# Capítulo 1

# Ameaças de Segurança, Defesas e Análise de Dados em IoT Baseada em SDN

**Nelson G. Prates Jr.**[1], **Mateus Pelloso**[1,3], **Ricardo T. Macedo**[1,2], **Michele Nogueira**[1]

Centro de Ciência de Segurança Computacional e
Universidade Federal do Paraná[1]
Universidade Federal de Santa Maria[2]
Instituto Federal Catarinense[3]

***Abstract***

*Software Defined Networks (SDN) have been proposed to solve problems related to scalability, management and mobility in the traditional network model. Hence, emerging network paradigms, –Internet of Things (IoT) and Internet of Everything (IoE),– have integrated SDN into their infrastructure for benefiting from its advantages. However, these networks are susceptible to threats against the authenticity, confidentiality, integrity and availability of data and/or network services. This chapter presents the main threats and defenses in SDN-based IoT, following a theoretical and a practical perspectives. The theoretical perspective introduces the main concepts, threats and countermeasures of conventional IoT and SDN-based IoT. The practical perspective addresses a study case related to Denial of Service (DoS) attacks, involving simulation and network traffic analysis based on the extraction and analysis of data by statistical learning methods. This chapter offers an opportunity to identify the main research challenges related to SDN-based IoT, and to the analysis of data focused on network security.*

***Resumo***

*As Redes Definidas por Software (SDN) foram propostas para solucionar problemas relacionados à escalabilidade, gerenciamento e mobilidade do modelo de redes tradicional. Desta forma, paradigmas emergentes de redes, – Internet das Coisas (IoT) e Internet de Todas as Coisas (IoE), – vêm integrando a SDN em suas infraestruturas para usufruir dos seus benefícios. No entanto, estas redes são suscetíveis a ameaças contra a confidencialidade, integridade e disponibilidade dos dados e/ou serviços da rede. Este capítulo apresenta as principais ameaças e defesas em redes IoT baseadas em SDN, seguindo uma perspectiva teórica e uma prática. A parte teórica introduz os principais conceitos, ameaças e contramedidas no contexto de redes IoT convencionais e IoT baseadas em SDN. A*

*perspectiva prática aborda um estudo de caso sobre ataques de negação de serviço, envolvendo a simulação e a análise do tráfego de rede através da extração de dados usando métodos de aprendizagem estatística. Este capítulo oferece a identificação dos principais desafios de pesquisa na segurança da IoT e da IoT baseada em SDN, e na análise de dados focando na segurança de redes.*

## 1.1. Introdução

O paradigma de Internet de Todas as Coisas (*Internet of Everything* – IoE) vêm expandindo o conceito de Internet das Coisas (*Internet of Things* - IoT) visando proporcionar serviços ainda mais relevantes para as pessoas. A IoT, muitas vezes considerada como um sinônimo de IoE, representa uma vasta gama de dispositivos (coisas) capazes de se conectar à Internet para prover serviços inteligentes por meio da troca de uma grande quantidade de dados em tempo real [Atzori et al. 2010]. Estas coisas podem ser, por exemplo, computadores de bordo de um veículo, *smartphones*, refrigeradores ou fontes de energia elétrica. No entanto, a IoE estende o conceito de IoT ao considerar a associação de pessoas, processos, dados e coisas [Schatten et al. 2016b]. Por meio desta associação, a IoE explora a íntima relação entre estas entidades, podendo prover serviços ainda mais relevantes para as pessoas e gerando oportunidades econômicas sem precedentes para empresas, indivíduos e países [Miraz et al. 2015]. Entretanto, o advento da IoE está intrinsecamente atrelado com a solução de questões abordadas pela IoT [Iannacci 2018], sendo um dos principais pontos o gerenciamento do número massivo de coisas e sua integração com Internet atual [Lamaazi et al. 2014].

A integração da IoT com as Redes Definidas por Software (do inglês, *Software-Defined Network* - SDN) persegue uma forma de resolver o problema de gerenciamento das coisas na IoT. A SDN propõe um modelo de rede que desacopla o plano de controle (papel do controlador SDN) dos comutadores/dispositivos de encaminhamento (*switches*). Ao integrar a IoT com a tecnologia SDN, os procedimentos de gerenciamento dos dispositivos da IoT são centralizados no plano de controle da rede SDN, proporcionando como principal vantagem uma simplificação significativa do gerenciamento da rede [Bera et al. 2017]. A Figura 1.1 ilustra esta integração. As aplicações da IoT, tais como identificação, sensoriamento, comunicação e computação, são consideradas para a rede SDN como logicamente localizadas acima do limite norte da interface do controlador SDN. Os *switches* atuam na interface sul apenas encaminhando os dados destas aplicações conforme as regras instaladas pelo controlador [Bizanis and Kuipers 2016]. Deste modo, as funções de gerenciamento da rede são concentradas no controlador, permitindo a programação das regras da rede em tempo real e possibilitando rápidas adaptações da topologia da rede diante da dinamicidade dos dispositivos da IoT.

Todavia, em paralelo a estas características e avanços, diferentes relatórios de incidentes de segurança vêm apresentando um crescimento significativo do número de ataques e ameaças contra as redes de computadores tradicionais no cenário nacional e global. As estatísticas disponibilizadas pelo CERT.br, por exemplo, mostram que em 2015 foram relatados cerca de 722 mil incidentes de segurança em 2015, aumentando para 647 mil em 2016 e atingindo a casa de 833 mil em 2017 [CERT.br 2018]. Considerando a escala global, existem estatísticas ainda mais alarmantes neste mesmo período. Em 2015, a Cisco reportou que cerca de 43% dos setores públicos falhavam em prestar serviços

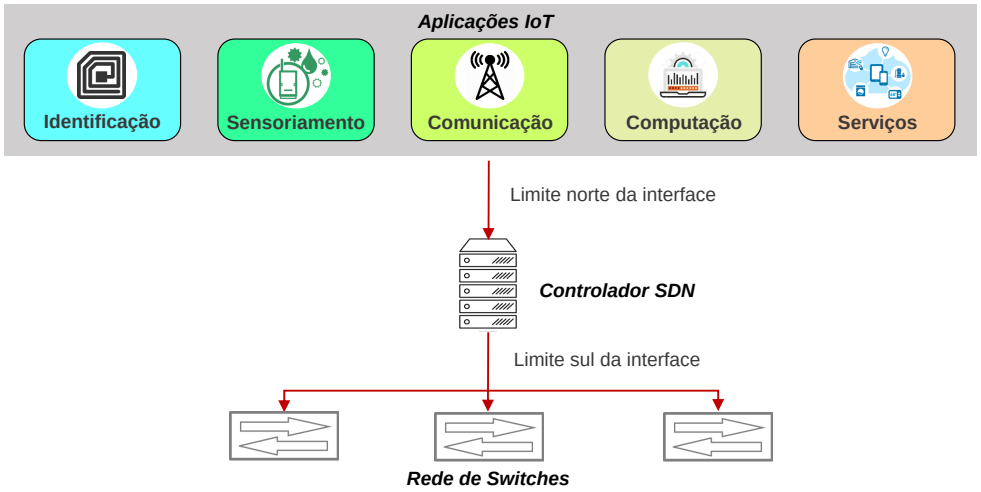


Figura 1.1: Rede IoT Baseada em SDN

de segurança para suas infraestruturas [Cisco 2018]. Em 2016, a Akamai confirmou 19 mega ataques, sendo dois deles os maiores ataques de negação de serviços já registrados, alcançando 623 Gbps e 555 Gbps, respectivamente [Akamai 2016]. Em 2017, os ataques envolvendo *ransomwares* cresceram 36% e 6,5% das pessoas foram vítimas de fraudes de identidades, resultando em prejuízos de 16 bilhões de dólares [Mason 2018]. Em 2018, este cenário acrescentou uma nova característica, pois foi registrado um aumento de 600% de ataques envolvendo dispositivos da IoT [Symatec 2018].

Em decorrência da presença de ameaças iminentes contra as redes de computadores tradicionais, cresce a preocupação com a segurança na integração entre IoT e SDN [Flauzac et al. 2015]. Os principais desafios de segurança para o advento da IoT abrangem questões de privacidade, autorização, verificação, controle de acesso, configuração de sistemas, armazenamento de informações e aspectos de gerenciamento [Alaba et al. 2017]. Em relação às SDNs, pesquisas acadêmicas revelam que as mesmas são suscetíveis a ameaças contra a autenticidade, a confidencialidade, a integridade e a disponibilidade dos dados e/ou componentes da rede [Scott-Hayward et al. 2016]. A captura e a análise dos pacotes do plano de controle das redes SDN podem fornecer informações privilegiadas ao atacante sobre as configurações dos *switches*, violando o princípio de confidencialidade. Ao obter conhecimento sobre a rede, o atacante pode falsificar pacotes para gerar um ataque de negação de serviço no controlador, ferindo os princípios de integridade dos pacotes e de disponibilidade do controlador, comprometendo consequentemente as aplicações da IoT.

Este capítulo apresenta os conceitos, a história, as principais ameaças e as defesas existentes para IoT tradicional e para as redes IoT baseadas em SDN, bem como as principais vantagens na relação entre estes dois modelos. O capítulo segue uma abordagem teórico/prática. A parte teórica introduz os fundamentos sobre a IoT e as SDNs, o histórico, as principais ameaças e as defesas. Também, a parte teórica apresenta como estes modelos se relacionam e porque essa relação traz vantagens significativas em termos de gerência de rede. Para cada uma dessas tecnologias existem uma diversidade de padronizações, neste documento optamos por seguir as tecnologias normatizadas pelas instituições e grupos de padronização pesquisas mais influentes encontrados na literatura, tais como a ***I**nternet **E**ngineering **T**ask **F**orce* (**IETF**), a ***I**nternational **O**rganization for*

***S**tandardization* (**ISO**) e o ***I**nstitute of **E**lectrical and **E**lectronics **E**ngineers* (**IEEE**). A parte de perspectiva prática descreve um estudo de caso da análise do comportamento de um ataque de negação de serviço, detalhando as ferramentas utilizadas para a simulação de uma rede sob ataque e a análise do seu tráfego. Neste capítulo, são descritas didaticamente as experiências do Centro de Ciência de Segurança Computacional (CCSC)[1] em extrair dados da rede, analisar esses dados através de métodos de aprendizado estatístico e concluir sobre os comportamentos da rede e de ataques.

O capítulo tem como objetivo incentivar a comunidade a desenvolver soluções de segurança para a IoT, destacando como as SDNs podem apoiar neste objetivo. Através deste capítulo, esperamos que os leitores possam compreender os principais conceitos e como relacionar a IoT e as SDNs, e identificar desafios de pesquisa em aberto quanto às ameaças e contramedidas existentes. Além disto, os leitores conhecerão um estudo de caso sobre a análise do comportamento da rede frente a um ataque de negação de serviço e terão a oportunidade de aprender sobre o uso de ferramentas para extração e análise de dados da rede, assim como métodos atuais de análise.

O restante do capítulo está organizado como segue. A Seção 1.2 introduz os principais conceitos e terminologias da IoT, sua arquitetura, a realação com a computação em nuvem e o conceito da IoE. A Seção 1.3 apresenta a história das SDNs, as principais definições e terminologias, a infraestrutura incluindo o protocolo *OpenFlow* e também como a IoT se relaciona com as SDNs para solucionar os desafios encontrados por estas redes. A Seção 1.4 descreve as principais ameaças encontradas nas SDNs, na IoT e IoT baseadas em SDN. A Seção 1.5 detalha as principais defesas. A Seção 1.6 finaliza o capítulo abordando uma perspectiva prática ao descrever a condução de uma simulação de uma rede IoT baseada em SDN sob ataque DDoS, mostrando a análise dos dados extraídos da rede em busca de padrões que revelem o ataque realizado.

## 1.2. Internet das Coisas e Internet de Todas as Coisas

A evolução das tecnologias de redes de sensores sem fio resultou na inclusão da IoT no cotidiano do cidadão moderno. Estudos realizados pela Cisco estimam que em 2021 haverá cerca de 8,3 bilhões de dispositivos pessoais portáteis conectados à rede mundial de computadores, gerando um tráfego de até 49 exabytes por mês [Cisco 2016]. Este novo modelo de rede conecta os mais variados dispositivos computacionais à Internet, exigindo redes com flexibilidade para acomodar um alto nível de escalabilidade e heterogeneidade. Além disso, esses dispositivos são distribuídos oferecendo diferentes tipos de aplicações em tempo real [Bera et al. 2017]. Estes aspectos somados à escalabilidade, heterogeneidade e aos serviços distribuídos geram grandes quantidades de dados e potencializam desafios relacionados a *Big Data* e dados em *Stream*. Esses desafios demandam serviços que atualmente vão além das conexões entre os dispositivos compondo a IoT, expandindo as possibilidades de desenvolvimento de soluções tanto para tecnologias nas camadas mais baixas da rede até a camada de aplicação, mas também soluções que considerem o comportamento das pessoas e dos processos envolvidos, ou seja, a IoE. Esta perspectiva tem aquecido o mercado, trazendo expectativas de investimentos em torno de US$ 1,7 trilhões em 2020 [Tayyaba et al. 2017].

[1]Centro de Ciência de Segurança Computacional (CCSC): Página Web `www.ccsc-research.org`

As principais soluções para os desafios da IoT estão fundadas em tecnologias como a computação em nuvem. O principal desafio de uma rede IoT consiste na limitação da capacidade dos recursos computacionais nos dispositivos de borda (dispositivos finais). A computação em nuvem trata os desafios relacionados ao *Big Data*, fornecendo recursos computacionais pela Internet e seguindo o modelo cliente/servidor. No entanto, o desafio de gerenciar a escalabilidade permanece. Para tratá-lo, uma tentativa consiste na proposição do conceito de computação em névoa (*fog computing*). Esta abordagem simplifica a disseminação dos dados e serviços da nuvem, aproximando-os da borda. Essas soluções requerem a definição e padronização de um conjunto de protocolos e tecnologias de rede, as quais podem aumentar a complexidade dessas redes dificultando a gerência da rede e agravando o problema de heterogeneidade. Esse fato gera uma demanda por sistemas de gerência igualmente escaláveis capazes de simplificar o ônus da manutenção da rede ao promover a heterogeneidade.

Esta seção detalha os principais conceitos da IoT, sua respectiva relação com a computação em nuvem e o advento da IoE. A Subseção 1.2.1 apresenta a arquitetura conceitual de rede para IoT e descreve como a IoT se conecta com a Internet. A Subseção 1.2.2 aborda a relação entre a computação em nuvem e IoT. A Subseção 1.2.3 introduz a relação entre IoT e IoE.

### 1.2.1. Arquitetura Conceitual da IoT

A IoT e a IoE suportam uma ampla variedade de aplicações e vêm sendo apoiadas pela evolução das tecnologias de rede, protocolos, meios de comunicação, dispositivos e serviços de rede. Esta evolução vem incentivando a proposição de diversas arquiteturas de redes IoT [Atzori et al. 2010, Khan et al. 2012, Bandyopadhyay and Sen 2011]. A arquitetura de rede conceitual mais popular está dividida em camadas de **aplicação**, **rede** e **percepção** [Palattella et al. 2013], como ilustra a Figura 1.2. A camada de aplicação utiliza as informações adquiridas e tratadas respectivamente pelas camadas de percepção e rede. Entretanto, a aplicação acaba por determinar como os dispositivos da IoT são organizados. A camada de rede realiza a comunicação dispositivo-a-dispositivo. A camada de percepção compreende dispositivos sensores responsáveis por coletar dados e por interagir com o ambiente físico. Esta organização em camadas suporta o desempenho de atividades direcionadas aos requisitos dos diferentes contextos, como casas inteligentes, cidades inteligentes e outros.

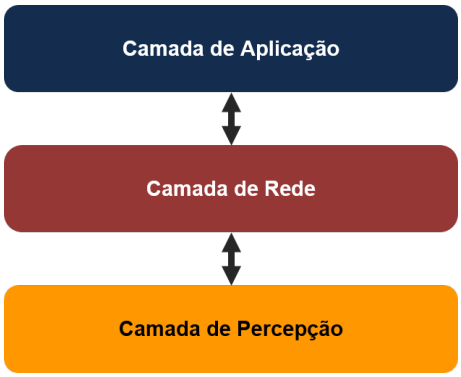


Figura 1.2: Arquitetura IoT [Palattella et al. 2013]

### Camada de Aplicação

A camada de aplicação na arquitetura conceitual da IoT, proposta por [Palattella et al. 2013], oferece um nível de abstração que engloba protocolos responsáveis por oferecer serviços aos usuários finais e uma interface entre a camada de aplicação e a camada de rede. Esta camada orienta como os dispositivos devem se comunicar para atender aos requisitos das aplicações suportadas pela IoT. Além disso, a camada de aplicação oferece serviços de gerenciamento da rede e de seus dispositivos. As funcionalidades de gerenciamento da rede assumem que a camada de aplicação está ciente do cenário e das aplicações apoiadas pela IoT. As aplicações em si e os cenários em que a IoT está inserida guiam quais tipos de sensores e atuadores são necessários para atender aos seus requisitos. A camada de aplicação, de posse dos dados coletados, podem utilizar técnicas de inteligência artificial e outras para analisar o estado da rede, tomar decisão e desempenhar atividades que cumpram os objetivos dos serviços oferecidos aos usuários.

Além disso, com base nos dados coletados, a camada de aplicação pode coordenar ações associadas ao usuário, tais como emitir alertas, ligar, desligar ou coordenar uma atividade através de um ou mais dispositivos. Na literatura, os principais usos de IoT estão associados ao projeto de soluções para cidades inteligentes e casas inteligentes [Feng et al. 2017, Pradhan et al. 2018, Vlacheas et al. 2013]. Em cada uma dessas aplicações, os dispositivos são heterogêneos, apresentando diferentes características em termos de capacidade computacional, energética e tecnologia de comunicação. Por exemplo, em [Feng et al. 2017], os autores propuseram integrar a IoT com um **S**istema **C**ognitivo **D**inâmico (CDS) e desenvolveram um estudo de caso sobre um cenário de casas inteligentes. Nesse trabalho, os autores apresentaram um exemplo em que o ambiente compreende dispositivos inteligentes, como a televisão, o sofá e o ar condicionado. No cenário, os dispositivos estão habilitados a coletar dados para identificar quando um usuário está cansado e prestes a dormir, analisando seus movimentos, gestos e temperatura do corpo. Este sistema ciber-humano (*cyber-human system – CHS*) permite que os dispositivos inicializem ações apropriadas para prover conforto ao usuário. Desta forma, a televisão pode baixar o volume, o ar condicionado pode ajustar a temperatura e o sofá se inclinar. Nessa estrutura, o CDS organiza técnicas computacionais para desempenhar um conjunto de funções baseadas nas capacidades e características humanas, com o objetivo de monitorar, organizar os dados, tomar decisões e interagir com o ambiente de forma coordenada. Esta técnica exige alta disponibilidade de recursos computacionais. Entretanto, os dispositivos possuem recursos limitados e, para contornar esta limitação, os autores sugerem que o CDS opere na nuvem computacional.

Com relação aos principais protocolos da camada de aplicação, podemos citar o ***Co**nstrained **A**pplication **P**rotocol* (CoAP) [Shelby et al. 2014] e o ***M**essage **Q**ueuing **T**elemetry **T**rans-port* (MQTT) [Banks and Gupta 2014]. O CoAP foi padronizado pela IETF (RFC 7252), sendo destinado às aplicações web para dispositivos com baixa capacidade computacional e energética. Sua principal característica consiste em utilizar o modelo REST (**RE**presentational **S**tate **T**ransfer), o qual permite que os sistemas solicitantes acessem e manipulem representações textuais dos recursos através de comandos básicos como GET, PUT, POST e DELETE. Diferente do CoAP, o protocolo MQTT utiliza o modelo *Publish/Subscribe*, onde os dispositivos geradores de dados (comummente sensores – *Publishers*) os enviam para um *broker* (normalmente um *gateway*), que por

sua vez os encaminha para os dispositivos interessados (*subscribers*). A ISO/IEC 20922 padronizou o MQTT e definiu regras destinadas às camadas inferiores para o controle da Qualidade do Serviço.

