerworks/library/os-cloud-anatomy/> Acesso em 10/12/2016.

J. SALIM. et al. **Linux Netlink as an IP Services Protocol,** 2003. Disponível em: <https://tools.ietf.org/html/rfc3549#section-1.1.1>. Acesso em 15/11/2016.

MCKEOWN, NICK. et al. **OpenFlow: Enabling Innovation in Campus Networks**, 2008. Disponível em: <http://ccr.sigcomm.org/online/files/p69-v38n2n-mckeown.pdf>. Acesso em 15/11/2016.

OPENDAYLIGHT.ORG. **OpenDaylight Controller: Architectural Framework.** Disponível em : <https://wiki.opendaylight.org/view/OpenDaylight_Controller :Architectural_Framework> Acesso em 17/11/2016.

OPENDAYLIGHT.ORG. **Platform Overview.** Disponível em : <https://www.opendaylight.org/platform-overview> Acesso em 10/12/2016.

OPEN NETWORKING FOUNDATION. **Software-Defined Networking:The New Norm for Networks**, 2012. Disponível em: <https://www.opennetworking.org/ images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf> Acesso em 10/12/2016.

OPEN NETWORKING FOUNDATION. **OpenFlow Switch Specification**, 2012. Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf> Acesso em 10/12/2016.

OPEN NETWORKING FOUNDATION. **What is ONF?,** 2013. Disponível em: <https://www.opennetworking.org/images/stories/downloads/about/onf-what-why.pdf>. Acesso em 15/11/2016.

PICA8. **Bare Metal Networking Leveraging "White Box' 'Thinking**, 2014. Disponível em: <http://www.pica8.com/documents/pica-whitepaper-white-box.pdf> Acesso em 15/11/2016.

PLUMMER, DAVID C. **An Ethernet Address Resolution Protocol,** 1982. Disponível em: <https://tools.ietf.org/html/rfc826>. Acesso em 15/11/2016.

RYU, **Ryu application programming model**. Disponível em: <http://ryu.readthedocs.io/en/latest/ryu_app_api.html>. Acesso em 15/11/2016.

SUMITS, ARIELLE. **The History and Future of Internet Traffic,** 2015 Disponiel em: <http://blogs.cisco.com/sp/the-history-and-future-of-internet-traffic>. Acesso em 13/06/2016.

TANENBAUM, ANDREW S. **Computer networks**, 3° Edição, 1996.

THE MCKEOWN GROUP. **About our group.** Disponível em <http://yuba.stanford.edu/group_wp/>. Acesso em 03/09/2016.

URS, BURG VON. **The thiumph of Ethernet: technological communities and the battle for the LAN standard,** 2002.

VAHDAT, AMIN. **Pulling Back the Curtain on Google's Network Infrastructure,** 2015**.** Diponivel em : <https://research.googleblog.com/2015/08/pulling-back-curtain-on-googles-network..html> . Acesso em 13/06/2016.

