UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

BRUNO HENRIQUE SCHWENGBER

ANÁLISE DA RESILÊNCIA SOBRE TOPOLOGIAS DE REDES DEFINIDAS POR SOFTWARE EM ATAQUES IMINENTES

TRABALHO DE CONCLUSÃO DE CURSO

Santa Helena, Paraná

2018

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE CIÊNCIA DA COMPUTAÇÃO

BRUNO HENRIQUE SCHWENGBER

ANÁLISE DA RESILIÊNCIA SOBRE TOPOLOGIAS DE REDES DEFINIDAS POR SOFTWARE EM ATAQUES IMINENTES

Trabalho de Conclusão apresentado ao Curso de Ciência da Computação da Universidade Tecnológica Federal do Paraná, Câmpus Santa Helena, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Euclides Peres Farias Junior

Coorientadora: Prof. Drª. Michele Nogueira Lima

Santa Helena, Paraná

2018

TERMO DE APROVAÇÃO

"ANÁLISE DA RESILIÊNCIA SOBRE TOPOLOGIAS DE REDES DEFINIDAS POR SOFTWARE EM ATAQUES IMINENTES"

por

"Bruno Henrique Schwengber"

Este Trabalho de Conclusão de Curso foi apresentado às 13h30 do dia 03 de dezembro de 2018 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação, pela Universidade Tecnológica Federal do Paraná – Câmpus Santa Helena. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Me. Euclides Peres Farias Junior (Presidente - UTFPR/Santa Helena)	Profª. Drª. Giani Carla Ito (Avaliador 1 – UTFPR/Santa Helena)
Profª. Drª. Michele Nogueira Lima (Coorientadora – UFPR/Curitiba)	Prof. Dr. Joilson Alves Junior (Avaliador 2 – UTFPR/Curitiba)
Profª. Drª. Arlete Teresinha Beuren (Coordenador do curso de Bacharelado em Ciência da Computação – UTFPR/Santa Helena)	Prof. Dr. Franck Carlos Vélez Benito (Professor Responsável pelo TCC – UTFPR/Santa Helena)

A folha de aprovação assinada encontra-se na Coordenação do Curso

AGRADECIMENTOS

Aos meus pais pelo amor, incentivo e apoio incondicional.

Aos meus amigos pela compreensão e paciência nas conversas de fim de tarde.

Ao meu orientador e coorientadora, pela amizade, suporte, correções, apontamentos e incentivos para a produção e formação acadêmica.

A universidade e seu corpo docente, que oportunizaram a janela que hoje vislumbro na carreira acadêmica.

E a todos que direta ou indiretamente fizeram parte da minha formação, meu muito obrigado.

RESUMO

SCHWENGBER, Bruno Henrique. **Análise da Resiliência Sobre Topologias de Redes Definidas por Software em Ataques Iminentes**. 2018. 24f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Tecnológica Federal do Paraná. Santa Helena.

Este trabalho apresenta a averiguação da resiliência de topologias em redes definidas por software em ataques iminentes. Foram avaliadas três topologias, a anel, a árvore e a torus, pois apresentam correlação aos ambientes utilizados em redes clássicas. Para obtenção dos dados foram construídos cenários emulados usando a ferramenta Mininet, cada cenário foi repetido 31 vezes para a captura de dados, os ataques foram gerados utilizando a ferramenta de ataque de negação de serviços T50 e a coleta de dados foi realizada utilizando a ferramenta de captura de pacotes Tcpdump e *Shell Scripts* para calcular os tempos entre cada evento definidos como início do experimento, morte da topologia e retorno a funcionamento da topologia. Os dados obtidos foram analisados utilizando estatística descritiva e análise de sobrevivência utilizando R para execução das análises. Por fim, observou-se que a topologia torus oferece maior resiliência, seguida pela anel e árvore, respectivamente, entre os cenários a principal diferença que pode ser observada são as rotas redundantes que são apresentadas nas topologias em anel e torus, onde a topologia em anel apresenta duas rotas entre cada switch e a torus apresenta quatro rotas entre cada switch de comunicação.

Palavras-chave: Redes definidas por *software*. Topologia. Resiliência.

ABSTRACT

SCHWENGBER, Bruno Henrique. **Analysis of Resilience of Topologies in Software Defined Networs in Imminent Attacks**. 2018. 57p. Work of Conclusion Course (Graduation in Computer Science) – Federal Technology University – Paraná. Santa Helena.

This work presents the research of the resilience of topologies in software defined networks in imminent attacks. It was evaluate three topologies, ring, tree and torus, because they were corelate for classical network environments. For geting the data were built emulated scenarios using the tool Mininet, each scenario were performed 31 times for the date capture, the atacks were generate using T50 denial of service tool, the data collection were made using Tcpdump, a packet analyzer and Shell Scripts to calculate the time between each event defined as the start of the experiment, the time death of the topology and the time of return to operation of the topology. The data capturaded were analyze using descriptive statistics and survival analysis using R to perform the analyzes. Finally, it was observed that the torus topology offers greater resilience, followed by topology ring and tree, between scenarios the main difference that can be reached is the redundant routes that topology torus and ring presents, this way the topology ring presents two routes in each switch and the torus topology presents four routes in each switch of comunication.

Keywords: Software defined network. Topology. Resilience.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

CPU: Central Process Unit

DDoS: Distributed Denial of Service

DoS: Denial of Service

HTTP: HyperText Transfer Protocol

ICMP: Internet Control Message Protocol

IDS: Intrusion Detection System

IPS: Intrusion Prevent System

IP: Internet Protocol

LLDP: Link Layer Descobert Protocol

NOS: Network Operational System

OFDP: OpenFlow Descobert Protocol

SDN: Software Defined Network

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

VM: Virtual Machine

SUMÁRIO

1 INTRODUÇÃO

Com a evolução tecnológica, foram alcançadas diversas melhorias para o conforto da sociedade. Uma das evoluções notáveis foi a Internet, que permite realizar as conexões de dois pontos distintos por meio de cabos ou ondas de rádio. Dada esta comunicação deve-se levar em conta a segurança dos dados que trafegam na rede, por este motivo os princípios básicos da segurança estão sendo tratados com mais cautela, a fim de manter os dados seguros para trafegarem nas redes.

A resiliência de uma rede computacional é definida pela capacidade de passar por momentos de perturbação, superar e se recuperar. Esta é uma das características fundamentais para apresentar uma rede em que os dados possam trafegar de forma estável.

Em vista da garantia mínima de segurança, para que seja possível determinar as necessidades de segurança de conteúdo, é preciso determinar os objetivos de preservação e perfil dos riscos, por meio destas informações, há a possibilidade de decidir sobre os controles de gerenciamento, controles operacionais e controles técnicos, de forma que seja o suficiente para reduzir os riscos a um nível aceitável.

Além das novas tecnologias estarem evoluindo na área de telecomunicações, são criados também novos modelos e conceitos, como é o caso das redes definidas por software (SDN), que visa possibilitar a programação dos dispositivos. Na utilização de SDN são definidas topologias para criação das redes, a partir destas considerações se viu a oportunidade de explorar o ambiente no sentido de analisar qual topologia de rede possui mais resiliência em ataques iminentes do tipo Negação de Serviço (DoS) ou Negação de Serviços Distribuídos (DDoS).

Pretende-se então explorar a ferramenta de ataque, T50 em cenários construídos com topologias especificas para a extração de relatórios de incidência. Partindo da hipótese de que as diferentes topologias apresentam diferentes resiliências sobre os ataques DoS e DDoS, tem-se então a busca de um trabalho científico para apresentar os conceitos e definições para a análise de resiliência de topologias na utilização de redes definidas por *software* quando expostas a ataques de natureza de negação de serviços.

1.1 OBJETIVOS

Expõem-se a seguir o objetivo geral e específicos que se pretende atingir com o trabalho.

1.1.1 Geral

Averiguar a resiliência de topologias de redes definidas por software em um ataque iminente de negação de serviços de forma distribuída.

1.1.2 Específicos

1) Construir as topologias anel, árvore e torus para estudo de resiliência;
2) Aplicar método de ataque DDoS;
3) Configurar e manipular o protocolo de descoberta de *links* e camadas (LLDP);
4) Configurar e manipular o protocolo *spanning tree*;
5) Criar *scripts* para os ataques e para coleta de dados;
6) Realizar testes de ataques nos cenários construídos;
7) Aplicar métricas estatísticas para análise dos resultados;
8) Analisar os relatórios dos ataques, utilizando métodos estatísticos.

1.2 JUSTIFICATIVA

A discussão sobre os impactos de ataques DoS ou DDoS em topologias de redes definidas por *software,* além do aspecto prático relevante, possui importância para o meio acadêmico. Nesse contexto, a principal produção de conteúdo se dá na apresentação de algumas topologias de forma a avaliar a resiliência sobre ataques iminentes. Desta forma, pelo contexto a ser apresentado em redes e segurança computacional, este trabalho tem pertinência para o curso de Ciência da Computação e para a área de teleinformática, uma vez que são assuntos correlatos.

1.3 DELIMITAÇÕES DO TRABALHO

Este trabalho tem como objetivo em aplicar técnicas de construção de topologias em redes, bem como a usabilidade de ferramentas de ataques DDoS com o objetivo da verificação da resiliência das topologias, árvore, anel e torus. Para que o experimento tenha o resultado mais próximo do real foram utilizadas redes definidas por *software*. Desta forma, não faz parte deste estudo, a contenção e prevenção de ataques no âmbito de segurança computacional, mas a análise por meio de técnicas estatísticas dos resultados obtidos em ataques iminentes e o quão resiliente é uma infraestrutura de rede de acordo com sua topologia.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em cinco (5) sessões. A primeira realiza a introdução do texto, a segunda apresenta a revisão da literatura para a construção do trabalho, a terceira apresenta a metodologia utilizada para alcançar os resultados, a quarta apresenta as análises dos resultados obtidos e a quinta sessão apresenta a conclusão do trabalho.

2 REVISÃO DA LITERATURA

Expõem-se a seguir a revisão da literatura que abrange os conteúdos necessários para a conclusão satisfatória desde trabalho.

2.1 ESTADO DA ARTE

Redes definidas por *software* é um dos paradigmas de redes de computadores, que apresentam a possibilidade de programar os dispositivos, dessa forma quebrando o paradigma clássico onde o plano de controle é acoplado aos dispositivos (KREUTZ et al, 2015). De acordo com Menezes et al. (2018), os trabalhos envolvendo SDN são categorizados em: desenvolvimento de SDN, segurança em SDN, SDN aplicado a segurança e performance em SDN.

Segundo Montibeler, Farias e Abelém (2017), os trabalhos relacionados a SDNs, comumente, utilizam somente uma métrica de avaliação para obtenção da resiliência da rede, e mesmo quando utilizam múltiplas métricas as otimizações visam a redundância e posicionamento dos controladores disponíveis na rede, sem realizar a verificação topológica da comunicação dos dispositivos da rede.

Dessa forma tem-se as propostas de Heller, Sherwood e McKeown (2012); Ros e Ruiz (2014); Müller (2014); Hock (2013) e Lange (2015) que apresentam argumentos para o aumento de resiliência utilizando os controladores de SDN como principal métrica de avaliação.

Nos trabalhos de Ros e Ruiz (2014) e Müller (2014), são levadas em consideração apenas uma métrica avaliativa para a determinação da resiliência da rede, em Ros e Ruiz (2014), é investigado qual o impacto do posicionamento dos controladores na capacidade de sobrevivência da SDN, realizando testes para definir a quantidade de controladores em uma topologia de forma a garantir um valor mínimo de confiabilidade da rede. Em Müller (2014), é proposta uma metodologia de balanceamento de cargas para os controladores para otimizar a resiliência tendo em vista os caminhos disjuntos entre nós.

Nos trabalhos de Hock (2013) e Lange (2015), são feitas propostas com múltiplas métricas, como latência entre dispositivos, balanceamento de carga de controladores, conectividade entre os planos da SDN, capacidade de sobrevivência, e a latência entre os dispositivos. Todas as métricas são avaliadas com o objetivo de induzir aumento significativo na resiliência da rede, porém, ainda se mantém somente na otimização do plano de controle da SDN.

2.2 REDES CLÁSSICAS

Uma rede de computadores é definida como a conexão entre dois ou mais dispositivos que permitem o compartilhamento de recursos e envio de dados entre os ativos conectados. Os dispositivos finais são conectados entre si por enlaces, que são links de comunicação, e por comutadores de pacotes, que roteiam e disseminam um link de comunicação. Há muitos tipos de enlaces de comunicação, eles são constituídos de diferentes tipos de meios físicos, podem ser de cabos coaxiais, fios de cobre, fibras óticas e ondas de rádio. Cada um dos tipos de enlace transmite os dados em taxas diferentes, a taxa de transmissão de um enlace é medida em bits por segundo (KUROSE; ROSS, 2014).

O encaminhamento dos pacotes pode ser feito por roteadores ou comutadores da camada de enlace, esses dois tipos de dispositivos redirecionam os pacotes para os destinos finais. Os comutadores da camada de enlace geralmente são utilizados em redes de acesso, enquanto os roteadores têm como função criar as rotas a partir do núcleo da rede. As redes comutadas por pacotes são muito semelhantes às redes de transportes rodoviário, pois como um produto precisa ser transportado a partir da fábrica para os clientes, os pacotes de rede também transitam nas redes a fim de serem entregues ao seu destino (KUROSE; ROSS, 2014).

Para ter acesso à Internet são necessários provedores de serviço de Internet, como exemplo de provedores temos, empresas de TV a cabo, empresas de telefonia ou até empresas específicas do ramo de provedores de Internet. Cada provedor é uma rede de dispositivos, que tem comutadores e enlaces de comunicação (KUROSE; ROSS, 2014).

Contudo, temos os protocolos de comunicação, o Internet Protocol (IP) e o protocolo de controle e transmissão (TCP) que são os dois protocolos mais importantes da Internet.

Kurose e Ross (2014) definem os protocolos como, “o IP especifica o formato dos pacotes que são enviados e recebidos entre roteadores e sistemas finais; o TCP controla o envio e o recebimento de informações”.

2.2.1 Topologias de Redes Clássicas

A topologia de uma rede apresenta o modo de conexão entre os dispositivos, a forma de processamento e a troca de informações. As topologias têm como função garantir a redução dos custos e aumentar a eficiência do sistema por meio da combinação de recursos (MENDES, 2015).

Segundo Mendes (2015), as principais topologias de redes clássicas são a topologia estrela, topologia linear e a topologia anel. Ainda, segundo Gomes (2012), temos a topologia híbrida que é a mais utilizada em redes grandes.

A topologia estrela é caracterizada por um dispositivo central que controla o fluxo dos dados da rede, estando conectado diretamente com todos os dispositivos, assim formando um desenho de estrela, na Figura 1 temos o exemplo da configuração da topologia. Todas as informações que transitam na rede devem passar pelo controlador central, assim o processo se torna eficaz, pois os dados passarão por todos os dispositivos finais (MENDES, 2015). Ainda a manutenção da rede é facilitada, pois cada ativo possui uma conexão direta com o controlador central. A topologia em estrela, pode ser representada graficamente em formato de estrela, no entanto também pode ser considerada uma topologia de barramento (TORRES, 2015).

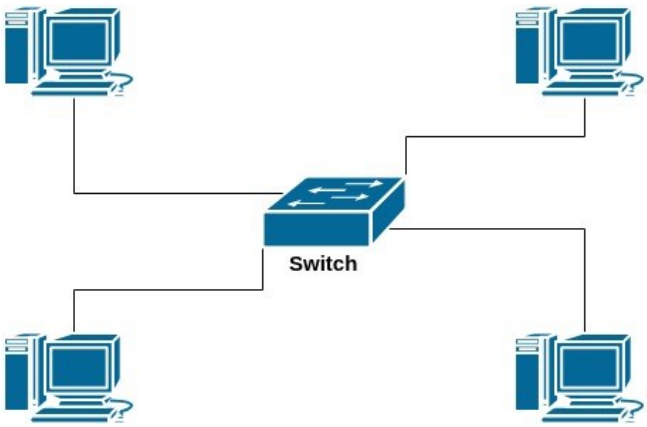

Figura 1– Topologia Estrela.

Fonte: Adaptado de Mendes (2015).

A topologia linear é simples e de fácil implementação. As redes que utilizam esse padrão possuem todos os computadores interligados por meio de um cabo contínuo como pode ser visto na figura 2, desse modo, os dados enviados serão entregues a todos os computadores até que chegue no destino. Essa topologia é recomendada para redes simples, devido ao barramento único ser limitado (TORRES, 2015).

Figura 2 – Topologia Linear.

Fonte: Adaptado de Mendes (2015).

Na topologia em anel cada dispositivo possui dois cabos, um conectado ao dispositivo anterior e o outro ao próximo, a figura 3 exemplifica um cenário com quatro computadores conectados entre si em uma topologia anel. Para que um computador possa se comunicar com outro ele deve percorrer o caminho entre todos os outros (TORRES, 2015). Essa topologia como o nome representa, é um circuito lógico fechado, centrado com um ativo de roteamento que forma o anel de comunicação e tem como vantagem a ausência de atenuação, já que o sinal é regenerado a cada computador que passa (MENDES, 2015).

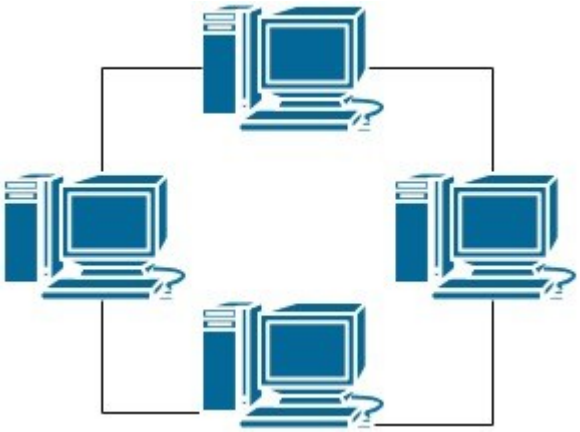

Figura 3. Topologia Anel.