### Camada de Rede

A camada de rede na arquitetura conceitual da IoT proposta por [Palattella et al. 2013] é responsável por controlar como as mensagens são transmitidas do emissor para o receptor e abstrai três principais tipos de comunicação: *i*) uma comunicação direta entre dois dispositivos da IoT, *ii*) a comunicação de um dispositivo da IoT (*gateway*) com outro dispositivo através da Internet, e a *iii*) a comunicação virtual fim-a-fim entre dois dispositivos. Ou seja, a camada de rede do modelo conceitual para a IoT proposto por [Palattella et al. 2013] acaba por agregar funcionalidades e serviços presentes nas camadas física/enlace, rede e transposte da arquitetura TCP/IP. Para cada tipo de comunicação mencionado, diferentes protocolos e padrões de comunicação são encontrados na literatura para transmitir mensagens. Essas mensagens por sua vez podem ser apenas de controle e coordenação da rede ou podem ser mensagens que carregam dados coletados na camada de percepção. Neste capítulo, nos referimos ao primeiro tipo de mensagem como *mensagem de controle* e ao segundo tipo, como *mensagem de dados*.

Salienta-se que a camada de rede precisa considerar as limitações de recursos presentes em grande parte dos dispositivos da IoT. Este aspecto é relevante principalmente para a comunicação direta entre dois dispositivos da IoT. Os principais padrões de comunicação direta entre dois dispositivos da IoT consistem no IEEE 802.15 e no *Bluetooh Low Energy* (LE). A Cisco estimou que 46% das conexões seguirão o IEEE 802.14 e *Bluetooth*. Assim como o IEEE 802.15, o *Bluetooth* LE é uma tecnologia de comunicação sem fio arquitetada para operar com dispositivos de baixa capacidade energética, ou seja, prioriza a economia no consumo de energia. Esses padrões atendem às especificidades das redes sem fio com uma abrangência de área pessoal (*Wireless Personal Area Network*). Em geral, esse tipo de comunicação se dá entre dispositivos com baixo recurso energético. Esses padrões especificam a camada física e o controle de acesso ao meio.

Como na IoT mais de um meio de comunicação físico pode estar presente, soluções que tornem a heterogeneidade transparente são necessárias. Devido a este fato, para se conectar com a Internet alguns dispositivos precisam do suporte de um outro protocolo ou de uma adaptação nos protocolos de comunicação direta para seguir os padrões estipulados pela camada de rede da arquitetura TCP/IP. Para solucionar esse problema surgiu o padrão que implementa IPv6 sobre Redes Sem Fio de Área Pessoal e Baixo Consumo de Energia (do inglês, *IPv**6** over **L**ow **Po**wer **W**ireless **P**ersonal **A**rea **N**etworks* - 6LoWPAN). O 6LoWPAN foi padronizado pela IETF (RFC 4919) e pode ser implementado diretamente nos dispositivos ou, no caso de dispositivos que utilizam outros meios de comunicação, através de um *gateway*. A principal função do 6LoWPAN é permitir que a estrutura de rede de curto alcance e pouca disponibilidade de banda se comunique com dispositivos na Internet através do protocolo IPv6. A principal técnica utilizada é a definição e compressão dos cabeçalhos IPv6, diminuindo o tamanho dos pacotes. Isso permite que mais dados sejam inseridos no *payload* dos pacotes sem exceder a ***M**aximum **T**ransmission **U**nit* (MTU) do protocolo responsável pela camada física e enlace na IoT.

As abstrações propostas pelo 6LoWPAN habilitam os dispositivos a estabelecer conexões entre dispositivos com tecnologias diferentes, utilizando o 6LoWPAN como tecnologia de comunicação comum. O 6LoWPAN foi desenvolvido para operar sobre o IEEE 802.15.4. No entanto, a RFC 7668 normatiza a integração do IPv6 sobre o *Bluetooth LE* [Nieminen et al. 2015]. A Figura 1.3 compara o projeto original original do *Bluetooth* LE com o projeto de integração do 6LoWPAN e o *Bluetooth LE*. Conforme ilustra a Figura 1.3$(a)$, o projeto original consistia na camada física, camada de enlace e uma interface modo de teste direto. A camada física transmite e recebe os pacotes atuais. A camada de enlace é responsável por prover o controle de acesso ao meio, estabelecimento de conexões, controle de erros e controle de fluxo. A interface de modo de teste direto é usada somente em testes. Nas camadas superiores encontram-se o controle do enlace lógico e o protocolo de adaptação, o Protocolo de Atributo, o Gerenciador de segurança, o Perfil de Atributo Genérico e o Perfil de Acesso Genérico. A Interface Controladora do Host (HCI) separa as camadas inferiores, sendo geralmente implementada na pilha do host. A Figura 1.3$(b)$ mostra os componentes herdados da pilha original do *Bluetooth*, destacando os componentes que agregam o 6LoWPAN.

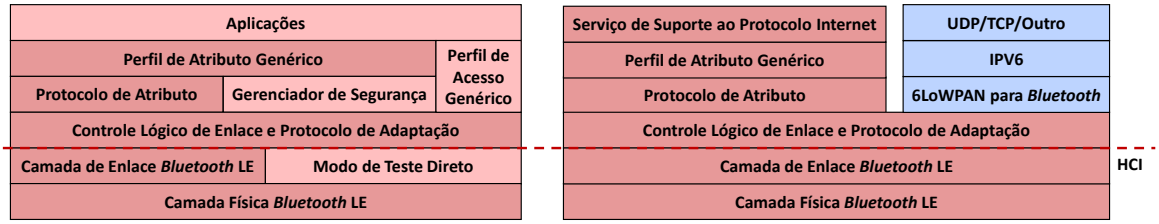


Figura 1.3: Comparação entre o *Bluetooth* e o 6LoWPAN [Nieminen et al. 2015]

Os dispositivos da IoT com menor capacidade, tais como os sensores, não realizam todas as funções para o tratamento dos dados sensoreados. Para isso, eles precisam transmiti-los para um dispositivo com maior capacidade computacional para então processar os dados necessários. Esses dispositivos com maior capacidade podem estar dentro da própria IoT ou podem estar fora da mesma como por exemplo em um ambiente em nuvem. Desta forma, são necessárias comunicações de múltiplos saltos e a comunicação com a Internet. Diante desta necessidade, a IETF criou o grupo ROLL (*Routing Over Low Power and Lossy Networks*). Este grupo definiu o RPL (*IPv6 Routing Protocol for Low-Power and Lossy Networks*)(RFC 6550) [Thubert et al. 2012], um protocolo de roteamento para redes IoT fundado no IPv6. Este protocolo suporta os três tipos de comunicação especificados.

O RPL constrói uma rede com topologia em árvore como um Gráfico Acíclico Direcionado Orientado ao Destino (do inglês, **D**estination-**O**riented **D**irected **A**cyclic **G**raph - DODAG). Um DODAG está ligado a um ou mais nós raízes, que servem como um ponto de trânsito que vincula a rede IoT às redes IPv6. Então as rotas são traçadas sempre de/-para um nó raiz. O RPL possui quatro valores de instância, ou seja, ID da instância, ID DODAG, número da versão DODAG e classificação. Esses quatro valores de instâncias são usados para manter uma topologia DODAG. Em particular, qualquer nó no RPL pode ser identificado exclusivamente com esses quatro valores de instâncias. A instância RPL é usada para identificar os DODAGs compartilhando o mesmo tipo de serviço. Os nós

conectados à mesma raiz têm o ID DODAG comum. O número da versão DODAG é atualizado conforme a topologia das alterações do DODAG. A classificação é usada para representar a distância relativa de um nó a raiz e é um parâmetro muito importante para nós no RPL. Os nós com as menores classificações indicam que eles estão mais próximos da raiz. O RPL também define rotas descendentes como as rotas da raiz para outros nós, enquanto as rotas ascendentes são definidas como rotas de nós para raiz. Para traçar as rotas e montar o grafo, o RPL implementa um padrão de trocas de mensagens que realizam a manutenção das rotas e inclusão de novos nós na estrutura [Zhao et al. 2017].

A comunicação virtual fim-a-fim segue um dos dois modelos: orientada à conexão e não-orientada à conexão. Quando quando a aplicação exige maior confiabilidade, o protocolo ***T**ransmition **C**ontrol **P**rotocol* (TCP) da Internet é empregado para oferecer uma comunicação virtual fim-a-fim orientada à conexão. O TCP foi padronizado pela IETF (RFC 793). Ele garante que os dados sejam entregues entre dois dispositivos. O ***U**ser **D**atagram **P**rotocol* (UDP) (RFC 768) tem como principal característica oferecer uma comunicação virtual fim-a-fim não orientada à conexão. Em geral, ele pode ser aplicado quando a aplicação não possui restrições estritas em relação à confiabilidade da entrega e, focando na IoT, quando a rede é formada por dispositivos com alta mobilidade e baixa capacidade computacional e energética. Dispositivos com restrições de recursos, como energia, precisam fazer um uso eficiente dos mesmos. Desta forma, ao utilizar o protocolo UDP, os dispositivos não precisam manter conexões, fazendo com que os dispositivos possam poupar energia entrando em estado de hibernação sem prejudicar as aplicações e seus serviços [Sonar and Upadhyay 2014].

### Camada de Percepção

A camada de percepção determina como os dispositivos interagem com o ambiente ao seu redor. Estes dispositivos podem atuar como sensores, atuadores ou de maneira híbrida. Os sensores captam dados do ambiente e os enviam para serem interpretados por um dispositivo com capacidade de processar esses dados e obter informações. Os atuadores interagem com o ambiente fisicamente, então eles recebem comandos para realizar atividades, por exemplo, ligar uma lâmpada inteligente. Os dispositivos híbridos normalmente já contêm uma determinada capacidade computacional, onde ele se torna capaz de perceber, processar os dados, interagir com o ambiente e compartilhar suas ações com a rede. Alguns dispositivos híbridos dotados de maior poder computacional (também chamados de coordenador) podem assumir o papel de *gateway*, executando atividades de gerenciamento da rede.

A Figura 1.4 mostra alguns exemplos de dispositivos utilizados na camada de percepção.As aplicações para casas inteligentes implementam redes IoT através dos móveis e eletrodomésticos, eles são equipados com sensores de temperatura, ruído, temporizadores. Uma geladeira, por exemplo, pode identificar a falta de insumos e gerar uma lista de compras e enviar para o *smartphone* do usuário. Os dispositivos pessoais podem se equipar de sensores de localização (GPS) ou acompanhamento cardíaco e promover um acompanhamento completo da saúde do usuário. Os dispositivos para controle de energia elétrica podem identificar oscilações na transmissão de energia elétrica e desligar os dispositivos, ou controlar o consumo excessivo de energia elétrica. Para diminuir os cus-

tos de mão de obra ou até otimizar a produção, as aplicações para fábricas inteligentes utilizam dispositivos como braços eletrônicos, equipados com sensores e atuadores de movimentação e recebem ordens remotamente. Os dispositivos de segurança permitem a monitoração remota de ambientes à distância, para isso câmeras IP capturam imagens e as transmitem através da rede, ou fechaduras que recebem ordens remotas.


Figura 1.4: Dispositivos IoT

### 1.2.2. Computação em Nuvem e IoT

Mesmo com alguns dispositivos assumindo a posição de coordenador da topologia da rede, nem sempre eles possuem a capacidade de realizar tarefas mais complexas. Incluir dispositivos mais robustos pode sair muito custoso, tanto financeiramente como também em termos de tempo com instalação e manutenção. Isso trouxe a necessidade de implementar arquiteturas que ofereçam recursos extras e sob demanda para a IoT. Esses recursos incluem poder de processamento, armazenamento, serviços de rede, aplicações completas e até ganho no consumo de bateria, visto que a IoT terceirizaria os recursos e as atividades de processamento. O modelo de computação em nuvem proporciona o fornecimento de recursos computacionais necessários para a IoT através da Internet e seguindo o modelo cliente/servidor. Além de ser um modelo amadurecido, a nuvem oferece formas de tratar Big Data e a heterogeneidade de dispositivos e tecnologias [Botta et al. 2016]. Os serviços em nuvem são adquiridos através de contratos de nível de serviço (do inglês, *Service Level Agreement* - SLA) que quantificam os recursos e especificam as regras de uso e valores entre o fornecedor do serviço e o cliente. A principal vantagem de utilizar a computação em nuvem consiste na possibilidade de integrar recursos computacionais e serviços que na maior parte das estruturas IoT são escassos.

No entanto, devido ao constante aumento no número de dispositivos, a demanda de serviços de rede também cresceu e acabou gerando problemas de escalabilidade e latência para os serviços ofertados em nuvem. Desta forma, o paradigma de computação em névoa (*fog*) surgiu para aliviar a sobrecarga dos servidores e enlaces responsáveis por interligar a borda da Internet com a nuvem. A *fog* aproxima da borda parte dos serviços oferecidos originalmente em nuvem [Alrawais et al. 2017], permitindo o pré-processamento de dados ou a pré-seleção dos recursos da nuvem. Além disso, a *fog* alivia a sobrecarga e permite um melhor desempenho em aplicações em tempo real. A Figura 1.5 apresenta uma visão geral sobre a organização dos serviços (nuvem e *fog*), e como os dispositivos se organizam para se conectar à Internet.

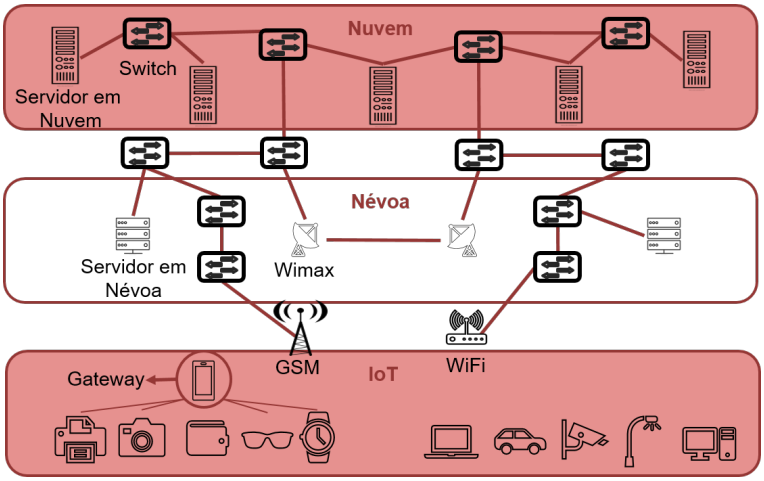


Figura 1.5: Exemplo de uma Arquitetura de Provimento de Serviços para IoT

### 1.2.3. Internet de Todas as Coisas

Devido à alta capacidade de impactar o dia-a-dia do cidadão moderno, a Cisco alavancou a discussão sobre a iteração das "coisas"com as pessoas [Bradley et al. 2013]. Neste contexto, além de considerar as comunicações diretas entre os dispositivos, eles também consideram iterações entre as pessoas e os dispositivos, e entre as pessoas e as pessoas. Assim como a banda larga consistiu em um fator crítico de crescimento econômico, é esperado que a IoE ocasionará um impacto semelhante ao abranger a inclusão social, a melhoria na prestação de serviços e a criação de muitas novas oportunidades. Outro benefício da IoE consiste no seu alto potencial de coletar e analisar dados de milhões de dispositivos, habilitando a automatização dos processos baseados nas pessoas [Mitchell et al. 2013]. Para organizar essa quantidade de informações, dispositivos e pessoas [Evans 2012], considera-se quatro pilares para construção de sistemas eficientes IoE, são eles; pessoas, dados, coisas e processos (Figura 1.6).

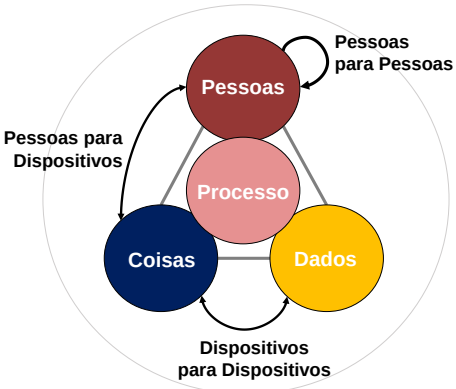


Figura 1.6: Internet de Todas as Coisas [Evans 2012]

As pessoas interagem com esses sistemas se conectando à Internet através de seus dispositivos e gerando dados. Os dispositivos implementam funcionalidades associadas a aplicações como cuidados com a saúde e iteração social (redes sociais). Os dados normalmente coletados por dispositivos são transmitidos através da Internet para serem analisados. Estima-se que a medida que as tecnologias desses dispositivos e sistemas de análise

de dados forem se desenvolvendo, a capacidade de cruzar dados e trazer soluções efetivas também aumentará. A IoT como relatada nesta seção está em constante desenvolvimento, elevando a capacidade dos dispositivos a desenvolverem ciência do contexto em que estão inseridos ou dos objetivos que devem cumprir. Os processos envolvem a gestão dos outros pilares, tornando os objetivos que levam a relação entre pessoas, dados e dispositivos mais eficientes.

Alguns autores como em [Schatten et al. 2016a] levantam questionamentos de como esses sistemas que envolvem milhões de dispositivos devem ser construídos. No mesmo trabalho os autores sugerem que além de inteligência, os dispositivos também precisam considerar as relações sociais. Na literatura, já existem propostas que definem arquiteturas e formas de relacionamento social entre as coisas [Bernabe et al. 2016, Bassi et al. 2013], onde os dispositivos se agrupam através do contexto em que estão inseridos. Em [Atzori et al. 2017], os autores propõem um *framework* para gerência de multidões de dispositivos móveis conscientes de sociabilidade, para isso ele utiliza uma ferramenta de virtualização de dispositivos que segue um algoritmo de controle de recursos. Esses trabalhos demonstram a evolução para uma IoE inteligente e organizada.

Esta seção mostrou os principais fatores que devem ser levados em consideração ao desenvolver soluções IoT, bem como a sua arquitetura, os principais protocolos utilizados, modelo de provimento de serviço mais utilizado e também um breve estudo sobe a IoE. Ao somar diferentes tecnologias, a complexidade acaba aumentando, dificultando o desenvolvimento de soluções para gerência de redes e segurança, principalmente quando consideramos aspectos iterativos entre as pessoas e os dispositivos como na IoE. Nas próximas seções, apresentamos os conceitos básicos de redes definidas por software, as quais trazem vantagens na gerência de redes, bem como as principais ameaças de segurança e defesas para essas arquiteturas.

## 1.3. Redes Definidas por Software

O propósito fundamental das redes de comunicação consiste em transferir dados de um ponto para outro. Uma das principais funcionalidades envolvidas na transferência dos dados consiste no encaminhamento de pacotes (unidade de dados tratada na camada de rede da arquitetura TCP/IP). Esta funcionalidade determina o modo como os pacotes são trafegados entre os diferentes equipamentos de rede intermediários. Tipicamente, as redes são construídas com muitos equipamentos, compreendendo roteadores, comutadores e dispositivos intermediários, tais como *firewalls*, balanceadores de carga e sistemas de detecção de intrusão. Cada um destes diferentes equipamentos necessita ser configurado de maneira específica para desempenhar suas respectivas tarefas associadas com a manipulação de pacotes [Feamster et al. 2014, Nunes et al. 2014]. Desta forma, encaminhar pacotes de maneira eficiente entre os equipamentos é essencial.

Nas redes tradicionais, a configuração referente às decisões de encaminhamento de pacotes e a configuração física dos dispositivos são combinadas no mesmo equipamento da rede. Através dessa abordagem, após a definição inicial do gerenciamento de fluxo (política de encaminhamento), a única maneira de ajustar esta política é através da configuração individual dos equipamentos. Essa característica ocasiona limitações quanto à administração de redes em larga escala, pois demanda a configuração individual de cada

equipamento da rede [Sezer et al. 2013], requerendo um nível elevado de conhecimento dos seus operadores [Nunes et al. 2014]. Além disso, novas demandas para configuração do tráfego de rede têm sido originadas por tendências emergentes nas tecnologias de comunicação, tais como mobilidade, relações sociais e *Big Data* [Xia et al. 2015]. Em virtude das limitações existentes nas redes tradicionais e as novas demandas para configuração do tráfego de rede, emerge a necessidade em repensar a forma como os pacotes são encaminhados na rede.