APENDICE A - Código fonte do programa Flow-manager.

```
#!/bin/bash
#################################################################
#Programa criado para demonstracao dos conceitos apresentados          #
#no sobre a programabilidade da rede                                   #
#                                                                      #
#Autor: Tiago Viana.                                                     #
#Versao: 0.1                                                           #
#################################################################

#Funcao list-flows - inicio.

        function list-flow {
        RANDOM1=$RANDOM
        printf "***LISTAGEM DE FLOWS***\n\n"
        read -p "Digite o nome do switch que deseja listar os flows e aperte [ENTER]:
" SWITCH
        /usr/bin/curl -u admin:admin -X GET -H "Content-Type: application/xml" -H
"Accept: application/xml" -H "Cache-Control: no-cache" \
        "http://10.0.0.10:8181/restconf/config/opendaylight-
inventory:nodes/node/$SWITCH/flow-node-inventory:table/0" 2> /dev/null\
        | sed -e 's/ xmlns="urn:opendaylight:flow:inventory"//g' > /tmp/$RANDOM1-
$SWITCH-flows.xml
        printf "\n \n%-30s %-10s %-20s %-20s %-10s %-10s %-30s\n" "ID"
"PRIORITY" "SRC-IP" "DST-IP" "SRC-PORT" "DST-PORT" "ACTION"
        for ID in $(xmlstarlet sel -T -t -v "table/flow/id" -n /tmp/$RANDOM1-$SWITCH-
flows.xml 2> /dev/null) ;
                do
                PRI=$(xmlstarlet sel -T -t -v "table/flow[id="'"$ID"'"]/priority" -n
/tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                SRCIP=$(xmlstarlet sel -T -t -v "table/flow[id="'"$ID"'"]/match/ipv4-
source" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
```

```
                DSTIP=$(xmlstarlet sel -T -t -v "table/flow[id="'$ID']/match/ipv4-
destination" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                SRCPORT=$(xmlstarlet sel -T -t -v "table/flow[id="'$ID']/match/tcp-
source-port" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                DSTPORT=$(xmlstarlet sel -T -t -v "table/flow[id="'$ID']/match/tcp-
destination-port" -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                for ORDER in $(xmlstarlet sel -T -t -v
"table/flow[id="'$ID']/instructions/instruction/apply-actions/action/order"\
                -n /tmp/$RANDOM1-$SWITCH-flows.xml 2> /dev/null)
                        do
                        ACTION=$(xmlstarlet sel -T -t -v
"table/flow[id="'$ID']/instructions/instruction/apply-
actions/action[order="'$ORDER']\
                        /output-action/output-node-connector" -n /tmp/$RANDOM1-
$SWITCH-flows.xml 2> /dev/null)
                        printf "%-30s %-10s %-20s %-20s %-10s %-10s %-30s\n" "$ID"
"$PRI" "$SRCIP" "$DSTIP" "$SRCPORT" "$DSTPORT" "$ACTION"
                        done
                done
        printf "\n \n%s" "Press enter to continue..."
        read
        }
#Funcao list flows - fim.

#Funcao add flows - inicio.
        function add-flow {
        printf "\n\nInsira os dados do flow\n"
        read -p "SWITCH: " SWITCH
        printf "\n"
        read -p "ID: " ID
        printf "\n"
        read -p "PRIORITY: " PRI
        printf "\n"
```

```
read -p "SRC-IP: " SRCIP
printf "\n"
read -p "DST-IP: " DSTIP
printf "\n"
read -p "SRC-PORT: " SRCPORT
printf "\n"
read -p "DST-PORT: " DSTPORT
printf "\n"
curl -u admin:admin -X PUT -H "Content-Type: application/xml" -H "Accept: application/xml" -H "Cache-Control: no-cache"\
-d '<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
        <priority>'$PRI'</priority>
        <flow-name>teste1401</flow-name>
        <match>
<ipv4-source>'$SRCIP'</ipv4-source>
        <ipv4-destination>'$DSTIP'</ipv4-destination>
        <ip-match>
                <ip-protocol>6</ip-protocol>
                <ip-dscp>2</ip-dscp>
                <ip-ecn>2</ip-ecn>
</ip-match>
<tcp-source-port>'$SRCPORT'</tcp-source-port>
<tcp-destination-port>'$DSTPORT'</tcp-destination-port>
<in-port>0</in-port>
        </match>
        <id>'$ID'</id>
        <table_id>0</table_id>
        <instructions>
                <instruction>
                        <order>0</order>
                                <apply-actions>
                                <action>
```

```
                                        <order>0</order>
                                        <dec-nw-ttl/>
                                    </action>
                                    </apply-actions>
                </instruction>
                </instructions>
        </flow>' "http://10.0.0.10:8181/restconf/config/opendaylight-
inventory:nodes/node/"$SWITCH"/table/0/flow/"$ID""

        if [ "$?" -eq "0" ]
                then
                printf "Flow adcionado com sucesso !!"
                else
                printf "Falha ao adcionar flow"
        fi
        printf "\n \n%s" "Press enter to continue..."
        read
        }
#Funcao add flows - fim.

#Funcao remove flow - inicio
        function del-flow {
        printf "\n***REMOCAO DE FLOW***\n\n"
        read -p "Informe o switch do qual o flow sera removido: " SWITCH
        printf "\n"
        read -p "ID: " ID
        curl -u admin:admin -X DELETE -H "Content-Type: application/xml" -H
"Accept: application/xml" -H "Cache-Control: no-cache"\
        "http://10.0.0.10:8181/restconf/config/opendaylight-
inventory:nodes/node/"$SWITCH"/table/0/flow/"$ID""
        if [ "$?" -eq "0" ]
            then
            printf "Flow removido com sucesso !!"
```

```
            else
            printf "Falha ao remover flow"
        fi
        printf "\n \n%s" "Press enter to continue..."
        read
          }
#Funcao remove flow - fim.

#Funcao quit - inicio
          function quit {
          printf "\n\n"
          exit
          }

#Funcao qui - fim

#Inicio Programa.
          while [ true ]
                    do
                    clear
                    OPTION=""
                    printf "MENU\n\n"
                    printf " 1 para listar flows\n"
                    printf " 2 para adcionar flow\n"
                    printf " 3 para remover flow\n"
                    printf " 4 para sair\n\n"
                    read -p " Digite numero da opcao: " OPTION
                    case "$OPTION" in
                    1)
                              list-flow
                              ;;
                    2)
                              add-flow
```

```
                ;;
        3)
                del-flow
                ;;
        4)
                quit
                ;;
        esac
        done
#Fim do Programa.
```