Fonte: Adaptado de Mendes (2015).

A topologia híbrida é a mais utilizada em grandes redes, com ela é possível adequar a topologia conforme o ambiente, incrementando melhoria de custos, expansibilidade, flexibilidade e funcionalidade de cada segmento da rede. Em topologias hibridas seu esquema gráfico é a combinação de duas ou mais topologias de rede, esse grupamento de topologias

aumenta as vantagens de cada uma das topologias que integram esta topologia, a integração de topologias pode ser vista na figura 4 onde é apresentado um cenário com topologias anel e estrela interligadas (GOMES, 2012).

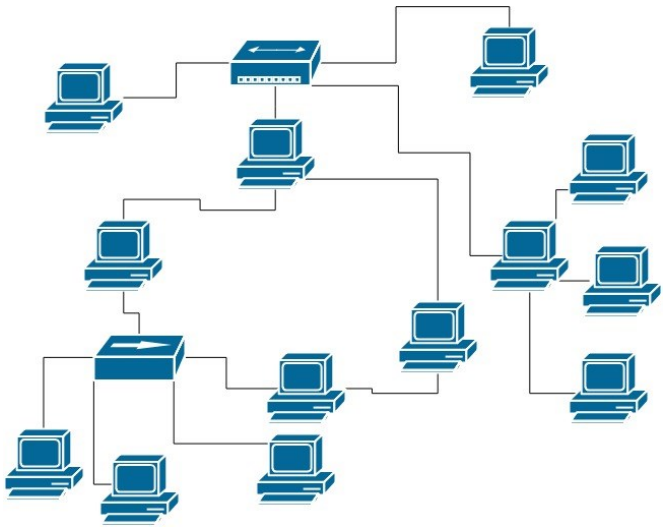

Figura 4. Topologia Híbrida.

Fonte: Adaptado de Gomes (2012).

2.3 REDES DEFINIDAS POR SOFTWARE

As SDNs foram originadas na definição da arquitetura de redes *Ethane*, que definia uma forma de implementação de políticas de controle de acesso de forma distribuída, partindo de um controlar centralizado. Na arquitetura *Ethane*, cada dispositivo deveria consultar o controlador para identificação do novo fluxo de dados, assim o controlador com base nas políticas globais de encaminhamento decidia sobre como o dispositivo deveria tratar o dado. Desde o surgimento, diversos pesquisadores de redes de computadores e empresas voltadas ao desenvolvimento de equipamentos e sistemas para gerência de redes têm voltado sua atenção para o paradigma de SDN. Isso é resultado da alta procura de encontros e congressos na área (GUEDES et al., 2012).

Na figura 5 é possível observar as diferenças entre uma rede clássica e uma rede definida por *software*, onde na rede clássica tem-se os controladores acoplados a todos os dispositivos configurados no ambiente e na rede definida por *software* tem-se apenas um controlador que gerencia as comunicações com as aplicações que utilizam os elementos configurados no ambiente da rede.

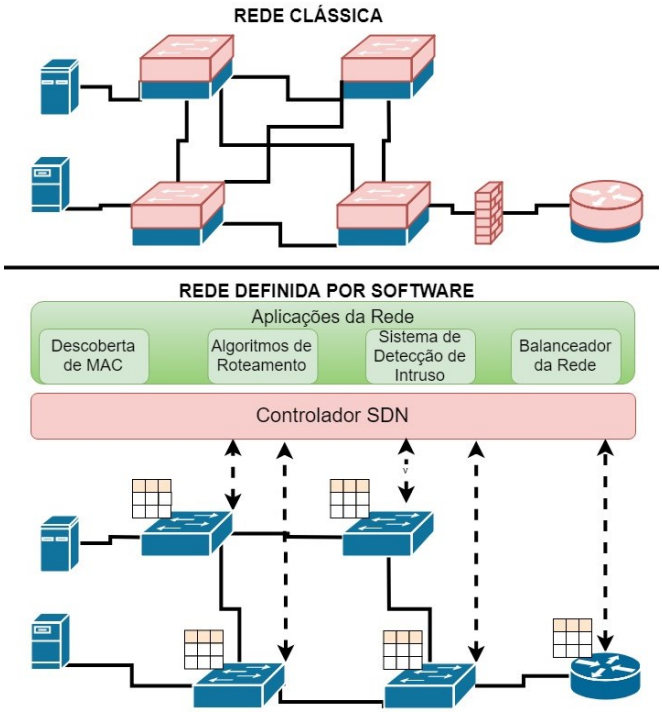

Figura 5. Comparação entre uma rede clássica e uma SDN.

Fonte: Adaptado de GUEDES et al. (2012).

O conceito de redes definidas por software foi inicialmente criado para representar novas ideias e contornar a limitação da programação do protocolo OpenFlow, que de acordo com Nunes et al. (2014), trata-se de um protocolo de comunicação, que controla e gerência o encaminhamento dos pacotes na orientação por fluxo das SDNs, uma vez que este protocolo atua como uma camada intermediadora entre o controle e a infraestrutura do ambiente. SDN refere-se a uma arquitetura de rede, onde o *status* de encaminhamento dos pacotes no plano de dados, é gerenciado por um controlador desacoplado da camada anterior (KREUTZ et al.2015).

2.3.1 Arquitetura de Redes Definidas por Software

No contexto de redes simuladas, as SDNs possuem uma arquitetura, que incorporam as decisões de encaminhamento de pacotes nos dispositivos da rede, assim, são feitas as decisões pelo dispositivo denominado controlador SDN. O controlador SDN é o dispositivo que contém os algoritmos programados para realizar os encaminhamentos nas direções corretas e realizar a orientação pelo fluxo. Além disso, para que o controlador realize suas

tarefas de forma ótima, se faz necessário a utilização de protocolos de comunicação, comumente, o protocolo utilizado é o OpenFlow (NUNES et al. 2014).

A interface de programação do protocolo OpenFlow é simples, esta permite o controle da tabela de encaminhamentos que deve ser utilizada pelos dispositivos da rede. Inicialmente esse protocolo foi proposto na Universidade de Stanford, com o objetivo de atender à demanda de validação de novos modelos de arquitetura e protocolos de rede que agissem em equipamentos comerciais (MCKEOWN et al. 2008).

O OpenFlow possui um importante papel na arquitetura de SDN, pois atua como uma interface entre as camadas de controle e infraestrutura, bem como faz a definição das regras de encaminhamento, portanto, com essas características é possibilitada a criação das SDNs por meio de implementações realizadas pelo OpenFlow (KREUTZ et al. 2015).

As redes definidas por software, de acordo com Kreutz et al (2015), podem ser definidas em quatro pilares que são o plano de controle e dados são desacoplados, as decisões de encaminhamentos são baseadas no fluxo de pacotes, o controle lógico é feito em uma entidade externa e a rede é programável. A seguir serão discutidos estes tópicos.

- Os planos de controle e dados são desacoplados: A funcionalidade de controle foi removida dos dispositivos de rede que se tornam simples equipamentos de encaminhamento de pacotes.
- As decisões de encaminhamentos são baseadas no fluxo de pacotes: Um fluxo é definido por um conjunto de campos valorados atuando como um critério de compatibilidade em um conjunto de instruções. No contexto SDN, um fluxo é uma sequência de pacotes entre uma origem e um destino. Todos os pacotes de um fluxo recebem políticas de serviços iguais nos dispositivos de encaminhamento. A abstração do fluxo permite unificar o comportamento de diferentes tipos de dispositivos da rede, incluindo roteadores, switches e firewalls. A programação do fluxo de encaminhamento de pacotes, permite uma grande flexibilidade que é limitada apenas pelas capacidades das tabelas de fluxo de pacotes implementadas.
- O controle lógico é movido para uma entidade externa: denominada controlador SDN ou sistema operacional da rede (NOS). NOS é uma plataforma de software que executa um servidor primário de tecnologia e fornece os recursos essenciais e abstrações para facilitar a programação dos dispositivos da SDN. A sua finalidade pode ser comparada a um sistema operacional tradicional.

A rede é programável por meio de aplicações de software executados no NOS que interagem com os dispositivos nos planos de dados subjacentes. Essa é a característica fundamental da SDN.

Contudo, o princípio básico das redes definidas por software é a viabilidade de programar os dispositivos da rede. Essa programação se restringe a manipulação básica de pacotes utilizando como base o conceito de fluxos, que são os conjuntos de pacotes que compartilham atributos com valores. Dessa forma, um fluxo é uma função que compreende os recursos oferecidos pela interface de programação (GUEDES et al., 2012).

A arquitetura das SDN contempla um sistema de controle, que tem como objetivo controlar o mecanismo de encaminhamento dos elementos de roteamento da rede, utilizando uma interface de programação, dessa forma permite o controlador averiguar, definir e modificar as entradas da tabela de roteamento por meio do OpenFlow (GUEDES et al., 2012). Ainda, o OpenFlow atua como controlador SDN que é a interface entre as camadas de controle e infraestrutura como pode ser visto na Figura 6, também dita as regras de encaminhamento. Com isso possibilita a criação de SDNs com implementações arquitetadas pelo OpenFlow (KREUTZ et al., 2013).

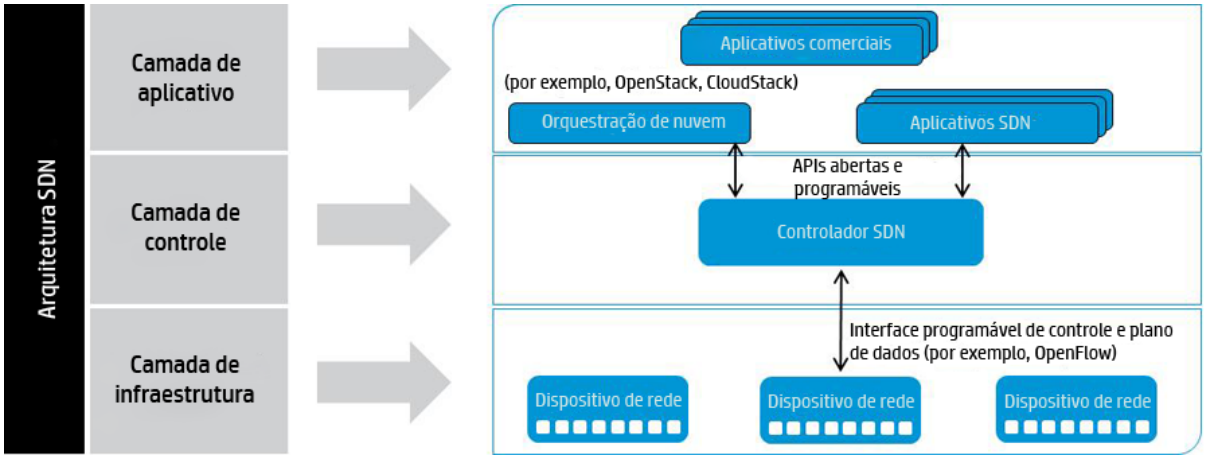

Figura 6. Arquitetura SDN.

Fonte: JAMMAL et al. (2014).

Na Figura 7 pode se observar a segmentação dos limites de operabilidade dos componentes das SDNs. No limite Norte existe a implementação de interface que opera entre as aplicações e os controladores, pois apresenta uma visão abstrata da rede, além de receber informações do comportamento da rede. Por outro lado, o limite Sul é a interface definida entre um controlador e a infraestrutura de rede, ou seja, é um ativo da SDN, que pode ser

inserido via programação OpenFlow. Assim, tem-se que as interfaces Norte e Sul são implementações abertas e independentes (SEZER et al., 2013).

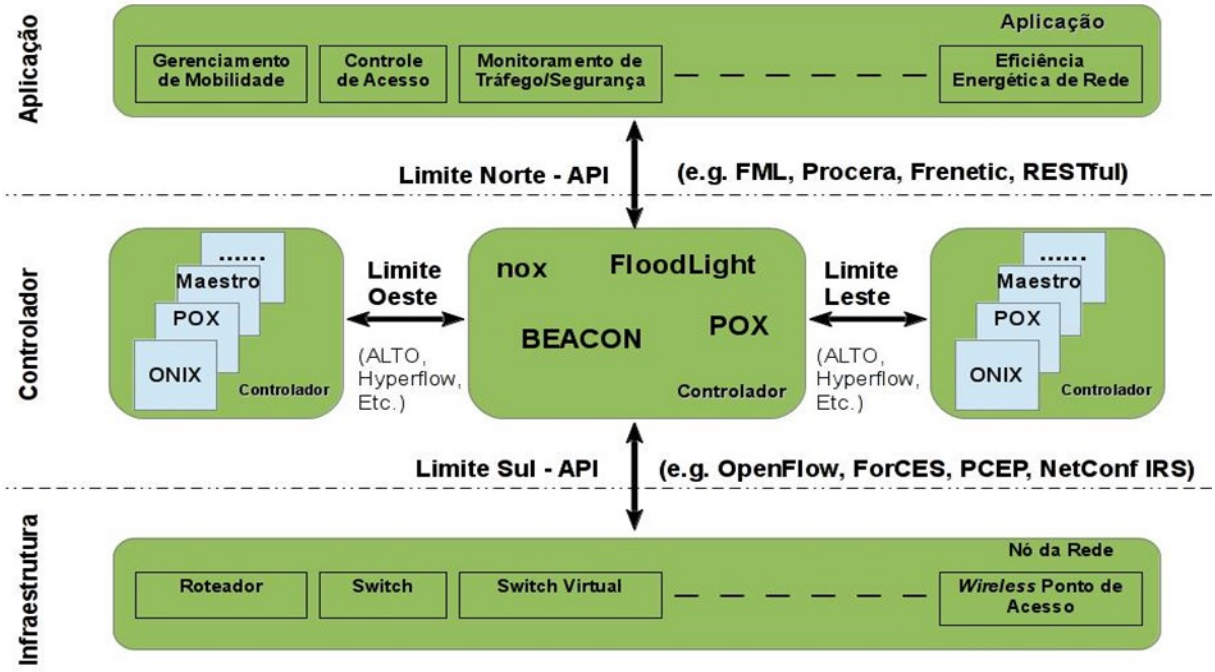

Figura 7. Arquitetura funcional da SDN ilustrando a infraestrutura, controle e outros elementos de aplicação que compõe este tipo de rede.
Fonte: Adaptado de Sezer et al. (2013).

A arquitetura funcional da SDN elaborado por Sezer et al. (2013), representada acima na Figura 7, que apresenta a ilustração sobre a infraestrutura de controle e outros elementos de aplicação que compõe as redes SDN, e assim Lins (2015), apresenta de forma sucinta a definição desta infraestrutura, como segue:

- **Controladores SDN**: trata-se do principal componente de uma rede SDN, onde também é chamado de Sistema Operacional da Rede, este foi o ambiente idealizador que definiu a natureza do paradigma SDN, que por sua vez, é o responsável por concentrar a comunicação com todos os elementos programáveis da rede, oferecendo uma visão unificada da rede. Entretanto, existem diversos tipos de controladores aplicáveis para as redes SDN;
- **Aplicações**: Na arquitetura SDN, as aplicações passam a ser ativas nas redes, função esta que a difere das redes convencionais, uma vez que a rede toma ciência da aplicação que está sendo executada. Coisa que em redes convencionais, as aplicações devem descrever indiretamente seus requisitos para o funcionamento adequado, o que

normalmente envolve diversas etapas e processamentos para negociar recursos que estejam disponíveis, bem como definir uma política de controle de forma a atender o aplicativo em execução. Em SDN as aplicações podem monitorar o estado da rede de forma a adaptar-se conforme sua estrutura e o plano de controle é de forma lógica e centralizada, onde o controlador SDN efetua um resumo do estado da rede, de forma a comunicar os aplicativos, bem como faz a tradução dos requisitos das aplicações em execução;

- **Limites Norte e Sul**: Trata-se de interfaces, onde o limite Norte é uma interface entre as aplicações e os controladores, pois tem a característica de oferecer uma visão abstrata da rede, bem como recebe as informações diretamente do comportamento e requisitos da rede. Já a interface do limite Sul, é a interface definida entre um controlador e a infraestrutura, ou seja, nada mais é que um dispositivo SDN, pois através de uma linguagem como por exemplo OpenFlow, onde a API tem o objetivo de fornecer o controle programático de todas as operações, capacidades de notificação, estatísticas e relatórios são disponibilizadas e construídas nesta interface. De forma análoga as interfaces Norte e Sul são implementações construídas de forma aberta, independentes de fornecedores e com interoperabilidade;
- **Dispositivos programáveis SDN**: trata-se de dispositivos que normalmente são switches e roteadores que recebem a informação do controlador através da interface do limite Sul para montar seu plano de dados. Onde após o recebimento das informações do controlador, as decisões já podem ser tomadas do que será feito com os pacotes. Além dos dispositivos suportados pelos ambientes livres GPL, existem também muitos dispositivos comerciais que suportam o padrão OpenFlow, como por exemplo HP, NEC dentre outros. Esta é uma evolução eminente do SDN, pois tenderá mais fabricantes adotarem este padrão.

2.3.2 Topologias de Redes Definidas por Software

As topologias de SDN são configuráveis, quem controla e cria é o próprio usuário. Utilizando uma máquina virtual de SDN, por exemplo o emulador Mininet, pode-se criar diversas topologias, desde aglomerados de dispositivos simples até conjuntos de *switches* formando topologias muito complexas (BOMBAL, 2018).

Ainda segundo Bombal (2018), segue alguns exemplos de topologias possíveis de serem criadas em SDN. Em relação as topologias de redes clássicas, a única diferença é que as SDNs são simuladas em ambiente virtual, mas de certa forma imitam os modelos tradicionais com a opção de criar topologias mais complexas e utilizando mais *links* entre os dispositivos.