As redes SDN surgiram como uma iniciativa para contornar estas limitações e atender às novas demandas de configuração de tráfego de rede. Estas redes foram citadas no relatório da *IEEE Computer Society* como uma das 23 tecnologias que prometem mudar o mundo até 2022 [Alkhatib et al. 2015]. A *Open Networking Foundation* define SDN como um tipo de arquitetura de rede que desacopla o plano de controle do plano de dados, centraliza de modo lógico a inteligência e o estado da rede, abstraindo a infraestrutura de rede das aplicações [Open Networking Foundation 2012]. No plano de dados são tratadas as necessidades da infraestrutura física para o encaminhamento de dados, envolvendo os equipamentos de rede como comutadores e roteadores [Nunes et al. 2014, Sezer et al. 2013]. O plano de controle compreende a tomada de decisão no encaminhamento de pacotes na rede, possibilitando a programação dos caminhos que serão usados pelos fluxos de dados e representando inteligência da rede. Através das redes SDN, espera-se simplificar o gerenciamento de redes em larga escala, de maneira a alcançar os requisitos das novas demandas de configuração do tráfego de rede.

Uma rede SDN típica é composta por três partes principais: aplicação, plano de controle e plano de dados [Farhady et al. 2015], como ilustra Figura 1.7. A *aplicação* indica a parte que explora o desacoplamento do plano de controle e o plano de dados para alcançar objetivos específicos, tais como mecanismos de segurança ou soluções de monitoramento de rede. As aplicações se comunicam com o controlador do plano de controle através da interface *limite norte* do plano de controle. O *plano de controle* consiste na parte que manipula os dispositivos de encaminhamento de fluxo de dados através de um controlador visando alcançar objetivos específicos de uma dada aplicação. O controlador usa a interface *limite sul* do *switch* SDN para se conectar com o plano de dados. O *plano de dados* consiste na parte que suporta um protocolo compartilhado (por exemplo o *OpenFlow*) com o controlador e manipula os pacotes atuais com base nas configurações que são manipuladas pelo controlador.

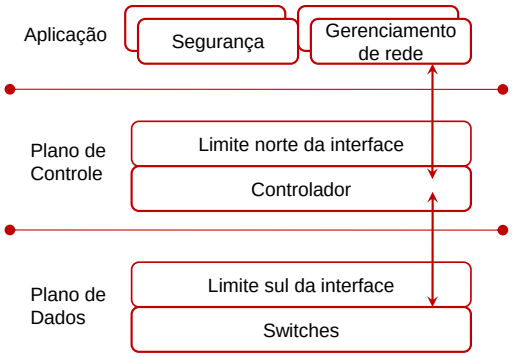


Figura 1.7: Componentes de uma Rede SDN. Adaptado de [Farhady et al. 2015].

Considerando o advento das SDNs, atualmente os paradigmas de rede podem ser divididos em três tipos de acordo com a implantação dos planos de controle e dados [Farhady et al. 2015]. A Figura 1.8 ilustra em alto nível os paradigmas de rede. No paradigma tradicional, onde a rede é centrada no hardware, os *switches* são usualmente sistemas fechados que agregam em si mesmos os planos de controle e dados e suportam interfaces de controle específicas desenvolvidas por seus fabricantes. Portanto, na abordagem de rede centrada no hardware, a implantação de novos protocolos e serviços (e mesmo novas versões de protocolos existentes) se torna desafiadora, pois todos os *switches* precisam ser atualizados ou substituídos. Em contraste, nas redes SDN, os *switches* se tornam simples dispositivos de encaminhamento de dados e controladores centralizados são responsáveis pelo gerenciamento da rede como um todo. Este desacoplamento dos planos de dados e controle facilita a implantação de novos protocolos e serviços, pois a decomposição possibilita que os *switches* sejam programados através dos controladores. Finalmente, a abordagem híbrida suporta tanto o modelo tradicional de rede, quanto o modelo SDN.

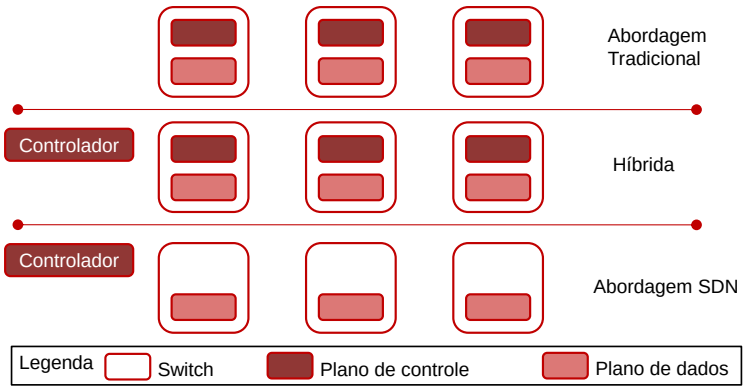


Figura 1.8: Paradigmas de Rede. Adaptado de [Farhady et al. 2015].

### 1.3.1. Evolução das Redes Programáveis

As redes definidas por software vêm recebendo atenção da indústria e academia. Todavia, vale salientar que as ideias da programação de redes de computadores através do desacoplamento do controle lógico das redes vêm sendo discutidas durante anos. Esta seção provê uma visão geral das abordagens precursoras das redes SDN as quais contribuíram para o amadurecimento das ideias que sustentam o paradigma atual das SDNs. As principais iniciativas que contribuíram para o surgimento das SDNs são o grupo de trabalho *Open Signaling* (OPENSIG) [Campbell et al. 1999], IETF *Network Configuration* (NETCONF) [R. Enns 2006], *Active Networking* [Tennenhouse et al. 1997], *Devolved Control of ATM Networks* (DCAN) [Leslie et al. 2015], Projeto 4D [Greenberg et al. 2005] e *Ethane* [Casado et al. 2007]. O grupo OPENSIG iniciou em 1995 promovendo uma série de *workshops* dedicados a tornar as redes ATM, Internet e as redes móveis mais abertas, extensíveis e programáveis [Campbell et al. 1999]. Eles acreditaram na necessidade de implantar a comunicação com o hardware através do controle baseado em software, mas que esta tarefa apresentaria desafios em termos práticos. O núcleo da proposta do OPENSIG era prover acesso ao hardware de rede através de interfaces de rede programáveis visando à criação de novos tipos de serviços baseados em um ambiente distribuído pro-

gramável. O protocolo *General Switch Management Protocol* (GSMP) consistiu em um dos principais resultados desta iniciativa. Esta iniciativa encontra-se oficialmente concluída e a última versão do protocolo GSMP foi publicada em Junho de 2002.

A iniciativa *Active Networking* também iniciou em 1995 com objetivo de proporcionar infraestruturas de rede que poderiam ser programáveis para melhor se adaptar a serviços específicos [Tennenhouse et al. 1997, Moore et al. 2001]. Para alcançar este objetivo, o *Active Networking* considerava duas abordagens: (1) os *switches* programáveis pelos usuários; e (2) cápsulas. A primeira abordagem previa o gerenciamento de canais para transferir as bandas de entrada e saída. A segunda abordagem advogava que programas poderiam ser fragmentados e carregados nas mensagens dos usuários para serem interpretados e executados pelos roteadores. Apesar dos diversos esforços desta iniciativa, o *Active Networking* nunca chegou a ser considerado para a adoção pela indústria devido às limitações de desempenho e segurança [Tennenhouse and Wetherall 2002].

Ainda na metade dos anos 90 surgiu a iniciativa DCAN visando projetar e desenvolver uma infraestrutura necessária para gerenciamento e controle escalável das redes ATM [Leslie et al. 2015]. A principal premissa do DCAN consistia em desacoplar o controle e o gerenciamento das funções de encaminhamento dos pacotes, este consiste basicamente no conceito fundamental por trás das redes SDN. O DCAN também assumia um protocolo minimalista entre o gerenciador e a rede, o que de modo geral consiste no papel desempenhado pelo *OpenFlow*. Esta iniciativa foi oficialmente encerrada em 1998.

O projeto 4D iniciou em 2004 e enfatizava a separação entre a decisão lógica do roteamento de dados e os protocolos que governavam a interação entre os dispositivos de rede [Rexford et al. 2004, Greenberg et al. 2005]. Este projeto defendia que o controle das redes deveria ser separado em três planos: decisão, disseminação e descobrimento. Através desta organização o plano de decisão possuía uma visão geral da rede, contando com serviços oferecidos pelo plano de disseminação e descobrimento. Estes três planos controlariam o plano de dados, responsável por encaminhar o tráfego de rede.

O NETCONF surgiu em 2006 tendo como objetivo melhorar o ***S**imple **N**etwork **M**anagement **P**rotocol* (SNMP) para configuração de dispositivos da rede [R. Enns 2006]. O protocolo SNMP foi proposto na década de 80 e se mostrou muito popular como um protocolo de gerenciamento de rede ao usar a ***S**tructured **M**anagement **I**nterface* (SMI) para encontrar e alterar os dados contidos no ***M**anagement **I**nformation **B**ase* (MIB) [J. D. Case and M. Fedor and M. L. Schoffstall and J. Davin 1990]. Através do SNMP era possível alterar variáveis na MIB para modificar aspectos de configuração. Com o passar do tempo, o SNMP se proporcionou contribuições mais eficazes quando utilizado como uma ferramenta de monitoramento de desempenho e gerenciamento de faltas. Além disto, foram detectadas limitações na concepção do SNMP, principalmente referente a aspectos de segurança. O NETCONF visava corrigir as limitações do SNMP ao prover uma simplificação da configuração e reconfiguração dos dispositivos, atuando como um bloco de gerenciamento, onde não existia a separação entre o plano de dados e o plano de controle. Atualmente esta iniciativa continua ativa.

O *Ethane* também foi iniciado em 2006 e consiste no predecessor imediato do *OpenFlow* [Nunes et al. 2014]. O *Ethane* focou na utilização de um controlador centralizado para gerenciar a segurança na rede [Casado et al. 2007]. Esta iniciativa empregava

dois componentes principais: um controlador e *switch ethane*. O componente controlador era responsável por decidir se um pacote deveria ser encaminhado e o *switch ethane* mantinha uma tabela de fluxos e um canal seguro para o controlador. Uma das características mais marcantes desta iniciativa consistiu no paradigma de controle de acesso baseado em identidades, o qual contribuiu para a implementação de controladores SDN largamente utilizados atualmente, tais como o *NOX* [Gude et al. 2008], *Maestro* [Ng 2010] e *Beacon* [Erickson 2013].

### 1.3.2. Principais arquiteturas SDN

Esta seção revisa as arquiteturas *ForCES* e *OpenFlow*, as quais seguem o princípio básico da separação entre o plano de controle e o plano de dados. Além disto, ambas as arquiteturas padronizam a troca de informação entre estes planos. No entanto, elas são diferentes em termos de projeto, modelo de encaminhamento e interface de protocolo.

#### *ForCES*

Consiste na abordagem proposta pelo grupo de trabalho da *IETF Forwarding and Control Element Separation* (ForCES) para refinar a arquitetura interna dos dispositivos de rede tendo o controle separado dos elementos de encaminhamento de dados [Nunes et al. 2014]. Nesta arquitetura, o dispositivo de rede permanece representado como uma única entidade. O principal objetivo deste consistia em combinar novos hardwares para encaminhamento de dados com um controle provido por uma terceira parte dentro de um único dispositivo de rede. Assim, os planos de controle e de dados são mantidos próximos, por exemplo, na mesma caixa ou sala. Em contraste, o plano de controle é tirado inteiramente do dispositivo de rede, seguindo o mesmo estilo dos sistemas SDN baseados em *OpenFlow*. O ForCES define duas entidades lógicas denominadas *Forwarding Element* (FE) e *Control Element* (CE), sendo que ambas se comunicam através do protocolo *ForCES*. O FE é responsável por prover a manipulação dos pacotes. O CE executa o controle, sinaliza funções e emprega o protocolo *ForCES* para instruir os FEs em como manipular pacotes. O protocolo funciona de acordo com o modelo mestre/escravo, onde os FEs atuam como escravos e CEs atuam como mestres. Um dos principais componentes da arquitetura *ForCES* consiste no *Logical Function Block* (LFB). O LFB consiste em um componente armazenado nos FEs que é controlado pelos CEs através do protocolo *ForCES*. O LFB possibilita que os CEs controlem a configuração dos FEs, especificando como os FEs devem processar os pacotes. O grupo de trabalho do *ForCES* foi oficialmente declarado como concluído em 2015 e contribuíram com uma variedade de documentos, por exemplo, um arcabouço definindo as principais entidades e suas interações, um modelo de linguagem que define funções lógicas dentro do dispositivo de encaminhamento e um protocolo para comunicação entre o controlador e os elementos de encaminhamento.

#### *OpenFlow*

O *OpenFlow* padroniza a troca de informação entre os planos de controle e de dados [McKeown et al. 2008]. Na arquitetura *OpenFlow* ilustrada na Figura 1.9 o dispositivo de encaminhamento, ou *switch OpenFlow*, contém uma ou mais tabelas de fluxos e uma camada de abstração que possibilita a comunicação segura com o controlador através do

protocolo *OpenFlow*. As tabelas de fluxos consistem de fluxos de entrada, onde cada uma determina como os pacotes pertencentes a um fluxo serão processados e encaminhados. Os fluxos de entrada consiste tipicamente de (1): campos de correspondência, ou regras de correspondência; campos de correspondência podem conter informações encontradas nos cabeçalhos dos pacote, porta de ingresso e metadados; (2) contadores, usados para coletar estatísticas para um fluxo particular, tal como o número de pacotes recebidos, número de *bytes* e a duração do fluxo; e (3) um conjunto de instruções, ou ações para serem aplicadas sobre uma correspondência, elas ditam como manipular pacotes correspondentes. Quando um pacote chega a um *switch OpenFlow*, os cabeçalhos do pacote são extraídos e comparados com os campos de correspondência da tabela de fluxos. Se existir uma correspondência, o *switch* aplica o conjunto de instruções ou ações associadas com a entrada do fluxo correspondente. Se não existir uma correspondência na tabela de fluxos, a ação tomada pelo *switch* dependerá das instruções definidas nas entradas de falhas de correspondência de fluxos. Estas entradas especificam o conjunto de ações a serem tomadas quando nenhuma correspondência para um dado fluxo é encontrada para um pacote de entrada, por exemplo, o descarte do pacote, o encaminhamento para outra tabela de fluxos ou o encaminhamento do pacote para o controlador *OpenFlow*.

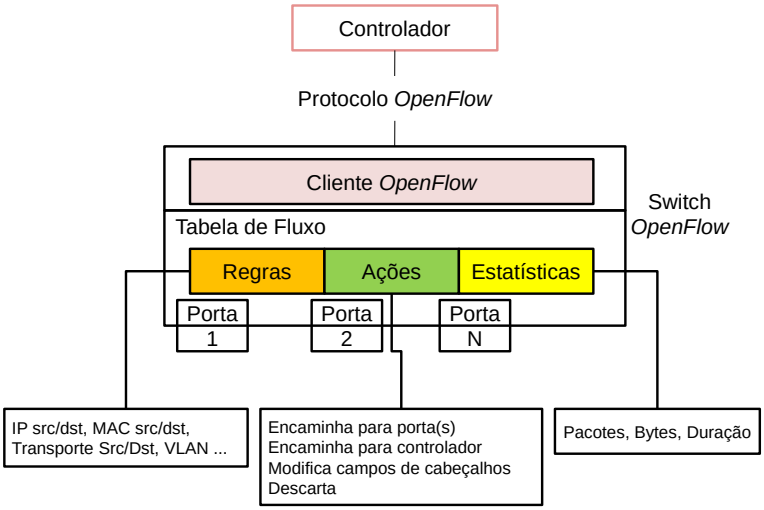


Figura 1.9: Arquitetura *OpenFlow*. Adaptado de [Nunes et al. 2014].

A comunicação entre o controlador e o *switch* acontece via protocolo *OpenFlow*, o qual define um conjunto de mensagens que podem ser trocadas entre estas entidades através de um canal seguro. Usando o protocolo *OpenFlow* um controlador remoto pode executar operações de gerenciamento, como por exemplo adicionar, atualizar ou remover fluxos de entrada das tabelas de fluxos dos *switches*. Estas operações podem ocorrer de modo reativo ou proativo. No modo proativo, as operações de gerenciamento são executadas em resposta a chegada de um pacote. Enquanto que o modo reativo compreende na predefinição das operações de gerenciamento para fluxos previamente esperados. Embora protocolos como o *OpenFlow* especifiquem que um *switch* é controlado por um controlador, implicando em uma centralização, as redes SDN podem possuir um plano de controle centralizado ou distribuído. Como a comunicação entre controladores não é definida pelo *OpenFlow*, torna-se necessário algum tipo de distribuição ou redundância

no plano de controle. O controlador fisicamente centralizado representa um ponto único de falhas para toda a rede. Portanto, o *OpenFlow* permite a conexão de múltiplos controladores para um *switch* o qual deveria possibilitar a utilização de controladores backups para serem ativados em caso de uma falha. Muitos pesquisadores adotam manter o plano de controle logicamente centralizado, mas fisicamente distribuído [Nunes et al. 2014].

### 1.3.3. IoT baseada em SDN

Os princípios das redes SDNs podem ser direcionados para diferentes aspectos das estruturas de provimento de serviços na IoT [Bera et al. 2017]. A Figura 1.10 demonstra o funcionamento de uma rede IoT baseada em SDN considerando a divisão entre os planos de gerência, controle, dados, serviço e IoT. O plano de gerência compreende sistemas responsáveis por controlar as operações de rede em nível de aplicação. O plano de controle atua como o cérebro da estrutura, sendo composto pelos controladores SDN. Os controladores são dispositivos logicamente posicionados no centro da estrutura e realizam as principais abstrações para que o plano de gerência se relacione com os dispositivos do plano de dados. O plano de dados agrega os dispositivos responsáveis pelo encaminhamento de dados. Esses dispositivos detêm tabelas de regras responsáveis por controlar os fluxos de tráfego da rede. Seguindo esta organização, sempre que um comutador do plano de dados não souber lidar com um fluxo ele solicita auxílio ao plano de controle. O plano de serviço representa a computação em nuvem e fog, são serviços para IoT fornecidos através da Internet. Por fim, o plano IoT envolve os dispositivos com capacidade de se conectar com a Internet. É importante destacar que os planos que representam as SDNs (Planos de gerência, controle e dados) estão presentes nos planos de serviço e IoT, e eles realizam as tarefas de distribuição e controle do tráfego de rede. A arquitetura representada na Figura 1.10 mostra que o modelo SDN pode estar presente na expectativa do servidor online (Nuvem), da disseminação dos dados desses serviços (fog) ou IoT.

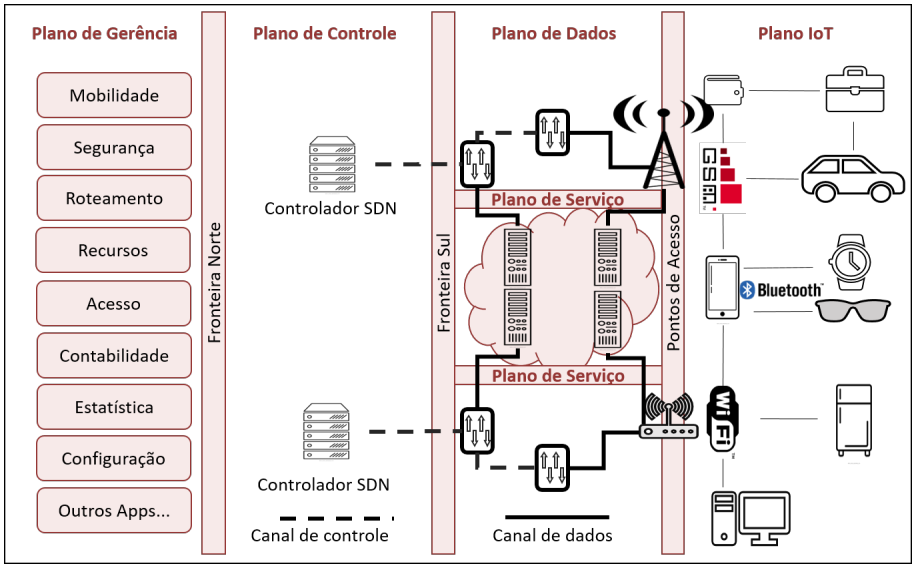


Figura 1.10: Exemplo de IoT Baseada em SDN

Devido ao rápido crescimento no número de dispositivos conectados a Internet, a gerência de redes se torna um fator importante para as estruturas IoT. Administrar redes com grandes quantidades de dispositivos gerando dados massivos, somado com a escas-

sez de recursos, trazem a necessidade de processamento de dados remotos. Isso exige da estrutura mecanismos para administrar o tráfego da rede de forma eficiente e rápida. As SDNs oferecem benefícios para a gerência de redes de computadores. Centralizar o plano de controle da rede permite uma visão global da estrutura, o que traz agilidade para mecanismos que realizam atividades como distribuição, controle de tráfego e alocação de recursos. Outro ponto em que as SDNs beneficiam a IoT consiste na simplificação para implementar soluções baseadas em virtualização de funções de rede (do inglês, *Network Function Virtualization* - NFV), que habilitam os dispositivos, originalmente projetados para executar uma única função, para desempenhar múltiplas funções [Bera et al. 2017].

Ao agregar estas tecnologias (SDN, Nuvem e IoT), novas funções para esse conjunto vêm sendo desenvolvidas. O modelo tradicional para as SDNs implicam em um controlador fixo para cada sub-rede, ou seja, um dispositivo servidor exclusivo para controle das regras de rede. No entanto, no modelo IoT as sub-redes são móveis, sem fio e muitas vezes são interconectados por conexões de um único salto e organizadas através de um *gateway* (não existe nenhum de dispositivo de encaminhamento entre o *gateway* e o dispositivo final); ou seja, não necessitam de regras de roteamento. Por outro lado, devido a característica de escassez de recursos, precisam de sistemas de administração de recursos mais eficientes. Levando em consideração esses parâmetros, na literatura existem trabalhos que implementam o modelo de controladores SDN no dispositivo final. Lee *et al.* propõem administrar o consumo dos recursos através de uma hierarquia de controladores composta por controladores móveis e um controlador global [Lee et al. 2014]. O controlador global desempenha atividades administrativas nos controladores móveis, sendo posicionado de forma fixa. Os controladores móveis administram o consumo dos recursos de acesso considerando a disponibilidade dos canais sem fio através do tempo de transmissão para a comunicação direta entre os dispositivos, controlam as especificações de qualidade de serviço para o padrão IEEE 802.11 por meio das regras instaladas sob demanda pelo controlador global. Esta organização permite que as SDNs possam se tornar um serviço para gerência de redes IoT ofertados como um serviço de nuvem, demonstrando a flexibilidade oferecida por este conjunto de tecnologias.

## 1.4. Principais Ameaças

A segurança em redes vem sendo um dos principais focos para o desenvolvimento de pesquisas em IoT e SDN na atualidade[Bera et al. 2017]. Estes problemas muitas vezes comprometem a segurança dos dados trafegando na rede. Os três principais pilares da segurança da informação consistem na confidencialidade, integridade e disponibilidade. Uma estrutura de rede considerada segura deve manter estes pilares consonância com seus objetivos. Os usuários mal-intencionados (atacantes) realizam análises para encontrar vulnerabilidades do sistema podendo resultar em vazamento de dados, acesso não autorizado nos serviços e operações da rede, modificação de dados [Scott-Hayward et al. 2016]. Existem ameaças de segurança para cada plano da arquitetura de IoT baseada em SDN apresentada na Figura 1.10. Estas ameaças são classificadas de acordo com o tipo de rede alvo, podendo compreender redes estruturadas (SDN) e redes não estruturadas (IoT e IoT Baseada em SDN). As ameaças de segurança podem ser classificadas em intrusões e ataques. Esta seção aborda estas ameaças.

### 1.4.1. Intrusões e Ataques

Um dos principais problemas da IoT baseada em SDN é a segurança contra intrusões e ataques [Farris et al. 2018]. As intrusões visam explorar vulnerabilidades nos sistemas que garantem a confidencialidade dos dados na rede. Os usuários mal-intencionados usualmente falsificam ou clonam dados de identificação e por meio de um acesso não autorizado a rede (intrusão), remoto ou físico, capturam e manipulam os dados que trafegam na rede (ataque). Existem diversos tipos de ataques, cada um atinge diferentes planos da IoT baseada em SDN (Tabela 1.1). As aplicações tanto da IoT como das SDNs e o plano de controle são atingidas por ataques que manipulam, falsificam ou negam o fornecimento de informações. O plano de rede e a camada de rede são prejudicados por ataques que exploram os pontos de acesso e a transmissão dos dados. A camada de percepção é atingida por ataques que envolvem a falsificação de dispositivos, encaminhamento seletivo de informações ou esgotamento dos recursos.

Para realizar uma intrusão, o atacante pode personificar usuários, administradores, dispositivos ou até aplicações. Para realizar a intrusão ao sistema, o atacante pode utilizar as técnicas de análise ou força bruta. As técnicas de análise consistem em adquirir informações da estrutura de rede. As mais comuns em estruturas de rede baseadas em SDNs são os escaneadores de WLAN, *sniffers*, escaneadores de portas, *phishing* e engenharia social. Os escaneadores de WLAN são dispositivos equipados com placas de redes sem fio que buscam pontos de acesso sem fio adaptando a frequência de transmissão, essa técnica de escaneamento se torna útil para descobrir a localização dos dispositivos e consequentemente informações sobre a topologia. Os *sniffers* capturam todo o tráfego de rede que não é criptografado e realizam análises para descobrir os serviços ativos na rede. Os escaneadores de portas se conectam a servidores e exploram o TCP e o UDP para realizar um mapeamento das portas, através disso eles conseguem planejar quais serviços podem ser atacados. No *fishing* o atacante explora a inocência dos usuários legítimos da rede, com o objetivo de adquirir os dados de acesso, para isso ele insere um código malicioso nos dispositivos das vítimas através de links de acesso, imagens ou páginas web falsificadas. Os métodos de engenharia social também exploram a inocência dos usuários, então os atacantes tentam adquirir informações dos usuários como cargos, horários de ponto, datas, ou seja, qualquer informação que revele senhas ou oportunidades de acesso físico à estrutura. Na força bruta, os atacantes testam todas as possibilidades de combinações de credenciais de acesso na tentativa de burlar as defesas da rede. Além disso, o atacante pode combinar as técnicas utilizando as informações adquiridas através das análises para direcionar os dados da força bruta. Por exemplo, utilizar uma lista dos nomes, cargos e datas de nascimento dos funcionários de uma empresa e programar um *script* para testar todas as combinações possíveis entre esses dados.

Ao conseguir penetrar as defesas da rede, o atacante pode desempenhar ataques considerando especificidades das SDNs e IoT. Nas SDNs, a personificação pode ocorrer no plano de controle ou no plano de dados. A personificação de um controlador (plano de controle) pode comprometer toda a arquitetura de rede, pois por meio deste dispositivo o atacante consegue manipular todas as operações de rede. Ao personificar um servidor provedor de serviços, o atacante pode capturar uma quantidade significativa de dispositivos que utilizam o serviço e desempenhar ataques de negação de serviço contra outro

alvo através do redirecionamento das requisições. Considerando uma rede IoT, a personificação pode ocorrer no dispositivo ou nos serviços. A personificação de um dispositivo (camada de percepção) implica no comprometimento do dispositivo em si e pode abrir brechas para outros ataques. Em nível de serviço, este ataque permite aos atacantes o controle dos dispositivos para prejudicar a rede ou divulgar dados indevidamente. As principais variações deste ataque estão classificados na Tabela 1.1.

| **Ataques** | **SDN** | | | **IoT** | | |
|---|---|---|---|---|---|---|
| | *Aplicação* | *Plano de controle* | *Plano de Rede* | *Aplicação* | *Camada de Rede* | *Camada de Percepção* |
| *Sybil* | • | • | | • | | • |
| *Homem no Meio* | | | • | | • | |
| *Buraco Negro* | | • | • | | • | • |
| *Falsificação e Manipulação dos dados* | • | • | • | • | • | • |
| *Negação de Serviço* | • | • | • | • | • | • |

Tabela 1.1: Categorização dos Ataques por Camada/Plano

Os ataques de homem no meio podem ocorrer em infraestruturas SDN de redes IoT que empregam a computação em névoa [Li et al. 2017]. A computação em névoa resulta em benefícios para IoT ao proporcionar uma etapa preliminar de processamento dos dados antes de os enviar para a nuvem e ao reduzir a latência na troca de dados de dispositivos IoT. As funcionalidades da computação em névoa e das redes SDN são integradas ao representar os nós fog como *switches SDN*. Todavia, esta combinação de soluções em rede torna-se vulnerável à ataques homem no meio. Neste cenário, atacantes interceptam o canal de comunicação entre o controlador e os *switches* do protocolo *OpenFlow*. Este canal possibilita o envio de comandos e requisições do controlador, bem como as estatísticas coletadas dos *switches*. A ocorrência de um ataque homem no meio neste canal ocasiona circunstâncias desastrosas para a rede e para seus usuários. Considerando a perspectiva os usuários, um atacante pode coletar informações sensitivas dos seus usuários, resultando no vazamento de dados confidenciais. Do ponto de vista da rede, o atacante pode enviar pacotes falsos para o controlador se passando pelos *switches*, contaminando a visão global que o controlador possui da rede. Como consequência, o controlador pode configurar de maneira equivocada os demais *switches*, gerando problemas de conectividade na rede.

Os ataques Sybil prejudicam as redes IoT através da inclusão de um dispositivo falso na estrutura para manipular regras de roteamento ou opiniões coletivas [Rajan et al. 2017]. A principal característica do ataque Sybil consiste em falsificar a identidade dos dispositivos. Nesse ataque o atacante utiliza dispositivos com falsas credenciais para acessar uma rede IoT. Para isso o atacante precisa conhecer as características das aplicações e dispositivos que utilizam a rede. Com objetivo de obter descobrir como funcionam as aplicações de autenticação, o atacante pode utilizar técnicas como *sniffing*, *phishing* e engenharia social. Ao descobrir detalhes sobre o método de autenticação do dispositivo em uma aplicação, o atacante falsifica os dados de um dispositivo e tenta conectá-lo

à estrutura. Ao conseguir, o dispositivo falso passa a simular o comportamento de um nó legítimo para agregar confiança e por fim inserir novos dispositivos maliciosos. Este tipo de ataque fere diretamente os pilares de confidencialidade e integridade da rede. O objetivo desse ataque normalmente consiste em adquirir dados confidenciais e manipular a opinião geral inserindo dados falsos na estrutura de rede. Em uma estrutura de IoT baseada em SDN, caso o atacante consiga incorporar um controlador, ele consegue manipular o tráfego de rede e direcionar para um alvo, a fim de realizar um ataque DDoS, transformando todos os dispositivos subordinados ao controlador em uma rede zumbi. A integração da infraestrutura SDN em uma rede IoT alvo deste ataque pode facilitar a proposição de mecanismos para identificar dispositivos com comportamento malicioso [Bull et al. 2016a].

Os ataques buraco de negro e de falsificação e manipulação de dados se aproveitam de regras de roteamento da rede. Para desempenhar estes ataques, o atacante utiliza os escaneadores de rede, de portas e *sniffers* para identificar os dispositivos e serviços em conjunto com seus respectivos dados temporais e informações de roteamento. Os dados temporais são importantes para identificar quando um dispositivo para de transmitir dados. Então o atacante se aproveita deste intervalo para transmitir dados maliciosos ou instalar regras de roteamento. No ataque buraco de minhoca o atacante cria um tunelamento para armazenar o tráfego da rede através de um dispositivo de rede capturado ou falsificado. Ao conseguir capturar registros suficientes do tráfego da rede o atacante pode desempenhar o ataque *replay*. Nessa variação do ataque buraco de minhoca o atacante consegue fazer com que a rede fique repetindo as ações passadas e burlando os sistemas de detecção de intrusão. Ao falsificar os dados de roteamento o atacante pode desempenhar atividades como camuflar um dispositivo malicioso ou instalar um *looping* de roteamento. O atacante também pode desempenhar o ataque do sumidouro, onde ele manipula os protocolos de roteamento para atrair todo o tráfego da rede, prejudicando a recepção dos dados no ponto de coleta. Uma variação dos ataques de falsificação consiste no ataque do buraco negro, onde o atacante instala uma regra em um dispositivo de encaminhamento capaz de ocasionar a eliminação de toda nova entrada de fluxos para isolar parte da rede dos sistemas de segurança.

### 1.4.2. Ataques de Negação de Serviço

Um ataque de negação de serviço (DoS) visa tornar indisponível um serviço ou informação da rede aos usuários legítimos. A abordagem mais tradicional para alcançar este objetivo consiste em sobrecarregar intencionalmente recursos da rede ou dos servidores, tais como a largura de banda da rede, processamento ou memória dos equipamentos de rede, a fim de impedir que usuários legítimos acessem determinados recursos computacionais [Bartholemy and Chen 2015]. Em sua abordagem tradicional, um atacante fabrica uma carga maliciosa e a dispara contra um serviço alvo com objetivo de comprometer seu desempenho e/ou torná-lo indisponível. A característica mais marcante deste ataque consiste na exaustão dos recursos computacionais do serviço ou sistema alvo. Dentre os principais alvos destes ataques encontram-se as infraestruturas de bancos, as empresas que oferecem serviços *on-line* e as organizações governamentais. Iniciativas vêm sendo realizadas para tratar a ocorrência este ataque no contexto de redes IoT baseadas em SDN [Wani and Revathi 2018]. Nestes ataques, o atacante consegue atingir todas os

planos da arquitetura da IoT baseada em SDN. Nas aplicações. tanto da IoT como das SDNs, o atacante explora as vulnerabilidades dos protocolos, através da falsificação de requisições. No plano de controle o atacante pode exaurir os recursos do controlador, negando o serviço de instalação de regras e consequentemente inviabilizando o funcionamento do plano de rede. Os ataques DoS que objetivam prejudicar a camada de rede da IoT, sobrecarregam o canal físico [Sonar and Upadhyay 2014].

O ataque de negação de serviço distribuído, comumente referenciado como DDoS (*Distributed Denial of Service*), é uma variação do ataque tradicional de negação de serviço. Esta variação do ataque emprega de forma coordenada uma grande quantidade de dispositivos previamente infectados para intensificar o poder disruptivo do ataque e dificultar a identificação do atacante [Wang et al. 2015, Paxson 2001]. Usualmente, os atacantes utilizam dispositivos interconectados através de infraestruturas conhecidas como *botnets* para executar ataques DDoS. Essas redes são compostas por computadores infectados, normalmente referenciados como escravos, *bot* ou zumbis, os quais atuam como geradores de carga maliciosa para um sistema alvo [Zargar et al. 2013].

Os ataques de negação de serviço foram observados pela comunidade acadêmica no início dos anos 80. No verão de 1999, o *Computer Incident Advisory Capability* (CIAC), grupo de especialistas responsável por relatar incidentes de segurança, reportou a primeira ocorrência deste ataque [Criscuolo 2000]. Desde então, muitos ataques DDoS vêm sendo disparados contra diferentes organizações com o objetivo de tornar indisponível determinados serviços alvos e gerar prejuízos financeiros ao incrementar custos com a restauração de serviços. Por exemplo, no mês de Fevereiro de 2000, a empresa *Yahoo!* vivenciou um dos primeiros ataques DDoS de inundação, o qual tornou indisponível os serviços prestados pela companhia durante cerca de duas horas, ocasionando uma perda financeira significativa [Wired 2000].

Em Outubro de 2002, um ataque DDoS tornou indisponível por uma hora 9 dos 13 servidores que proviam o serviço *Domain Name System* (DNS) para os usuários da Internet [Naraine 2002]. Outro grande ataque DDoS ocorreu em Fevereiro de 2004, tornando o site da *SCO Group* inacessível para usuários legítimos [Vijayan 2004]. Este ataque foi disparado através de sistemas que foram previamente infectados pelo vírus *Mydoom*. O vírus continha um código que instruiu milhares de computadores infectados para acessar o site da SCO no mesmo instante. O código do vírus *Mydoom* foi reutilizado para disparar ataques DDoS de inundação contra os maiores sites de finanças e redes de notícias dos governos da Coreia do Sul e dos Estados Unidos em Julho de 2009 [Myd 2009, Sudworth 2009]. Em Dezembro de 2010, um grupo auto intitulado como "*Anônimos*" orquestrou ataques DDoS responsáveis por inviabilizar o acesso aos sites das organizações *Mastercard*, *PostFinance* e *Visa* [Wik 2010]. Em 2013, o serviço de banco online dos 9 maiores bancos dos Estados Unidos (*Bank of America*, *Cititgroup*, *Wells Fargo*, *U.S. Bancorp*, PNC, *Capital One*, *Fifth Third Bank*, BB&T e HSBC) foram alvos de uma série de poderosos ataques DDoS disparados por um grupo denominado "*Izz ad-Din al-Qassam Cyber Fighters*" [Kitten 2013].

Em 2013 ocorreu o ataque DDoS contra a *Spamhaus*, uma organização anti-spam sem fins lucrativos [Prince 2013]. No dia 18 de Março de 2013, a empresa identificou um tráfego de entrada massivo em seus servidores de aproximadamente 10 Gbps como

pertencente a um ataque DDoS. No dia seguinte, os relatórios indicaram que o tráfego de entrada tinha aumentado, atingindo picos de 90 Gbps. Nos dias seguintes o ataque desapareceu por um período curto de tempo, retornando no dia 22 de Março com uma intensidade ainda maior, alcançando 120 Gbps. Através de investigações envolvendo peritos da área de segurança computacional, foi descoberto que o autor do ataque consistia em um adolescente residente na cidade de Londres.

Um ataque motivado por razões políticas atingiu 500 Gbps e foi denominado como Occupy Central [Olson 2014]. ELe teve como objetivo remover o poder eleitoral de um pequeno grupo leal aos interesses de Pequim, favorecendo os cidadãos de Hong Kong. Durante este acontecimento, foi realizada uma pesquisa de opinião pública através da Internet para realizar uma reforma constitucional. O ataque teve como objetivo prejudicar a consolidação da democracia em Hong Kong ao impedir o acesso ao sistema de votação do grupo pró-democracia. Alguns indícios mostram que o ataque foi originado internamente no país. Em 2016 o blog KrebsOnSecurity [2] sofreu um ataque que atingiu cerca de 620 Gbps. O ataque foi motivado pelo fato de o blog ter divulgado informações de um serviço que cobrava para realizar ataques DDoS. Os donos do serviço foram presos logo em seguida. A empresa de Internet Akamai que mitigou o ataque descobriu que a botnet era composta por câmeras de segurança. Até então o maior ataque registrado tinha atingido cerca de 363 Gbps. Ainda em 2016 um *malware* que explora dispositivos com usuário e senha padrão foi descoberto ao realizarem análises em um ataque realizado contra empresa francesa de serviços em nuvem a OVH [Goodin 2016]. O ataque atingiu cerca de 1.1 Tbps de tráfego gerado por mais de 145 mil câmeras de segurança. Segundo estudos realizados pela Akamai [Mckeay 2016] os dispositivos desta botnet estão espalhados em cerca de quatro países.