Topologia linear, consiste em um conjunto de *switches* ou qualquer tipo de ativo de rede conectados entre si, como pode ser visto na Figura 2.

Topologia estrela, é apresentada em um esquema onde é utilizado somente um *switch* e diversos *hosts* conectados, conforme a Figura 1.

Topologia em árvore, apresenta um *switch core* que se conecta a vários outros *switches* que farão a conexão com os *hosts*, como pode ser visto na Figura 8.

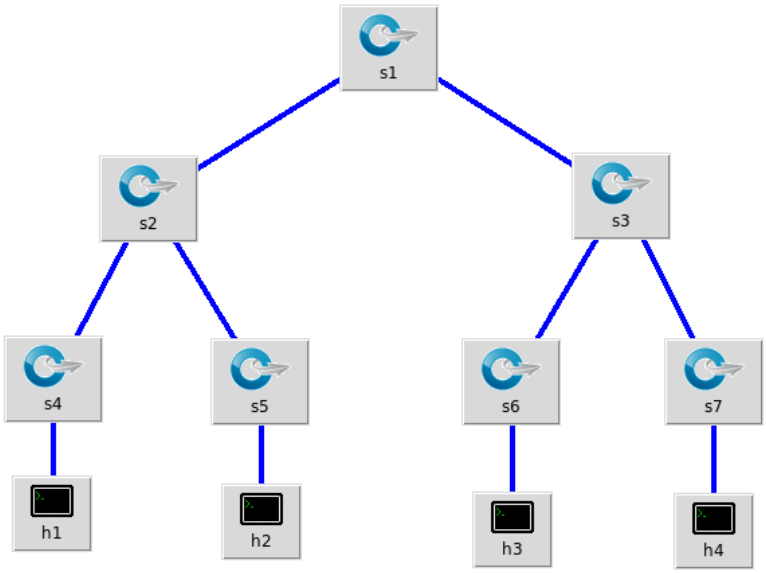

Figura 8. Topologia em árvore.
Fonte: Autoria Própria (2018).

A topologia t*orus*, utiliza diversos *switches* para comunicação, a conexão dos switches é quase completa, dessa forma são criados vários *links* de comunicação entre *switches*, para obtenção de mais rotas de redundância e aumento da resiliência da rede, a Figura 9 apresenta o desenho de uma topologia Torus.

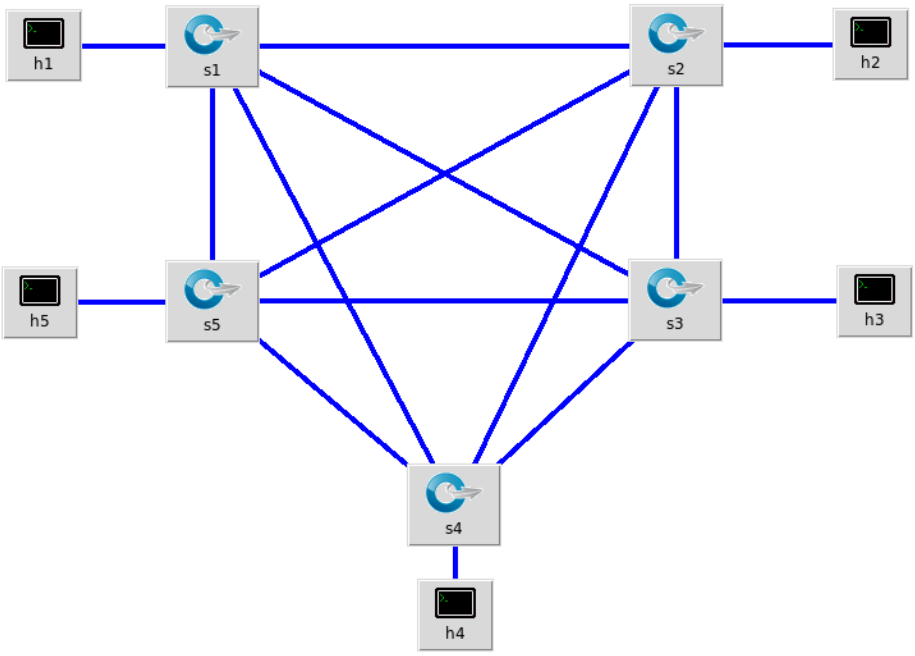

Figura 9. Topologia t*orus*.

Fonte: Autoria Própria (2018).

Estas topologias representam a capacidade e flexibilidade que um ambiente SDN proporciona aos recursos de redes, e desta forma, vem se consolidado cada vez mais no meio acadêmico cujas pesquisas estão presentes em diversas Universidades no mundo todo, e no meio corporativo, é uma realidade de aplicação, cujas maiores usuárias e disseminadoras desta tecnologia, atualmente são as empresas como a Google, Cisco, HP, Juniper dentre outras.

2.3.3 Aplicações de SDN

O conceito SDN está em pauta das industrias de tecnologia e as projeções do mercado giram na casa de bilhões. Especialistas destacam que, levando em conta o crescimento a nuvem, a SDN é o caminho adequado para encaminhar novas aplicações. Estimado pela IHS Technology, provedora de pesquisas comerciais, as receitas globais envolvendo SDN devem atingir 11 bilhões de dólares até 2019 (JUNIPER, 2015).

Segundo Melo (2018), a SDN propõe uma arquitetura dinâmica, gerenciável, adaptável e com custo-benefício adequado, ainda seu desemprenho é alinhado com as principais tendências do mercado:

- Nuvem Híbrida que são sistemas distribuídos geograficamente por meio de nuvens públicas e privadas que demandam gerenciamento de tráfego extremamente flexível e acesso à largura de banda sob demanda;
- Consumerização de tecnologia da informação que é a tendência de que cada usuário tenha seu próprio dispositivos de acesso e armazenagem de dados também requer redes flexíveis e seguras;
- *Data center,* pois, com a gigantesca quantidade de dados a largura de banda necessária se torna maior para a conectividade e estabilidade dos sistemas de *data center.*

2.4 EMULADOR DE SDN MININET

Mininet é uma ferramenta de emulação que permite executar inúmeros *hosts* virtuais, controladores, switches e conexões. Utiliza a virtualização baseada em contêineres para criar um único sistema que funcione como uma rede completa. É um instrumento muito simples e robusto para desenvolver e testar aplicações fundamentadas em OpenFlow. A máquina virtual Mininet pode criar complexas topologias de rede para testes, sem configurar redes físicas. Ainda, suporta topologias customizadas e programação Python (KAUR; SINGH; GHUMMAN, 2014).

O código que é desenvolvido para Mininet, também pode ser executado em redes reais sem quaisquer modificações, suportando até redes de grande escala. Em suma, todos os dispositivos gerados para o Mininet são como dispositivos reais, a diferença é que são softwares e não hardwares (MININET TEAM, 2018).

De acordo com o mesmo autor a arquitetura completa de contêineres do Linux adiciona cabines isolantes do sistema raiz, unidade central de processamento (CPU) e limites de memória para garantir completa virtualização no nível do sistema operacional. Mininet pode criar kernel e espaço de usuários, OpenFlow *switches* e *hosts* para comunicar na rede simulada. Mininet conecta os ativos da rede utilizando pares de conexões simulando placas de redes físicas. O Mininet depende do kernel do Linux, porém, no futuro poderá suportar outros sistemas operacionais com virtualização baseada em processos (MININET TEAM, 2018).

2.5 SEGURANÇA EM REDES

A partir da criação da Internet datada no final dos anos 60, uma das maiores preocupações nessa área era a conectividade e a segurança dos dados. Atualmente com a rápida evolução e expansão do uso da tecnologia em redes e compartilhamento da informação, bem como sua compatibilidade com qualquer ambiente, neste cenário a preocupação na área de segurança aumentou significativamente, em especial os princípios básicos que são a disponibilidade, confidencialidade e integridade dos dados aumentou significativamente. Essa preocupação com a segurança está diretamente relacionada ao fato de que, atualmente, a informação possui demasiada importância nos negócios para as organizações. Dado que o conhecimento é gerado a partir dos dados que trafegam na rede, assim se tornou de extrema importância para as organizações determinar meios para a proteção dos sistemas computacionais contra as ações não autorizadas, que tem como foco a obtenção da inteligência empresarial (MARTIMIANO, 2006).

De acordo com Stallings e Brown (2014) p. 437:

> As necessidades de proteção podem ser determinadas a partir das seguintes três perguntas: Quais ativos devo proteger? Como esses ativos são ameaçados? O que podemos fazer para contrapor essas ameaças? Para responder a essas perguntas formalmente utilizamos das técnicas de gerenciamento de segurança de Tecnologia da Informação (TI).

Contudo, deve-se determinar uma visão clara dos objetivos da segurança e o perfil dos riscos gerais da organização. A seguir, é preciso uma avaliação dos riscos para cada ativo corporativo que exija proteção. Para garantir a proficiência da proteção é necessário que a avaliação responda as três perguntas da lista apresentada. Com essas informações deverá ser possível decidir quais controles de gerenciamento, operacionais e técnicos são necessários para reduzir os riscos identificados a um nível aceitável (STALINGS; BROWN, 2014).

Martimiano (2006), afirma que é indispensável para o estabelecimento da proteção, apontar as possíveis ameaças e riscos que uma organização pode sofrer. Desta forma, é dever da organização definir as condições de segurança para ter com clareza o que pode e o que não pode ser permitido e gerenciado no ponto de vista da segurança. A partir desse momento, com uma política de segurança estipulada, pode-se mensurar o grau de segurança se está a um nível desejável.

Como definição, a política de segurança deve determinar como cada parte do sistema deve operar e deve ser aplicada, além disso deve descrever os deveres e direitos de cada componente que manuseia o sistema e como os ativos devem ser protegidos. Ainda com a política de segurança, é possível arquitetar as ações a serem tomadas caso ocorram problemas de segurança, bem como apontar quais os métodos para aplicação da garantia de segurança acordado nas políticas. Assim, o estado de confiança do sistema, deve ser extraído com base na comparação entre a forma de como reage e a forma específica de como ele deveria reagir no ponto de vista da segurança (KUROSE; ROSS, 2014).

Para compor um sistema de segurança eficiente, é necessário integrar técnicas de segurança física e lógica. Como método de segurança física pode-se utilizar dispositivos de detecção corpórea como: sensores e alarmes, ainda existem diversas ferramentas de prevenção como: travas e barreiras físicas. Como forma de prover efetividade da segurança física, existem diversos grupamentos de medidas para unir sistemas, dispositivos eletrônicos e ambientes computadorizados (KUROSE; ROSS, 2014).

A implementação da segurança lógica, tem seu princípio dado pelo controle de acesso, assim, utilizando um único cartão de identificação para acesso físico e lógico, inscrição e anulação de um usuário ao mesmo tempo em todos os bancos de dados, utilizam um sistema central de gerenciamento de identidades e monitoramento de eventos em forma unificada, cujo propósito é garantir o máximo possível uma proteção lógica integrada com a segurança física de modo efetivo (STALLINGS; BROWN, 2014).

A segurança de rede, parte do princípio da segurança da informação, pois o objetivo básico se dá nas premissas descritas nos parágrafos anteriores deste capítulo, e no contexto de redes, as aplicações pertinentes à rede deve primeiramente responder algumas questões que são primordiais aos requisitos de conectividades, de acordo com Kurose e Ross (2014), estes requisitos são:

Confidencialidade: trata-se de garantir que somente o remetente e o destinatário pretendido devem ter a capacidade e a autorização para entender o conteúdo da mensagem transmitida. Desta forma, o fato de estranhos terem acesso às informações, acaba por exigir que esta mensagem deva ser cifrada de alguma forma, para que não seja possível sua interceptação. O aspecto de confidencialidade é provavelmente, o principal significado no que diz respeito à comunicação segura (KUROSE; ROSS, 2014);

Autenticação do ponto final: O remetente e o destinatário, devem confirmar a identidade da outra parte envolvida na comunicação. Deve-se entender que a comunicação pessoal entre pessoas, é facilmente resolvida através de reconhecimento visual, o que não ocorre no âmbito computacional. Quando entidades comunicantes trocam mensagens através de qualquer meio pelo qual não é possível ver a outra parte, a autenticação então não é uma tarefa trivial (KUROSE; ROSS, 2014);

Integridade da mensagem: Mesmo que o remetente e o destinatário tenham a condição de se autenticarem reciprocamente na rede, eles também devem ter o desejo de assegurar que o conteúdo trocado entre ambos na comunicação não tenha condições de sofrer alterações, ou seja, tenha condições de ser preservados por qualquer tipo de incidente, como acidentes ou má intenções durante a transmissão propriamente ditas. Extensões das técnicas de soma de verificação comumente disponíveis em protocolos de transportes e de enlaces confiáveis, podem ser utilizados para proporcionar integridades nas mensagens trocadas na comunicação (KUROSE; ROSS, 2014);

Segurança Operacional: Como quase todas as organizações (empresa, universidades etc.) estão conectadas à rede mundial de computadores Internet, é eminente a possibilidade de a rede ser comprometida potencialmente por atacantes mal-intencionados, com o objetivo de obter acessos a essas redes por meio deste ambiente público e distribuído. Os atacantes podem tentar colocar vírus hospedeiros nas redes, como *worms*, cavalo de tróia dentre outros tipos variantes de vírus ou metodologias que possam lhes permitirem adquirir segredes corporativos ou pessoais, mapeamento e configurações de uma rede interna, para ataques iminentes DoS ou DDoS (KUROSE; ROSS, 2014).

Dessa forma, é de extrema importância tratar protocolos IP de segurança, cujo mais conhecido é o Ipsec, uma vez que o mesmo provê segurança na camada de rede. Este protocolo, tem a finalidade de proteger datagramas IP entre quaisquer entidades da camada de rede, incluindo hospedeiros e roteadores (KUROSE; ROSS, 2014). Outro aspecto a ser considerado na segurança de redes, é a implementação de Firewall cujo objetivo se dá pela definição da combinação de hardware e software para isolar a rede interna de uma organização da Internet em geral, de forma a permitir que alguns pacotes possam ser bloqueados ou liberados de acordo com uma regra de política de segurança (já discutida nos capítulos anteriores deste trabalho), uma vez que os firewalls têm a capacidade de ver que um filtro de pacote (tradicional ou de estado) seja inspecionado no campo de cabeçalho de um

protocolo quer seja IP, TCP, protocolo *user datagram* (UDP) ou protocolo de controle internet-mensagem (ICMP). Entretanto, é necessário que se faça uma inspeção profunda de pacotes que trafegam na rede, este então é a necessidade de se ter um sistema de monitoramento reativo e proativo nos aspectos de segurança na rede, tais como o sistema de detecção de intrusos (IDS) e o sistema de prevenção a intrusos (IPS).

IDS é uma ferramenta de segurança, que objetiva detectar intrusões na rede por meio da monitoração do tráfego de dados, lança alarmes para informar caso exista a possibilidade de a segurança do sistema ser violada. A sua utilização é dada em conjunto com organizações de Firewall para acrescentar um nível maior de segurança aos serviços (MORAIS, 2011). O IPS é uma ferramenta que utiliza os alarmes gerados pelo IDS para executar ações de contenção para interromper o ataque e evitar dados à rede. Essas ações podem ser: cancelar a conexão aberta, bloquear o endereço de origem do ataque, limitar a banda das requisições do atacante ou redirecionar o tráfego para uma quarentena. O IPS também mantém uma lista com perfis de tráfego malicioso para realizar a contenção colaborativa do ataque (BRITO et al., 2018).

2.5.1 Tipos de Ataques em Redes

Esta seção versa sobre os tipos de ataques DoS e DDoS que é tema de estudos deste trabalho, uma vez que existem diversos tipos de ataques que não serão explorados e também não é o objetivo deste estudo.

Segundo Kim e Solomon (2014) um ataque em um sistema computacional ou em um ativo de rede atinge o sucesso quando consegue explorar uma vulnerabilidade. Os ataques podem ser classificados em quatro categorias e, ainda um ataque pode ser constituído em todas ou algumas das quatro seguintes categorias: fabricações, interceptações, interrupções e modificações.

- **Fabricações**: Uma fabricação envolve a criação de alguma fraude de modo a enganar usuários não suspeitos.
- **Interceptações**: Uma interceptação envolve estudar transmissões e redirecioná-las para o uso não autorizado.
- **Interrupções**: Uma interrupção causa uma quebra em um canal de comunicação, o que bloqueia a transmissão de dados.

- **Modificações**: Uma modificação é a alteração de dados contidos em transmissões ou arquivos.

Um ataque DoS é do tipo ativo, o qual resulta em usuários do sistema sem acesso aos recursos que deviam estar disponíveis. Um ataque DoS é uma tentativa coordenada de negar um serviço, fazendo com que a vítima pare de realizar suas tarefas produtivamente. As atividades excessivas realizadas pelo ataque são responsáveis pela indisponibilidade dos sistemas para operações e qualquer tipo de utilização por parte dos usuários. Por fim, na iminência de esgotamento de um disco, uma vez que este se torna preenchido, faz com que o dispositivo efetue o travamento da CPU, pois ocorre o chamado gargalo na rede, pelo excesso de gravação em disco (KIM; SOLOMON, 2014).

Os ataques DoS concentram-se em sobrecarregar os elementos de rede, como servidores e roteadores, a fim de impedir o serviço de usuários legítimos do sistema. Os recursos afetados pelos ataques incluem qualquer ativo que estiver conectado à rede atacada (RABIE; DRISSI, 2018). Embora todos os tipos de ataques têm seus níveis, os ataques DoS são os mais poderosos. Comumente as abordagens adaptadas e incorporadas pelos ataques DoS envolvem o envio de mensagens ICMP para a rede de vítimas, como pode-se observar na figura 10, usando o endereço IP da vítima como fonte de pedido que inicia uma tempestade de respostas do ICMP (ALDAEJ, 2017).

Figura 10. Representação de um ataque DoS.

Fonte: Autoria Própria (2018).