O maior ataque DDoS registrado até o momento atingiu 1.35 Tbps e foi contra o servidor de um dos maiores serviços de hospedagem de códigos fontes do munto, o *GitHub* [Newman 2018]. Os atacantes utilizaram servidores de memória distribuída (*memcache*) para amplificar a potência do ataque. Os autores e os motivos não foram revelados, no entanto o ataque conseguiu deixar offline os servidores da empresa por 10 minutos. Como defesa a empresa utilizou os serviços de mitigação do provedor de internet Akamai, o Akamai Prolexic. Esse tipo de ataque utiliza de técnicas de falsificação de cabeçalhos para realizar requisições aos servidores *memcache*, assim as respostas são direcionadas para o alvo.

### Categorização da Motivação dos Atacantes

Existem diversas razões motivadoras para os atacantes desencadearem ataques DDoS. Estas razões podem ser resumidas em cinco categorias, ganho econômico, vingança, crença ideológica, desafio intelectual e guerras cibernéticas [Zargar et al. 2013]. Os atacantes motivados por razões econômicas têm como objetivo obter ganhos financeiros de corporações através de extorsão. Devido à natureza desta motivação, os atacantes desta categoria apresentam usualmente alto nível de habilidades técnicas e muita experiência. Os ataques motivados por ganhos financeiros causam os maiores danos e visam desencadear falhas de funcionamento em um curto espaço de tempo nos sistemas alvo.

[2] https://krebsonsecurity.com/

Os atacantes cuja principal razão consiste na vingança são geralmente indivíduos frustrados, possivelmente com baixo nível de habilidade técnica e usualmente executam ataques DDoS em resposta a uma aparente injustiça. Outros atacantes são inspirados por suas convicções ideológicas para escolher seus alvos [Prasad et al. 2014]. Atualmente, a maioria dos atacantes é incentivada por estas razões. Dentre os principais motivos encontrados encontram-se as opiniões políticas, religiosas e aspectos morais. A principal característica dos atacantes motivados pelo desafio intelectual compreende em colocar em prática e aprender como executar vários ataques. Estes indivíduos usualmente são hackers iniciantes que querem mostrar suas capacidades. Atualmente, existem várias ferramentas fáceis de usar e *botnets* que podem ser alugadas, possibilitando que até mesmo amadores podem ser capazes de executar ataques DDoS bem-sucedidos.

Os atacantes motivados por guerras cibernéticas geralmente pertencem a organizações militares ou terroristas e estão politicamente encorajados para atacar uma larga faixa de serviços críticos um grupo, região ou país rival. Estes atacantes empregam uma grande porção de tempo e recursos para planejar quais os serviços essenciais de uma região causam maior impacto econômicos em situações de falha ou que podem paralisar um país. Os alvos em potencial destes ataques podem incluir departamentos civis executivos, serviços críticos conectados com a Internet, tais como as redes elétricas inteligentes, os sistemas de monitoramento de tráfego e/ou os serviços de provimento de gás. Estes atacantes podem ser considerados muito bem treinados equipados com amplos recursos.

### Escopo e Classificação

Os ataques DDoS são classificados de acordo com a camada onde ocorrem. De modo geral, estes ataques ocorrem na camada de rede/trasporte ou em nível de aplicação. A seguir são descritos os principais ataques DDoS nestas camadas. Os ataques DDoS em nível de rede/transporte podem ser volumétricos, em nível de sistema operacional, reflexão e amplificação. Os ataques DDoS volumétricos focam em interromper a conectividade dos usuários legítimos ao exaurir a largura de banda da rede da vítima, por exemplo, gerando inundações, fabricando tráfego através de protocolos como *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP) ou *Internet Control Message Protocol* (ICMP). Os ataques DDoS em nível de sistema operacional se aproveitam da forma como os protocolos são implementados no sistema operacional para ocasionar uma sobrecarga, um exemplo muito conhecido deste tipo de ataque consiste no *ping* da morte [Douligeris and Mitrokotsa 2004]. Nos ataques DDoS baseados em reflexão, os atacantes usualmente enviam requisições forjadas (tais como, *ICMP echo request*) ao invés de direcionar para os refletores, assim esses refletores enviam suas respostas para a vítima [Paxson 2001], exaurindo seus recursos (por exemplo, ataques *Smurf* e *Fraggle*). Os ataques de amplificação são conhecidos por empregar serviços ou dispositivos para aumentar a intensidade dos efeitos de negação de serviço. As principais técnicas utilizadas consistem em responder requisições com mensagens muito maiores ou com múltiplas respostas, visando amplificar o tráfego que a vítima processará. As *botnets* têm sido constantemente usadas para propósitos de reflexão e amplificação, principalmente considerando as *botnets* móveis formadas por dispositivos da IoT [Santos et al. 2017]. As técnicas de reflexão e amplificação são empregadas usualmente em casos de ataques *Smurf*, onde os atacantes

enviam requisições com endereço *Internet Protocol* (IP) de origem falsificado (reflexão) para um grande número de refletores através de mensagens *broadcast* (amplificação).

Os ataques DDoS em nível de aplicação focam na interrupção dos serviços oferecidos para usuários legítimos ao exaurir os recursos dos dispositivos, por exemplo, *sockets*, CPU, memória, disco/base de dados e largura de banda de entrada e/ou saída. Quando comparados com os ataques DDoS em nível de rede/transporte, os ataques DDoS executados em nível de aplicação geralmente consomem menos largura de banda, sendo mais fácil de ocultar no tráfego de rede legítimo. Entretanto, os ataques DDoS de inundação em nível de aplicação usualmente possuem o mesmo impacto nos serviços alvos, pois eles são projetados considerando características específicas dos protocolos de aplicação, tais como o HTTP, MQTT ou DNS. Os ataques DDoS em nível de aplicação são categorizados como baseados em amplificação/reflexão e baseados em geração de carga explorando características do protocolo HTTP. Os ataques em nível de aplicação baseados em amplificação/reflexão usam as mesmas técnicas aplicadas nos níveis de rede/transporte, tais como, enviar requisições forjadas de protocolos em nível de aplicação para um grande número de refletores. Por exemplo, a amplificação de ataque DNS emprega técnicas de reflexão e amplificação. Os atacantes geram pequenas consultas DSN forjando o endereço de IP de origem no qual podem gerar um grande volume de tráfego de resposta. Na sequência, este grande volume de tráfego é redirecionado para o sistema alvo com o objetivo de paralisá-lo.

Existem quatro principais tipos de ataques DDoS baseados em geração de carga HTTP, inundação de sessão, inundação de requisições, assimétricos e com requisições e/ou respostas lentos [Wang et al. 2015]. Nos ataques DDoS de inundação de sessão as taxas de requisição de conexão de sessão dos atacantes são maiores que as requisições dos usuários legítimos. Assim, as requisições maliciosas exaurem os recursos dos servidores. Um dos mais famosos ataques desta categoria consiste no ataque de inundação GET/POST, o qual gera uma grande quantidade de requisições HTTP do tipo GET/POST para o servidor web vítima [Nazario 2015]. Os métodos GET e POST estão previstos no protocolo HTTP para possibilitar a interação dos clientes e servidores [Fielding et al. 1999]. Através do método GET os clientes recuperam conteúdos dos servidores e através do método POST os clientes enviam dados para os servidores. Os atacantes exploram estes métodos para sobrecarregar um servidor HTTP vítima usualmente empregam *botnets*. Através destes ataques, cada *bot* gera uma grande quantidade de requisições HTTP GET/POST válidas, sem necessariamente falsificar endereços IP.

Nos ataques de inundação de requisições, os atacantes enviam sessões que contém um número maior de requisições do que o usual, gerando uma inundação no servidor. Um dos ataques mais conhecidos desta categoria consiste na inundação de requisições HTTP GET/POST com sessão única. Este ataque consiste em uma variação do ataque de inundação de requisições HTTP GET/POST que utiliza características da versão 1.1 do HTTP para permitir múltiplas requisições dentro de uma única sessão HTTP. Assim, o atacante pode limitar a taxa de sessão de um ataque HTTP e ultrapassar mecanismos de defesa que limitam a taxa da sessão de muitos sistemas de segurança.

Nos ataques assimétricos os atacantes enviam sessões que contem requisições com alta carga de trabalho. Os ataques mais famosos desta categoria compreendem na inunda-

ção através de múltiplas requisições HTTP *GET/POST* e na injeção de códigos *Structured Query Language* (SQL). No primeiro caso o atacante cria múltiplas requisições HTTP ao formar um único pacote. Desta forma, o atacante pode manter altas cargas de trabalho no servidor vítima com uma baixa taxa de pacotes, tornando o atacante quase invisível para técnicas de detecção de anomalias no tráfego de rede. Através da injeção de código SQL, os usuários maliciosos tiram vantagem de sites Web com um projeto pobre ou integração imprópria com a base de dados. Quando uma vítima é identificada com estas características, o atacante injeta códigos SQL para executar consultas complexas no banco de dados, consumindo uma grande quantidade de recursos do servidor, tais como memória e CPU.

Os ataques DDoS com requisição/resposta lentos objetivam iniciar diversas sessões em um servidor alvo, obrigando-o a mantê-las. Os ataques desta categoria contam com rotinas programadas para realizar requisições periódicas explorando o limite máximo de tempo de vida das sessões. Ao incrementar significativamente o número de sessões em um sistema vítima, um usuário malicioso obriga sua vítima a consumir grande parcela de seus recursos computacionais, resultando na eventual indisponibilidade dos serviços prestados para usuários legítimos. Os ataques mais conhecidos desta categoria são o *Slowloris*, a fragmentação HTTP, *Slowpost* e *Slowreading* [Zargar et al. 2013].

## 1.5. Principais Defesas

Manter um sistema em rede estável e seguro compreende na maior motivação que levam os profissionais em redes a desenvolverem sistemas de defesa. As principais linhas de defesa ao desenvolver um sistema de segurança em redes são prevenção, reação e tolerância [Lima et al. 2009]. Esta seção apresenta as principais defesas para os ataques contra redes IoT baseadas em SDN obedecendo esta organização lógica. A Subseção 1.5.1 descreve as defesas preventivas. A Subseção 1.5.2 aborda as defesas reativas. A Subseção 1.5.3 detalha as defesas capazes de tolerar os ataques.

### 1.5.1. Prevenção

As técnicas prevenção procuram se antecipar aos ataques, ou seja, implementam soluções que impedem ou predizem o acontecimento de ataques. Esta subseção apresenta as principais técnicas utilizadas para sistemas de prevenção e os principais trabalhos que propõe a utilização dessas técnicas para IoT baseada em SDN.

#### Autenticação e Criptografia

Os dispositivos precisam se autenticar para estabelecer uma comunicação confiável em nível de segurança e estabilidade da conexão. Essa autenticação identifica o dispositivo ou a origem dos dados e define os parâmetros da criptografia para as partes obterem acesso às informações íntegras e privadas, garantindo os pilares da confidencialidade, integridade e disponibilidade. A maior parte dos sistemas de autenticação compreendem a troca de chaves de criptografia entre os dispositivos. No entanto, essas técnicas normalmente exigem dos dispositivos muito poder computacional (capacidade de processamento) e energia. Isso ocorre porque para gerar uma chave de criptografia os dispositivos realizam cálculos de tal forma que os dados calculados só podem ser acessados através de uma chave. Essa chave consiste em uma sequência de caracteres e pode ser uma informação única do

dispositivo, um código de acesso ou uma informação que as partes compartilham. A principal motivação que leva ao desenvolvimento de soluções de autenticação compreende na possibilidade do vazamento das chaves de autenticação, pois a obtenção da chave por parte do atacante implica no acesso legítimo das informações de um sistema até mesmo da estrutura da rede.

As soluções capazes de definir como os dispositivos devem se organizar durante a autenticação são interessantes para garantir a autenticação segura dos dispositivos e usuários. Seguindo esta linha, Sciancalepore *et al.* desenvolveram o OAuth-IoT, um *framework* para controle de acesso baseado em padrões abertos [Sciancalepore et al. 2017]. Neste trabalho a rede IoT compreende diversos dispositivos inteligentes interligados por uma conexão sem fio de baixa potência. Nesta rede, um dispositivo coordenador desempenha as atividades de *gateway* e coordena os clientes (por exemplo, estabelecimento de canal TLS com o cliente, autenticação e controle de acesso) em conjunto com um servidor de autorização, como proposto pelo *framework* OAuth2.0. Então, as tabelas do *gateway* são atualizadas como consequência da mútua autenticação dos os dispositivos. A partir desta etapa, o servidor de autenticação realiza o controle dos recursos disponíveis e instala regras de roteamento para o *gateway* acessar os recursos. Neste *framework* o usuário também se autentica para poder solicitar um dos recursos disponíveis. Estas autenticações são realizadas através de *tokens*, assim, quando um usuário solicitar recursos, ele recebe um *token* do servidor para acessar os recursos disponíveis através do *gateway*.

[Bernabe et al. 2016] propõem um sistema de controle de acesso flexível e confiável para IoT (TACIoT). O sistema é baseado em um *framework* de segurança para manutenção da privacidade [Bernabe et al. 2014]. Este *framework*, propõe uma versão melhorada do Modelo de Referência Arquitetural (ARM) [Bassi et al. 2013], onde inclui um gerenciador de contexto, um gerenciador de identidades e um gerenciador de grupos. Através desta estrutura, o TACIoT propõe um controle de acesso flexível e leve, com um modelo de confiança multidimensional capaz de quantificar as dimensões de forma eficiente para aplicar a lógica *fuzzy*. Os nós utilizam como parâmetro de autenticação informações contextuais, como a quem eles pertencem.

**Análise de Dados e Predição dos Ataques**

A análise de dados consiste no processo de identificação e avaliação dos dados capazes de descrever as características mais relevantes sobre o comportamento do sistema alvo de um ataque DDoS. Dessa forma, a análise de dados compreende as seguintes etapas: (*i*) medição dos dados, (*ii*) identificação das características e (*iii*) aplicação de artefatos (ex. estatísticos). A etapa de medição ou registro dos dados extraí os dados da rede, sendo executada no início do processo de análise. A etapa de identificação das características depende da definição de objetivos de análise e foca em características diretamente relacionadas com aspectos referentes à ataques DDoS como exemplo, tamanho de pacotes, volume de requisições, entre outras. A etapa de aplicação de artefatos consiste em selecionar as técnicas como exemplos modelos estatísticos ou matemáticos para compor o *board* de artefatos aplicados na análise de dados. Durante a condução destas etapas torna-se crucial desempenhar de modo satisfatório o registro, a extração, a manipulação dos dados e a seleção das ferramentas adequadas no processo de predição de ataques DDoS.

A predição de ataques DDoS se apropria de técnicas utilizadas para a predição

do comportamento de sistemas complexos e dinâmicos. Em geral, esses sistemas são voláteis, metaestáveis e também dispõem de significativos volumes de dados. As redes de dados se assemelham aos sistemas complexos devido à alta dinamicidade do seu fluxo de dados. Diversas pesquisas estão em andamento para desenvolver a predição de ataques DDoS. Estas pesquisas geralmente exploram técnicas estatística, inteligência artificial ou aprendizagem de máquina para antecipar o comportamento do fluxo dos dados em redes de computadores. O estudo de [Xiao et al. 2006] propõem um sistema para alertar a ocorrência de um ataque DDoS em seus estágios iniciais. O sistema possui um módulo cliente e outro servidor. Nesta pesquisa, o sistema emprega um filtro de *Bloom* modificado para monitorar *handshakes* anormais e utiliza uma estrutura de dados baseada em *hash*. O filtro de *Bloom* identifica se determinado elemento pertence a um conjunto de dados e armazena a informação de forma probabilística emitindo o alerta do ataque DDoS em seus estágios iniciais. Outra proposta observada em [Tsai et al. 2010] apresenta um sistema de alertas antecipados de multicamadas para a geração de alertas antecipados baseado em redes neurais e implementado sobre um IDS. Neste sistema, os computadores atuam de forma colaborativa para monitorar seus nós vizinhos. Cada nó da rede envia os dados coletados para um módulo que analisa e compara combinações com os padrões de DDoS.

O trabalho de [Kwon et al. 2017] sugere um método para estimar a quantidade de ataques DDoS. Os pesquisadores visam superar os limites impostos pelos sistemas de segurança reativos na detecção de intrusão e avaliar a necessidade de implantação de sistemas IPSs na rede. Os autores estimam o número de *bots* com base no número de usuários da rede em associação com os dados disponibilizados por uma pesquisa que informa a porcentagem de *bots* estimada para o país. O método proposto estima a quantidade de ataques DDoS na rede por meio de regressão e de correlação ao considerar as características do tráfego da rede, o número de usuários e a estimativa de *bots*. A pesquisa de [Nijim et al. 2017] propõe um sistema para predizer e previnir os ataques na camada de aplicação. O sistema proposto prioriza requisições legítimas em detrimento do tráfego de ataque através de um mecanismo automático de priorização baseado na classificação das requisições e no histórico do uso dos recursos como memória, espaço em disco, tempo de CPU e tráfego da rede.

Com a aplicação de modelos orientados a dados que capturam o comportamento temporal e espaço-temporal dos ataques DDoS, caracterizados por seus comportamentos e dinâmicas, o estudo de [Wang et al. 2017] aplica modelos estatísticos para predizer as características comportamentais de *botnets* e ocorrência de ataques DDoS. A predição é realizada por meio de análises de engenharia reversa sobre traços de ataques DDoS observados a partir de operações de mitigação calculam correlações temporais, espaciais e espaço-temporais das propriedades de ataques (ex. número de *bots* e duração do ataque). Outra técnica passível de ser aplicada na predição de ataques DDoS é exposta por [Azzouni and Pujolle 2017] e é caracterizada pelo uso de rede neural sobre um modelo de séries temporais (*Long Short-Term Memory*) para processar, classificar e prever a matriz de tráfego em grandes redes. Como a matriz de tráfego tem entre suas aplicações favorecer o gerenciamento da rede, esta coopera com a detecção de anomalias. Assim sendo, é possível predizer a matriz de tráfego aplicando-a na predição de ataques DDoS.

Nas pesquisas de [Zan et al. 2009, Holgado et al. 2017], são propostos métodos baseados em Modelos Ocultos de Markov (*HMM - Hidden Markov Model*) para predi-

zer ataques de múltiplas etapas, como ataques DDoS. As múltiplas etapas compreendem as fases realizadas durante um ataque, como busca de vulnerabilidade e de preparação. Dessa forma, os métodos objetivam prever os próximos passos dos ataques com base nos passos anteriores e dados históricos. Para esse fim, o processo estocástico oculto do modelo de Markov foi demonstrado pela sequência de diferentes etapas de ataques observados nos alertas emitidos por IDSs. Esses alertas são transformados em observações e agrupados em *clusters* conforme a intensidade. Uma vez treinado, o modelo é capaz de predizer a probabilidade dos passos dos ataques.

[Nanda et al. 2016] adotaram uma abordagem baseada em algoritmos de aprendizagem de máquina para predizer padrões de ataques em redes SDN. Os algoritmos foram treinados com base em dados históricos de ataques a rede para identificar conexões com potencial malicioso e potenciais alvos do ataque. Entre os algoritmos aplicados no estudo estão *C4.5, Bayesian Network, Decision Table e Naive-Bayes* para predizer o *host* que será atacado com base nos dados históricos da rede. Na pesquisa de [Mousavi and St-Hilaire 2015], a proposta é detectar ataques DDoS contra controladoras SDN em seus estágios iniciais. Dessa forma, aplicam o conceito de entropia em que define um valor limite com base em simulação. Na etapa seguinte é calculado um valor de entropia baseando-se em uma janela de 50 pacotes. Se o resultado da entropia calculada for menor que o valor limite estipulado, um alerta é enviado indicando a detecção do ataque.

As soluções propostas por estes estudos têm características que demandam conhecimento prévio estado da rede, de registros históricos do fluxo de dados ou ainda de treinamento prévio de algoritmos de aprendizagem de máquina. Assim, inviabiliza a predição de ataques DDoS desconhecidos ou que ainda não foram identificados seus padrões. Os dispositivos que compõem a IoT geram volumes de fluxo de dados significativos. Dessa forma a técnica de predição de ataques apresentada por meio do sistema STARK (*Prediction **S**ys**T**em against DDoS **A**ttack on Netwo**RK***) [Pelloso et al. 2018], contribui com a prevenção de ataques DDoS em redes IoT através da identificação antecipada da iminência de ataques, minimizando ou evitando os seus impactados.