Nos casos de ataques DoS ou DDoS, no Brasil, as incidências são reportadas a uma entidade responsável, no caso o CERT.br, que é um grupo de resposta a incidentes de segurança para a internet no Brasil. É responsável por tratar incidentes de segurança em computadores que envolvam redes conectadas à internet no Brasil. Ainda opera como ponto central para notificação de incidentes de segurança no Brasil e também realiza por meio do trabalho de conscientização sobre os problemas de segurança, da análise de tendências e correlação entre eventos na internet brasileira (CERT.br, 2018).

Na figura 11 é possível identificar os tipos de ataques e a quantidade de ocorrências relacionadas com o total informado ao CERT.br no ano de 2017.

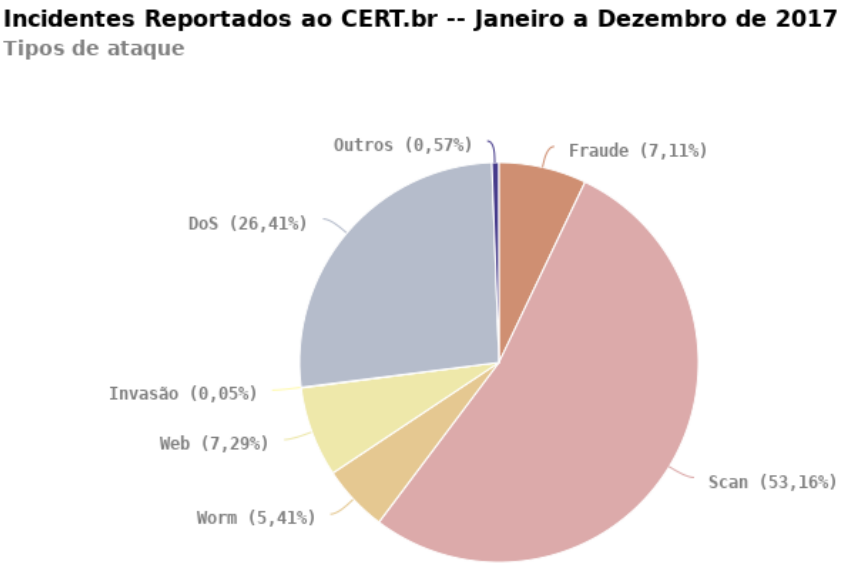

Figura 11 – Gráfico de Incidência dos tipos de ataques.

Fonte: CERT.BR (2018).

Segundo Kim e Solomon (2014) ainda como forma de ataque, tem-se os ataques DDoS, é uma variação de ataque DoS que envolve a inundação de um ou mais computadores com solicitações falsas que geram uma sobrecarga e impede que o computador realize suas tarefas legitimas. Em escopo os ataques DDoS são diferentes dos ataques DoS, no ataque DDoS o invasor penetra em centenas ou milhares de computadores na internet, plantando agentes de ataque automáticos, esses agentes quando instruídos bombardeiam o endereço alvo com mensagens falsas, a chave desse tipo de ataque é a força bruta, isto é, quantidade de solicitações por segundo.

Comumente empresas grandes e universidades são atacadas com ataques DDoS. Nas instituições mais visadas pelos invasores, tem-se como alta prioridade a prevenção para impedir esses ataques. Os ataques DDoS são mais difíceis de evitar que os DoS, pois se originam de diferentes fontes, como pode ser visualizado na figura 12 onde tem-se diversos *hosts* para realização de um ataque em um indivíduo da rede. A proteção DDoS exige diversas camadas de segurança. Uma invasão bem-sucedida pode causar milhões em receita perdida de acordo com Kim e Solomon (2014), este então é o fato gerador de que se deve cuidar o

máximo possível para que ataques destas naturezas não estejam presentem na infraestrutura computacional.

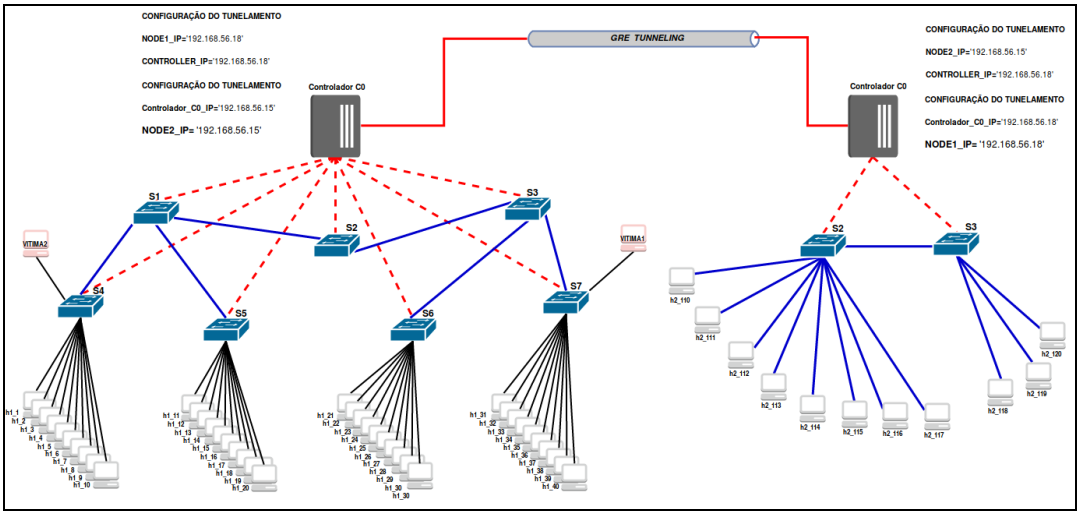

Figura 12. Exemplo de ataque de negação de serviço distribuído, simulado em uma SDN.

Fonte: ARAÚJO et al. (2017).

2.5.2 Ferramentas de Ataque DDoS

A maior parte dos ataques de negação de serviço utilizam ferramentas ou softwares específicos para esse tipo de tarefa (CORRÊA; MARTINS, 2013). Será abordado sobre uma ferramenta de ataques de negação de serviço, o T50.

A ferramenta T50 é utilizada para a realização de ataques de negação de serviços, criada com o propósito de efetuar testes de qualidade de variados tipos de estruturas de redes de computadores. No entanto, como seu código fonte é aberto, foi modificado por *hackers* com o objetivo de viabilizar a ferramenta para a execução de ataque DDoS. Seu grande diferencial é a possibilidade de utilizar vários números de protocolos, dessa forma aumentando as funcionalidades e efetividade nos ataques (CORRÊA; MARTINS, 2013).

2.5.3 Ferramentas de Coleta de Dados

Quando há a ocorrência de um ataque, é necessário que se faça a coleta dos dados para análise posterior e a também para identificá-los na incidência ocorrida. O assunto de coleta de dados, será discutido nos capítulos seguintes deste trabalho, uma vez que serão utilizadas algumas ferramentas para a coleta de dados em redes.

O Tcpdump é um analisador de pacotes, utilizado via terminal, descreve os conteúdos dos pacotes em uma interface de rede fornecida como parâmetro na sua utilização (TCPDUMP, 2017).

2.5.4 Segurança em Redes Definidas por Software

Uma SDN também deve ter a atenção voltada no aspecto de segurança, pois as funcionalidades da segurança deste tipo de redes, equivalem aos algoritmos implementados no controlador central, a lógica implementada para a segurança pode ser com definição de fluxos baseados em estado e segurança que se concentram apenas no fluxo, como: algoritmos de detecção de intrusão ou de anomalias (KREUTZ et al., 2013).

Segundo Mattos et al. (2014) “Os desafios de segurança de uma rede definida por software se dividem em três categorias: negação de serviço, ausência de confiança entre componentes e vulnerabilidades de componentes.”.

A negação de serviço pode ocorrer nos planos de dados e de controle. No plano de dados, qualquer ativo que gere fluxos falsos pode exaurir os recursos de banda, recursos de memória ou tabela de fluxos dos comutadores da rede. No plano de controle a negação pode atingir dois pontos na rede, são eles: o controlador e a comunicação do controlador com os comutadores. Com isso é possível esgotar toda a capacidade de processamento do controlador de rede enviando uma grande quantidade de pacotes com diferentes cabeçalhos (MATTOS et al., 2014).

A ausência de confiança entre componentes da rede pode comprometer uma SDN, pois as funcionalidades que são executadas no controlador podem conter comportamentos maliciosos. Dessa maneira, o controlador deve conseguir verificar quais são as aplicações confiáveis e quais possuem comportamento malicioso. Ainda, a ausência de confiança afeta o registro de ações que ocorreram na rede, pois quando é feito, não apresenta nenhuma garantia de que o acontecido não tenha sido executado por uma aplicação maliciosa (MATTOS et al., 2014).

Por último, temos a vulnerabilidade de componentes que está relacionada com a ausência de confiança entre componentes, pois se um ativo controlador está vulnerável, toda a rede está. Uma vulnerabilidade em um dispositivo permite que um atacante ganhei acesso a

um comutador ou controlador e aplique um ataque contra o plano de controle (MATTOS et al., 2014).

2.6 DESCOBERTA DE TOPOLOGIAS DE REDE

O gerenciamento de topologias é um dos recursos de SDNs, este permite ao controlador facilitar o uso das aplicações no plano virtual. O controlador pode descobrir uma topologia por meio de: descoberta de *host*, *switches* de distribuição e *links* de comunicação entre comutadores. Um *host* é descoberto quando receber um pacote de mensagem do comutador, os *switches* são descobertos na distribuição dos pacotes na rede e os *links* de comunicação são descobertos pelo protocolo de descoberta OpenFlow (OFDP) (KHAN et al., 2017).

2.6.1 Descoberta de Topologias em Redes Clássicas

Os ataques às topologias em redes clássicas são frequentes, estes ataques têm como objetivo modificar uma topologia e causar um distúrbio nas operações da rede em termos de controle e gerenciamento. Se um roteador infectado informar suas configurações de roteamento para seus vizinhos, isso teria como resultado a disseminação do tráfego de rede falsificado. As informações enviadas por meio de um roteador contaminado podem atualizar o banco de dados de links do roteador vizinho com informações erradas, assim afetando todo o roteamento dos pacotes (KHAN et al., 2017).

A Tabela 1 apresenta as ferramentas para descoberta de topologias em rede clássicas, cada uma com uma característica de descoberta diferente.

Tabela 1. Ferramentas para descoberta de topologia em redes clássicas.

Característica	**Descoberta em redes clássicas**
Descoberta de *Hosts*	NMAP
Descoberta de *Switches*	SNMAP
Descoberta de *Link*	Análise dos protocolos (RIP, OSPF, LSA, OLSR)
Gerência do Controle	Independente
Escalabilidade	Número de *Switches*

Fonte: KHAN et al. (2017).

2.6.2 Descoberta de Topologias em Redes Definidas por Software

O gerenciamento de topologias é uma das característica que as SDNs possuem. O desacoplamento do plano de controle do plano de dados permite as SDNs ter uma lógica de controle centralizado. Para alcançar o controle centralizado o controlador da rede SDN deve ter uma visão global da rede. O controlador incorpora diversos módulos de assistência durante a execução de aplicações em SDN. Dentre os módulos, o gerenciador de topologias cria a topologia de toda a infraestrutura da SDN. Para minimizar a sobrecarga da rede, pode-se utilizar um único pacote LLDP para cada *switch* OpenFlow, que deve enviar em *broadcast* para todas as portas ativas. Por meio deste protocolo o processo de descoberta diminui em 40% o envio de mensagens do controlador, dessa forma não causando sobre carga nos *links* (KHAN et al., 2017).

A Tabela 2 apresenta as ferramentas de descoberta de topologias em redes definidas por *software*, bem como as características usadas para a descoberta.

Tabela 2. Ferramentas para descoberta de topologia em redes definidas por *software*.

Característica	**Descoberta em redes definidas por s*oftware***
Descoberta de *Hosts*	Pacote de mensagem de encaminhamento
Descoberta de *Switches*	Distribuição dos pacotes
Descoberta de *Link*	LLDP
Gerencia do Controle	Controlador
Escalabilidade	Número de *Switches* OpenFlow

Fonte: KHAN et al. (2017).

2.7 PROTOCOLO SPANNING TREE

Nos projetos atuais de redes de computadores é comum a utilização de múltiplos *switches*, ou seja, construção de caminhos redundantes para os pacotes. O objetivo principal dessa estratégia é manter a rede funcionando se houver algum incidente como o *switch* falhar ou algum cabo se romper (GOMES, 2005).

O protocolo *spanning tree* é utilizado em *switches* que possuem rotas de redundância, para que estes possam manejar os pacotes entre as rotas possíveis evitando os *loops* de rede. O algoritmo realiza o cálculo de todos os possíveis caminhos para o pacote, mantendo como a

rota principal somente o mais estável para o envio, os *links* de comunicação restantes ficam em condição de *backup* no caso da rota principal se tornar inviável (GOMES, 2005).

Segundo Odom (2004), o algoritmo *spanning tree*, separa as portas do *switch* em encaminhamento ou bloqueada. Toda a porta que estiver definida como encaminhamento estão habilitadas para transmitir dados e toda a porta que estiver definida como bloqueada não realiza tarefa alguma.

2.8 ANÁLISE ESTATÍSTICA

A necessidade do conhecimento numérico se apresenta na sociedade desde sempre, com isso as primeiras operações estatísticas tinham como objetivo conhecer as características das populações através de contagens. Hoje em dia, com a alta dependência da informação, a estatística tornou-se uma ferramenta imprescindível na tomada de decisões em diversas áreas como: agricultura, medicina e engenharia (SANTOS, 2007).

Em geral, a estatística descritiva, permite descrever e entender dados de um cenário utilizando medidas de tendências e medidas de dispersão, como: média, mediana e desvio padrão (SANTOS, 2007).

Análise de sobrevivência é uma das áreas da estatística que mais cresceu nas últimas décadas do século passado. Uma evidência desse sucesso é o número de aplicações da análise de sobrevivência na medicina. Na análise de sobrevivência, a variável de resposta é o tempo até a ocorrência de um evento de interesse. A principal característica da técnica de análise de sobrevivência é a presença de censura, que é a observação parcial da resposta, ou seja, por algum motivo especifico o processo de observação foi interrompido antes do acontecimento do evento de interesse (CUNHA, 2014).

Para a realização da análise, faz-se necessário a utilização de um estimador da função de sobrevivência, podem ser utilizados estimadores paramétricos ou não paramétricos (CUNHA, 2014).

3 METODOLOGIA

O objetivo dessa seção é apresentar os materiais utilizados e a metodologia adotada para o desenvolvimento do trabalho.

A Figura 13 apresenta um fluxograma de como foram executadas as atividades para construção do trabalho. Na primeira fase foi feita a instalação e configuração das ferramentas que serão utilizadas no decorrer das atividades. Na segunda fase foram feitas as definições do ambiente e definidas as regras de avaliação onde são executadas as topologias no ambiente virtual. Na terceira etapa foram efetuados os ataques e incidentes a serem avaliados e a coleta dos dados, para cada topologia foram coletadas 30 amostras para cada parâmetro definido. Na quarta fase foi feita a análise dos dados e construção dos resultados para apresentação.

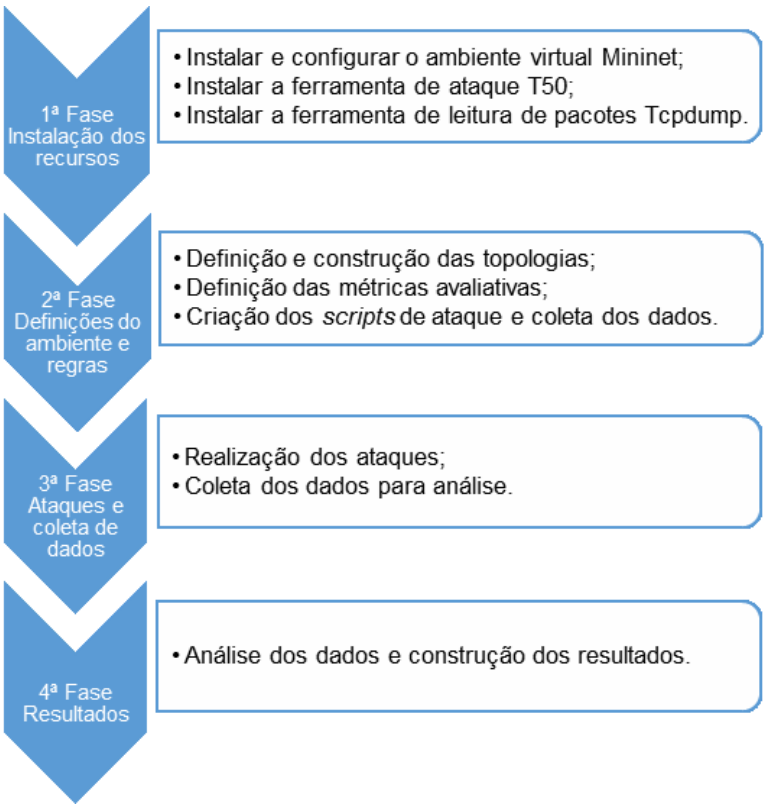

Figura 13 - Fluxograma da metodologia.

Fonte: Autoria Própria (2018).

3.1 RECURSOS UTILIZADOS

Os recursos que foram utilizados neste trabalho são o MININET como emulador de SDN, as ferramentas a serem utilizadas são o T50 e *Shell Scripts* para realização de testes com ataques DDoS sobre o cenário construído com o emulador, e para coleta de dados para as análises será utilizado o Tcpdump.

3.1.1 Mininet

Para este trabalho, foi utilizado o ambiente simulado Mininet, por meio de uma Máquina Virtual (VM) disponibilizada pelo MININET TEAM (2018). O sistema operacional disponibilizado nesta VM é o Ubuntu Server 14.04 LTS – 64 bits. Dessa forma optou-se por utilizar as seguintes tecnologias:

- Controlador POX;
- Protocolo de descoberta *Link Layer,* para exploração da rede;
- Configuração de *spanning tree* para as topologias anel e torus que possuem multi-caminhos;
- A utilização do *learning switch* para construir as tabelas de fluxo do controlador.