O sistema STARK segue três etapas: (*i*) medições e preparação dos dados, (*ii*) cálculo dos indicadores estatísticos, e (*iii*) análise dos indicadores e emissão de alertas. As entradas do sistema compreendem dados de medições da rede contendo características (*features*), como por exemplo tamanho dos pacotes, número de requisições. Com base nas medições, o sistema extrai as *features* relevantes para o cálculo dos indicadores a fim de compor as séries temporais. A composição das séries temporais se dá através dos dados brutos do fluxo da rede, ou seja, não se limitam de maneira específica ao fluxo proveniente da vítima, para a vítima, ou por um endereço IP e porta específicos. Após a composição das séries temporais, o sistema analisa os valores dos indicadores.

A etapa de **medições e preparação dos dados** engloba *(i) a coleta do fluxo de dados da rede*, *(ii) o estabelecimento do tamanho da janela de tempo* e *(iii) a filtragem dos dados* (extração da característica). Dessa forma, o projeto do sistema STARK requer que a interface de rede do *hardware* subjacente opere em modo promíscuo. As análises ocorrem sobre janelas de coleta de tamanho $N$. O tamanho da janela de dados deve ser definido por tempo, por exemplo, $X$ segundos (ou minutos). O tamanho da janela se ajusta automaticamente, uma vez que pode sofrer variações conforme o volume de dados

do fluxo da rede.

De acordo com os dados contidos na janela de tamanho $N$ , o sistema STARK efetua o processo de cálculo dos indicadores estatísticos. Desta forma, se o sistema emitir a mensagem que indica ausência de recurso computacional para calcular sobre os dados que compõem a janela ou ainda que os dados são insuficientes para alimentar os indicadores, o STARK descarta o conjunto de dados dessa janela e submete uma nova janela de tamanho $N$+$P$ ou $N$-$P$ (onde $P$ é o valor percentual sobre $N$), ampliando ou reduzindo o tamanho da janela conforme a demanda, auto ajustando o valor de $N$. Dessa forma, a filtragem dos dados ocorre sobre cada janela de onde são extraídas as características desejadas (ex. tamanho do pacote). As séries temporais são compostas com base nos dados contidos na janela de tamanho $N$ e as amostras são utilizadas como entrada para a função $F$. A função $F$ extrai as características desejadas através de parâmetros específicos, neste estudo, o tamanho dos pacotes. A saída da função $F$ gera séries temporais evidenciando o tamanho dos pacotes (em *bytes*) relacionado ao respectivo tempo $T$. Assim, essas séries temporais são as saídas da etapa de medição e preparação dos dados e servem como entrada para a etapa de cálculo dos indicadores estatísticos.

Cada série temporal resultante da etapa anterior é utilizada como entrada da etapa de **cálculo dos indicadores estatísticos**. Nesta etapa são calculados os indicadores *taxa de retorno, auto correlação, coeficiente de variação e assimetria*. O processo de cálculo ocorre com a conversão dos valores amostrados na série temporal em um vetor ($V$). O sistema faz uso de um parâmetro $W$ para expressar em porcentagem uma janela de rolamento. Essa janela tem a função de indicar a fração do conjunto de dados contido no vetor $V$ que é utilizado como carga para o início do cálculo dos respectivos indicadores estatísticos. Em geral, é utilizado como valor padrão para a janela de rolamento ($W$) cinquenta por cento das amostras contidas no vetor. Assim, o sistema submete as observações contidas no vetor $v$ e o valor $W$ que determina a janela de rolamento a cada um dos indicadores. O resultado desta etapa mostra a tendência de comportamento de cada um dos indicadores, que pode ser crescente ou decrescente, bem como a intensidade desta tendência através do respectivo *Kendall tau* do indicador estatístico. Tais parâmetros são descritos por $kTR$ (*Kendall tau* da taxa de retorno), $kAC$ (*Kendall tau* da auto correlação), $kCV$ (*Kendall tau* do coeficiente de variação), $kAS$ (*Kendall tau* da assimetria).

Na etapa de **análise dos indicadores e emissão de alerta** o sistema STARK avalia se os indicadores estatísticos apresentam determinado comportamento para emissão do alerta. Quando a taxa de retorno apresenta uma tendência decrescente, a auto correlação e o coeficiente de variação demonstram uma tendência crescente em associação com a assimetria. Esta, ao revelar tanto uma tendência crescente ou decrescente ao deslocar as médias para os extremos, indica que a rede está com dificuldades de ser recuperar de uma perturbação, neste caso, perturbações provocadas por atividades pré-ataque, como exemplo, varreduras na rede. Ao observar este comportamento associado dos indicadores o sistema emite um alerta para informar que a rede está sofrendo ruptura no seu estado metaestável, ou seja, saindo de um estado metaestável para outro, e que por isso há a aproximação de um ataque DDoS. O sistema avalia o $Kendalltau$ dos indicadores para avaliar a intensidade da tendência da ruptura. Os indicadores são avaliados em conjunto conforme a função limiar resultante da Equação 1. Quando valor resultante da função limiar $FL$ for maior ou igual a 0,7 o sistema emite o alerta.

$$FL = \frac{-(kTR) + (kAC) + (kCV) + \sqrt{(kAS^2)}}{nTI} \quad (1)$$

Dessa forma, o sistema STARK reconhece tendências de aproximação de ataques DDoS com base no comportamento de indicadores estatísticos genéricos, ou seja, sem o conhecimento prévio do estado da rede. Por meio da antecipação de um ataque DDoS, ferramentas de monitoramento da rede, podem disparar gatilhos para ajustes das configurações em recursos específicos como exemplo um *firewall*, com objetivo de parametrizar a rede para contingenciar, evitar ou conter o ataque.

### 1.5.2. Reação

Os sistemas de reação a ataques compreendem a detecção do ataque e a reação do sistema contra ele. Para isso, esses sistemas realizam verificações para a detecção de anomalias e assinaturas. Estas verificações compreendem a análise do fluxo de rede, identificação das fontes do tráfego malicioso e a realização do fechamento das portas de acesso ou bloqueio das sessões. Esta subseção apresenta um estudo sobre os sistemas de identificação de dispositivos e do tráfego malicioso junto com as formas de impedir eventuais ameaças.

[Bull et al. 2016b] propuseram o uso de um *gateway* SDN como meio distribuído de monitoramento do tráfego originado e direcionado para dispositivos baseados em IoT. Esse *gateway* detecta comportamentos anômalos e executa uma resposta apropriada, tais como bloquear, encaminhar ou aplicar a Qualidade de Serviço. Essa abordagem abrange ataques baseados em inundações TCP e ICMP. [Nobakht et al. 2016] propõem o IoT-IDM, um *framework* de identificação de intrusão e mitigação para IoT. Este *framework* utiliza a SDN para fornecer proteção em nível de rede para dispositivos inteligentes implantados em ambientes domésticos. O IoT-IDM monitora as atividades de rede dos dispositivos inteligentes planejados e investiga se há alguma atividade suspeita ou mal-intencionada. Uma vez que uma intrusão é detectada, o conjunto de funções propostos pelo *framework* bloqueia o invasor em tempo real. Os autores empregam técnicas personalizadas de aprendizado de máquina para detecção com base em padrões de assinatura de ataques conhecidos. Em [Le et al. 2011], os autores propuseram outra abordagem baseada em especificação, focada na detecção de ataques contra o RPL. O comportamento do RPL foi modelado em uma máquina de estados finitos para monitorar a rede e detectar ações maliciosas. Uma extensão desta proposta foi apresentada em [Le et al. 2016] com o objetivo de suportar arquivos de rastreamento de simulação e gerar a máquina de estados finitos para o protocolo RPL. Além disso os autores também apresentaram uma técnica estatística para traçar perfis de comportamento da rede. Por fim, o autor une todos os estados e transições com as estatísticas correspondentes e implementa como um conjunto de regras para detectar dispositivos atuando maliciosamente. [Cervantes et al. 2015] desenvolveu o sistema de detecção de invasão INTI (**I**nstrusion detection for Si**N**khole attacks over 6LoWPAN for Interne**T** of Th**I**ngs). O INTI visa prevenir, detectar e isolar os efeitos dos ataques de roteamento sumidouro. Para isso, o autor utilizou a combinação de uma estratégia de monitoramento com técnicas de reputação e confiança. Esse conjunto de técnicas oferecem propriedades de auto-organização e auto-reparo. A auto-organização visa a coordenação e cooperação dos dispositivos para a configuração da rede. A se-

gunda propriedade permite a detecção de nós suspeitos e o reagrupamento para manter a estabilidade da rede.

### 1.5.3. Tolerância

Os sistemas desenvolvidos para tolerar ataques utilizam técnicas de mitigação e redistribuição de tráfego para aliviar os efeitos dos ataques. Essas técnicas de mitigação compreendem o controle de portas de acesso, das rotas em rede e agrupamento entre servidores em rede. Esta subseção descreve um estudo sobre sistemas de segurança que implementam soluções tolerantes à os tráfegos maliciosos que um ataque pode causar. Em [Zhang et al. 2016], os autores contribuiram com um esquema para manter o alvo em movimento utilizando salto de portas. Através do *port hopping based DoS mitigation* (PH-DM), o controlador troca as portas de acesso dos serviços em determinados intervalos de tempo, ele também assume o papel de *gateway*, descartando as conexões com as portas dessincronizadas. O esquema implementa um algoritmo de sincronização das portas de acesso do serviço com o usuário.

[Sattar et al. 2016] calculam o limite de tráfego que cada servidor pode suportar e o tráfego que todos os servidores podem suportar juntos. Quando um servidor atinge seu limite de acessos, o controlador envia um sinal de *reset* para os *switches* responsáveis pela rota, que removem todas as tabelas de fluxo, isso força os fluxos a voltarem para o controlador, que vai reinstalá-los de forma adaptativa. [Belyaev and Gaivoronski 2014] propõem um modelo de distribuição de carga na camada 4 do modelo OSI. Visto que uma rede pode ter mais de um caminho para um mesmo destino, os autores mapearam toda a rede em canais e conforme esses canais fossem atingindo um limite de tráfego, o controlador distribui os fluxos pelos demais canais. Esta abordagem, pode solucionar o problema estrangulamento de banda que um ataque pode causar. Porém, esse modelo de distribuição também gera uma grande carga no canal de controle. [Macedo et al. 2016] desenvolveram um protocolo de mitigação dos efeitos dos ataques DDoS contra redes SDN com múltiplos controladores. O protocolo compreende três fases. A primeira fase consiste no monitoramento entre os dispositivos. Nesse monitoramento os controladores trocam mensagens a fim de identificar controladores sobrecarregados por ataques DDoS. Ao identificar, a segunda fase é iniciada, realizando uma eleição entre os controladores com o objetivo de eleger controladores mais ricos em recursos disponíveis a serem compartilhados, esses controladores são organizados em uma hierarquia; o líder, o vice-líder e o grupo de elite. Em seguida, na terceira fase o controlador líder assume parte da carga de trabalho do servidor sob ataque. Caso o controlador líder também entre em estado de sobre carga, o vice-líder realiza uma nova eleição.

## 1.6. Prática

Esta seção detalha um estudo de caso prático envolvendo a criação das topologias de redes SDN e a simulação de ataques DoS contra estas infraestruturas de redes. Os ataques são gerados por meio de *scripts* em *Python* existentes na literatura [Mousavi 2014]. As ferramentas utilizadas no estudo de caso consistem no *Mininet*[3], no *POX*[4] e no *Wireshark*[5].

[3]http://mininet.org

[4]https://openflow.stanford.edu/display/ONL/POX+Wiki

[5]https://www.wireshark.org/

O *Mininet* é um emulador/simulador de rede em modo texto usado na criação de topologias customizadas, sendo compatível com o protocolo *OpenFlow*. O *MiniEdit* consiste em uma interface gráfica para o *Mininet*. O POX *Controller* implementa um controlador SDN por meio da linguagem *Python*, sendo largamente utilizado na academia. O *software Wireshark* possibilita a análise do tráfego de rede gerado entre os *switches* e o controlador SDN. Estas ferramentas possibilitam criar topologias de redes SDN usando uma interface gráfica, gerar carga de trabalho e analisar o tráfego gerado.

### 1.6.1. Simulando uma SDN

Testar soluções para SDNs envolve atividades como simular/emular uma rede, pois experimentações evolve muitas vezes um alto investimento financeiro e de tempo. O *Mininet* foi criado por professores da Universidade de Stanford[6] para a realização de simulações de SDNs e se tornou popular por suportar o protocolo *OpenFlow* e pela eficiência nas configurações, possibilitando chegar mais próximo da realidade. O *Mininet* pode ser utilizando pela linha de comandos ou pela interface gráfica (*Miniedit*). Inicialmente, vamos analisar o *Miniedit* bem como as suas principais funcionalidades. Em seguida, vamos utilizar o Mininet para simular uma SDN.

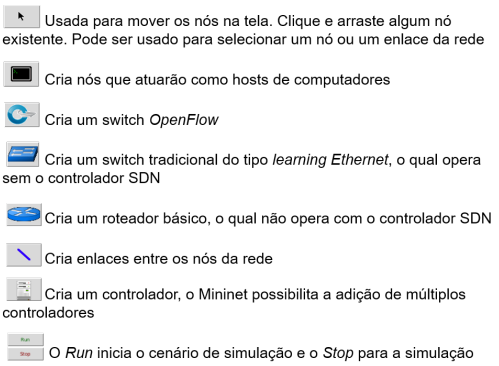


Figura 1.11: Funcionalidades do Miniedit

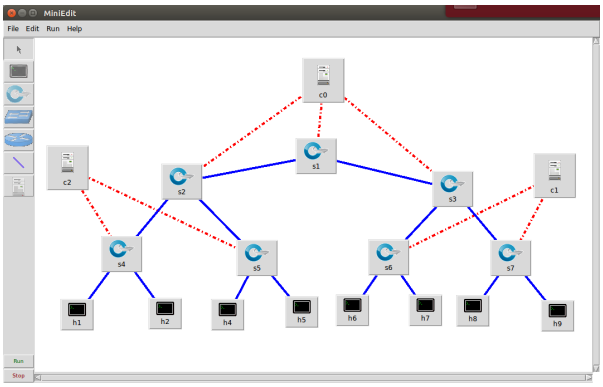


Figura 1.12: Topologia SDN no Miniedit

#### Executando o Miniedit

Para iniciar o *Miniedit* digite o seguinte comando:

```
$ sudo ./mininet/examples/miniedit.py
```

Este comando abrirá a interface do *Miniedit*. O *MiniEdit* tem uma interface de usuário simples. No lado esquerdo da janela, encontra-se uma lista de ícones referentes as funcionalidades e uma barra de menu na parte superior. A Figura 1.11 explica cada funcionalidade da interface inicial.

[6] http://mininet.org/

### Criando uma Topologia

Primeiro, vamos criar uma topologia ao adicionar alguns *hosts* ao cenário de rede conforme a Figura1.12. Para isto, clique no ícone do *host*, mova o ponteiro para o local na tela do *MiniEdit* onde deseja que o *host* apareça e clique novamente. Um *host* aparecerá na tela. Para adicionar mais *hosts*, com o ícone selecionado, basta continuar clicando na posição desejada. Adicione oito *switches* e três controladores usando o mesmo método: Clique na ferramenta *Switch* e adicione *switches*, clique na ferramenta controlador e adicione os controladores. Em seguida, adicione *links* entre os nós na tela. Para realizar esta tarefa, clique na ferramenta *NetLink* (representada por um traço), clique em um nó e arraste o *link* para outro nó. Por exemplo: conecte um *host* a um *switch* ou um *switch* a outro *switch*. Conecte também cada *host* em pelo menos um *switch*. Em seguida, conecte os *switches* juntos para criar uma rede. Por fim, conecte cada switch a um dos controladores. No exemplo mostrado na Figura 1.12 inserimos três controladores. Então, usaremos o controlador de referência *OpenFlow* padrão que vem embutido no *Mininet*. No entanto, precisamos configurar cada controlador para que use uma porta diferente, como mostra a Figura 1.13. Clique com o botão direito do mouse em cada controlador e selecione *Propriedades* no menu exibido. O número da porta padrão para cada controlador é 6633. Altere este valor para que os números das portas usadas pelos controladores c0, c1 e c2 sejam 6633, 6634 e 6635, respectivamente.

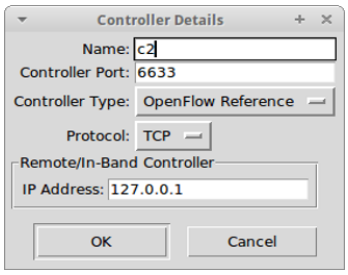


Figura 1.13: Configurando o Controlador

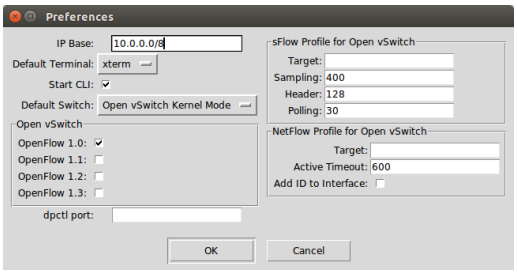


Figura 1.14: Menu *Preferences* do *Mininedit*

### Ativando o CLI

O *Mininet* possibilita a realização de testes via linha de comando (CLI) durante a simulação. Usaremos esta opção em nosso estudo de caso e portanto precisamos habilitá-la. Para definir as preferências do *MiniEdit*, use o comando do menu superior como segue, *Edit* $\rightarrow$ *Preferences*. Abrirá uma caixa de diálogo como mostra na Figura1.14, marque a opção "*Start CLI*", então você terá uma CLI tal como a ilustrada na Figura 1.15.

### Salvando a Topologia e Testando a Simulação

Antes de executar a simulação salve a topologia. Para salvar o arquivo *Mininet Topology* (.mn), clique em *File* na barra de menu superior e selecione *Save* no menu suspenso. Digite um nome de arquivo e salve o arquivo. Por fim, execute o cenário a ser simulado. Para desempenhar esta tarefa, clique no botão *Executar*. Na janela do terminal a partir da qual você iniciou o *MiniEdit*, você verá algumas mensagens mostrando o progresso da inicialização da simulação e em seguida, o terminal na linha de comandos do *Mininet*. Para

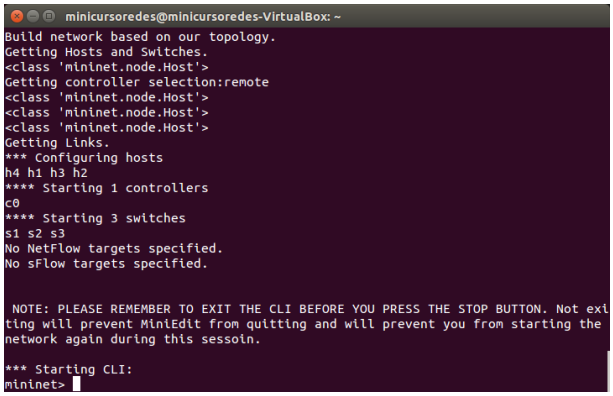


Figura 1.15: CLI do *Mininet*

verificar o funcionamento do *Mininet* vamos executar alguns comandos. Primeiramente vamos executar o comando "*pingall*", como segue:

```
mininet> pingall
```

Através deste comando todos os *hosts* enviam *pings* entre si. Através dos *pings*, os *switches* que, até então não tinham recebido nenhuma regra de fluxo, solicitam a instalação de regras para o controlador. Para verificar as regras instaladas nos *switches* é necessária a abertura de terminal virtual da máquina raiz da simulação. Para abrir o terminal o *miniedit* disponibiliza uma opção no menu superior em "*Run*"→ "*Root Terminal*". Ao executar o terminal, para solicitar a tabela de regras do *switch* 1 execute o comando abaixo.

```
# ovs-ofctl dump-flows s1
```

### Simulando a Quebra de um Link

O *Miniedit* permite também a iteração com a interface gráfica durante a simulação. Para simular a quebra de um *link*, volte para a interface gráfica. A seguir com o botão direito do mouse selecione um *link* e clique na opção "*link down*", como mostra a Figura 1.16. Volte na CLI e execute um teste através do *ping*. Para executar o comando *ping* de forma individual, execute o seguinte comando:

```
mininet> h1 ping h8
```

### Terminal Virtual dos *Hosts*

As redes simuladas pelo *Mininet* criam *hosts* reais com a capacidade de executar códigos e aplicativos de rede padrão *Unix/Linux*, pois ele virtualiza o *kernel* e a pilha de rede reais desses sistemas. Para acessar terminal virtual do *host* 1 basta executar o comando abaixo:

```
mininet> xterm h1
```

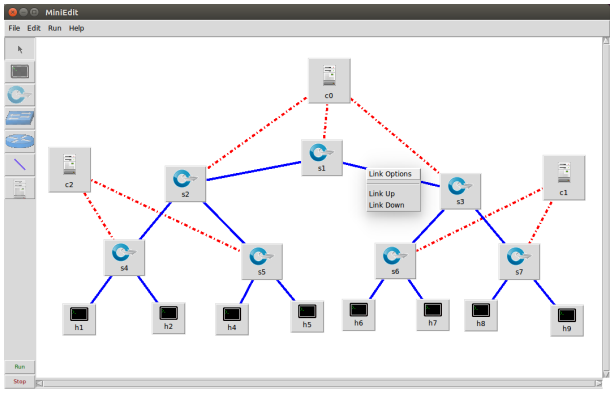


Figura 1.16: Simulando a Quebra de um *Link*

No terminal, verifique as placas de rede do *host* com o comando "*ifconfig*". Note que o *host* contém as configurações de rede semelhantes a de um sistema Linux normal. Isso capacita a realização de simulações realistas através do *Mininet*.