3.1.2 Cenários

Os cenários utilizados para os ataques foram com as topologias: anel ilustrado na Figura 14 e o *dump* de dispositivos e links no Apendice I, árvore ilustrado na Figura 15 e e o *dump* de dispositivos e links no Apendice II e torus ilustrado na Figura 17 e o *dump* de dispositivos e links no Apendice III. Todos continham cem (100) hosts, um (1) controlador POX, a quantidade de switches na topologia torus e anel foram utilizados cinco (5) switches e na topologia árvore utilizou-se treze (13) switches. Esta diferença se dá pela forma como são estruturadas as topologias, dessa forma, a topologia árvore necessita mais *switchs* para realizar a conexão entre os *hosts* definidos.

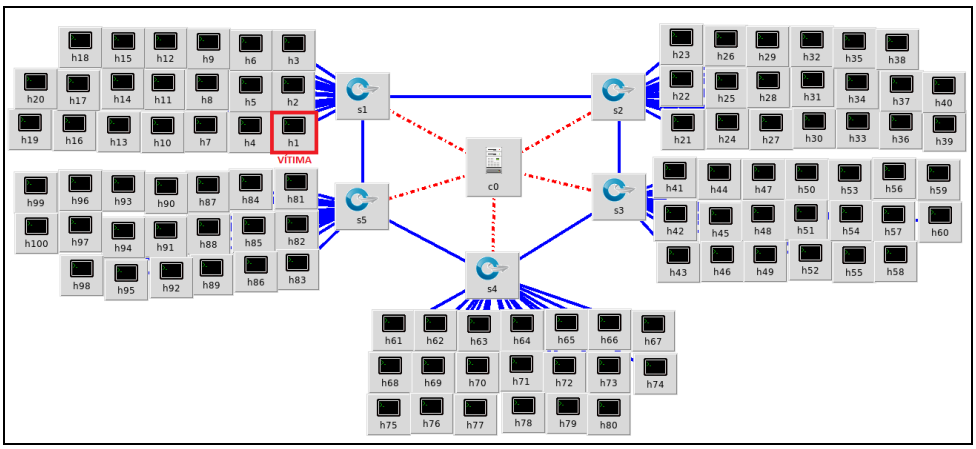

Figura 14 – Cenário da topologia anel.
Fonte: Autoria Própria (2018).

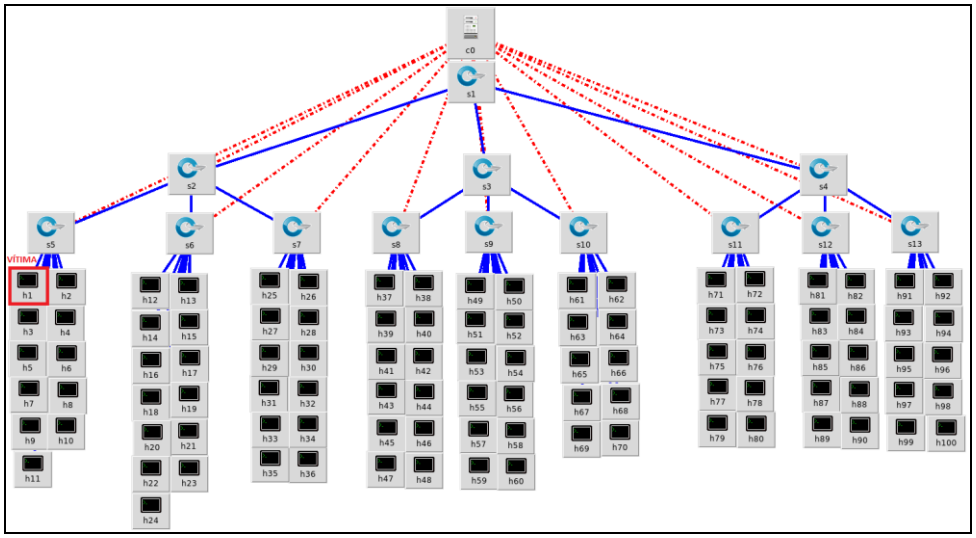

Figura 15 – Cenário da topologia árvore.
Fonte: Autoria Própria (2018).

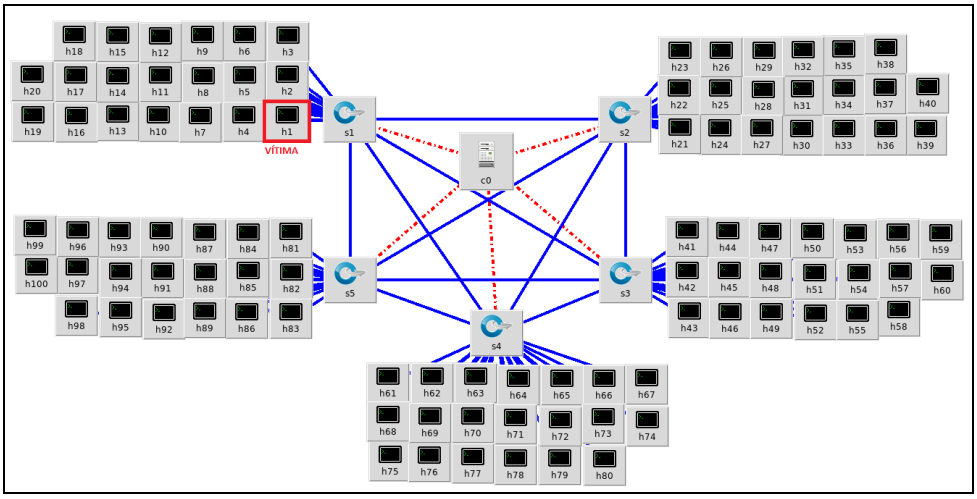

Figura 16 – Cenário da topologia torus.
Fonte: Autoria Própria (2018).

3.1.3 Ferramenta de Ataque DDoS

A fim de testar os cenários que foram construídos no emulador Mininet, foi feito uso da ferramenta T50 para realizar os ataques. Os ataques têm como objetivo a injeção de pacotes na rede para causar lentidão e até derrubar o serviço da rede. A ferramenta funciona utilizando terminal Linux com parâmetros específicos.

Para efetuar os ataques no *host* determinado como vítima, foi utilizado o seguinte comando:

"./t50 IP_VÍTIMA --flood --turbo --dport 80"

O comando apresentado é explicado na Tabela 3.

Tabela 3. Explicação do comando de ataque.

Comando	**Explicação**
./t50	Executa a ferramenta.
IP_VITIMA	Deve ser escrito o endereço IP da vítima.
--flood	Parâmetro que realiza a injeção de um grande número de pacotes.
--turbo	Parâmetro que potencializa o parâmetro --flood.
--dport 80	Parâmetro que define a porta do ataque.

Fonte: Autoria Própria (2018).

Foram realizados dois tipos de testes, o primeiro com sete (7) *hosts* atacantes iniciados a partir do início da contagem do tempo, o segundo com quatorze (14) *hosts* iniciados gradativamente de cinco (5) em cinco (5) segundos. No segundo tipo, se o *host* vítima parasse de responder antes do início dos 14 *hosts* atacantes, só eram iniciados os *hosts* com tempo menor de inicialização do que o tempo de morte da rede.

Quando a rede parava de responder um dos *scripts* desenvolvidos era responsável por parar o ataque, segue abaixo o *script,* representado em no Algoritmo 1 e no fluxograma da Figura 17 que em um *loop* verificava contiguamente se o arquivo morto.txt já estava preenchido, enquanto não estivesse o ataque não era parado.

Algoritmo 1 – Script de parada do ataque.

```
#!/bin/bash
contador=0
while [ $contador -lt 1 ]; do
        if [ ! -s ~/scripts/morto.txt ]; then
                sleep 1
        else
                A=`ps -ef | grep t50 | awk '{print $2}'`;sudo kill -9 $A
                contador=1
        fi
done
```

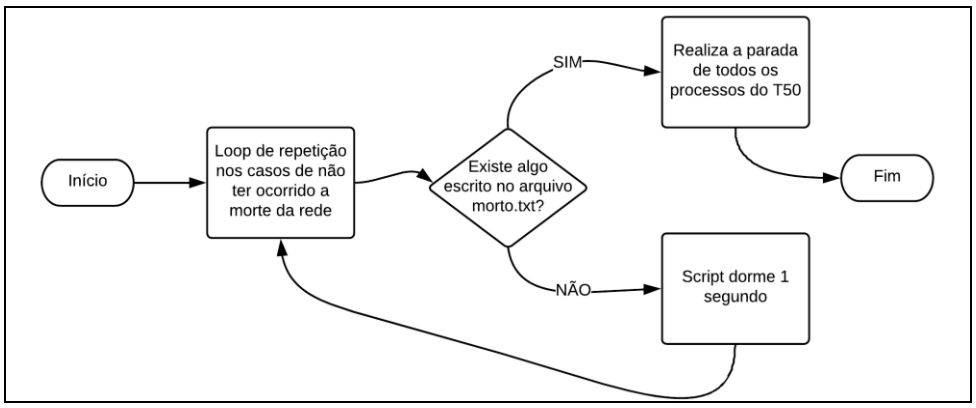

Figura 17 – Fluxograma do algoritmo de parada do ataque.
Fonte: Autoria Própria (2018).

3.1.4 Coleta de Dados

A coleta dos dados para análise foi feita utilizando a ferramenta Tcpdump e *Shell Scripts*.

O Tcpdump foi utilizado apenas no *host* vítima a fim de verificar a quantidade de pacotes trafegados na interface de rede durante o ataque, a ferramenta foi utilizada com o seguinte comando: "tcpdump –i INTERFACE", onde o comando tcpdump inicia a ferramenta, -i é um parâmetro para especificação da interface de rede seguido da interface desejada.

Foi utilizado para medir dos tempos de morte e de recuperação um *script*, como segue.

Algoritmo 2 – Script de coleta de tempos.

```
#!/bin/bash
date >> teste_date.txt
contador=0
contador1=0
while [ $contador -lt 1  ]; do
      ping -c 1 10.0.0.1 | grep -i  Unreachable > ping.txt
      if [ ! -s  ping.txt  ]; then
            sleep 1
      else
            date >> morto.txt
            sleep 1
            while [ $contador1 -lt 1  ]; do
                  ping -c 1 10.0.0.1 | grep -i Unreachable > ping.txt
                  if [ ! -s ping.txt ]; then
                        date >> funcionando2.txt
                        sleep 1
                        contador=1
                        contador1=1
                  fi
            done
      fi
done
```

Na Figura 18 pode-se observar o fluxograma do *script* que realizou a captura dos tempos para cada teste realizado.

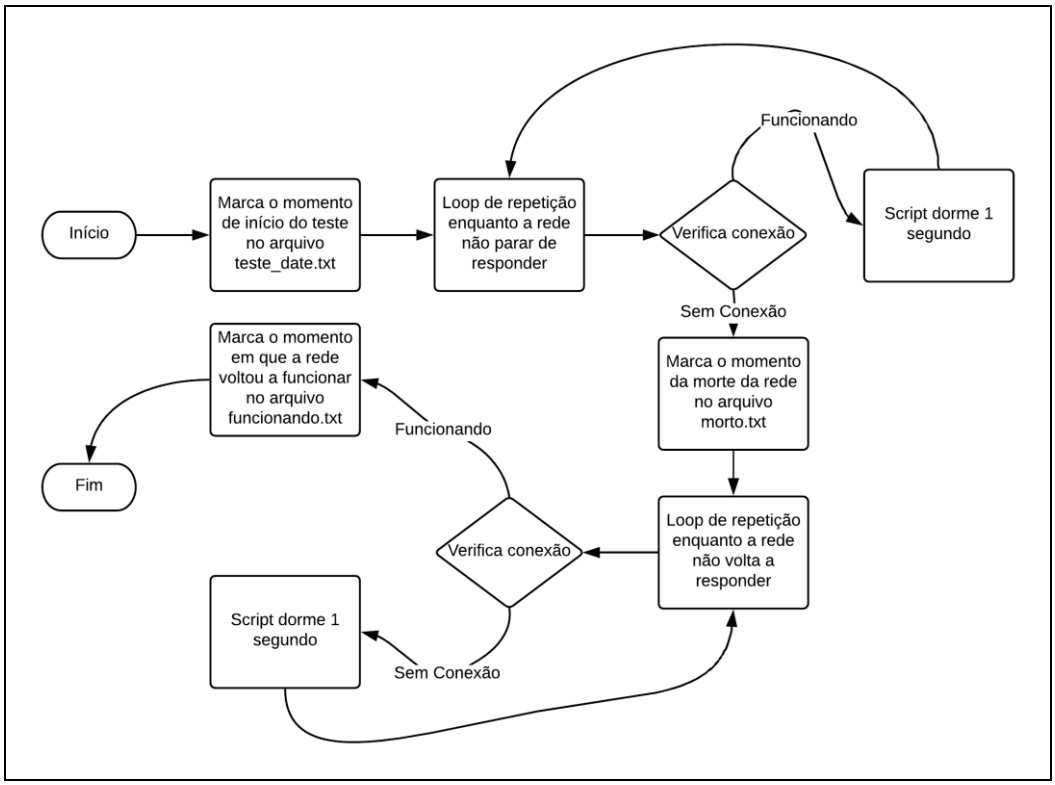

Figura 18 – Fluxograma do algoritmo de captura de tempos.
Fonte: Autoria Própria (2018).

O Iperf foi utilizado para realizar a medição da largura de banda das topologias. Para realizar o teste foram necessários dois (2) hosts aleatórios, um (1) deles executando o comando "iperf -s" para definir como servidor do teste e outro com o comando "iperf –c IP_SERVIDOR" para verificar a largura de banda entre os dois hosts selecionados, a Figura 19 apresenta um teste realizado entre os hosts um (1) e dois (2) da topologia árvore.

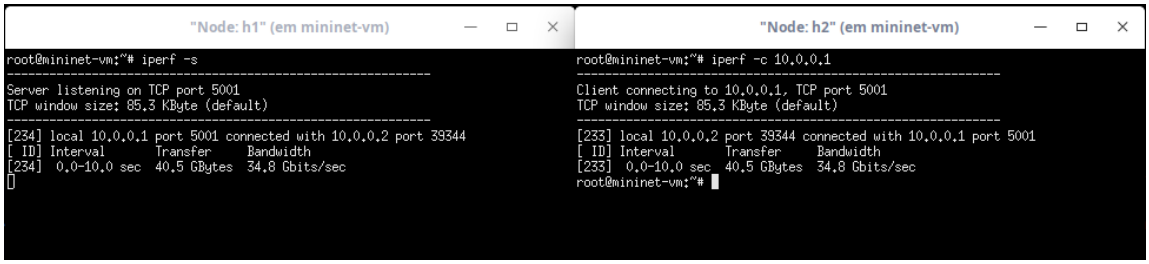

Figura 19 – Teste de largura de banda.
Fonte: Autoria Própria (2018).

3.1.5 Análise Estatística

Para efetuar a análise dos dados foi utilizado o *software* Rstudio, que é um ambiente livre para computação estatística e gráfica (THE R FUNDATION, 2018).

O *software* Rstudio utiliza a linguagem R e pode ser utilizado com uma ampla variedade estatística, que competem: modelamento linear e não linear, classificação estatística, análise de tempo, classificação, clusterização, entre outros (THE R FUNDATION, 2018). O *software* R foi utilizado com os dados obtidos dos testes das topologias, aplicando técnicas de análise de sobrevivência utilizando estimadores não paramétricos. Para a realização da análise foram utilizados os seguintes comandos:

- require(survival) para o *download* dos arquivos do pacote survival;
- require(readr) para o *download* dos arquivos do pacote readr para leituras;
- analise <- read_csv(‘CAMINHO_ARQUIVO.csv’) para passar os dados do arquivo csv para a variável analise;
- km <- survfit(Surv(analises$TR, analises$status)~TOPOLOGIA, data = analises) para realizar os cálculos da análise de sobrevivência sobre os dados separando-os por topologia;
- plot(km,xlab = 'Tempo de Recuperação, ylab = 'S(t)', col=curvas) para desenhar o gráfico da análise;
- legend("topright", legendas, col=curvas, lty =1) para desenhar a legenda do gráfico;

3.1.6 Protocolo de Descoberta *Link-Layer*

A fim de verificar a topologia em uso durante a realização dos testes de ataque foi utilizado o protocolo de descoberta *link-layer*. Este deve enviar pacotes para cada *switch* OpenFlow na rede que envia os pacotes em *broadcast* para todas as portas ativas. Dessa forma, tem-se o conhecimento sobre os *links* de ligação entre os ativos da rede, com os *links* o protocolo determinará a topologia da rede em análise. O protocolo foi utilizado por meio do algoritmo de LLDP no controlador POX utilizado nas topologias.

3.1.7 Protocolo *Spanning Tree* e *Learning*

O protocolo *spanning tree* foi utilizado no controlador para a criação das melhores rotas disponíveis na topologia em conjunto com o algoritmo *learning* que constrói a tabela de fluxos do controlador, com isto, o controlador evita que os pacotes de tráfego entrem em *loop* na rede, o que causa a perda de pacotes e falha de comunicações.

4 ANÁLISE DE RESULTADOS

Para avaliar a resiliência das topologias foram capturados dados de largura de banda, que apresentam qual a capacidade da topologia de tráfego, tempo de morte, que apresenta o tempo que a topologia passou resistindo ao ataque e tempo de recuperação, que apresenta o tempo que a topologia levou para voltar a funcionar depois do ataque e a quantidade de pacotes que trafegaram pela interface atacada.

Os testes foram efetuados no ambiente emulado Mininet em três cenários diferentes: topologia em árvore, topologia em anel e topologia torus, todos com um controlador POX que executava com os algoritmos LLDP, *spanning tree* e *learning*.

A coleta de dados foi obtida em 31 experimentos com cada uma das topologias selecionadas. Os dados obtidos em cada experimento seguem nas Tabelas 4, 5 e 6, apresentando a identificação do experimento na topologia, a largura de banda da topologia em Gigabits por segundos, o tempo de morte da topologia em segundos, o tempo de recuperação da topologia em segundos e a quantidade de pacotes trafegados na interface da vítima.

Tabela 4 – Dados de amostras da topologia Anel.

id	Largura de banda (gbits/s)	Tempo de morte (s)	Tempo de recuperação (s)	Quantidade de pacotes
1	30,2	13	115	45337
2	31,9	19	116	43811
3	29,9	48	648	142563
4	29,6	16	737	157864
5	30,1	51	366	124012
6	28,8	13	32	24591
7	29,4	15	271	58267
8	30,2	19	549	10928
9	29,2	15	859	111789
10	28,3	20	203	35432
11	28,1	14	198	27663
12	30,7	19	110	46310
13	31,4	12	98	41994
14	31,9	13	122	46140
15	32	13	114	44170
16	31,4	25	98	41288
17	32,1	69	305	73524
18	29,9	78	421	144663
19	32,5	14	353	132342
20	32	18	256	75329
21	32,9	21	187	47652

22	29,4	59	343	129482
23	30,2	5	250	72703
24	29,2	34	196	244544
25	28,3	20	72	15312
26	28,1	48	653	37193
27	30,7	53	140	61942
28	31,4	56	60	48465
29	31,9	48	191	50641
30	32	46	1320	195084
31	31,4	57	344	21964

Fonte: Autoria Própria (2018).

Tabela 5 – Dados de amostras da topologia Torus.

id	Largura de banda (gbits/s)	Tempo de morte (s)	Tempo de recuperação (s)	Quantidade de pacotes
1	33	6	288	78975
2	33,3	56	120	35963
3	32	58	90	37627
4	32,2	68	144	38783
5	32,1	49	98	45713
6	32,4	13	148	62533
7	32	22	370	96730
8	33,3	17	163	69975
9	31,3	11	306	48704
10	33,6	12	305	49052
11	32,6	16	537	36460
12	33,3	3	35	19518
13	33,2	11	108	45912
14	33,5	19	570	200681
15	31,9	60	112	41519
16	33,8	13	129	134821
17	33,8	15	23	28870
18	34,8	18	194	16305
19	34,6	17	30	37754
20	33,8	43	93	40368
21	33,4	60	95	38746
22	31,3	48	214	50588
23	33,6	37	215	58730
24	32,6	59	51	24231
25	33,3	70	1320	174901
26	33,2	63	233	59568
27	33,5	39	434	94117
28	31,9	51	50	32370
29	33,8	5	271	50284
30	33,8	44	18	17517
31	34,8	48	14	70910

Fonte: Autoria Própria (2018).

Tabela 6 – Dados de amostras da topologia Árvore.

id	Largura de banda (gbits/s)	Tempo de morte (s)	Tempo de recuperação (s)	Quantidade de pacotes
1	24,5	20	420	15211
2	28,3	56	64	66249
3	27,3	54	576	46616
4	28	42	662	34516
5	27,5	53	540	32707
6	26,7	47	936	66300
7	26,4	36	276	44441
8	27,5	33	402	35196
9	26,5	45	247	49428
10	27,3	48	750	32560
11	28,2	52	522	33715
12	22,9	29	80	45701
13	24,9	11	59	59667
14	27,8	15	796	200794
15	26,8	51	1296	219517
16	27,4	12	807	164829
17	23,9	14	158	136366
18	26,2	13	780	221587
19	27	15	1061	240792
20	25,9	12	898	228852
21	26,5	23	237	194769
22	24,5	57	27	84482
23	28,3	15	1320	6368
24	27,3	57	210	60697
25	28	63	11	10346
26	27,5	16	1320	88929
27	26,7	63	205	49748
28	26,4	16	100	2083
29	27,5	4	48	27268
30	26,5	56	240	53725
31	27,3	30	185	58236

Fonte: Autoria Própria (2018).

Na Figura 20 pode-se observar as médias e desvio padrão dos dados obtidos da largura de banda das topologias. De acordo com o gráfico a topologia que teve destaque foi a topologia torus com maior média de largura de banda, o que representa maior capacidade de tráfego e estabilidade. Em seguida pode-se observar a topologia anel com resultados melhores que a topologia árvore.

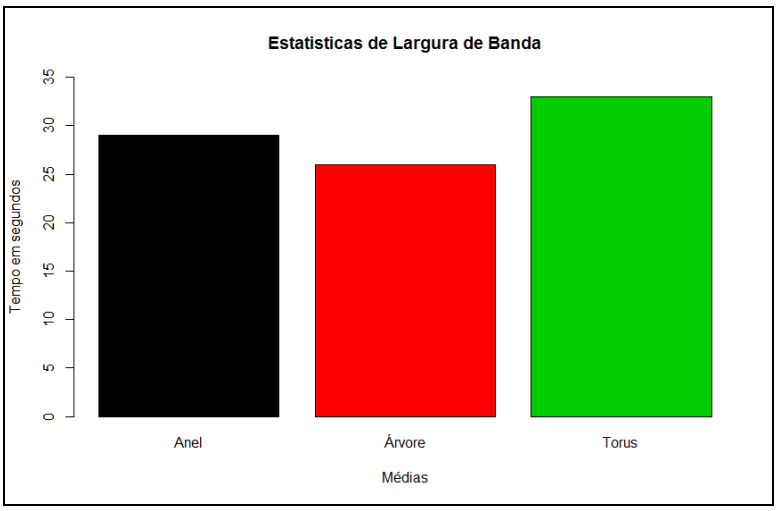

Figura 20 – Gráfico de largura de banda.
Fonte: Autoria Própria (2018).

Também foram obtidos os tempos de resistência da topologia antes de parar de responder as requisições, como pode ser visto na Figura 21. Observando os dados pode-se concluir que as diferenças entre as médias não são discrepantes ficando em torno de 10% de diferença, portanto não apresentam alto nível de significância para a tomada de decisão quanto a resiliência da topologia.

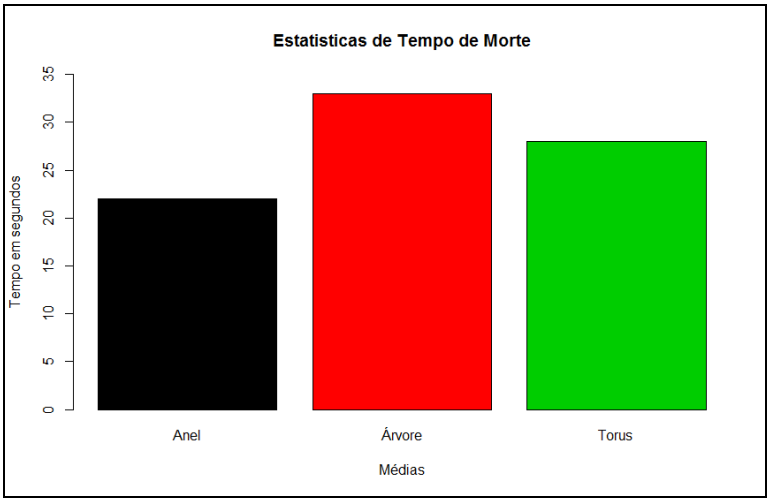

Figura 21 - Gráfico de tempo de morte das topologias.
Fonte: Autoria Própria (2018).

Nas medições de tempo de recuperação ilustrados na Figura 22, pode-se observar que a topologia torus se destacou em comparação com as topologias árvore e anel, pois, apresentou uma média de recuperação da rede menor. Portanto a diferença de mais de 100% em relação a topologia árvore e em torno de 30% em relação a topologia anel é significativa para sustentar que a topologia torus possui maior resiliência, seguida da topologia anel e

árvore com a menor resiliência. O principal motivo da diferença entre as resiliências encontradas são as rotas de redundância presentes nas topologias anel e torus. Na topologia anel com duas rotas entre cada *switch* e na topologia torus com quatro rotas entre cada *switch*. A topologia árvore não apresenta nenhuma rota de redundância para os pacotes.

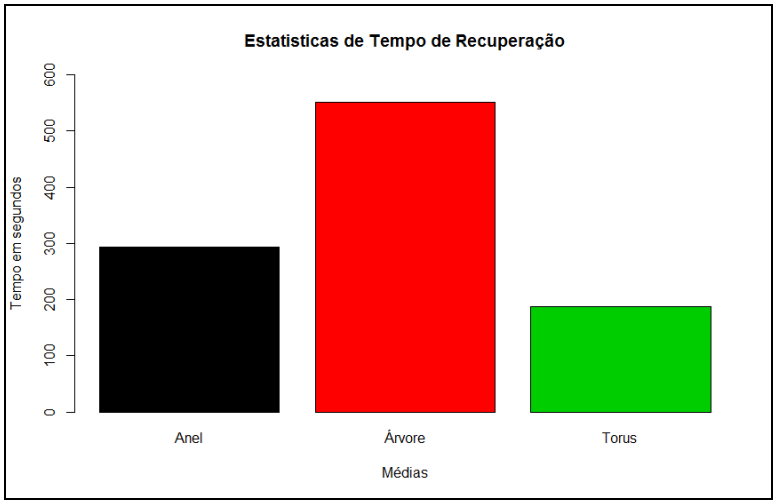

Figura 22 – Gráfico de tempo de recuperação das topologias.
Fonte: Autoria Própria (2018).

Para verificar o desempenho de cada topologia em relação ao tempo de recuperação, pode-se observar na figura 23, o gráfico gerado pela aplicação estatística da análise de sobrevivência, onde cada linha desenhada representa uma topologia e as amostras de tempo de recuperação. A análise de sobrevivência apresenta a probabilidade da topologia levar um determinado tempo para a sua recuperação. Dessa forma, na Figura 19, pode-se observar que a topologia torus apresenta a probabilidade de levar mais de 600 segundos para sua recuperação menor do que 10%, a topologia anel apresenta a probabilidade de levar mais de 600 segundos em torno de 20% e a topologia árvore apresenta a probabilidade de levar mais de 600 segundos em torno de 40%. Com isso pode-se definir a topologia torus como mais resiliente em relação as topologias anel e árvore.

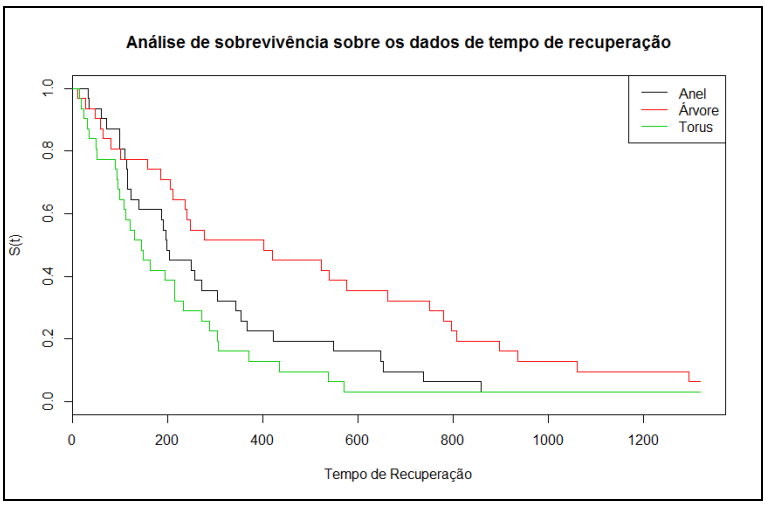

Figura 23 – Gráfico da análise de sobrevivência sobre os dados de tempo de recuperação.
Fonte: Autoria Própria (2018).

Contudo, pode-se afirmar que a adição de rotas redundantes entre os *switches* auxilia no aumento da resiliência da topologia utilizada. Nesse contexto a topologia em destaque foi a torus que apresentou maior resiliência em relação as outras testadas, isso ocorre em função da sua infraestrutura lógica robusta, dada pela utilização de vários caminhos de redundância entre cada *switch* de comunicação.

5 CONCLUSÕES

Neste trabalho, procurou-se apresentar vários conceitos que envolvem redes de computadores, como as características de redes clássicas, características de redes definidas por *software*, como funcionam e as diferenças entre as arquiteturas de redes clássicas e SDNs, além destes também foram exploradas características de segurança das redes clássicas e SDNs. Para a apresentação dos resultados foram aplicadas técnicas de estatística descritiva básica e estatística de sobrevivência, que é comumente utilizada em amostras de doenças, estas auxiliaram para a explanação dos resultados.

Partindo destes conceitos e técnicas, foram estudados artigos atuais publicados em revistas e periódicos de renome como o Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) e a *Association for Computing Machinery* (ACM).

Dessa forma, as topologias com suas dimensões, a construção de *scripts* para mensura de tempos, que garantem credibilidade para a extração dos dados, estudos sobre o paradigma de SDN e na área de segurança computacional, foram de grande valia para o aprendizado de redes de computadores e segurança computacional, exigindo conhecimento aplicado de disciplinas da área de comunicação de dados, redes de computadores, segurança computacional, arquitetura de comunicações e sistemas operacionais.

Com isso, pode-se observar que existe diferença de resiliência entre as topologias apresentadas, tendo em destaque a topologia torus, que apresentou os melhores resultados nas análises estatísticas, seguida pela topologia anel e por último a topologia árvore, que apresentou os piores resultados. Nas topologias anel e torus a principal diferença é a utilização de rotas de redundância, pois o controlador POX utilizado realizava cálculos de rota a cada pacote para avaliar a melhor rota até o endereço de destino do pacote. Entre as topologias torus e anel, a única diferença destacável é a quantidade de rotas de redundância utilizadas, que na topologia anel são utilizadas duas rotas de redundância enquanto na topologia torus existem quatro rotas entre cada *switch* de comunicação.

Com base nos resultados obtidos dos experimentos e estudos, os objetivos foram atingidos e pode-se recomendar o uso da topologia torus como a mais resiliente dentre as topologias testadas neste trabalho.

Para trabalhos futuros sugere-se a utilização de outras ferramentas e estratégias de ataque para realização do DDoS na topologia, também a utilização de mecanismos de segurança para prevenir e remediar os ataques em execução, bem como IDSs ou IPSs. A utilização de algoritmos para posicionamento e aumento do número de controladores em conjunto com a incorporação de *links* de redundância entre ativos da rede também pode ser explorada. Além disso também pode-se aplicar os testes em outras topologias especializadas como: redes Fat-Tree, infraestruturas de redes com *Inband Telemetry*, bem como redes com roteamento, redes com multi-caminhos e avaliação de controle de congestionamento. Dessa forma pode-se melhorar a pesquisa no sentido de definir as topologias com maior resiliência baseado em experimentos, que é a contribuição máxima deste trabalho.

REFERÊNCIAS

ALDAEJ, Abdulaziz. Information security and distributed denial of service attacks: A survey. **In: Electrical and Computing Technologies and Applications (ICECTA), 2017 International Conference on**. IEEE, 2017. p. 1-6.

ARAÚJO, Márcio José et al. Uma avaliação empírica dos efeitos de ataques de negação de serviço sobre redes definidas por software. **In: SEMANA TECNOLÓGICA ACADÊMICA DE CIÊNCIA DA COMPUTAÇÃO DA UTFPR-SH, 4., 2017, Santa Helena, PR**. Anais..., Santa Helena: UTFPR, 2017. p.48-60. Disponível em: < http://sh.utfpr.edu.br/setac/Anais2017.pdf>.

ARAÚJO, Tiago Emílio de Sousa; MATOS, Fernando Menezes; MOREIRA, Josilene Aires. Intrusion detection systems' performance for distributed denial-of-service attack. **In: Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 2017 CHILEAN Conference on**. IEEE, 2017. p. 1-6.

BEZERRA, Adonel. **Evitando Hackers: Controle seus sistemas computacionais antes que alguém o faça.** Rio de Janeiro, Ciência Moderna. 2012.

BOMBAL, David. **SDN 101: Using Mininet and SDN Controllers**. 2018. Disponível em: <http://pakiti.com/sdn-101-using-mininet-and-sdn-controllers/>. Acesso em 3 de maio de 2018.

BRITO, Italo Valcy S.; RIBEIRO, Adriana Viriato; SAMPAIO, Leobino N. **SDN-IPS: Uma Ferramenta para Contenção Automatizada e Colaborativa de Ataques Cibernéticos Baseada em SDN**. 2018.

CERT.br. **Recomendações para Melhorar o Cenário de Ataques Distribuídos de Negação de Serviço**. 2016. Disponível em: <https://www.cert.br/docs/whitepapers/ddos/#6.3>. Acesso em 7 de maio de 2018.

CERT.br. **Incidentes Reportados ao CERT.br – Janeiro a Dezembro de 2017**. 2018. Disponível em: <https://www.cert.br/stats/incidentes/2017-jan-dec/tipos-ataque.html>. Acesso em 10 de junho de 2018.

CERT.br. **Sobre o CERT.br.** 2018. Disponível em: <https://www.cert.br/sobre/>. Acesso em 10 de junho de 2018.

CHEN, Li-Chiou; LONGSTAFF, Thomas A.; CARLEY, Kathleen M. Characterization of defense mechanisms against distributed denial of service attacks. **Computers & Security**, v. 23, p. 665e678, 2004.

CORRÊA, André Luiz Riccó; MARTINS, Henrique Pachioni. Monitoramento de ataques de negação de serviço: Um caso prático utilizando slowloris. **Faculdade de Tecnologia de Bauru (FATEC),** 2013.

CUNHA, Franciely Farias da. **ANÁLISE DE SOBREVIVÊNCIA**. 2014. Disponível em: <http://www.ufpa.br/heliton/arquivos/aplicada/seminarios/M2_06_Analise_Sobrevivencia_Franciely.pdf>. Acesso em 18 de novembro de 2018.