### Topologias Customizadas

Apesar do *Miniedit* oferecer uma interface gráfica que simplifica muitas das atividades, ele não oferece todas as funcionalidades do *Mininet*. Nesta parte da simulação vamos encerrar a arquitetura criada. Para isso vamos encerrar a linha de comandos do *Mininet* com o comando abaixo. Em seguida, vamos parar a simulação através da interface gráfica, clicando no "*Stop*"(canto inferior esquerdo). Por fim, clique no menu superior em "*File*"→ "*New*".

```
mininet> exit
```

Abra novamente a topologia salva no momento anterior ("*File*"→ "*Open*"). O *Mininet* permite a criação de *scripts* customizados que permitem um nível de customização mais elevado. Com a topologia sugerida criada, ative a linha de comandos como anteriormente (1.14). Mova o mouse até o menu superior, clique nas opções "*File*"→ "*Save Level 2 Script*". Digite o nome do arquivo como "teste.py"no diretório "*home/minicursoredes/*". Por fim, vamos abrir um novo terminal, dar as devidas autorizações para o *script* salvo e executar a simulação. Os comandos ficam como segue:

```
$ sudo chmod +x teste.py
$ sudo ./teste.py
```

Note que o *script* salvo configura a topologia e o terminal do *Mininet*. A vantagem de executar um *script* de topologia personalizado *Mininet* é que você pode editar o *script* originalmente criado pelo *MiniEdit* para criar cenários mais complexos e usar os recursos do *Mininet* não suportados pelo *MiniEdit*. Execute os testes sugeridos anteriormente como o comando "*pingall*". Por fim, encerre a topologia com o comando abaixo:

```
mininet> exit
```

### Topologias Instantâneas

Outra funcionalidade do *Mininet* é a possibilidade de criar topologias instantâneas. Isso permite a realização de testes rápidos que compreendam topologias simples. O comando abaixo representa uma topologia em árvore com profundidade três e dois filhos por nó. Por padrão ele inicializa a linha de comandos e o controlador embutido no *Mininet*. A opção "*–controller remote*"significa que a topologia vai esperar um controlador se conectar no *ip* 127.0.0.1, na porta 6633 (lembre-se que a barra invertida significa um espaço).

```
$ sudo mn --topo tree,depth=3,fanout=2 --controller
\ remote,ip=127.0.0.1,port=6633
```

### Controlador

Os controladores desempenham o papel mais importante na estrutura de rede. No entanto existem diversas propostas para sistemas controladores SDN. No primeiro momento da parte prática utilizamos o controlador nativo do *Mininet*. Nesta seção, para tornar a simulação mais realista, vamos simular uma arquitetura com o controlador externo. Para isso, vamos utilizar o controlador POX. O Controlador POX[7] oferece uma interface de comunicação para SDN. Para isso, ele suporta os protocolos *OpenFlow* e OVSDB. O objetivo geral dos controladores SDN, incluindo o Controlador POX, é permitir o desenvolvimento de aplicações para SDN. Neste caso, aplicações desenvolvidas em *Python*. O controlador POX, vem com programas adicionais que implementam funcionalidades de redes SDN, como a instalação de regras de encaminhamento de pacotes em *switches* SDN.

### Executando o Controlador

Nesta parte vamos simular uma topologia sob controle do controlador POX. Primeiramente inicie uma topologia instantânea como na demonstrado anteriormente. Execute alguns testes com o comando *ping*. Note que o comando não funciona, devido a falta do controlador. Verifique também que as tabelas de fluxos dos *Switches* estão vazias. Para executar o controlador com o componente de instalação de regras dos *switches*, inicie um terminal em paralelo e digite o comando abaixo (Note que é um único comando, a barra invertida significa um espaço). O comando executa o controlador POX com o os componentes de encaminhamento, incluindo a demonstração detalhada dos *logs* no terminal. A tela fica como mostra a Figura 1.17. Em seguida, volte ao terminal que está executando a topologia e realize os testes novamente. Note que o comando "*ping*" voltou a funcionar, também que as tabelas de fluxos dos *switches* foram preenchidas. Verifique também o terminal do controlador mostrando as ações que estão sendo executadas.

```
$ sudo ~/pox/pox.py forwarding.l3_learning info.packet_dump
\samples.pretty_log log.level --DEBUG
```

[7] https://noxrepo.github.io/pox-doc/html/

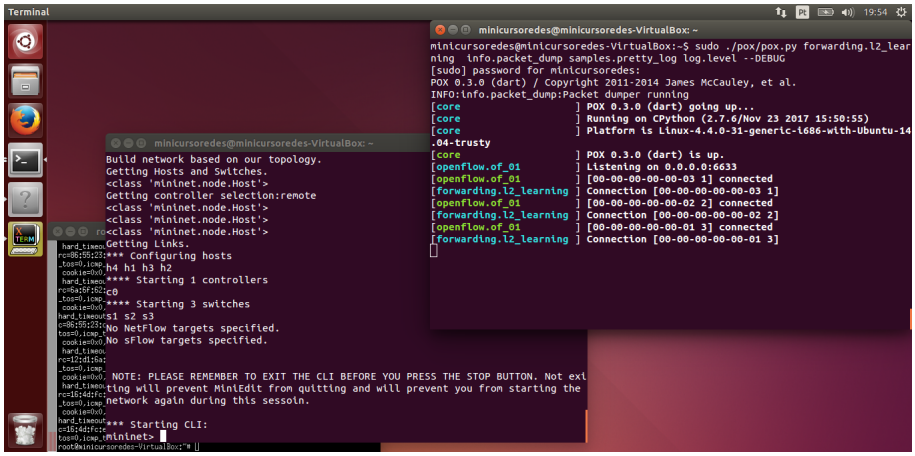


Figura 1.17: Terminal do Controlador

### Wireshark

A simulação de uma rede também envolve a captura e análise dos pacotes que trafegam na rede simulada. O *Wireshark*consiste em um analisador de protocolos capaz de capturar de pacotes *OpenFlow*. Esta ferramenta também possibilita salvar capturas para a realização de análises futuras. Primeiramente vamos iniciar uma simulação através da topologia instantânea de rede e o controlador POX conforme explicado anteriormente. Em seguida, abra um terminal virtual para o *host "h1"* (*xterm h1*) e digite o comando:

```
# wireshark
```

A interface gráfica do *Wireshark* inicializa. O Wireshark oferece diversas opções de captura. Para iniciar uma captura, selecione a interface de rede "*h1-eth0*" e clique na opção "*Start*". Por fim, realize alguns testes na topologia com o comando *ping*. Note que os pacotes são capturados pela linha do tempo do *Wireshark*[8].

### 1.6.2. Simulando e Analisando os Dados de um Ataque de Negação de Serviço

O desenvolvimento de soluções de segurança para redes de computadores envolve também o desenvolvimento e simulação dos ataques almejados pelo desenvolvedor. Um ataque DoS em um sistema simples pode ser desenvolvido em poucas linhas de código. Nesta seção vamos simular uma versão simples de um DoS. O *script* do ataque está disponível na pasta "Documentos"do usuário "minicursoredes"na máquina virtual disponibilizada. A função deste *script* consiste basicamente em falsificar os *IPs* do cabeçalho de pacotes UDP e os enviar repetidamente. Os *IPs* falsificados geram um trafego desconhecido pelo *switches*, forçando-os a enviarem requisições ao controlador. Para simular este ataque primeiramente inicie uma topologia instantânea, o controlador POX e o *Wireshark* como na simulação anterior. Em seguida, inicie um terminal virtual para o *host "h2"* (*xterm h2*). Por fim, inicie o *script* do ataque através do comando abaixo no terminal virtual aberto. Note o aumento na atividade dos *logs* do controlador. Execute alguns testes com comando "*ping*"e você perceberá a queda de desempenho nas respostas, indicando os efeitos do ataque DoS.

[8]Não encerre a execução para a próxima simulação

```
# ./Documentos/ataque.py
```

### Analisando Dados da Rede em Busca de Padrões

A análise de dados da rede em busca de padrões é determinada pelas etapas de (*i*) medições e preparação dos dados, do (*ii*) cálculo dos indicadores estatísticos, e da (*iii*) análise dos indicadores e emissão de alertas [Pelloso et al. 2018]. Na primeira etapa o *TcpDump* efetua o registro do fluxo de dados bruto da rede, resultando em conjuntos de dados definidos em janelas de tempo $T$ pré-determinadas.Portanto, assume-se a interface de rede operando em modo promíscuo. A janela de tempo tem como valor padrão 60 segundos, porém o sistema avalia o volume de dados gerado pelo fluxo da rede, ajustando-se de acordo com a demanda. Se as amostras de dados da janela for insuficiente o sistema expande o tempo. Contudo, se a quantidade de observações disponível na janela for grande e comprometer a capacidade de análise, o sistema realiza o ajuste, diminuindo o tempo da janela, conforme exposto na Subseção 1.6.2. Após o registro das amostras em janelas, o sistema STARK realiza a extração da característica sobre a qual os dados são analisados, utilizando o *tshark*. Neste caso, representado pelo tamanho do pacotes *versus* o tempo. Este processo resulta em uma estrutura contendo a respectiva série temporal. Com base nas séries temporais geradas, a etapa de cálculo dos indicadores estatísticos efetua o cálculo da **taxa de retorno, autocorrelação, coeficiente de variação** e **assimetria**, aplicando a biblioteca *Early Warning Signals Toolbox - EWS*, implementada sobre a ferramenta R. Para calcular os indicadores, a EWS tem como parâmetro a janela de rolamento, neste caso representada por $W$. Em geral, a janela $W$ é fixada em cinquenta por cento. Esta indica a quantidade de amostras da série temporal usada como carga do indicador. O resultado desta etapa expõe a tendência e intensidade do comportamento dos indicadores estatísticos. O resultado dessa etapa é submetida à análise dos indicadores e emissão de alertas. Nesta etapa o comportamento dos indicadores é avaliado.

A operacionalização das etapas do sistema ocorre ao assumir a interface de rede em modo promíscuo, assim os dados do fluxo da rede são registrados em formato pcap. A etapa (*i*) medições e preparação dos dados consiste no registro dos dados do fluxo da rede. Para fins didáticos assumimos que o resultado dessa etapa é um *log* do fluxo da rede em formato *pcap*, conforme instruções a seguir:

```
# tcpdump -i eth0 -w capture_file
```

Ainda nesta etapa é realizada a preparação dos dados, que inclui o fracionamento do *log* em arquivos de tamanhos adequados à extração da característica a ser avaliada pelos indicadores estatísticos. Neste caso, o tamanho dos pacotes foi definido em 60 segundos. O fracionamento do *log pcap* é realizado aplicando a ferramenta *editcap* e a extração da característica para cada pacotes para submeter aos indicadores é realizada pelo *tshark*.

```
# editcap -i 60 -F pcap capture_file output-packets-XX.pcap

# tshark -r output-packets-XX.pcap -t e -T fields -e frame.time_relative -e frame.len
-E header=n -E quote=n -E occurrence=f > output.dat
```

A etapa de análise dos indicadores consiste em submeter os pacotes nomeados como .dat resultantes da etapa anterior aos indicadores estatísticos submetendo a biblioteca EWS por meio do software R, como segue:

```
data <- read.table(output.dat, header = FALSE, sep = "", col.names = c('time', 'frame_size'))
timeseries <- ts(data)
generic_ews(timeseries, winsize = 50)
```

A Figura 1.18 ilustra o resultado desta etapa demonstrando o comportamento esperado dos indicadores estatísticos indicando a aproximação de ruptura nos dados, ou seja, a aproximação de um ataque DDoS. Na etapa de (*iii*) análise dos indicadores e emissão de alertas, o sistema calcula o limiar estipulado para a emissão do alerta com base nos valores do *Kendall tau* de cada um dos indicadores e, ao superar o limiar o alerta é emitido.

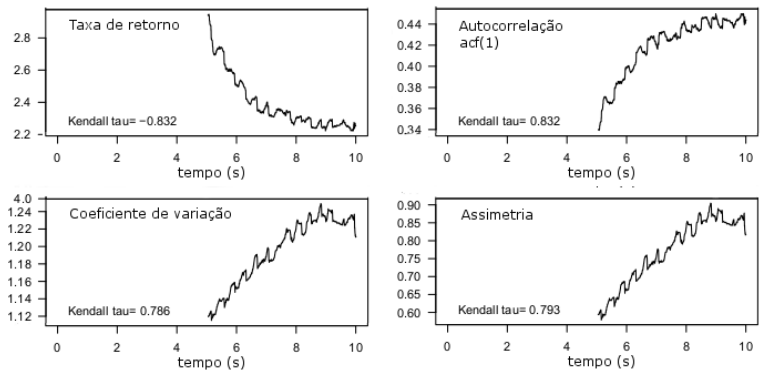


Figura 1.18: Comportamento esperado demonstrando a predição no conjunto de dados

## 1.7. Conclusões

As redes IoT baseadas em SDN vêm se consolidando como uma solução dos principais desafios característicos da IoT tradicional e da IoE, tais como heterogeneidade, escalabilidade e *Big Data*. No entanto, esta integração torna-se suscetível a ameaças contra a confidencialidade, a integridade e a disponibilidade dos dados, serviços e das aplicações em rede. Este capítulo apresentou as principais ameaças de segurança e defesas em redes IoT baseadas em SDN. Para alcançar este objetivo, o capítulo seguiu uma abordagem teórico/prática. Na parte teórica foram apresentados os conceitos, terminologias das redes IoT e SDN. Além disso, o capítulo detalhou as principais ameaças e defesas das redes IoT baseadas em SDN. Estes ataques foram apresentados seguindo as camadas da arquitetura das IoTs baseadas em SDNs, destacando suas especialidades. As soluções de defesa foram divididas em sistemas de prevenção, reação e tolerância, as quais apresentam diferentes técnicas para evitar, identificar ou mitigar os ataques. Na parte prática foi realizado um estudo de caso envolvendo a simulação de uma SDN sob ataque de negação de serviço, a captura e análise do tráfego gerado pelo ataque. O simulador *Mininet* foi usado para criação das topologias, juntamente com a ferramenta *MiniEdit*. Os *scripts* de ataque foram extraídos de trabalhos encontrados na literatura da área. A análise de dados foi conduzida por meio do sistema STARK em combinação com funções do *software* R.

## Referências


[Myd 2009] (2009). Lazy Hacker and Little Worm Set Off Cyberwar Frenzy. https://www.computerworld.com/article/2574799/security0/mydoom-lesson--take-proactive-steps-to-prevent-ddos-attacks.html. Último Acesso: Agosto de 2018.

[Wik 2010] (2010). Wikileaks supporters disrupt Visa and MasterCard sites in 'Operation Payback'. http://www.theguardian.com/world/2010/dec/08/wikileaks-visa-mastercard-operation-payback. Último Acesso: Maio de 2018.

[Akamai 2016] Akamai (2016). Akamai's [state of the internet]/security q3 2016 report. https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q3-2016-state-of-the-internet-security-report.pdf. Último Acesso: Agosto de 2018.

[Alaba et al. 2017] Alaba, F. A., Othman, M., Hashem, I. A. T., and Alotaibi, F. (2017). Internet of Things security: A survey. *Journal of Network and Computer Applications*, 88:10 – 28.

[Alkhatib et al. 2015] Alkhatib, H., Faraboschi, P., Frachtenberg, E., Kasahara, H., Lange, D., Laplante, P., Merchant, A., Milojicic, D., and Schwan, K. (2015). What Will 2022 Look Like? the IEEE CS 2022 Report. *IEEE Computer*, 48(3):68–76.

[Alrawais et al. 2017] Alrawais, A., Alhothaily, A., Hu, C., and Cheng, X. (2017). Fog computing for the Internet of Things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42.

[Atzori et al. 2017] Atzori, L., Girau, R., Martis, S., Pilloni, V., and Uras, M. (2017). A SIoT-aware approach to the resource management issue in mobile crowdsensing. In *Innovations in Clouds, Internet and Networks*, Páginas 232–237. IEEE.

[Atzori et al. 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15):2787 – 2805.

[Azzouni and Pujolle 2017] Azzouni, A. and Pujolle, G. (2017). A long short-term memory recurrent neural network framework for network traffic matrix prediction.

[Bandyopadhyay and Sen 2011] Bandyopadhyay, D. and Sen, J. (2011). Internet of Things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1):49–69.

[Banks and Gupta 2014] Banks, A. and Gupta, R. (2014). Mqtt version 3.1. 1. *OASIS standard*, 29.

[Bartholemy and Chen 2015] Bartholemy, A. and Chen, W. (2015). An Examination of Distributed Denial of Service Attacks. In *IEEE International Conference on Electro/Information Technology*, Páginas 274–279.

[Bassi et al. 2013] Bassi, A., Bauer, M., Fiedler, M., Kramp, T., Van Kranenburg, R., Lange, S., and Meissner, S. (2013). *Enabling things to talk*. Springer.

[Belyaev and Gaivoronski 2014] Belyaev, M. and Gaivoronski, S. (2014). Towards load balancing in SDN-networks during DDoS-attacks. In *Science and Technology Conference (Modern Networking Technologies)*, Páginas 1–6. IEEE.

[Bera et al. 2017] Bera, S., Misra, S., and Vasilakos, A. V. (2017). Software-Defined Networking for Internet of Things: A survey. *IEEE Internet of Things Journal*, 4(6):1994–2008.

[Bernabe et al. 2014] Bernabe, J. B., Hernández, J. L., Moreno, M. V., and Gomez, A. F. S. (2014). Privacy-preserving security framework for a social-aware Internet of Things. In *International conference on ubiquitous computing and ambient intelligence*, Páginas 408–415. Springer.

[Bernabe et al. 2016] Bernabe, J. B., Ramos, J. L. H., and Gomez, A. F. S. (2016). TACIoT: multidimensional trust-aware access control system for the Internet of Things. *Soft Computing*, 20(5):1763–1779.

[Bizanis and Kuipers 2016] Bizanis, N. and Kuipers, F. A. (2016). SDN and virtualization solutions for the Internet of Things: A survey. *IEEE Access*, 4:5591–5606.