GOMES, Daniel Cardoso; ALVES, Rêmulo Maia; DOS SANTOS, Anderson Bernardo. Proposta de Otimização do Tráfego da Rede da Universidade Federal de Lavras Utilizando a Técnica de Spanning Tree Protocol. **INFOCOMP**, v. 4, n. 4, p. 48-57, 2005.

GOMES, João Maria. **Topologia de Redes.** 2012. Disponível em: <https://estudoderedes.wordpress.com/tag/hibrida/>. Acesso em 19 de junho de 2018.

GUEDES, Dorgival et al. Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012**, v. 30, n. 4, p. 160-210, 2012.

HELLER, Brandon; SHERWOOD, Rob; MCKEOWN, Nick. The controller placement problem. In: **Proceedings of the first workshop on Hot topics in software defined networks**. ACM, 2012. p. 7-12.

JUNIPER. **SDN, o caminho para a aplicação na nuvem**. 2015. Disponível em: <http://www.junipernetworks.com.br/sdn-o-caminho-para-a-aplicacao-na-nuvem>. Acesso em 26 de novembro de 2018.

HOCK, David et al. Pareto-optimal resilient controller placement in SDN-based core networks. In: **Teletraffic Congress (ITC), 2013 25th International**. IEEE, 2013. p. 1-9.

JAMMAL, Manar et al. Software defined networking: State of the art and research challenges. **Computer Networks**, v. 72, p. 74-98, 2014.

JANTSCH, Rodrigo. **Mitigação de ataques DDoS em redes baseadas em infraestruturas SDN/NFV**. 2016.

KHAN, Suleman et al. Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art. **IEEE Communications Surveys & Tutorials**, v. 19, n. 1, p. 303-324, 2017.

KAUR, Sukhveer; SINGH, Japinder; GHUMMAN, Navtej Singh. Network programmability using POX controller. **In: ICCCS International Conference on Communication, Computing & Systems, IEEE**. 2014.

KIM, David; SOLOMON, Michael G.. **FUNDAMENTOS DE SEGURANÇA DE SISTEMAS DE INFORMAÇÃO**. Rio de Janeiro: Ltc, 2014. 386 p. Tradução de: Daniel Vieira.

KREUTZ, Diego et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14-76, 2015.

KREUTZ, Diego; RAMOS, Fernando; VERISSIMO, Paulo. Towards secure and dependable software-defined networks. In**: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking**. ACM, 2013. p. 55-60.

KUROSE, James F.; ROSS, Keith W.. **Redes de Computadores e a Internet: Uma abordagem top-down**. 6. ed. São Paulo: Pearson Education do Brasil Ltda, 2014. 634 p.

LANGE, Stanislav et al. Heuristic approaches to the controller placement problem in large scale SDN networks. **IEEE Transactions on Network and Service Management**, v. 12, n. 1, p. 4-17, 2015.

LINS, Theo. **Redes Definidas Por Software (Software Defined Networks) SDN**. 2015. Disponível em: <http://www.decom.ufop.br/imobilis/redes-definidas-por-software-software-defined-networks-sdn/>. Acesso em 19 de junho de 2018.

LYON, Gordon. **Nmap Network Scanning**. 2018. Disponível em: <https://nmap.org/book/preface.html#preface-intro>. Acesso em 10 de junho de 2018.

MARTIMIANO, Luciana Andréia Fondazzi. **Sobre a estruturação de informação em sistemas de segurança computacional: o uso de ontologias.** 2006. Tese de Doutorado. Universidade de São Paulo.

MATTOS, Diogo Menezes Ferrazani; DUARTE, Otto Carlos Muniz Bandeira. **AuthFlow: Um Mecanismo de Autenticação e Controle de Acesso para Redes Definidas por Software**. Rio de Janeiro – RJ – Brasil 2014.

MCKEOWN, Nick et al. OpenFlow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review,** v. 38, n. 2, p. 69-74, 2008.

MENDES, Douglas Rocha. **REDES DE COMPUTADORES: Teoria e Prática**. 2. ed. São Paulo: Novatec, 2015.

MENEZES, Pablo Marques et al. Estudo Catalográfico sobre Redes Definidas por Software (SDN) no âmbito brasileiro. **Semana de Pesquisa da Universidade Tiradentes-SEMPESq**, n. 18, 2018.

MELO, Ronaldo. **SDN (Software Defined Network) realidade do mundo virtual?**. 2018. Disponível em: <http://www.vert.com.br/blog-vert/sdn-software-defined-network-realidade-do-mundo-virtual/>. Acesso em 26 de novembro de 2018.

MININET TEAM. **Mininet Overview**. 2018. Disponível em: <http://Mininet.org/overview/> Acesso em 14 de abril de 2018.

MONTIBELER, Pedro; FARIAS, Fernando; ABELÉM, Antônio. Fator de Resiliência para Aprimoramento Topológico em Redes Definidas por Software. **Workshop de Gerência e Operação de Redes e Serviços (WGRS_SBRC)**, [S.l.], v. 22, n. 1/2017, may 2017. ISSN 2595-2722. Disponível em: <http://portaldeconteudo.sbc.org.br/index.php/wgrs/article/view/2593>. Acesso em: 17 nov. 2018.

MORAIS, Guilherme Filipe Zorego Rodrigues. **Análise e Implementação de Sistemas IDS e IPS**. 2011. Tese de Doutorado.

MÜLLER, Lucas F. et al. Survivor: an enhanced controller placement strategy for improving SDN survivability. In: **Global Communications Conference (GLOBECOM), 2014 IEEE**. IEEE, 2014. p. 1909-1915.

NUNES, Bruno Astuto A. et al. A survey of software-defined networking: Past, present, and future of programmable networks. **IEEE Communications Surveys & Tutorials**, v. 16, n. 3, p. 1617-1634, 2014.

ODOM, Wendell. **Cisco CCNA ICND Exam Certification Guide**. Cisco Press, 2004.

RABIE, Rashed; DRISSI, Maroua. Applying sigmoid filter for detecting the low-rate denial of service attacks. **In: Computing and Communication Workshop and Conference (CCWC)**, 2018 IEEE 8th Annual. IEEE, 2018. p. 450-456.

ROS, Francisco Javier; RUIZ, Pedro Miguel. Five nines of southbound reliability in software-defined networks. In: **Proceedings of the third workshop on Hot topics in software defined networking**. ACM, 2014. p. 31-36.

SANTOS, Carla. Estatística descritiva. **Manual de Auto-aprendizagem, Lisboa, Edições Sílabo**, 2007.

SEZER, Sakir et al. Are we ready for SDN? Implementation challenges for software-defined networks. **IEEE Communications Magazine**, v. 51, n. 7, p. 36-43, 2013.

STALLINGS, William; BROWN, Lawrie. **SEGURANÇA DE COMPUTADORES: PRINCÍPIOS E PRÁTICAS**. 2. ed. Rio de Janeiro: Elsevier, 2014. 726 p. Tradução de: Arlete Simille Marques.

TCPDUMP. **TCPDUMP**. 2017. Disponível em: <https://www.tcpdump.org/tcpdump_man.html>. Acesso em 19 de junho de 2018.

THE R FUNDATION. **GETTING STARTED.** 2018. Disponível em: <https://www.r-project.org/>. Acesso em 19 de junho de 2018.

THE R FUNDATION. **What is R?.** 2018. Disponível em: <https://www.r-project.org/about.html>. Acesso em 19 de junho de 2018.

TORRES, Gabriel. **REDES DE COMPUTADORES**. 2. ed. Rio de Janeiro: Novaterra Editora e Distribuidora Ldta, 2015.

WEIDMAN, Georgia. **Testes de Invasão: Uma introdução prática ao hacking**. Novatec Editora, 2014.

YU, Shui et al. Can we beat DDoS attacks in clouds?. **IEEE Transactions on Parallel and Distributed Systems,** v. 25, n. 9, p. 2245-2254, 2014.

APÊNDICE I

<Host h76: h76-eth0:10.0.0.76 pid=1625>
<Host h44: h44-eth0:10.0.0.44 pid=1627>
<Host h50: h50-eth0:10.0.0.50 pid=1629>
<Host h84: h84-eth0:10.0.0.84 pid=1631>
<Host h68: h68-eth0:10.0.0.68 pid=1633>
<Host h8: h8-eth0:10.0.0.8 pid=1635>
<Host h16: h16-eth0:10.0.0.16 pid=1637>
<Host h100: h100-eth0:10.0.0.100 pid=1639>
<Host h66: h66-eth0:10.0.0.66 pid=1641>
<Host h35: h35-eth0:10.0.0.35 pid=1643>
<Host h30: h30-eth0:10.0.0.30 pid=1645>
<Host h57: h57-eth0:10.0.0.57 pid=1647>
<Host h61: h61-eth0:10.0.0.61 pid=1649>
<Host h25: h25-eth0:10.0.0.25 pid=1651>
<Host h99: h99-eth0:10.0.0.99 pid=1653>
<Host h93: h93-eth0:10.0.0.93 pid=1655>
<Host h80: h80-eth0:10.0.0.80 pid=1657>
<Host h86: h86-eth0:10.0.0.86 pid=1659>
<Host h37: h37-eth0:10.0.0.37 pid=1661>
<Host h83: h83-eth0:10.0.0.83 pid=1663>
<Host h26: h26-eth0:10.0.0.26 pid=1665>
<Host h79: h79-eth0:10.0.0.79 pid=1667>
<Host h67: h67-eth0:10.0.0.67 pid=1669>
<Host h40: h40-eth0:10.0.0.40 pid=1671>
<Host h36: h36-eth0:10.0.0.36 pid=1673>
<Host h17: h17-eth0:10.0.0.17 pid=1675>
<Host h87: h87-eth0:10.0.0.87 pid=1677>
<Host h9: h9-eth0:10.0.0.9 pid=1679>
<Host h19: h19-eth0:10.0.0.19 pid=1681>
<Host h42: h42-eth0:10.0.0.42 pid=1683>
<Host h10: h10-eth0:10.0.0.10 pid=1685>
<Host h97: h97-eth0:10.0.0.97 pid=1687>
<Host h58: h58-eth0:10.0.0.58 pid=1689>
<Host h95: h95-eth0:10.0.0.95 pid=1691>
<Host h90: h90-eth0:10.0.0.90 pid=1693>
<Host h62: h62-eth0:10.0.0.62 pid=1695>
<Host h63: h63-eth0:10.0.0.63 pid=1697>
<Host h11: h11-eth0:10.0.0.11 pid=1699>
<Host h18: h18-eth0:10.0.0.18 pid=1701>
<Host h69: h69-eth0:10.0.0.69 pid=1703>
<Host h15: h15-eth0:10.0.0.15 pid=1705>
<Host h92: h92-eth0:10.0.0.92 pid=1707>
<Host h1: h1-eth0:10.0.0.1 pid=1709>
<Host h59: h59-eth0:10.0.0.59 pid=1711>
<Host h96: h96-eth0:10.0.0.96 pid=1713>
<Host h38: h38-eth0:10.0.0.38 pid=1715>

<Host h91: h91-eth0:10.0.0.91 pid=1717>
<Host h74: h74-eth0:10.0.0.74 pid=1719>
<Host h12: h12-eth0:10.0.0.12 pid=1721>
<Host h24: h24-eth0:10.0.0.24 pid=1723>
<Host h22: h22-eth0:10.0.0.22 pid=1725>
<Host h45: h45-eth0:10.0.0.45 pid=1727>
<Host h55: h55-eth0:10.0.0.55 pid=1729>
<Host h89: h89-eth0:10.0.0.89 pid=1731>
<Host h71: h71-eth0:10.0.0.71 pid=1733>
<Host h53: h53-eth0:10.0.0.53 pid=1735>
<Host h20: h20-eth0:10.0.0.20 pid=1737>
<Host h73: h73-eth0:10.0.0.73 pid=1739>
<Host h98: h98-eth0:10.0.0.98 pid=1741>
<Host h13: h13-eth0:10.0.0.13 pid=1743>
<Host h77: h77-eth0:10.0.0.77 pid=1745>
<Host h85: h85-eth0:10.0.0.85 pid=1747>
<Host h72: h72-eth0:10.0.0.72 pid=1749>
<Host h14: h14-eth0:10.0.0.14 pid=1751>
<Host h43: h43-eth0:10.0.0.43 pid=1753>
<Host h41: h41-eth0:10.0.0.41 pid=1755>
<Host h46: h46-eth0:10.0.0.46 pid=1757>
<Host h51: h51-eth0:10.0.0.51 pid=1759>
<Host h23: h23-eth0:10.0.0.23 pid=1761>
<Host h65: h65-eth0:10.0.0.65 pid=1763>
<Host h28: h28-eth0:10.0.0.28 pid=1765>
<Host h48: h48-eth0:10.0.0.48 pid=1767>
<Host h29: h29-eth0:10.0.0.29 pid=1769>
<Host h56: h56-eth0:10.0.0.56 pid=1771>
<Host h54: h54-eth0:10.0.0.54 pid=1773>
<Host h82: h82-eth0:10.0.0.82 pid=1775>
<Host h39: h39-eth0:10.0.0.39 pid=1777>
<Host h78: h78-eth0:10.0.0.78 pid=1779>
<Host h21: h21-eth0:10.0.0.21 pid=1781>
<Host h3: h3-eth0:10.0.0.3 pid=1783>
<Host h49: h49-eth0:10.0.0.49 pid=1785>
<Host h31: h31-eth0:10.0.0.31 pid=1787>
<Host h4: h4-eth0:10.0.0.4 pid=1789>
<Host h47: h47-eth0:10.0.0.47 pid=1791>
<Host h34: h34-eth0:10.0.0.34 pid=1793>
<Host h81: h81-eth0:10.0.0.81 pid=1795>
<Host h5: h5-eth0:10.0.0.5 pid=1797>
<Host h88: h88-eth0:10.0.0.88 pid=1799>
<Host h52: h52-eth0:10.0.0.52 pid=1801>
<Host h32: h32-eth0:10.0.0.32 pid=1803>
<Host h64: h64-eth0:10.0.0.64 pid=1805>
<Host h2: h2-eth0:10.0.0.2 pid=1807>
<Host h33: h33-eth0:10.0.0.33 pid=1809>
<Host h70: h70-eth0:10.0.0.70 pid=1811>

<Host h6: h6-eth0:10.0.0.6 pid=1813>
<Host h94: h94-eth0:10.0.0.94 pid=1815>
<Host h27: h27-eth0:10.0.0.27 pid=1817>
<Host h75: h75-eth0:10.0.0.75 pid=1819>
<Host h60: h60-eth0:10.0.0.60 pid=1821>
<Host h7: h7-eth0:10.0.0.7 pid=1823>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None,s5-eth3:None,s5-eth4:None,s5-eth5:None,s5-eth6:None,s5-eth7:None,s5-eth8:None,s5-eth9:None,s5-eth10:None,s5-eth11:None,s5-eth12:None,s5-eth13:None,s5-eth14:None,s5-eth15:None,s5-eth16:None,s5-eth17:None,s5-eth18:None,s5-eth19:None,s5-eth20:None,s5-eth21:None,s5-eth22:None pid=1608>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,s2-eth5:None,s2-eth6:None,s2-eth7:None,s2-eth8:None,s2-eth9:None,s2-eth10:None,s2-eth11:None,s2-eth12:None,s2-eth13:None,s2-eth14:None,s2-eth15:None,s2-eth16:None,s2-eth17:None,s2-eth18:None,s2-eth19:None,s2-eth20:None,s2-eth21:None,s2-eth22:None pid=1611>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-eth4:None,s4-eth5:None,s4-eth6:None,s4-eth7:None,s4-eth8:None,s4-eth9:None,s4-eth10:None,s4-eth11:None,s4-eth12:None,s4-eth13:None,s4-eth14:None,s4-eth15:None,s4-eth16:None,s4-eth17:None,s4-eth18:None,s4-eth19:None,s4-eth20:None,s4-eth21:None,s4-eth22:None pid=1614>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-eth6:None,s1-eth7:None,s1-eth8:None,s1-eth9:None,s1-eth10:None,s1-eth11:None,s1-eth12:None,s1-eth13:None,s1-eth14:None,s1-eth15:None,s1-eth16:None,s1-eth17:None,s1-eth18:None,s1-eth19:None,s1-eth20:None,s1-eth21:None,s1-eth22:None pid=1617>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None,s3-eth4:None,s3-eth5:None,s3-eth6:None,s3-eth7:None,s3-eth8:None,s3-eth9:None,s3-eth10:None,s3-eth11:None,s3-eth12:None,s3-eth13:None,s3-eth14:None,s3-eth15:None,s3-eth16:None,s3-eth17:None,s3-eth18:None,s3-eth19:None,s3-eth20:None,s3-eth21:None,s3-eth22:None pid=1620>
<Controller c0: 127.0.0.1:6633 pid=1600>

APÊNDICE II

<Host h52: h52-eth0:10.0.0.52 pid=13966>
<Host h24: h24-eth0:10.0.0.24 pid=13968>
<Host h90: h90-eth0:10.0.0.90 pid=13970>
<Host h4: h4-eth0:10.0.0.4 pid=13972>
<Host h61: h61-eth0:10.0.0.61 pid=13974>
<Host h14: h14-eth0:10.0.0.14 pid=13976>
<Host h96: h96-eth0:10.0.0.96 pid=13978>
<Host h18: h18-eth0:10.0.0.18 pid=13980>
<Host h36: h36-eth0:10.0.0.36 pid=13982>
<Host h64: h64-eth0:10.0.0.64 pid=13984>
<Host h78: h78-eth0:10.0.0.78 pid=13986>
<Host h6: h6-eth0:10.0.0.6 pid=13988>
<Host h67: h67-eth0:10.0.0.67 pid=13990>
<Host h73: h73-eth0:10.0.0.73 pid=13992>
<Host h51: h51-eth0:10.0.0.51 pid=13994>
<Host h84: h84-eth0:10.0.0.84 pid=13996>
<Host h82: h82-eth0:10.0.0.82 pid=13998>
<Host h35: h35-eth0:10.0.0.35 pid=14000>
<Host h53: h53-eth0:10.0.0.53 pid=14002>
<Host h68: h68-eth0:10.0.0.68 pid=14004>
<Host h27: h27-eth0:10.0.0.27 pid=14006>
<Host h9: h9-eth0:10.0.0.9 pid=14008>
<Host h63: h63-eth0:10.0.0.63 pid=14010>
<Host h12: h12-eth0:10.0.0.12 pid=14012>
<Host h5: h5-eth0:10.0.0.5 pid=14014>
<Host h28: h28-eth0:10.0.0.28 pid=14016>
<Host h80: h80-eth0:10.0.0.80 pid=14018>
<Host h17: h17-eth0:10.0.0.17 pid=14020>
<Host h89: h89-eth0:10.0.0.89 pid=14022>
<Host h7: h7-eth0:10.0.0.7 pid=14024>
<Host h58: h58-eth0:10.0.0.58 pid=14026>
<Host h37: h37-eth0:10.0.0.37 pid=14028>
<Host h72: h72-eth0:10.0.0.72 pid=14030>
<Host h98: h98-eth0:10.0.0.98 pid=14032>
<Host h55: h55-eth0:10.0.0.55 pid=14034>
<Host h54: h54-eth0:10.0.0.54 pid=14036>
<Host h3: h3-eth0:10.0.0.3 pid=14038>
<Host h60: h60-eth0:10.0.0.60 pid=14040>
<Host h34: h34-eth0:10.0.0.34 pid=14042>
<Host h32: h32-eth0:10.0.0.32 pid=14044>
<Host h71: h71-eth0:10.0.0.71 pid=14046>
<Host h39: h39-eth0:10.0.0.39 pid=14048>
<Host h38: h38-eth0:10.0.0.38 pid=14050>
<Host h29: h29-eth0:10.0.0.29 pid=14052>
<Host h81: h81-eth0:10.0.0.81 pid=14054>

<Host h99: h99-eth0:10.0.0.99 pid=14056>
<Host h15: h15-eth0:10.0.0.15 pid=14058>
<Host h19: h19-eth0:10.0.0.19 pid=14060>
<Host h97: h97-eth0:10.0.0.97 pid=14062>
<Host h57: h57-eth0:10.0.0.57 pid=14064>
<Host h46: h46-eth0:10.0.0.46 pid=14066>
<Host h88: h88-eth0:10.0.0.88 pid=14068>
<Host h75: h75-eth0:10.0.0.75 pid=14070>
<Host h44: h44-eth0:10.0.0.44 pid=14072>
<Host h42: h42-eth0:10.0.0.42 pid=14074>
<Host h50: h50-eth0:10.0.0.50 pid=14076>
<Host h83: h83-eth0:10.0.0.83 pid=14078>
<Host h23: h23-eth0:10.0.0.23 pid=14080>
<Host h95: h95-eth0:10.0.0.95 pid=14082>
<Host h41: h41-eth0:10.0.0.41 pid=14084>
<Host h40: h40-eth0:10.0.0.40 pid=14086>
<Host h69: h69-eth0:10.0.0.69 pid=14088>
<Host h77: h77-eth0:10.0.0.77 pid=14090>
<Host h70: h70-eth0:10.0.0.70 pid=14092>
<Host h48: h48-eth0:10.0.0.48 pid=14094>
<Host h91: h91-eth0:10.0.0.91 pid=14096>
<Host h43: h43-eth0:10.0.0.43 pid=14098>
<Host h65: h65-eth0:10.0.0.65 pid=14100>
<Host h21: h21-eth0:10.0.0.21 pid=14102>
<Host h26: h26-eth0:10.0.0.26 pid=14104>
<Host h22: h22-eth0:10.0.0.22 pid=14106>
<Host h10: h10-eth0:10.0.0.10 pid=14108>
<Host h16: h16-eth0:10.0.0.16 pid=14110>
<Host h79: h79-eth0:10.0.0.79 pid=14112>
<Host h94: h94-eth0:10.0.0.94 pid=14114>
<Host h47: h47-eth0:10.0.0.47 pid=14116>
<Host h49: h49-eth0:10.0.0.49 pid=14118>
<Host h93: h93-eth0:10.0.0.93 pid=14120>
<Host h20: h20-eth0:10.0.0.20 pid=14122>
<Host h8: h8-eth0:10.0.0.8 pid=14124>
<Host h74: h74-eth0:10.0.0.74 pid=14126>
<Host h25: h25-eth0:10.0.0.25 pid=14128>
<Host h56: h56-eth0:10.0.0.56 pid=14130>
<Host h85: h85-eth0:10.0.0.85 pid=14132>
<Host h66: h66-eth0:10.0.0.66 pid=14134>
<Host h86: h86-eth0:10.0.0.86 pid=14136>
<Host h76: h76-eth0:10.0.0.76 pid=14138>
<Host h11: h11-eth0:10.0.0.11 pid=14140>
<Host h92: h92-eth0:10.0.0.92 pid=14142>
<Host h45: h45-eth0:10.0.0.45 pid=14144>
<Host h100: h100-eth0:10.0.0.100 pid=14146>
<Host h33: h33-eth0:10.0.0.33 pid=14148>
<Host h30: h30-eth0:10.0.0.30 pid=14150>

<Host h87: h87-eth0:10.0.0.87 pid=14152>
<Host h13: h13-eth0:10.0.0.13 pid=14154>
<Host h62: h62-eth0:10.0.0.62 pid=14156>
<Host h1: h1-eth0:10.0.0.1 pid=14158>
<Host h59: h59-eth0:10.0.0.59 pid=14160>
<Host h31: h31-eth0:10.0.0.31 pid=14162>
<Host h2: h2-eth0:10.0.0.2 pid=14164>
<OVSSwitch s6: lo:127.0.0.1,s6-eth1:None,s6-eth2:None,s6-eth3:None,s6-eth4:None,s6-eth5:None,s6-eth6:None,s6-eth7:None,s6-eth8:None,s6-eth9:None,s6-eth10:None,s6-eth11:None,s6-eth12:None,s6-eth13:None,s6-eth14:None pid=13925>
<OVSSwitch s9: lo:127.0.0.1,s9-eth1:None,s9-eth2:None,s9-eth3:None,s9-eth4:None,s9-eth5:None,s9-eth6:None,s9-eth7:None,s9-eth8:None,s9-eth9:None,s9-eth10:None,s9-eth11:None,s9-eth12:None,s9-eth13:None pid=13928>
<OVSSwitch s8: lo:127.0.0.1,s8-eth1:None,s8-eth2:None,s8-eth3:None,s8-eth4:None,s8-eth5:None,s8-eth6:None,s8-eth7:None,s8-eth8:None,s8-eth9:None,s8-eth10:None,s8-eth11:None,s8-eth12:None,s8-eth13:None pid=13931>
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None,s7-eth4:None,s7-eth5:None,s7-eth6:None,s7-eth7:None,s7-eth8:None,s7-eth9:None,s7-eth10:None,s7-eth11:None,s7-eth12:None,s7-eth13:None pid=13934>
<OVSSwitch s12: lo:127.0.0.1,s12-eth1:None,s12-eth2:None,s12-eth3:None,s12-eth4:None,s12-eth5:None,s12-eth6:None,s12-eth7:None,s12-eth8:None,s12-eth9:None,s12-eth10:None,s12-eth11:None pid=13937>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None,s3-eth4:None pid=13940>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None pid=13943>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=13946>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None,s5-eth3:None,s5-eth4:None,s5-eth5:None,s5-eth6:None,s5-eth7:None,s5-eth8:None,s5-eth9:None,s5-eth10:None,s5-eth11:None,s5-eth12:None pid=13949>
<OVSSwitch s13: lo:127.0.0.1,s13-eth1:None,s13-eth2:None,s13-eth3:None,s13-eth4:None,s13-eth5:None,s13-eth6:None,s13-eth7:None,s13-eth8:None,s13-eth9:None,s13-eth10:None,s13-eth11:None pid=13952>
<OVSSwitch s10: lo:127.0.0.1,s10-eth1:None,s10-eth2:None,s10-eth3:None,s10-eth4:None,s10-eth5:None,s10-eth6:None,s10-eth7:None,s10-eth8:None,s10-eth9:None,s10-eth10:None,s10-eth11:None pid=13955>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-eth4:None pid=13958>
<OVSSwitch s11: lo:127.0.0.1,s11-eth1:None,s11-eth2:None,s11-eth3:None,s11-eth4:None,s11-eth5:None,s11-eth6:None,s11-eth7:None,s11-eth8:None,s11-eth9:None,s11-eth10:None,s11-eth11:None pid=13961>
<RemoteController c0: 127.0.0.1:6633 pid=13918>

APÊNDICE III

<Host h34: h34-eth0:10.0.0.34 pid=6546>
<Host h15: h15-eth0:10.0.0.15 pid=6548>
<Host h52: h52-eth0:10.0.0.52 pid=6550>
<Host h100: h100-eth0:10.0.0.100 pid=6552>
<Host h48: h48-eth0:10.0.0.48 pid=6554>
<Host h97: h97-eth0:10.0.0.97 pid=6556>
<Host h12: h12-eth0:10.0.0.12 pid=6558>
<Host h78: h78-eth0:10.0.0.78 pid=6560>
<Host h98: h98-eth0:10.0.0.98 pid=6562>
<Host h66: h66-eth0:10.0.0.66 pid=6564>
<Host h60: h60-eth0:10.0.0.60 pid=6566>
<Host h86: h86-eth0:10.0.0.86 pid=6568>
<Host h55: h55-eth0:10.0.0.55 pid=6570>
<Host h30: h30-eth0:10.0.0.30 pid=6572>
<Host h33: h33-eth0:10.0.0.33 pid=6574>
<Host h70: h70-eth0:10.0.0.70 pid=6576>
<Host h53: h53-eth0:10.0.0.53 pid=6578>
<Host h22: h22-eth0:10.0.0.22 pid=6580>
<Host h76: h76-eth0:10.0.0.76 pid=6582>
<Host h91: h91-eth0:10.0.0.91 pid=6584>
<Host h37: h37-eth0:10.0.0.37 pid=6586>
<Host h54: h54-eth0:10.0.0.54 pid=6588>
<Host h18: h18-eth0:10.0.0.18 pid=6590>
<Host h21: h21-eth0:10.0.0.21 pid=6592>
<Host h32: h32-eth0:10.0.0.32 pid=6594>
<Host h25: h25-eth0:10.0.0.25 pid=6596>
<Host h20: h20-eth0:10.0.0.20 pid=6598>
<Host h31: h31-eth0:10.0.0.31 pid=6600>
<Host h96: h96-eth0:10.0.0.96 pid=6602>
<Host h8: h8-eth0:10.0.0.8 pid=6604>
<Host h9: h9-eth0:10.0.0.9 pid=6606>
<Host h47: h47-eth0:10.0.0.47 pid=6608>
<Host h35: h35-eth0:10.0.0.35 pid=6610>
<Host h42: h42-eth0:10.0.0.42 pid=6612>
<Host h74: h74-eth0:10.0.0.74 pid=6614>
<Host h68: h68-eth0:10.0.0.68 pid=6616>
<Host h58: h58-eth0:10.0.0.58 pid=6618>
<Host h26: h26-eth0:10.0.0.26 pid=6620>
<Host h64: h64-eth0:10.0.0.64 pid=6622>
<Host h62: h62-eth0:10.0.0.62 pid=6624>
<Host h63: h63-eth0:10.0.0.63 pid=6626>
<Host h67: h67-eth0:10.0.0.67 pid=6628>
<Host h82: h82-eth0:10.0.0.82 pid=6630>
<Host h69: h69-eth0:10.0.0.69 pid=6632>
<Host h23: h23-eth0:10.0.0.23 pid=6634>

<Host h56: h56-eth0:10.0.0.56 pid=6636>
<Host h10: h10-eth0:10.0.0.10 pid=6638>
<Host h90: h90-eth0:10.0.0.90 pid=6640>
<Host h80: h80-eth0:10.0.0.80 pid=6642>
<Host h71: h71-eth0:10.0.0.71 pid=6644>
<Host h92: h92-eth0:10.0.0.92 pid=6646>
<Host h3: h3-eth0:10.0.0.3 pid=6648>
<Host h94: h94-eth0:10.0.0.94 pid=6650>
<Host h49: h49-eth0:10.0.0.49 pid=6652>
<Host h46: h46-eth0:10.0.0.46 pid=6654>
<Host h81: h81-eth0:10.0.0.81 pid=6656>
<Host h11: h11-eth0:10.0.0.11 pid=6658>
<Host h84: h84-eth0:10.0.0.84 pid=6660>
<Host h44: h44-eth0:10.0.0.44 pid=6662>
<Host h59: h59-eth0:10.0.0.59 pid=6664>
<Host h50: h50-eth0:10.0.0.50 pid=6666>
<Host h99: h99-eth0:10.0.0.99 pid=6668>
<Host h17: h17-eth0:10.0.0.17 pid=6670>
<Host h4: h4-eth0:10.0.0.4 pid=6672>
<Host h77: h77-eth0:10.0.0.77 pid=6674>
<Host h16: h16-eth0:10.0.0.16 pid=6676>
<Host h83: h83-eth0:10.0.0.83 pid=6678>
<Host h73: h73-eth0:10.0.0.73 pid=6680>
<Host h88: h88-eth0:10.0.0.88 pid=6682>
<Host h36: h36-eth0:10.0.0.36 pid=6684>
<Host h95: h95-eth0:10.0.0.95 pid=6686>
<Host h6: h6-eth0:10.0.0.6 pid=6688>
<Host h40: h40-eth0:10.0.0.40 pid=6690>
<Host h19: h19-eth0:10.0.0.19 pid=6692>
<Host h5: h5-eth0:10.0.0.5 pid=6694>
<Host h1: h1-eth0:10.0.0.1 pid=6696>
<Host h27: h27-eth0:10.0.0.27 pid=6698>
<Host h24: h24-eth0:10.0.0.24 pid=6700>
<Host h13: h13-eth0:10.0.0.13 pid=6702>
<Host h28: h28-eth0:10.0.0.28 pid=6704>
<Host h51: h51-eth0:10.0.0.51 pid=6706>
<Host h65: h65-eth0:10.0.0.65 pid=6708>
<Host h61: h61-eth0:10.0.0.61 pid=6710>
<Host h45: h45-eth0:10.0.0.45 pid=6712>
<Host h85: h85-eth0:10.0.0.85 pid=6714>
<Host h38: h38-eth0:10.0.0.38 pid=6716>
<Host h93: h93-eth0:10.0.0.93 pid=6718>
<Host h72: h72-eth0:10.0.0.72 pid=6720>
<Host h29: h29-eth0:10.0.0.29 pid=6722>
<Host h89: h89-eth0:10.0.0.89 pid=6724>
<Host h41: h41-eth0:10.0.0.41 pid=6726>
<Host h39: h39-eth0:10.0.0.39 pid=6728>
<Host h87: h87-eth0:10.0.0.87 pid=6730>

<Host h7: h7-eth0:10.0.0.7 pid=6732>
<Host h14: h14-eth0:10.0.0.14 pid=6734>
<Host h79: h79-eth0:10.0.0.79 pid=6736>
<Host h57: h57-eth0:10.0.0.57 pid=6738>
<Host h43: h43-eth0:10.0.0.43 pid=6740>
<Host h2: h2-eth0:10.0.0.2 pid=6742>
<Host h75: h75-eth0:10.0.0.75 pid=6744>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None,s2-eth4:None,s2-eth5:None,s2-eth6:None,s2-eth7:None,s2-eth8:None,s2-eth9:None,s2-eth10:None,s2-eth11:None,s2-eth12:None,s2-eth13:None,s2-eth14:None,s2-eth15:None,s2-eth16:None,s2-eth17:None,s2-eth18:None,s2-eth19:None,s2-eth20:None,s2-eth21:None,s2-eth22:None,s2-eth23:None,s2-eth24:None pid=6529>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None,s5-eth3:None,s5-eth4:None,s5-eth5:None,s5-eth6:None,s5-eth7:None,s5-eth8:None,s5-eth9:None,s5-eth10:None,s5-eth11:None,s5-eth12:None,s5-eth13:None,s5-eth14:None,s5-eth15:None,s5-eth16:None,s5-eth17:None,s5-eth18:None,s5-eth19:None,s5-eth20:None,s5-eth21:None,s5-eth22:None,s5-eth23:None,s5-eth24:None pid=6532>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None,s3-eth4:None,s3-eth5:None,s3-eth6:None,s3-eth7:None,s3-eth8:None,s3-eth9:None,s3-eth10:None,s3-eth11:None,s3-eth12:None,s3-eth13:None,s3-eth14:None,s3-eth15:None,s3-eth16:None,s3-eth17:None,s3-eth18:None,s3-eth19:None,s3-eth20:None,s3-eth21:None,s3-eth22:None,s3-eth23:None,s3-eth24:None pid=6535>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-eth6:None,s1-eth7:None,s1-eth8:None,s1-eth9:None,s1-eth10:None,s1-eth11:None,s1-eth12:None,s1-eth13:None,s1-eth14:None,s1-eth15:None,s1-eth16:None,s1-eth17:None,s1-eth18:None,s1-eth19:None,s1-eth20:None,s1-eth21:None,s1-eth22:None,s1-eth23:None,s1-eth24:None pid=6538>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None,s4-eth4:None,s4-eth5:None,s4-eth6:None,s4-eth7:None,s4-eth8:None,s4-eth9:None,s4-eth10:None,s4-eth11:None,s4-eth12:None,s4-eth13:None,s4-eth14:None,s4-eth15:None,s4-eth16:None,s4-eth17:None,s4-eth18:None,s4-eth19:None,s4-eth20:None,s4-eth21:None,s4-eth22:None,s4-eth23:None,s4-eth24:None pid=6541>
<RemoteController c0: 127.0.0.1:6633 pid=6522>