[Botta et al. 2016] Botta, A., De Donato, W., Persico, V., and Pescapé, A. (2016). Integration of cloud computing and Internet of Things: a survey. *Future Generation Computer Systems*, 56:684–700.

[Bradley et al. 2013] Bradley, J., Reberger, C., Dixit, A., Gupta, V., and Macaulay, J. (2013). Internet of Everything (IoE): Top 10 insights from cisco's IoE value at stake analysis for the public sector. Analysis, Cisco. Último Acesso: Agosto de 2018.

[Bull et al. 2016a] Bull, P., Austin, R., Popov, E., Sharma, M., and Watson, R. (2016a). Flow based security for IoT devices using an SDN gateway. In *International Conference on Future Internet of Things and Cloud*, Páginas 157–163.

[Bull et al. 2016b] Bull, P., Austin, R., Popov, E., Sharma, M., and Watson, R. (2016b). Flow based security for IoT devices using an SDN gateway. In *Future Internet of Things and Cloud*, Páginas 157–163. IEEE.

[Campbell et al. 1999] Campbell, A. T., Katzela, I., Miki, K., and Vicente, J. (1999). Open Signaling for ATM, Internet and Mobile Networks. *ACM SIGCOMM Computer Communication Review*, 29(1):97–108.

[Casado et al. 2007] Casado, M., Freedman, M. J., Pettit, J., Luo, J., McKeown, N., and Shenker, S. (2007). Ethane: Taking Control of the Enterprise. *ACM SIGCOMM Computer Communication Review*, 37(4):1–12.

[CERT.br 2018] CERT.br (2018). Estatísticas do cert.br – incidentes de segurança. `https://www.cert.br/stats/incidentes/`. Último Acesso: Agosto de 2018.

[Cervantes et al. 2015] Cervantes, C., Poplade, D., Nogueira, M., and Santos, A. (2015). Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things. In *IM*, Páginas 606–611.

[Cisco 2018] Cisco (2018). Cisco annual security report. `https://www.cisco.com/c/dam/assets/global/DE/unified_channels/partner_with_cisco/newsletter/2015/edition2/download/cisco-annual-security-report-2015-e.pdf`. Último Acesso: Agosto de 2018.

[Cisco 2016] Cisco, C. V. N. I. (2016). Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021 white paper. Último Acesso: Agosto de 2018.

[Criscuolo 2000] Criscuolo, P. (2000). Distributed Denial of Service, Tribe Flood Network 2000, and Stacheldraht CIAC-2319, Department of Energy Computer incident Advisory Capability (CIAC). Technical report, UCRL-ID-136939, Rev. 1., Lawrence Livermore National Laboratory.

[Douligeris and Mitrokotsa 2004] Douligeris, C. and Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5):643 – 666.

[Erickson 2013] Erickson, D. (2013). The Beacon OpenFlow Controller. In *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Páginas 13–18, New York, NY, USA. ACM.

[Evans 2012] Evans, D. (2012). The Internet of Everything: How more relevant and valuable connections will change the world. *Cisco IBSG*, 2012:1–9.

[Farhady et al. 2015] Farhady, H., Lee, H., and Nakao, A. (2015). Software-Defined Networking: A Survey. *Computer Networks*, 81:79 – 95.

[Farris et al. 2018] Farris, I., Taleb, T., Khettab, Y., and Song, J. S. (2018). A survey on emerging SDN and NFV security mechanisms for IoT systems. *IEEE Communications Surveys & Tutorials*.

[Feamster et al. 2014] Feamster, N., Rexford, J., and Zegura, E. (2014). The Road to SDN: An Intellectual History of Programmable Networks. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98.

[Feng et al. 2017] Feng, S., Setoodeh, P., and Haykin, S. (2017). Smart home: Cognitive interactive people-centric Internet of Things. *IEEE Communications Magazine*, 55(2):34–39.

[Fielding et al. 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1.

[Flauzac et al. 2015] Flauzac, O., González, C., Hachani, A., and Nolot, F. (2015). SDN based architecture for IoT and improvement of the security. In *International Conference on Advanced Information Networking and Applications Workshops*, Páginas 688–693.

[Goodin 2016] Goodin, D. (2016). Record-breaking ddos reportedly delivered by >145k hacked cameras. `https://arstechnica.com/information-technology/2016/09/botnet-of-145k-cameras-reportedly-deliver-internets-biggest-ddos-ever/`. Último Acesso: Agosto de 2018.

[Greenberg et al. 2005] Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., and Zhang, H. (2005). A Clean Slate 4D Approach to Network Control and Management. *ACM SIGCOMM Computer Communication Review*, 35(5):41–54.

[Gude et al. 2008] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). NOX: Towards an Operating System for Networks. *ACM SIGCOMM Computer Communication Review*, 38(3).

[Holgado et al. 2017] Holgado, P., VILLAGRA, V. A., and Vazquez, L. (2017). Real-time multistep attack prediction based on hidden markov models. *IEEE Transactions on Dependable and Secure Computing*.

[Iannacci 2018] Iannacci, J. (2018). Internet of Things (IoT); Internet of Everything (IoE); tactile Internet; 5g – a (not so evanescent) unifying vision empowered byEH-MEMS (energy harvesting MEMS) and RF-MEMS (radio frequency MEMS). *Sensors and Actuators A: Physical*, 272:187 – 198.

[J. D. Case and M. Fedor and M. L. Schoffstall and J. Davin 1990] J. D. Case and M. Fedor and M. L. Schoffstall and J. Davin (1990). Simple Network Management Protocol (SNMP).

[Khan et al. 2012] Khan, R., Khan, S. U., Zaheer, R., and Khan, S. (2012). Future Internet: the Internet of Things architecture, possible applications and key challenges. In *Frontiers of Information Technology*, Páginas 257–260. IEEE.

[Kitten 2013] Kitten, T. (2013). DDoS: Lessons from Phase 2 Attacks. `http://www.bankinfosecurity.com/ddos-attacks-lessons-from-phase-2-a-5420`. Último Acesso: Maio de 2018.

[Kwon et al. 2017] Kwon, D., Kim, H., An, D., and Ju, H. (2017). DDoS attack volume forecasting using a statistical approach. In *IFIP/IEEE Symposium on Integrated Network and Service Management*, Páginas 1083–1086. IEEE.

[Lamaazi et al. 2014] Lamaazi, H., Benamar, N., Jara, A. J., Ladid, L., and Ouadghiri, D. E. (2014). Challenges of the Internet of Things: IPv6 and Network Management. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Páginas 328–333.

[Le et al. 2016] Le, A., Loo, J., Chai, K. K., and Aiash, M. (2016). A specification-based IDS for detecting attacks on RPL-based network topology. *Information*, 7(2):25.

[Le et al. 2011] Le, A., Loo, J., Luo, Y., and Lasebae, A. (2011). Specification-based IDS for securing RPL from topology attacks. In *IFIP Wireless Days*, Páginas 1–3. IEEE.

[Lee et al. 2014] Lee, J., Uddin, M., Tourrilhes, J., Sen, S., Banerjee, S., Arndt, M., Kim, K.-H., and Nadeem, T. (2014). mesdn: Mobile extension of sdn. In *International workshop on Mobile cloud computing & services*, Páginas 7–14. ACM.

[Leslie et al. 2015] Leslie, I., Crosby, S., and Rooney, S. (2015). Devolved Control of ATM Networks. `https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/dcan/#pub`. Último Acesso: Agosto de 2018.

[Li et al. 2017] Li, C., Qin, Z., Novak, E., and Li, Q. (2017). Securing SDN Infrastructure of IoT–Fog Networks From MitM Attacks. *IEEE Internet of Things Journal*, 4(5):1156–1164.

[Lima et al. 2009] Lima, M. N., Dos Santos, A. L., and Pujolle, G. (2009). A survey of survivability in mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 11(1):66–77.

[Macedo et al. 2016] Macedo, R., de Castro, R., Santos, A., Ghamri-Doudane, Y., and Nogueira, M. (2016). Self-organized SDN controller cluster conformations against DDoS attacks effects. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, Páginas 1–6. IEEE.

[Mason 2018] Mason, J. (2018). Cyber security statistics. `https://thebestvpn.com/cyber-security-statistics-2018/`. Último Acesso: Agosto de 2018.

[Mckeay 2016] Mckeay, M. (2016). 620 gbps attack post mortem. `https://blogs.akamai.com/2016/10/620-gbps-attack-post-mortem.html`. Último Acesso: Maio de 2018.

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

[Miraz et al. 2015] Miraz, M. H., Ali, M., Excell, P. S., and Picking, R. (2015). A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT). In *Internet Technologies and Applications (ITA)*, Páginas 219–224.

[Mitchell et al. 2013] Mitchell, S., Villa, N., Stewart-Weeks, M., and Lange, A. (2013). The Internet of Everything for cities: connecting people, process, data and things to improve the livability of cities and communities. *San Jose: Cisco*.

[Moore et al. 2001] Moore, J. T., Hicks, M., and Nettles, S. (2001). Practical programmable packets. In *IEEE INFOCOM*, volume 1, Páginas 41–50 vol.1.

[Mousavi 2014] Mousavi, S. M. (2014). Early detection of DDoS Attacks in Software Defined Networks Controller. Master's thesis, Carleton University, Ottawa, Ontario, Canada. Último Acesso: Agosto de 2018.

[Mousavi and St-Hilaire 2015] Mousavi, S. M. and St-Hilaire, M. (2015). Early detection of DDoS attacks against SDN controllers. In *Computing, Networking and Communications*, Páginas 77–81. IEEE.

[Nanda et al. 2016] Nanda, S., Zafari, F., DeCusatis, C., Wedaa, E., and Yang, B. (2016). Predicting network attack patterns in SDN using machine learning approach. In *IEEE Conference on Network Function Virtualization and Software Defined Networks*, Páginas 167–172. IEEE.

[Naraine 2002] Naraine, R. (2002). Massive DDoS Attack Hit DNS Root Servers. `http://www.cs.cornell.edu/people/egs/beehive/rootattack.html`. Último Acesso: Agosto de 2018.

[Nazario 2015] Nazario, J. (2015). BlackEnergy DDoS Bot Analysis - Arbor Networks. `http://atlas-public.ec2.arbor.net/docs/BlackEnergy+DDoS+Bot+Analysis.pdf`. Último Acesso: Maio de 2018.

[Newman 2018] Newman, L. (2018). Github survived the biggest DDoS attack ever recorded. `https://www.wired.com/story/github-ddos-memcached/`. Último Acesso: Agosto de 2018.

[Ng 2010] Ng, E. (2010). Maestro: A System for Scalable OpenFlow Control. Technical report, TSEN Maestro-Technical Report TR10-08, Rice University.

[Nieminen et al. 2015] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and Gomez, C. (2015). Ipv6 over bluetooth(r) low energy. RFC 7668, IETF. Último Acesso: Agosto de 2018.

[Nijim et al. 2017] Nijim, M., Albataineh, H., Khan, M., and Rao, D. (2017). Fastdetict: A Data Mining Engine for predecting and preventing DDoS attacks. In *Technologies for Homeland Security*, Páginas 1–5. IEEE.

[Nobakht et al. 2016] Nobakht, M., Sivaraman, V., and Boreli, R. (2016). A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow. In *Availability, Reliability and Security*, Páginas 147–156. IEEE.

[Nunes et al. 2014] Nunes, B., Mendonca, M., Nguyen, X.-N., Obraczka, K., and Turletti, T. (2014). A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634.

[Olson 2014] Olson, P. (2014). The largest cyber attack in history has been hitting hong kong sites. `https://www.forbes.com/sites/parmyolson/2014/11/20/the-largest-cyber-attack-in-history-has-been-hitting-hong-kong-sites/#782acbf638f6`. Último Acesso: Agosto de 2018.

[Open Networking Foundation 2012] Open Networking Foundation (2012). Software-Defined Networking: The New Norm for Networks. White paper, Open Networking Foundation, Palo Alto, CA, USA.

[Palattella et al. 2013] Palattella, M. R., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L. A., Boggia, G., and Dohler, M. (2013). Standardized protocol stack for the Internet of (important) Things. *IEEE communications surveys & tutorials*, 15(3):1389–1406.

[Paxson 2001] Paxson, V. (2001). An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. *ACM SIGCOMM Computer Communication Review*, 31(3):38–47.

[Pelloso et al. 2018] Pelloso, M., Vergu, A., Santos, A., Nogueira, M., et al. (2018). Um sistema autoadaptável para predição de ataques ddos fundado na teoria da metaestabilidade. In *Simpósio Brasileiro de Redes de Computadores (SBRC)*, volume 36.

[Pradhan et al. 2018] Pradhan, M., Fuchs, C., and Johnsen, F. T. (2018). A survey of applicability of military data model architectures for smart city data consumption and integration. In *IEEE World Forum on Internet of Things*, Páginas 129–134. IEEE.

[Prasad et al. 2014] Prasad, K. M., Reddy, A. R. M., and Rao, K. V. (2014). DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms - A Survey. *Global Journal of Computer Science and Technology*, 14(7).

[Prince 2013] Prince, M. (2013). The DDoS That Almost Broke the Internet. `https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/`. Último Acesso: Agosto de 2018.

[R. Enns 2006] R. Enns (2006). NETCONF Configuration Protocol. rfc 4741. obsoleto pela rfc 6241.

[Rajan et al. 2017] Rajan, A., Jithish, J., and Sankaran, S. (2017). Sybil attack in IoT: Modelling and defenses. In *Advances in Computing, Communications and Informatics*, Páginas 2323–2327. IEEE.

[Rexford et al. 2004] Rexford, J., Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Xie, G., Zhan, J., and Zhang, H. (2004). Network-wide Decision Making: Toward a Wafer-thin Control Plane. In *Proceedings of HotNets*, Páginas 59–64.

[Santos et al. 2017] Santos, A. A., Nogueira, M., and Moura, J. M. F. (2017). A stochastic adaptive model to explore mobile botnet dynamics. *IEEE Communications Letters*, 21(4).

[Sattar et al. 2016] Sattar, D., Matrawy, A., and Adeojo, O. (2016). Adaptive Bubble Burst (ABB): Mitigating DDoS attacks in Software-Defined Networks. In *Telecommunications Network Strategy and Planning Symposium*, Páginas 50–55. IEEE.

[Schatten et al. 2016a] Schatten, M., Ševa, J., and Tomičić, I. (2016a). A roadmap for scalable agent organizations in the Internet of Everything. *Journal of Systems and Software*, 115:31–41.

[Schatten et al. 2016b] Schatten, M., Ševa, J., and Tomičić, I. (2016b). A roadmap for scalable agent organizations in the Internet of Everything. *Journal of Systems and Software*, 115:31 – 41.

[Sciancalepore et al. 2017] Sciancalepore, S., Piro, G., Caldarola, D., Boggia, G., and Bianchi, G. (2017). OAuth-IoT: An access control framework for the Internet of Things based on open standards. In *IEEE Symposium on Computers and Communications*, Páginas 676–681. IEEE.

[Scott-Hayward et al. 2016] Scott-Hayward, S., Natarajan, S., and Sezer, S. (2016). A survey of security in Software Defined Networks. *IEEE Communications Surveys & Tutorials*, 18(1):623–654.

[Sezer et al. 2013] Sezer, S., Scott-Hayward, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2013). Are We Ready for SDN? Implementation Challenges for Software-Defined Networks. *IEEE Communications Magazine*, 51(7):36–43.

[Shelby et al. 2014] Shelby, Z., Hartke, K., and Bormann, C. (2014). The constrained application protocol (coap). Technical report.

[Sonar and Upadhyay 2014] Sonar, K. and Upadhyay, H. (2014). A survey: DDoS attack on Internet of Things. *International Journal of Engineering Research and Development*, 10(11):58–63.

[Sudworth 2009] Sudworth, J. (2009). New 'cyber attacks' hit s Korea. `http://news.bbc.co.uk/2/hi/asia-pacific/8142282.stm`. Último Acesso: Agosto de 2018.

[Symatec 2018] Symatec (2018). 2018 Internet Security Threat Report. `https://www.symantec.com/security-center/threat-report`. Último Acesso: Agosto de 2018.

[Tayyaba et al. 2017] Tayyaba, S. K., Shah, M. A., Khan, O. A., and Ahmed, A. W. (2017). Software Defined Network (SDN) Based Internet of Things (IoT): A Road Ahead. In *Proceedings of the International Conference on Future Networks and Distributed Systems*, page 10. ACM.

[Tennenhouse et al. 1997] Tennenhouse, D. L., Smith, J. M., Sincoskie, W. D., Wetherall, D. J., and Minden, G. J. (1997). A Survey of Active Network Research. *IEEE Communications Magazine*, 35(1):80–86.

[Tennenhouse and Wetherall 2002] Tennenhouse, D. L. and Wetherall, D. J. (2002). Towards an Active Network Architecture. In *DARPA Active NEtworks Conference and Exposition*, Páginas 2–15.

[Thubert et al. 2012] Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., and Pister, K. (2012). Rpl: Ipv6 routing protocol for low power and lossy networks. RFC 6550, IETF. Último Acesso: Agosto de 2018.

[Tsai et al. 2010] Tsai, C.-L., Chang, A. Y., and Ming-Szu, H. (2010). Early Warning System for DDoS Attacking Based on Multilayer Deployment of Time Delay Neural Network. *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*.

[Vijayan 2004] Vijayan, J. (2004). Mydoom lesson: Take proactive steps to prevent DDoS attacks. https://www.wired.com/2009/07/mydoom/. Último Acesso: Agosto de 2018.

[Vlacheas et al. 2013] Vlacheas, P., Giaffreda, R., Stavroulaki, V., Kelaidonis, D., Foteinos, V., Poulios, G., Demestichas, P., Somov, A., Biswas, A. R., and Moessner, K. (2013). Enabling smart cities through a cognitive management framework for the Internet of Things. *IEEE communications magazine*, 51(6):102–111.

[Wang et al. 2017] Wang, A., Mohaisen, A., and Chen, S. (2017). An adversary-centric behavior modeling of DDoS attacks. In *International Conference on Distributed Computing Systems*, Páginas 1126–1136. IEEE.

[Wang et al. 2015] Wang, Y., Liu, L., Sun, B., and Li, Y. (2015). A Survey of Defense Mechanisms against Application Layer Distributed Denial of Service Attacks. In *IEEE International Conference on Software Engineering and Service Science*, Páginas 1034–1037.

[Wani and Revathi 2018] Wani, A. and Revathi, S. (2018). Analyzing Threats of IoT networks using SDN Based Intrusion Detection System (SDIoT-IDS). In *Communications in Computer and Information Science*.

[Wired 2000] Wired (2000). Yahoo on Trail of Site Hackers. http://www.wired.com/2000/02/yahoo-on-trail-of-site-hackers/. Último Acesso: Maio de 2018.

[Xia et al. 2015] Xia, W., Wen, Y., Foh, C. H., Niyato, D., and Xie, H. (2015). A survey on Software-Defined Networking. *IEEE Communications Surveys Tutorials*, 17(1):27–51.

[Xiao et al. 2006] Xiao, B., Chen, W., and He, Y. (2006). A novel approach to detecting DDoS attacks at an early stage. *J Supercomput*.

[Zan et al. 2009] Zan, X., Gao, F., Han, J., and Sun, Y. (2009). A hidden markov model based framework for tracking and predicting of attack intention. In *International Conference on Multimedia Information Networking and Security*, volume 2, Páginas 498–501. IEEE.

[Zargar et al. 2013] Zargar, S., Joshi, J., and Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys Tutorials*, 15(4):2046–2069.

[Zhang et al. 2016] Zhang, L., Guo, Y., Yuwen, H., and Wang, Y. (2016). A Port Hopping Based DoS Mitigation Scheme in SDN Network. In *International Conference on Computational Intelligence and Security*, Páginas 314–317. IEEE.

[Zhao et al. 2017] Zhao, M., Kumar, A., Chong, P. H. J., and Lu, R. (2017). A comprehensive study of RPL and P2P-RPL routing protocols: Implementation, challenges and opportunities. *Peer-to-Peer Networking and Applications*, 10(5):1232–1256.